UXQB
International Usability and UX Qualification Board

CPUX-DS Curriculum and Glossary

Certified Professional for Usability and User Experience – Designing Solutions

Version 1.02 EN: 1. Oktober 2025

Published by: UXQB e. V.

Contact: info@uxqb.org

www.uxqb.org

Edited by UXQB Working Group “CPUX-DS”

Acknowledgments

The editors express their gratitude to the UXQB CPUX-DS working group and to the reviewers of this and previous versions of the CPUX-DS curriculum, for their constructive contributions to the technical contents.

Editor: Knut Polkehn; Co-Editor: Daniela Keßner

Contributors and reviewers: Chris Bailey, Kay Behrenbruch, Andreas Bleiker, Holger Fischer, Stefan Freimark, Thomas Geis, John Goodall, Morten Borup Harning, Rüdiger Heimgärtner, Oliver Kluge, Corinna Laabs, Rolf Molich, Sandra Murth, Elvi R. Nissen, Michael Richter, Chris Rourke, Guido Tesch, Norbert Zellhofer

Research & Support: Sina Flohr, Janine Galka, Christiane Geffert, Nils Hofmann, Hannes Hölzl, Laura Janitz, Lara Mhethawi, Alica Strigl, Alica Thissen, Jennifer Lynn Tune, Kathrin Wienrank

Copyright 2025, UXQB e. V.

Table of contents

Preliminary notes....6
1 Important perspectives for design activities....9
1.1 The baseline for Designing Solutions....9
1.2 Overview of design activities....12
1.2.1 Early design: conceptual modelling....13
1.2.2 First drafts: information architecture and interaction design....16
1.2.3 Refined design: interface design, information design, and sensory design....19
1.3 Iterating as needed and as the project demands....21
1.3.1 Iterate design decisions....21
1.3.2 Create alternatives to make a selection....21
1.3.3 Evaluating continuously in a formative way....22
1.3.4 Decide how to iterate in the project....24
1.4 Considering the whole user experience across all touchpoints....25
1.4.1 Reflecting psychological needs....25
1.4.2 Designing aesthetic interaction....27
1.4.3 Considering whole ecosystems....27
2 Early design....30
2.1 Design of user interfaces for the achievement of goals....30
2.1.1 Task-related operation to achieve user goals....30
2.1.2 User assistance: Explicit action-guiding information in addition to task objects and executable functions....35
2.1.3 Intended and unintended consequences of user interface design....36
2.2 Design activity: conceptual modelling....37
2.2.1 Creating task models for design....37
2.2.2 Creating interaction specifications based on task models....40
2.2.3 Identifying task objects, attributes, and executable functions in interaction specifications....45
2.2.4 Considering dependencies and creating variations....48
2.2.5 Communicating use scenarios to users and stakeholders....50
3 First drafts....52
3.1 Design activity: information architecture....52
3.1.1 Development of the information architecture....52
3.1.2 Enhance task objects with signposts....54
3.1.3 Structure task objects by determining connection paths....56
3.1.4 Make the information architecture visible for evaluation....57
3.1.5 Create the navigation structure using connection paths and signposts....62
3.1.6 Evaluate the information architecture....66
3.2 Design activity: Interaction design....68
3.2.1 Define task-related interaction sequences....68
3.2.2 Visualise interaction sequences....68
3.2.3 Structure the user interface based on all required views....70
3.3 Make design decisions tangible to get feedback....71

Copyright 2025, UXQB e. V.

3.3.1 Characteristics of visualisations across design phases 71
3.3.2 Typical types of visualisations 72
3.3.3 Benefits of visualising design decisions early and continuously 76
3.3.4 Iterating visualisations through evaluation 76
3.3.5 Guidelines for creating low-fidelity prototypes 76
3.3.6 Criteria for selecting prototyping tools 77
4 Refined design 79
4.1 Design activity: Interface design 79
4.1.1 Create the interface design by selecting, arranging, combining, and defining the behaviour of user interface elements 79
4.1.2 Appropriate use of user interface elements 81
4.2 Design activity: Information design 84
4.2.1 Principles for the presentation of information 84
4.2.2 Information reading and comprehensibility of content 86
4.2.3 Specific design recommendations for comprehensibility 89
4.3 Design activity: Sensory design 91
4.3.1 Design the user interface regarding its perception through relevant sensory channels 91
4.3.2 Gestalt laws 91
4.3.3 Colours, font sizes, and white space 93
5 Specific human needs 95
5.1 Accessibility 95
5.1.1 The importance of accessible design 95
5.1.2 Assistive technologies 96
5.1.3 Laws, standards and guidelines for accessible design 96
5.2 Design ethics 98
5.2.1 Influence by design and ethical consequences 98
5.2.2 Use of nudges as design elements of influence 100
5.2.3 Typical forms of nudges 100
5.2.4 Responsible design by applying common ethical standards 102
5.3 Cultural diversity 103
5.3.1 Intercultural user interface design (IUID) 103
5.3.2 Methods for interculturalisation (IUID process) 103
6 Aspects beyond the design activities 106
6.1 Managing stakeholders 106
6.1.1 Setting quality objectives for the project 106
6.1.2 Agreement on user feedback 108
6.1.3 Involve stakeholders 109
6.2 Setting the frame for design work 111
6.2.1 Deciding on the design process and methods 111
6.2.2 Deciding on appropriate systems of design recommendations to be used 111
6.3 Attending to implicit design tasks 118
6.3.1 Search 118

Copyright 2025, UXQB e. V.

6.3.2 Help and documentation 119
6.4 Documenting design decisions 120
6.4.1 The need for documentation 120
6.4.2 Explicit approach 121
6.4.3 Types of documentation of design decisions 121
Appendix 1: Overview of the CPUX-DS terms and activities 123
Appendix 2: Model Seminar 125
Appendix 3: Important changes to this document 128
Appendix 4: References & Index 129

Copyright 2025, UXQB e. V.

Preliminary notes

This document covers the human-centred design activity, “Design: Produce design solutions to meet user requirements”, part of “Human-centred design”, as defined in ISO 9241-210: “Human-centred design for interactive systems”.

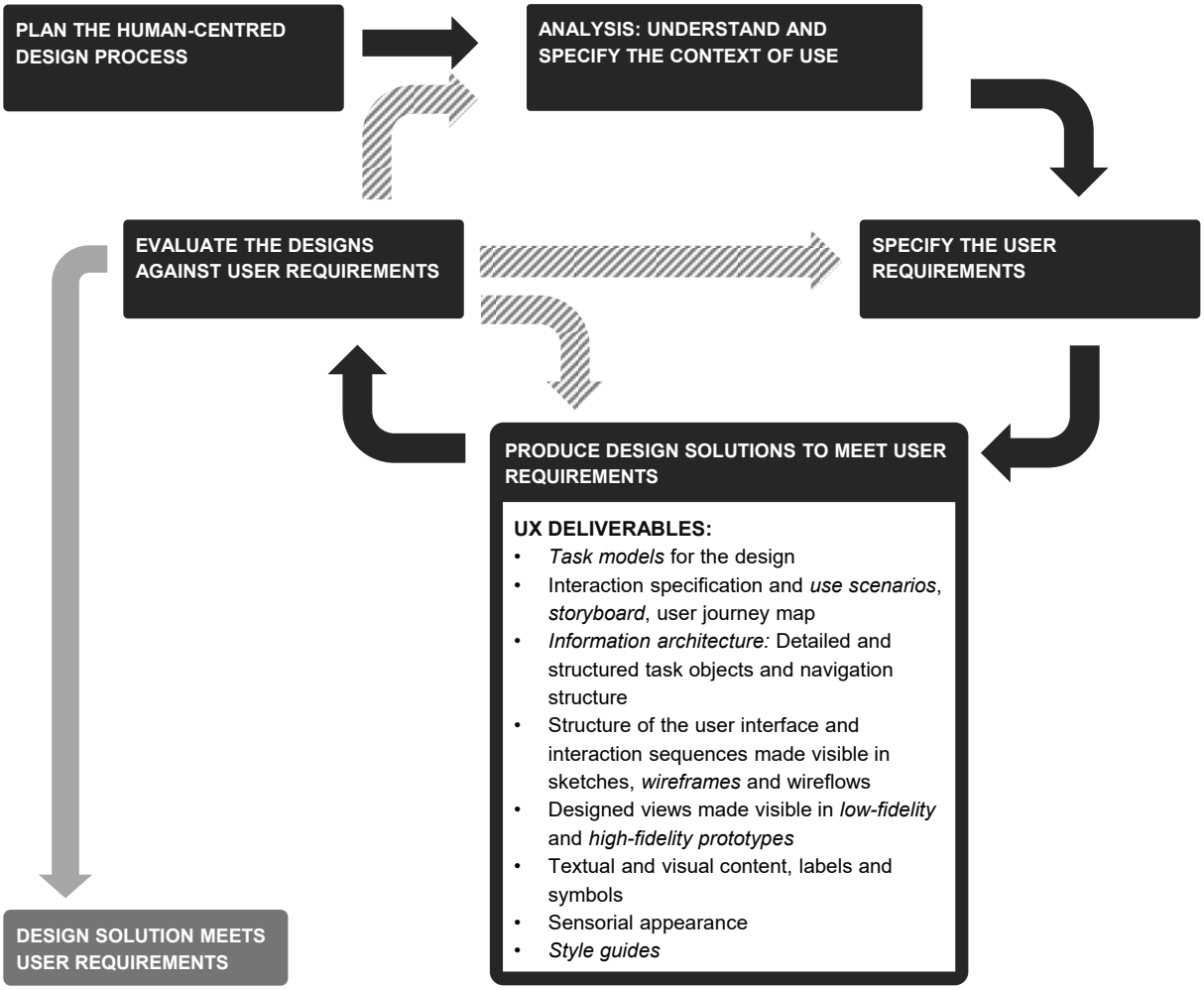


Figure 1. The activity “Designing Solutions” shown in the interdependence of human-centred design activities according to the ISO 9241-210 standard and related UX Deliverables (deliverables in *italics* are defined in CPUX-F)

The international standard ISO 9241-210 requires designers to understand the following sub activities for creating design solutions that meet the user requirements:

a. designing user tasks, user-system interactions, and user interfaces to meet user requirements, by considering the entire user experience
b. making design solutions more concrete (for example, creating use scenarios, simulations, or prototypes)
c. altering design solutions in response to user-centred evaluation and feedback
d. communicating design solutions to those responsible for their implementation.

This document defines what you need to know to attain the certification for “Certified Professional in Usability and User Experience – Advanced Level Design Solutions (CPUX-DS)”. The certification process will only cover your knowledge and understanding of the information and concepts in this document; it will not cover course material from CPUX-F.

The certification process aims to evaluate:

- your knowledge and understanding of the terms and topics in this curriculum
- your ability to apply your knowledge and understanding in practice.

Copyright 2025, UXQB e. V.

Who is CPUX-DS for?

CPUX-DS is for anybody involved in designing solutions, including

- usability and user experience professionals, who design interactive systems
- non-usability professionals involved in the design and implementation of interactive systems – including product managers, product owners, and system developers
- those learning about the field of human-centred design, who may be considering a position similar to those described above.

Holding a Foundation in Human-Centred Design (CPUX-F) certificate is not a precondition for taking the CPUX-DS exam. However, we strongly recommend familiarising yourself with the terms and concepts in the CPUX-F curriculum before studying the CPUX-DS curriculum.

What is the focus of CPUX-DS?

The scope of this curriculum is as follows:

- Fundamental HCD approach vs. Exploring current, popular design methods:
  - The curriculum is based on the human-centred design process, a process that needs to be understood to create human-centred design solutions based on the analysis of the context of use. A lot of currently discussed methods – including Lean UX and User Story Mapping – complement the HCD approach, but will not be discussed in this curriculum.
- Pragmatic approach vs. Systematically founded approach:
  - The curriculum focusses on a systematically founded approach so that inexperienced designers will find a basis for their work and experienced designers will know where and when to skip or adapt methods.
- Hardware vs. Software/digital interfaces:
  - The focus of the curriculum is on digital interfaces, which is reflected in the explanations and examples. All the approaches and methods in this curriculum can be applied to hardware design.

What does the CPUX-DS certification demonstrate?

Holders of the CPUX-DS certificate have demonstrated – through theoretical and practical examination – that they know, understand, and can apply relevant terms, design techniques, and methods in the context of human-centred design.

They have demonstrated their ability to

- apply a human-centred approach to producing design solutions
- create and develop interaction specifications and use scenarios to describe how an intended interactive system will be used, based on understanding the context of use and user requirements
- design information architectures that facilitate efficient access to information and functionality
- design navigation structures
- design effective ways of guiding and assisting the user
- specify and design the user interface based on interaction specifications and use scenarios
- design and iterate sketches, wireframes, wireflows, and low-fidelity and high-fidelity prototypes for evaluation.

Copyright 2025, UXQB e. V.

How to read the CPUX-DS curriculum

Each learning unit of the curriculum contains sections that comprise

- a short summary
- learning objective(s) at one or more of the levels, ‘know’, ‘understand’ and ‘be able to’
- learning content for the section, which includes term definitions, explanatory statements, and examples to support the learning objectives

Terms that are new in a learning unit appear in bold in the summary at the beginning and in the term definitions of a learning unit. Some sections may include terms from the CPUX-F glossary. If the definition of a term has been extended beyond its CPUX-F definition, the original CPUX-F definition (which will not be included in the certification test) will be clearly marked.

An example of a term from the CPUX-F Glossary extended in CPUX-DS.

Task model

CPUX-F Definition
A description of the subtasks within a task that have to be carried out to reach the user’s goals.

Task models form the basis for the development of interaction specifications and use scenarios are identified in the context of use analysis.

The index at the end of this document contains all glossary terms and their respective page numbers.

The appendix contains an overview of all learning units and a further reading list that includes all literature that has been referenced in the curriculum. It also includes a timetable for a typical training course, which may be used by training providers as a guide when developing their own seminars.

Preparational training courses

To prepare for the CPUX-DS certification test, we strongly recommend attending a training course delivered by one of our Recognised Training Providers, although this is not mandatory.

The UXQB.org website

All relevant information about CPUX certifications is freely available on UXQB.org – the website of the International Usability and User Experience Qualification Board.

The information on UXQB.org includes

- a complete list of recognised CPUX training providers and available courses
- the examination regulations and the checklist for the practical test
- public example tests (both theoretical and practical)
- informative chapters.

Copyright 2025, UXQB e. V.

1 Important perspectives for design activities

Designing Solutions describes the process of implementing the activity, “Design: Produce design solutions to meet user requirements” in a design project, as described in [ISO 9241-210]. The approach to designing user tasks, user-system interactions, and user interfaces to meet user requirements serves as the baseline for the human-centred design process.

The design process includes design activities within the phases of Early design, First drafts, and Refined design. The curriculum presents a compact and structured overview of the design activities in Designing Solutions, and every design activity and its sub-activities is explained in depth in the referenced chapters.

The most appropriate design for an interactive system cannot typically be achieved without iteration. Designers must decide which iterations are necessary.

Different UX deliverables help to make design solutions more concrete. They also influence design solutions in response to user-centred evaluation and feedback. Furthermore, these deliverables help in communicating design solutions to those responsible for their implementation. All design activities must consider the whole user experience.

1.1 The baseline for Designing Solutions

When **designing solutions,** the designer must incorporate the information provided by the **context of use analysis** to reach a match between the users’ **mental models**, the designer’s **conceptual model,** and the **system image** as the perceivable part of the interactive system.

Learning Objectives	
1.1.a	Know which deliverables from the context of use analysis are the baselines for the successful completion of design activities.
1.1.b	Know the relationship between users’ mental model, the system image, and the conceptual model of the designer.

Designing Solutions

A field of action within human-centred design where solutions are designed based on the results of context of use analysis, to meet user requirements**.**

Initial ideas are developed (Early design), the user interface is structured in line with the tasks that need to be supported (First drafts), and, finally, implemented (refined design). Transferring this work to use scenarios and prototypes enables users to be continuously involved during the iterative design process.

An understanding of the context of use information is important in getting the design right from the user’s perspective.

Context of use analysis

The process of planning, gathering, and documenting authentic context of use information, identifying the user needs contained therein, and specifying user requirements.

The designer needs to know the results of the context of use analysis. They are documented in context of use descriptions; for example, in user group profiles, personas, as-is scenarios, user journeys, task models, and – for every supported task – user requirements that are structured along the subtasks. If a designer is aware of the context of use descriptions, they

Copyright 2025, UXQB e. V.

can work on a meaningful user interface from the user's perspective throughout the design activities.

A user interface should always be designed in a way that the dialogue requires only minimal interaction knowledge or explicit training for users. The interaction should be as meaningful and natural as possible and resemble an interaction experienced in the real world or in familiar contexts. This way, tasks can be completed solely with the help of knowledge from the user's world, represented in their mental models.

Mental model

CPUX-F Definition
The perception people have of themselves, others, the environment, and the things with which they interact.

Humans build mental models from patterns retrieved by replicated experiences about the sequence of steps in a workflow, about where to find objects, or how these relate to one another.

The development of appropriate mental models depends on how the designer's conceptual model is understood and learned during users' interactions with the system image. For a meaningful interaction, the designer must create a suitable conceptual model which will be implemented as the system image. The conceptual model is appropriate if it matches the expectations of the users, which are based on their mental models.

Conceptual model

The designer's understanding of how each user's task should be performed and supported by the interactive system.

The conceptual model includes task models for the design and for the intended interaction for each task supported by the interactive system, the required task objects and their relationships, and the executable functions to create, modify and/or gather information about task objects as specified by the designer.

To support the user, designers must consider the user's mental model and build complex interactions aligned with it. The user should be able to easily assimilate the conceptual model into their mental model. Ideally, the designer's conceptual model reflect the user's mental model and the resulting system image will be a perfect reflection of both.

System image

All parts of an interactive system that are perceivable for the user, including associated information about the interactive system that is provided to the user before and during usage.

A system image is the representation (visual or non-visual) of the conceptual model of the designer(s) who created the interactive system. The translation of the conceptual model into a system image is sometimes flawed. Any mismatch between these three concepts (mental model, conceptual model, and system image) will inevitably lead to user interfaces that cause the user to struggle as their mental model is updated.

> Example: A new kettle shows a red glow while it is off, and a blue glow once switched on (system image). This aligns with the designer's conceptual model where the off-state is usually represented by red and the on-state by blue. Unfortunately, this conflicts with the user's mental model, according to which red indicates hot and blue indicates cold. This could cause users to burn their fingers when touching the kettle.

Copyright 2025, UXQB e. V.

Example: In the early days of online shops, the interaction of purchasing items often failed to effectively support users, leading to countless aborted shopping processes. As processes then were not as standardised as today, there were no explicit conventions about how the ordering process in an online shop should work and what role the shopping cart plays – there was no link between what happened online and what happened in the real world. The introduction of the shopping cart concept and the strict adherence to the user's mental model of a real-life shopping process improved the conversion rates in online shops significantly.

Innovation sometimes reverses the formation of mental models. This leads to brand new mental models that change real-world expectations.

Example: Our mental model of cooking with recipes is such that we decide on a recipe then buy the ingredients. However, some "Smart" applications allow us to input the ingredients we already have, before selecting a recipe from a list of suggestions.

To be at the right starting point in Designing Solutions, designers must ensure that they can execute design work that is informed by the user's perspective. The design should be carried out by people intimately familiar with the results of the context of use analysis or in close collaboration with people who conducted the research.

It might be seen as suboptimal to let one team analyse the context of use and derive the user requirements, then hand over the results to another team to do the design, but it is often necessary for larger projects or in companies with dedicated role models.

Copyright 2025, UXQB e. V.

1.2 Overview of design activities

When designing solutions, designers make implicit or explicit decisions at different levels, for example, about the appearance of a user interface element, or about how the user will achieve their goal in the future, with the help of the interactive system. Every **design decision** – whether made consciously or unconsciously – will be reflected in the future system.

This curriculum describes all the **design activities** and their sub-activities that lead to conscious design decisions. In practice, not all of these design activities are applied systematically and completely. Being aware of them helps the designer to make difficult design decisions consciously, to better reflect on unconsciously made design decisions, and to discuss them with stakeholders. Design activities are structured in three phases.

In Early design, the designer starts with the first design activity: making design decisions in **conceptual modelling** to determine the future interaction between the user and the interactive system.

In First drafts, the designer further develops the conceptual model by considering the **information architecture** and **interaction design**.

In Refined design, the designer develops the interface design by selecting user interface elements and defining their behaviour. **Information design** ensures the information presented to the user is comprehensible and consistent. The designer also employs aspects of **sensory design** to ensure that the user interface can be perceived by all users, regardless of any impairment.

Learning Objectives	
1.2.a	Understand which design activities are relevant for Designing Solutions.

The designer's actions and decisions in designing a solution can be summarised in design activities.

Design activity

A step in the multistep process of Designing Solutions. Each step focuses on specific design decisions.

This curriculum defines the following design activities for the three design phases:

- Early design: conceptual modelling
- First drafts: information architecture, interaction design
- Refined design: interface design, information design, sensory design

When performing a design activity, the designer manages several sub-activities during which they make design decisions.

> An example of a sub-activity: In interaction design, the designer defines task-related interaction sequences, visualises interaction sequences, and structures the user interface and the required views

In this curriculum, some very systematic procedures and methods are presented explicitly within all the design activities and their sub-activities.

In reality, time and resources often prevent designers from being able to perform all of these design activities in-depth, however it is valuable and helpful for designers to understand the system of design activities and their applicable methods in order to

- make conscious design decisions

Copyright 2025, UXQB e. V.

- carefully reflect alternative design decisions be able to question already (unconsciously) made design decisions
- be able to adapt the approach to the time and resources available
- communicate design decisions in a way that they can be understood by third parties, for example, the project team, users, other stakeholders
- discuss design decisions with stakeholders.

  Example: A designer works on a project with a tight schedule. Due to limited time, it is not possible to create task models for the design and interaction specifications in a detailed and systematic way to identify the task objects, attributes, and executable functions. Nevertheless, they listen carefully to the user interviews and pay close attention to which objects users work on, to understand the characteristics of those objects (attributes) that might play a role in the user interface. They consider those observations when creating wireframes. The latter discussion with the product owner about the wireframes goes surprisingly well. They understand the designer's arguments from the user's point of view and why certain objects must already be visible on the start page.

Design decision

A decision made during the execution of design activities regarding the interaction of the user with the user interface.

A substantial effort may be required in creating design deliverables that allow for sustainable design decision making. Each design decision represents a certain rule that the interactions should follow. The documentation of that decision ensures consistency in design and the reuse of decisions.

Figure 2 gives an overview of the design process, the design activities, their sub-activities, and deliverables. The appendix contains an alternative visualisation in Figure 51.

All ot the design activities presented, along with the associated methods are described in the corresponding chapters of this curriculum. These methods are neither mandatory nor exclusive. In everyday practice, other methods that are suitable to produce design solutions that satisfy user needs and the needs of other stakeholders may be used.

1.2.1 Early design: conceptual modelling

In Early design, the designer starts with activities for conceptual modelling by anticipating the future use of the interactive system.

Conceptual modelling

An activity that determines the task models for design, the interaction specifications, the task objects and their attributes, and the executable functions, and transforms them into tangible use scenarios for evaluation with users and other stakeholders.

Conceptual modelling is based on the context of use information. During conceptual modelling , dependencies between different tasks are considered and different alternatives are created.

Conceptual modelling follows the procedure outlined below.

Copyright 2025, UXQB e. V.

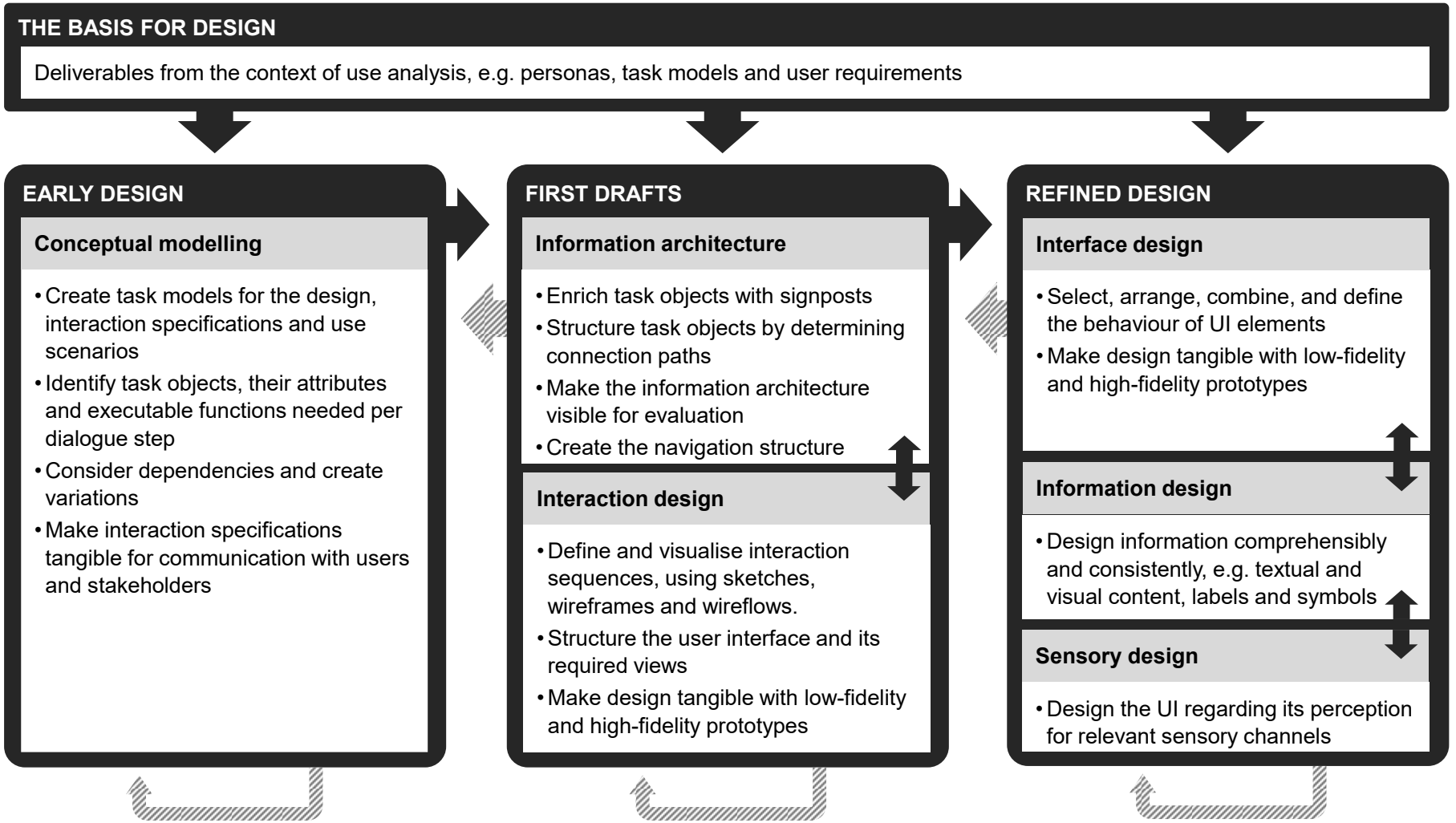


Figure 2 Design process in Designing Solutions (Overview of CPUX-DS activities)

Copyright 2025, UXQB e. V.

1.2.1.1 Create task models for the design, interaction specifications, and use scenarios

The designer uses the context of use information to develop ideas for future interactions that support the user in achieving their goals.

Alternative, optimal approaches to solutions that support users in achieving their goals in the user-system-interaction can be developed alongside the available task models, as-is-scenarios, story boards, or user journey maps. Any methods and notations to visualise processes, tell stories, or put oneself in the system's position are suitable to find good ideas. Different situations of use can be explored, but edge cases should be avoided.

This process results in task models for the design and interaction specifications for each task to be supported, which describe the interaction between the user and the system to support the task at hand as well as possible. These can be converted into narrative use scenarios to receive feedback from users and other stakeholders.

This sub-activity is covered in detail in 2.2.1 and 2.2.2.

1.2.1.2 Identify task objects, their attributes, and executable functions needed per dialogue step

Interaction specifications, embedded in the task model for design and the user requirements as well as use scenarios, help the designer to extract task objects and their attributes, and identify the functionality needed for the interactive system.

For each supported task, an overview of the required task objects is created, that is, the objects that the users create, modify, or gather information on when performing tasks. Overviews of attributes and associated executable functions are created for each task object.

The task models for the design, the interaction specifications for each task, the task objects and their attributes as well as the executable functions must be checked iteratively for their completeness, plausibility, and freedom from redundancies.

This sub-activity is covered in detail in 2.2.3.

1.2.1.3 Consider dependencies and create variations

While creating task models for design and interaction specifications, the dependencies between all the tasks that have to be supported by the interactive system must be considered. Often, these dependencies lead to adjustments of some task models for design.

Creating different variants of interaction specifications for the same task supports the designer in dealing with dependencies and choosing variants that can be transferred into use scenarios, which can be evaluated by users and other stakeholders.

This sub-activity is covered in detail in 2.2.4.

1.2.1.4 Make interaction specifications tangible for communication with users and stakeholders

The specified interaction can be used to derive use scenarios, storyboards, or user journey maps in order to evaluate ideas and models with users and other stakeholders. Use scenarios illustrate how users reach their goals step by step when using the interactive system. Task models, use scenarios, identified task objects, their attributes, and the executable functions serve as an input for subsequent design activities. Use scenarios that can be experienced by stakeholders help to prioritise the value during use from a user's perspective, instead of solely focusing on business impact, available resources, or technical feasibility.

Copyright 2025, UXQB e. V.

This sub-activity is covered in detail in 2.2.5.

1.2.2 First drafts: information architecture and interaction design

Based on the results of Early design, the designer works out the conceptual model in terms of the information architecture and a suitable interaction design.

Information architecture
CPUX-F Definition
The naming and structuring of the information that must be accessible to the user.
Based on the conceptual model.

The information architecture describes the labels and the structure within all task objects, their attributes, and the executable functions so that users can find and understand information.

To help the user anticipate the conceptual model of the designer in the system image, the objects represented in the user's mental model must be represented in the user interface in a comprehensible and discoverable way.

The information architecture is created by enhancing the task objects with signposts (required to locate task objects and executable functions), structuring those task objects, making this structure visible for evaluation, and creating the navigation structure.

In interaction design, based on the conceptual model and the information architecture, the interaction sequences are defined, the user interface is structured in required views, and the design is evaluated using prototypes.

1.2.2.1 Enhance task objects with signposts

Enhancing task objects means identifying the signposts by determining the task objects and executable functions that are related to the task object in focus. This sub-activity aims to prepare the overall structure across all the task objects and create the navigation structure.

Signposts must be titled to match the mental model of the user rather than being based on the system’s perspective. For example, when naming a signpost to an executable function, the designer chooses a label from the user’s point of view, such as “reschedule connection”, instead of using a technical term, such as “update connection information”.

Copyright 2025, UXQB e. V.

An example of an enhanced task object:

Table 1 A detailed task object enhanced with signposts

Task object: train connection	Details
Attributes	Departing station, departing platform, arrival station, date and time of departure, date, and time of arrival…
Executable functions (and the actions they support)	Edit connection • Select departing station, select arrival station, Input departure date… • Show connection • …
Signposts to executable functions and other task objects (calls to action):	Save a connection…
Signposts from other navigational points (trigger words):	find a connection…

Two different perspectives can be taken on signposts:

- "Calls to action":
  - Signposts from each task object to executable functions; for example, "reschedule connection" or
  - Signposts from each task object to other task objects; for example, "my tickets"
- "Trigger words":
  - Signposts from any other navigation point to the task object; for example, "my next journey".

By following the signposts to a task object, users can identify connection paths.

This sub-activity is covered in detail in 3.1.2.

1.2.2.2 Structure task objects by determining connection paths

To supply a semantic structure that corresponds to the user's mental model, the super- and subordination – the hierarchical relationship – of task objects must be considered (for example, products could be part of a shopping cart or product overview). Defining connection paths using the identified signposts helps to identify the required navigation structure (for example, it should be possible to navigate from a product to the detailed product description).

This sub-activity is covered in detail in 3.1.3.

1.2.2.3 Make the information architecture visible for evaluation

To evaluate an information architecture, it should be made visible for the internal team, the users, and other stakeholders.

Different types of presentations offer different perspectives on the created structure within all task objects, their attributes, and executable functions.

User tasks can be explored in the visualised structure. To check whether the resulting semantic structure of the user interface aligns with user expectations, it can be evaluated with users, for example, using the card sorting method.

This sub-activity is covered in detail in 3.1.4 and 3.1.6.

Copyright 2025, UXQB e. V.

1.2.2.4 Create the navigation structure

If the semantic structure of the task objects fits the users' mental model, the identified signposts are arranged into the navigation structure according to user expectations and requirements, by serving as labels for navigation elements within the navigation system. Each entry of a signpost in a navigation tree or path represents a guide for the user that helps in locating a task object or an executable function. The prioritisation and selection of entries within the navigation must be based on the results of the context of use analysis, for example, based on the prioritisation of user tasks.

For some interactive systems, the navigation structure depends on the delivery channel. For example, for an application running on a mobile device as well as on a desktop, the mobile navigation structure may differ from the desktop navigation structure.

The designer selects the appropriate navigation system based on their own professional experience and on the interactive system to be designed.

The navigation system and the selected navigation elements must be tested with users, for example, using the tree-testing method. Alternative navigation systems must be tested if the designer is in any doubt.

This sub-activity is covered in detail in 3.1.5 and 3.1.6.

1.2.2.5 Define and visualise interaction sequences with sketches, wireframes, and wireflows

Information architecture serves as the basis for the interaction. Interaction design defines the order of stages needed to complete tasks with the interactive system and the required views from the user's perspective.

Interaction design
The design activity of structuring the user interface in views, pages / screens as well as determining the interaction sequences.

The designer translates the dialogue steps specified in the interaction specification into a sequence of interactions that are represented in one or more views in the user interface. These interaction sequences need to be logical from the user's perspective.

The design decisions on the structure of the user interface and its views can be visualised in sketches, wireframes, wireflows, and low- and high-fidelity-prototypes, to get feedback from internal team members, users, and other stakeholders.

All the details on these sub-activities can be found in 3.2.1 and 3.2.2.

1.2.2.6 Structure the user interface and its required views

The views for all interaction sequences are what define the user interface structure.
An iterative refinement and adaptation of the user interface structure will be necessary, depending on the number and complexity of use scenarios involved. The interaction sequences specified by the structure of views must be evaluated with users.

This sub-activity is covered in detail in 3.2.3.

Copyright 2025, UXQB e. V.

1.2.2.7 Make the design tangible with low-fidelity and high-fidelity prototypes

Prototyping solutions helps to identify suitable alternatives and to get feedback from users and other stakeholders. Some rules need to be considered:

1. Rely as much as possible on the experience embedded in established design patterns, if they are applicable in the given context of use. The use of established design patterns must not restrict the scope of any solutions. Sometimes, a problem may require a different solution to any offered by existing design patterns.
2. Create as many low-fidelity prototypes as practical (divergence). Each prototype represents new, tried out aspects of the conceptual model. Failing early and often results in understanding what users truly need.
3. Evaluate the prototypes with users and other stakeholders to get feedback.
4. Based on the results, come up with new concepts as required and evaluate them.
5. Rule out the concepts with users and other stakeholders based on the results of usability tests, considering feasibility, practicality, and business impact. Merge concepts that have proved themselves in tests into the next generation of prototypes (convergence).
6. Create or adopt a style guide covering both syntactic and semantic aspects of design decisions. If applicable, regularly check to make sure prototypes conform to the style guide and define procedures for enforcing the style guide (governance).

The navigation structure, interaction sequences, required views, and the first prototypes are often developed in parallel (iteratively).

This sub-activity is covered in detail in 3.3.

1.2.3 Refined design: interface design, information design, and sensory design

Refined design is the process of designing every single screen or page in an interactive system by using the appropriate user interface elements, designing for comprehensibility and consistency of all kinds of information, and for user perception.

1.2.3.1 Select, arrange, combine and define the behaviour of UI elements

For each individual view and page that has been defined in the interaction design, it must be determined which interface elements are used and how they are specified.

Interface design

The selection, combination, arrangement, and the definition of the behaviour of user interface elements for all views as required for the interaction between user and user interface.

The user interface elements must be selected according to the user's expectations and user requirements so that the users are well supported in achieving their goals.

> An example of selecting, arranging, combining, and defining the behaviour of UI elements: Radio buttons were selected as user interface elements to support a single choice. They were combined with a field below to show instructions. After changing the selected radio button, the instruction was exchanged.

This sub-activity is covered in detail in 4.1.1 and 4.1.2.

Copyright 2025, UXQB e. V.

1.2.3.2 Design meaningful information comprehensibly and consistently

Design all the information in the views to be comprehensible and consistent.

Information design

Designing the presentation of meaningful information (text, labels, icons, symbols, etc.) to support the comprehensibility and interpretability of the contents presented in the task objects.

The designer must design all elements containing meaningful information such as textual and visual content, labels, symbols, or icons considering users' expectations, prior knowledge, and requirements.

This sub-activity is covered in detail in 4.2.

1.2.3.3 Design the UI regarding its perception across relevant sensory channels

The user interface is optimised to ensure it can be perceived by all users via their available sensory channels.

Sensory design

Designing the way users perceive the user interface across the available sensory channels.

Sensory channels are also referred to as “modalities” and include visual (seeing), auditory (hearing), tactile/haptic (touching), olfactory (smelling), and gustatory (tasting).

Perceptibility can be influenced both positively and negatively through all sensory channels.

This sub-activity is covered in detail in 4.3.

Copyright 2025, UXQB e. V.

1.3 Iterating as needed and as the project demands

The most appropriate design for an interactive system cannot typically be achieved without iteration. Designing solutions is an **iterative process**. This process includes:

- the approach of creating different solution variants and selecting the best alternatives, – often called **Design Darwinism**
- **validation** by questioning already made design decisions through continuous **formative evaluation,** which gathers **user feedback** and involves stakeholders across all design activities.

For a successful iterative process, designers must make decisions on which work products are iterated, with whom, when, and how, according to the specific needs of the project.

Learning Objectives	
1.3.a	Know what it means to iterate.
1.3.b	Know what is required to successfully follow the Design Darwinism approach.
1.3.c	Know how to find ideas and create variations for iterations.
1.3.d	Know how to evaluate design solutions with users or other stakeholders.
1.3.e	Know which aspects designers need to decide on to iterate appropriately.
1.3.f	Know when iterations can take place during Designing Solutions.

1.3.1 Iterate design decisions

It is crucial to think broadly and produce a variety of ideas on how users might interact with the user interface, especially during the Early design phase and while creating the first drafts of a solution. It is counterproductive to focus on just one or two ideas early in the process. Moreover, to iterate and include different perspectives, the designer must be open to involving third parties in the creative process.

Iterative process

CPUX-F Definition
Repetitive. An iterative process repeats steps in the human-centred design process until a usability evaluation of the user interface shows that the user requirements have been adequately met.

Iteration refers to assessing the already-made design decisions as well as any alternative design decisions from different perspectives (different parties), to select the most promising variants and work on them or revise them as a starting point for the following design activity.

1.3.2 Create alternatives to make a selection

A good approach to iteration is to create alternatives and select the best of them.

Design Darwinism

An approach in which different variants of work products are created in parallel and the best solution prevails against other solutions through several continuous iterations involving different parties (the internal team, users, and other stakeholders).

Copyright 2025, UXQB e. V.

Design Darwinism can only be successful under certain conditions: Finding the best alternative to a solution from the users' perspective depends on whether

- the set of possible solution variants includes variants that are valid from the perspective of the context of use
- the individual variants differ sufficiently from each other to create added value for users and other stakeholders
- the selection of a variant was fair with the participation of all parties.

To ensure that a variant is appropriate from the perspective of the users and other stakeholders, the designer must validate it.

Validation

A process of determining whether all stakeholder requirements have been effectively implemented from the perspective of all stakeholders.

Creating and selecting variants is a crucial factor in making good design decisions in any design activity and must be planned into every project. This process can start with the development of variants of task models for the design (see chapter 2.2.4).

> Example: A designer creates several different variants of task models for the design. In a workshop, the users and all the other stakeholders are enthusiastic about a particularly innovative variant of the resulting use scenarios. The designer selects this innovative variant to take forward, develops an information architecture based on it, and creates different alternatives of interaction sequences in paper prototypes. After some usability tests, they realise that there is another very important subtask. They add this subtask to the task model for design, adapt the information architecture, and create a new prototype.

In the process of creating variants of possible solutions,

- creative methods can be applied to boost creativity and thinking “outside the box”
- team members or stakeholders can be involved to widen the variability of ideas and perspectives
- users can be invited to take part in the creative process
- any neutral third party can serve as a sparring partner for discussing and creating ideas.

An initial assessment and clustering of ideas is often done by the people involved in their creation. The goal is to find a reasonable set of different but equally feasible ideas in terms of the support of user goals that subsequently should be evaluated with stakeholders, to critically question them from the point of view of user requirements and stakeholder requirements. The best solution will then be detailed in the course of the design process.

1.3.3 Evaluating continuously in a formative way

In Designing Solutions, formative evaluation can take place on any work product that is created during design activities.

Formative evaluation

The process through which information about the usability of an interactive system is gathered in order to improve the interactive system.

The feedback from users based on the formative evaluation of tangible UX deliverables of each design activity typically results in revising design decisions. It also unveils further details in the context of use and may lead to updated user requirements.

Copyright 2025, UXQB e. V.

User feedback

The user's responses on how they perceive the system under development or any work product that it is represented by. This feedback can result, for example, from a user survey, a focus group, or a usability test.

According to the mindset of continuous evaluation, the designer must evaluate UX deliverables with users and other stakeholders whenever the opportunity is there. Table 2 gives an overview of frequently produced UX deliverables.

Table 2: Frequently produced UX deliverables, recipients, and examples of evaluation methods

UX Deliverable	Evaluator(s)	Evaluation Method
• Use scenario, • User journey map	Internal Team, Stakeholders, Users	Informal feedback, Review, Focus group
• Information architecture	Users	Informal feedback, Card sorting
• Navigation structure	Users	Informal feedback, Tree testing
• Sketch	Designer, Internal Team	Informal feedback
• Wireframe, • Wireflow	Internal team, Stakeholders, UX-Professionals, Users	Informal feedback, Review, Inspection
• Low-fidelity prototype	Internal team, Stakeholders, Users	Informal feedback, Cognitive walkthrough, Usability test
• High-fidelity prototype, • Eventual Solution	UX-Professionals, Stakeholders, Users	Inspection, Review, Interview, Cognitive walkthrough, Usability test, User surveys

This approach of continuous evaluation of UX deliverables within Designing Solutions has several advantages:

- The defects and misunderstandings of scenarios, task models, user requirements, etc., can be caught early in the design phase before too much time is invested in unusable designs
- The need for guessing what is most important to users is eliminated by updating the designer with early and continuous feedback from the users and other stakeholders based on evaluations: for example, a usability test or inspection.
- Many design alternatives can be considered, and the human tendency to commit to one solution too early and miss out on suitable design alternatives is avoided.

Formative evaluation must be mandated and planned into the design project. From the point of view of the person initiating the design project, the participation of real users might not seem important, especially if it is considered time-consuming and costly. Nevertheless, it is the responsibility of the designer to highlight the importance of user feedback and demand the involvement of users, to avoid mistaking requests from stakeholders as valid user requirements. Evaluation results must be discussed and agreed upon with stakeholders (see chapter 6.1.3).

Copyright 2025, UXQB e. V.

1.3.4 Decide how to iterate in the project

Regarding iterations, designers have to make different decisions: Which variants of work products must be iterated? How and with whom will the iteration take place? When will iteration happen?

Not every design project requires all the design activities shown in Figure 2 to be performed to their full extent. Depending on the needs of the project (How does the product development process work? What is the project status? What resources are available?), designers should carefully choose which activities to perform, to what extent, and which to skip, and make conscious decisions.

> Example: A project team optimises the website of a theatre. A usability test has shown that the navigation structure fits the user needs, but users cannot interpret the information on ticket prices correctly. The design team starts to question the design decisions made in information design and decides to discuss the usability test results in a stakeholder workshop. Based on the discussions, the designers revise the text and the symbols, and make conscious decisions about the use of colours, to make the information on ticket prices more comprehensible. The design team now decides to conduct another round of usability tests and finds out that two of the variants work equally well. After consultation with the development team and other stakeholders, they decide to implement one variant.

Regardless of the selected design activities and their order, they should not represent isolated tasks. The work products resulting from each design activity inform subsequent design activities.

Iterations take place within each individual design activity and also between two design activities, to ensure that the design decisions satisfy the user needs within the specific context of use.

> Example: While working on first drafts, the designer may conclude that users are better supported in a different way than the one previously assumed. This leads to the modification of use scenarios which have already been agreed upon.

Iterations can even take place at whole-project level. The result of a design project may lead to the start of a new design project. In this respect, designed solutions can be iterated across projects.

Copyright 2025, UXQB e. V.

1.4 Considering the whole user experience across all touchpoints

To consider the whole user experience, the designer must make design decisions to help users achieve their goals and tasks, and to fulfil their expectations, considering all aspects of the context of use for design. This includes the resources used by the user and the environment(s) they are in while performing tasks. Additionally, they must consider the users' psychological needs and make decisions about the experiences they want to create by determining the feelings and emotions they want to provoke in users during the interaction.

All design activities must shape the interaction in the form of the whole user experience. They should consider aspects of the **aesthetics of interaction** and users' psychological needs.

The interaction between the user and the interactive system does not take place in a vacuum. It is embedded in an ecosystem of the use of a large number of other interactive products, systems, or services, in a wide variety of use environments (physical, social, technical), in which information is exchanged between user and system via different channels.

To consider the whole user experience, the designer must act from the perspective of various relevant **touchpoints** with the interactive product, system, or service. The use of **design thinking** methods can be helpful here.

Learning Objectives	
1.4.a	Understand which aspects are important for the whole user experience.

1.4.1 Reflecting psychological needs

To create a certain experience for the user, the designer must decide which psychological needs of the users they want to fulfil. Each psychological need can be supported in a specific way in interaction design and, thus, serves as a guideline for the identification of user requirements related to user experience. [LeDiHa14] describes the following psychological needs.

Copyright 2025, UXQB e. V.

Table 3 Psychological needs

Psychological Need	Description
Autonomy	Feeling that you are the cause of your own actions rather than feeling that external forces or pressures are the cause of your action. For example, deciding to leave the office as planned instead of waiting for an update to finish.
Competence	Feeling that you are very capable and effective in your actions rather than feeling incompetent or ineffective. For example, being able to use known shortcuts to perform a task in one step instead of following a multi-step wizard to do the same.
Relatedness	Feeling that you have regular and intimate contact with the people who care about you rather than feeling lonely and uncared for. For example, being able to see that colleagues start their work in the morning, even if they are not in the same office.
Popularity	Feeling that you are liked, respected, and have an influence over others rather than feeling like a person whose advice or opinion nobody is interested in. For example, being able to see that colleagues frequently use a document that you worked on for a long time.
Stimulation	Feeling that you get plenty of enjoyment and pleasure rather than feeling bored and under stimulated. For example, a visual representation of a growing tree symbolises the progress of your work. An unbalanced treetop signals missing content; the colour of the leaves, their relevance.
Security	Feeling safe and in control of your life rather than feeling uncertain and threatened by your circumstances. For example, knowing that all necessary steps related to specific demands have been taken instead of being nervous while waiting for feedback.
Meaning	Feeling that you are developing in line with your potential and making life meaningful rather than feeling stagnant, and that life does not have much meaning. For example, if a website donates to reforestation with money collected through searches, the user immediately sees how many trees will be planted thanks to their search activity, or the activity of others.

When considering individual psychological needs while designing the user experience, the designer should not only consider these needs in the context of use descriptions – for example, personas or as-is scenarios – but also consider them in each formative evaluation.

Copyright 2025, UXQB e. V.

1.4.2 Designing aesthetic interaction

In addition to the appearance and the look and feel of the designed user interface, the interaction itself can be subject to the user's experience of aesthetics.

Aesthetics of interaction

Results from the harmony between the execution of actions by the user and the intended experience during the interaction.

The aesthetics of interaction are based on the interplay of an aesthetic form, the design decisions for an interaction that fits the intended experience, and the technology that makes this interaction possible.

To achieve an intended experience, the designer has to consciously differentiate between certain attributes of the interaction (for example, slow, mediated, delayed) and the emerging experience (for example, a positive and meaningful moment). Creating a particular experience requires awareness and the purposeful combination of attributes at interaction level.

> Example: The iPod's scroll wheel enables users to scroll up and down their long song lists. This kind of interaction is perceived as smooth and easy because the finger can go around continuously and doesn't have to return to the top of the screen.

The designer can influence the aesthetics of the interaction by specifically adapting certain properties of the interaction, including

- temporal: the duration of the interaction, the sequence of interaction steps
- spatial: the use of space, the spatial distribution of elements, the direction of interaction
- action-reaction: the relation of action and reaction, feedback, response
- presentation: the way information is presented and the possibilities for interaction
- force: the force necessary to interact, the application of force that characterises the interaction.

> Example: A very fast system response generates a feeling of efficiency and has an activating effect. A slow system response, on the other hand, can create a feeling of calm and significance of the moment/product. Slow system response can also create a feeling of frustration, depending on the context of use.

Since the aesthetics of interaction lead to an emotional experience with regards to a person's subjective requirements, this aspect has a particular effect on the user experience and can exert an influence – positive or negative – upon it. An experience of a well-being-oriented design that aims to set meaningful moments in everyday life is, therefore, very important [LeDiHa2014].

1.4.3 Considering whole ecosystems

The interaction between the user and the interactive system is often embedded in an ecosystem of multiple interactive products, systems, or services or a wide variety of use contexts. Information is often exchanged via in parallel or different channels; for example, websites, emails, desktop software, mobile apps, or loudspeakers with voice input.

Addressing the whole user experience, the designer must consider all relevant touchpoints to the interactive system including other interactive products, systems, or services.

Copyright 2025, UXQB e. V.

Touchpoint

Any point of contact enabling interaction with the interactive system or the supplier of the interactive system across various channels (for example, website, product flyer, mobile app, PC software, telephone…) during the entire product life cycle (from the very first interaction to the last contact with the interactive system and/or its supplier).

Example: Imagine a user of an online shop, which the user discovers through a price comparison website (touchpoint) on their mobile phone. They call the shop using the browser on their computer (touchpoint) to make a purchase. To their surprise, the page looks different from their previous visit (touchpoint). They stop their purchase and save their shopping cart because they want to speak with their friends about the product. Their friends have only heard good things (touchpoint). So, the user returns to their mobile phone and completes the purchase. Two days later, they receive a confirmation of delivery (touchpoint).

To observe the relevant contexts of use, interactive systems, or channels, the designer needs to leave the narrow perspective on the current product which they are working on and think more outside the box. Using the design thinking approach can be helpful for this.

Design thinking

A systematic approach to solution finding for complex problems of all areas of life, focussing on human values before considering technological or business constraints. It is a tool to tackle the unknown and is as such used to create products, services, or process innovations.

The process includes the following phases: ethnographic research, the definition of the problem, ideation and generation of ideas and solutions, rapid prototyping, test and enhancement of prototypes, and testing of solutions with the people the design has been created for.

A variant is the design sprint, which is a 5-day, design-thinking project [Knap2016].

The design-thinking approach and the human-centred design process share the following basic principles and ideas. Both approaches

- focus strictly on humans and their needs, and emphasise not relying on the designer's own experiences as the source for possible solutions but to understand people's hidden needs through research
- require analysis of the initial situation and use the same or similar methods (personas, observations, interviews)
- recommend doing design work in interdisciplinary teams
- emphasise the importance of iterations and describe an iterative procedure
- make ideas and possible solutions tangible in work products, for example, prototypes
- evaluate work products with users and use feedback from all stakeholders to improve solutions

On the other hand, both approaches differ in certain relevant aspects:

- The type of problems the approach is applied to:
  Human-centred design addresses problems related to the use of interactive systems, whereas design-thinking addresses all kinds of fundamental human problems (social, economic, technical…).
- The realm of possible solutions:
  Human-centred design is about enhancing or creating the usability, accessibility, and user experience of a digital product, interface or service, whereas design-thinking may

provide innovative and creative services, policies, or processes in the analogue or digital world.

- The role of innovation:
  Design-thinking often focuses on innovations and new product ideas, whereas human-centred design projects aim at improvements or extensions of existing products as well
- The inclusion of an implementation phase:
  Human-centred design considers the implementation of the designed solution and the evaluation of the final product, whereas the design thinking process ends with the creation of prototypes.
- Standard as a basis:
  Human-centred design is based on published standards, and its procedures are well-defined, well-ordered and well-controlled, whereas design-thinking loosely describes the principles, methods, and techniques that support the focused but creative chaos of design work.

Despite all differences, possible synergies arise from the introduction of design-thinking into human-centred design. Ideation methods can help the creation of creative solutions, which is helpful in design projects that aim at innovation, especially in the early phases of a human-centred design project.

2 Early design

This design phase aims to transform the deliverables from the context of use analysis into a specification for future interaction. Here, the designer ensures that appropriate functionality is available and can easily be found by users, and determines how to guide users implicitly and explicitly to achieve their goal.

Early design includes design activity conceptual modelling, where the task models for the design are developed, from which interaction specifications can be derived. Based on these interaction specifications, the task objects, their attributes, and executable functions are identified. Use scenarios, story boards, and user journey maps make the intended interaction tangible.

Early design builds a solid basis for creating the first sketches of the new system.

2.1 Design of user interfaces for the achievement of goals

To achieve a goal (intended outcome), the user creates, modifies, or gathers information about the **task objects**. For this purpose, **executable functions** are provided by the interactive system. **Signposts** are used to make executable functions and task objects accessible to the user.

To complete all **tasks** necessary to achieve a goal (intended outcome), users perform **actions** on the **user interface**.

To ensure users can perform **task-related operations** with the interactive system, the user interface must include

- task objects with their attributes
- executable functions to create and/or modify task objects and/or gather information about task objects from different perspectives
- signposts for locating task objects and executable functions.

On top of a regular task-related operation, explicit **user assistance,** which includes (various forms of) system-initiated guidance, online help, and user documentation should be available in the user interface. During each design activity and across all touchpoints, designers must be aware that designing the user interface not only leads to intended user behaviour but can also lead to undesirable behaviour.

Learning Objectives	
2.1.a	Understand the relationship between tasks, task objects, executable functions, actions, signposts, task-related operation, and user assistance.
2.1.b	Understand the consequences of intended and undesirable user behaviour resulting from the user interface design.

2.1.1 Task-related operation to achieve user goals

2.1.1.1 Create, modify, and gather information about task objects

To achieve their goals, users create and/or modify task objects and/or gather information about the task objects while interacting with the system [StJW2005] [Wood2007].

Task object

An object and its attributes presented in the user interface are required during the completion of one or more tasks with the interactive system. The attributes can be derived directly by analysing the user requirements and/or by the dialogue steps (user actions and system reactions) within an interaction specification.

Examples for attributes of the task object “boarding pass”: flight number, destination, gate number, and seat number.

An example of creating, modifying, and gathering information about task objects:

- Create: the task object “boarding pass” is created by the user checking in for the flight.
- Modify: the task object “boarding pass” is modified by the user changing the seat number.
- Gather information: the user gathers information about the task object “boarding pass” by reading the information on it (for example, delayed flight time, changed gate number, seat number).

To create, modify, and gather information about task objects, user interfaces provide executable functions.

Executable function

A means in the user interface that allows users to create a task object, modify a task object, or gather specific information contained in a task object.

Users can easily access the executable functions via the design of signposts.

An example of an executable function: Add boarding pass to passbook, print boarding pass, save boarding pass as PDF file.

To make executable functions and task objects accessible to the user, the designer specifies signposts to the task objects.

Signpost

A hint – often realised as a navigation element – that supports the user in finding a particular task object or an executable function in the interactive system.

Example: A navigation menu can be characterised as a collection of signposts.

Task objects are described by a title (for example, “boarding pass”), the attributes that provide the required information about the task object, and the signposts to executable functions or other task objects (for example, “flight”).

An example illustrating the relationship of executable functions, task object, their attributes, and signposts:

For the task “reduce the size of a file”, the user needs to modify the task object “file” using the executable function “compress”, which allows users to specify a target location (action 1), and then save the reduced file at the target location (action 2). Another task-related action on the file might be “Increase the size of a (reduced) file”, which leads to the executable function “uncompress”.

To use executable functions on the task object (for example, “compress file”, “uncompress file”) or to get information about the task object (for example, “find out how big the file is”), the task object needs to be represented in the user interface with its attributes (for example, file name, file type, file size). After having selected a file, a context menu – accessible through a right-mouse click – might offer signposts to the executable functions “compress” and “uncompress”.

Copyright 2025, UXQB e. V.

When designing task objects and their attributes in the user interface, the designer should consider their knowledge about task objects that the user has built up from their known world. Nowadays, mental models are not only acquired in the analogue world but also from experience in the digital world.

The user's mental model is the basis for representing the relevant task objects in the user interface and the required executable functions on each task object. The designer should use metaphors that help the user identify the task object in order to complete the task and offer actions that support the user in carrying out the necessary dialogue steps.

Example of task objects and actions in the analogue and digital world:
Task objects in the analogue world: Customers visit a bookstore. Displays and bookshelves provide an overview of all books. You can choose a book and take a closer look at it.
Task objects in the digital world: Customers visit an online shop (bookstore). Overview pages give an overview of all books (displays and bookshelves). You can get a detailed view of a selected product (book) with more information about it.
Actions in the analogue world: In the bookstore, I pick up a book, flip through it, and take it to the checkout to buy it.
Actions in the digital world: In the online shop, I click on the book (pick up a book), view its description or read a preview (flip through it), and put it in the shopping cart (take it to the checkout to buy it).

2.1.1.2 Interaction of user and interactive system for the achievement of goals

In the user's world, several possible goals can be identified (for example, organise flight tickets as a gift for someone, visit a friend in a faraway country, get the cheapest flight tickets). Each goal also has tasks that need to be completed to achieve it.

Task
CPUX-F Definition
A set of activities undertaken in order to achieve a specific goal.

Most tasks can be subdivided into subtasks. A subtask is a necessary decision or physical activity that contributes to a user reaching a goal.

If the user has a specific goal (for example, visiting a friend in a faraway country), they must perform the actions as part of the subtasks in order to reach their intended outcome (the intended outcome could be, for example, to have arrived at the friend‘s home in the faraway country).

Action
A specific activity the user performs or initiates in the user interface to achieve a goal (intended outcome). Different actions may become necessary across the subtasks performed to achieve a goal.

Performing actions contributes to the completion of necessary subtasks. As shown in Figure 3, the user can perform these actions by interacting with the interactive system.

Copyright 2025, UXQB e. V.

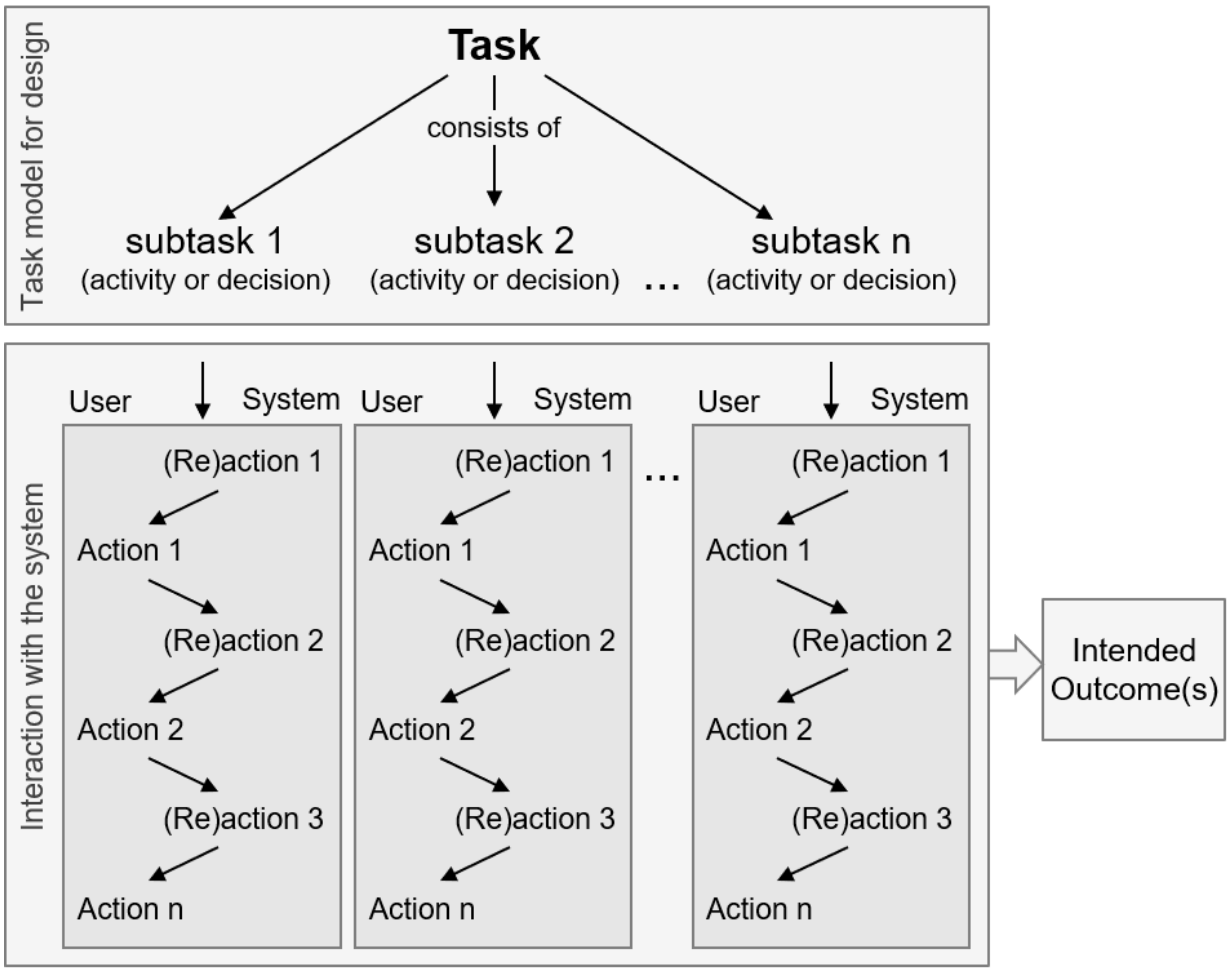


Figure 3 interaction between user and interactive system to achieve a goal

To get from one or more contextual pre-conditions to an intended outcome, the user performs each subtask in terms of a (physical) activity (for example, identifying alternative flights to a destination) or a decision (deciding which flights to book).

As part of each subtask, users take actions on the interactive system “flight booking site” (for example, “Show suitable flights”) and perceive the system’s reaction (for example, “three suitable flights”). Users also take actions on the resources in the context of use (for example, the flight schedule of their preferred airline in printed form). There are also actions that the interactive system performs by itself (for example, automatically notifying the user when the gate changes).

Actions required for task completion that are used in conjunction with each other are perceived by the user to belong to one executable function.

Example: The actions “select delivery address”, “select payment method”, and “purchase now” can be looked at as the executable function “proceed to checkout” and therefore, be visualised as such.

Example on how to get from a contextual pre-condition to an intended outcome:

To complete the subtasks “decide on a product to be purchased”, “prepare to purchase the product”, and “purchase”,

- the user first selects a task object in the interactive system (for example, a specific book in an online shop)

- they then take one or more actions for each subtask on the task object “book” to complete the subtasks (for example, “show table of contents”, “read table of contents”, “put the book into the shopping cart”, “purchase book”)
- finally, they use the appropriate executable function to let the system perform each selected action (for example, “show table of contents”, “add to shopping cart”).

2.1.1.3 User interfaces for the achievement of goals

To enable the user to perform actions on the task objects, they must know
- which actions the interactive system can perform on which task objects
- how the actions of the interactive system can be initiated.

If the user interface provides the user with this action-guiding information through action-guiding signposts for different relevant task objects, it will aid the user in fulfilling their goals.

User interface

CPUX-F Definition
All components of an interactive system (software or hardware) that provide information and controls for the user to allow them to accomplish specific tasks with the interactive system.

The core components of the user interface are task objects and executable functions that enable effective, efficient, and satisfactory task-related operation of the interactive system.

From the point of view of effective, efficient, and satisfactory achievement of goals, a suitable user interface
- provides the user with the required signposts, task objects, and their attributes as well as the executable functions required to achieve their goals during task completion
- enables a dialogue with the interactive system which requires minimal learning effort
- presents all information necessary in a perceivable and interpretable way for users and thereby enables actions in terms of inputs and choices efficiently

When designing the interaction between user and interactive system, the designer must include the design of task-related operation and user assistance.

Task-related operation

The interaction of the user with task objects and executable functions of the interactive system during task completion as intended by the manufacturer.

Task-related use is typically accompanied by user assistance that allows the continuation of task-related operation as soon as the user gets stuck or helps avoid or manage use errors.

> Example: A user's goal is to ensure they are at the desired destination as soon as possible. To achieve this goal, the user uses an app which offers a task object “real-time overview of the next arriving trains” and user assistance with instructions on how to interpret the overview for the subtask “find out the arrival time of the next train for a specific destination” (as part of the task “travelling by public transport”).

This curriculum focuses on the design of task-related operation (see Figure 4).

Task-related operation is the core of the users’ interaction with the interactive system while working towards the goal (intended outcome). To complete all relevant tasks with the help of the interactive system, a user must be able to locate and use all task objects and executable functions efficiently. The design of task objects provides the necessary information and signposts to executable functions in the appropriate scope.

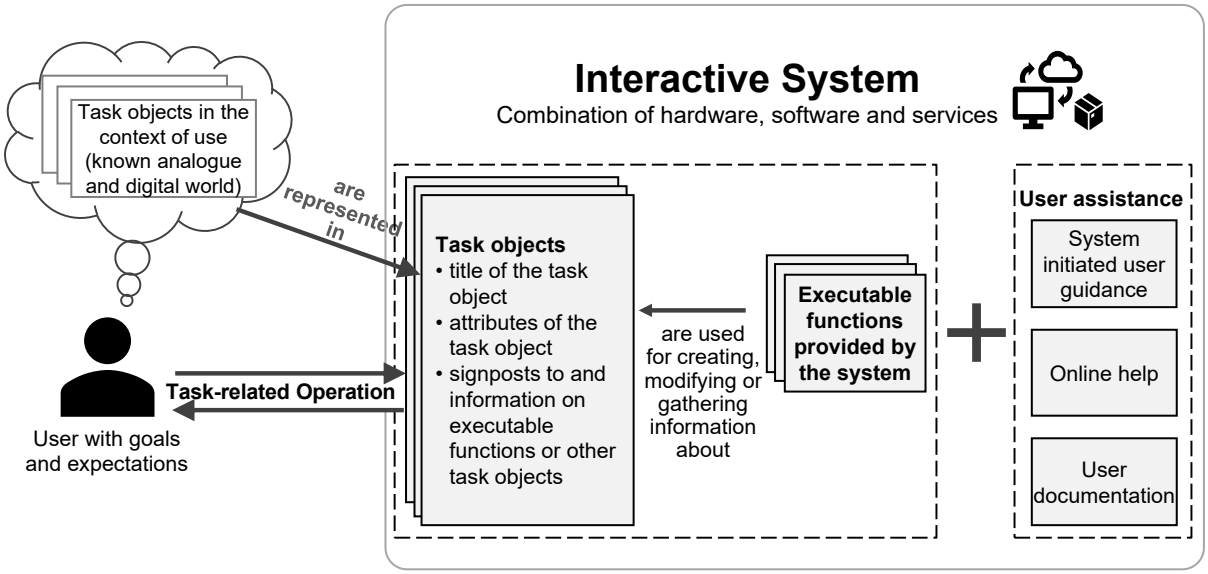


Figure 4: Task-related operation and components of an interactive system

2.1.2 User assistance: Explicit action-guiding information in addition to task objects and executable functions

Task objects with their designed attributes and signposts to executable functions and other task objects give users an implicit action-guidance. Implicit refers to the clear correspondence between functionality and the user's task, which makes the required interaction obvious to the user. Task objects need to be designed in a way that signals to users that they can perform a certain action (affordance) and allows them to understand which of the available actions is the intended one.

For complex tasks, it might not be sufficient to make the appropriate task objects, attributes, and executable functions available in the user interface. Users might need explicit user assistance for task-related operation.

User assistance
CPUX-F Definition
Information to help a user to interact with an interactive system.

In addition to task objects, attributes, and signposts, the user interface has to provide explicit action-guiding information to let the user know which dialogue step has to be performed next in the process of a complex task in the interactive system. Explicit action-guiding information is system-initiated user guidance (for example, feedback, status information, instructions), online help, and user documentation.

An example of the lack of explicit action-guiding: In gesture-driven user interfaces (for example, the app "Pinterest"), users must first perform an initial action like tapping or swiping to make the available functions visible. Because of the lack of explicit action-guiding information, the designer must include, for example, a three-step instruction after the installation of the app, so that the user can learn the necessary interaction.

Example for explicit action-guiding: A user interface provides information about how and when a button will reach the enabled state and displays an explicit action-guiding information "form cannot be sent because one or more mandatory fields are empty" so that the user has guidance on how to continue the task.

Copyright 2025, UXQB e. V.

2.1.3 Intended and unintended consequences of user interface design

Designing the task-related operation and user assistance by providing task objects and necessary executable functions has its consequences. Every design decision made by the designer influences the behaviour of users. User behaviour can be intended or unintended. Every use error represents unintended user behaviour. Both intended and unintended user behaviours must be considered across all touchpoints.

The user's behaviour can have its roots in the design of both the user interface and of the technical system, for example, system response times due to technology.

The interactive system should be designed in such a way that the user's behaviour is optimally supported and intended behaviour is stimulated, leading to effectiveness, increased efficiency, and user satisfaction.

Examples of positive consequences are

- users adapting their behaviour during their work routine to make full use of the interactive system because it is more efficient and satisfactory than using another interactive system or no interactive system at all.
- users now using the interactive system more often than before.

Other than the behaviour intended by the designer, unintended behaviour can also occur. Unintended behaviour is usually undesirable.

Examples of undesirable behaviour are

- repeated use errors
- avoiding using of the interactive system
- use of workarounds
- frequently asking for help from other users
- using the system for tasks for which it was not designed
- exclusive use of search instead of navigation because of a lack of information scent.

Possible reasons for undesirable behaviour are

- the system does not support the achievement of goals sufficiently.
- new and difficult system-initiated subtasks have to be performed (for example, registering) because of the use of the interactive system.
- unnecessary actions are added within necessary subtasks or take longer to perform than before.

Users' behaviour is not only influenced by the design of the user interface, but also by predefined procedures and work processes, the touchpoints available for the interaction with the interactive system, and contextual aspects of the interaction. Furthermore, other products and services available on the market can influence users' behaviour with an interactive system since they generate expectations concerning the functioning of the system (and thus influence users' mental models).

Work processes, touchpoints, or culture might have to be adapted when specifying the user requirements as a basis for designing solutions because they cannot be shaped by user interface design itself.

Copyright 2025, UXQB e. V.

2.2 Design activity: conceptual modelling

In conceptual modelling, the designer defines how the user interacts with the interactive system. During this design activity, task objects and their attributes, and executable functions needed for effective, efficient, and satisfactory task performance are identified.

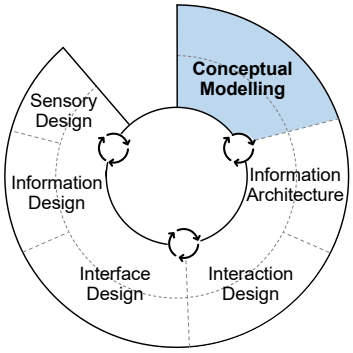


Figure 5 Design activity: Conceptual modelling

In the first step, the designer adapts the **task models** of the current context of use in accordance with the **user requirements,** to the **task models for design,** which are the basis for developing **dialogue steps** along all tasks (and subtasks).

Interaction specifications are particularly suitable for specifying the intended dialogue steps and to determine the task objects and executable functions systematically. While interaction specifications ensure that each dialogue matches the user's subtasks, narrative **use scenarios,** storyboards, and **user journey maps** are more suitable for communicating the developed dialogues with users and other stakeholders to obtain feedback.

Learning Objectives	
2.2.a	Understand how interaction specifications are derived from an understanding of the user's tasks and conformance with user requirements.
2.2.b	Be able to specify dialogue steps and identify task objects and their attributes, and executable functions using interaction specifications.
2.2.c	Know different forms of use scenarios used to communicate the specified interaction to certain roles in a design project.

2.2.1 Creating task models for design

Conceptual modelling starts with adapting the task models of the current context of use.

Task model

CPUX-F Definition
A description of the subtasks within a task that have to be carried out to reach the user's goals.

Task models form the basis for the development of interaction specifications and use scenarios, and are identified in the context of use analysis.

A task model identified in the context of use typically has to be adjusted as a task model for design, for the following reasons:

- the context of use for design is limited (for example, by focusing on a special user group)
- subtasks are added because of the use of a certain interactive system or
- users' tasks were changed in conceptual modelling; for example, if a subtask was removed because it was automated.

Since the task models identified during the context of use analysis reflect the current situation (as-is), it is likely that the way users perform tasks to achieve their goals will be optimised in terms of the (intended) design of the interactive system reflecting the derived user requirements.

Copyright 2025, UXQB e. V.

User requirement

CPUX-F Definition
A requirement for use that provides the basis for the design and evaluation of an interactive system to meet identified user needs.

In some cases, a task model may not have been identified during analysis of the context of use: for example, when the task is new. In this case, the designer develops ideas on how the task model could look like in the future and evaluates their ideas with users.

Taking business objectives, new technological capabilities, and identified user requirements into consideration, design decisions for a future interactive system may have to acknowledge that task models might change, i.e.

- certain subtasks have to be supported differently (for example, instead of creating a route using a map, users enter their starting and destination point and receive their route information)
- other subtasks will not require interaction at the user interface (for example, because they are automated)
- new subtasks will arise (for example, a required registration for using the interactive system).

Task model for design

An adaption of the task model from the context of use analysis that is made for one of the following reasons:

- the context of use for design has been limited (for example, by focusing on a special user group)
- there are added subtasks because of the use of a certain technology
- users' tasks were optimised during the conceptual modelling

The adaptation can include deleting, adding, or changing tasks and subtasks.

As changes in the task model have consequences for the design of the interactive system, the new or updated task models for design should be the basis for further design activities to support the users in achieving their goals. The following example shows how a task model for design is created.

> Example: A designer creates an interactive system (mobile app and website) for making reservations at restaurants. For this, the designer studies the task model of the current context of use (which either has been identified as part of a context of use analysis or is simply being assumed) to see how restaurant visitors are currently making reservations.

Copyright 2025, UXQB e. V.

Table 4 Task model

Task	Reserve a table in a known restaurant
User group	Visitor of the restaurant
Contextual pre-condition(s)	• The visitor has decided to go for dinner with a group of people • The visitor has decided on a specific restaurant • The visitor has decided on a specific date and time frame
Intended out-come(s)	The visitor has reserved a table • for the desired date • within the desired time frame • for the intended number of people. The visitor has a confirmation that they can present upon arrival at the restaurant.

The task model of the current context of use includes the subtasks:

- call the restaurant of choice
- explain that a reservation is required for a desired date and time
- communicate the number of people
- communicate your surname
- confirm that the table is available
- confirm that the table will be reserved
- write down the name of the restaurant, the date and time of the reservation, and the number of people
- inform other participants that the table has been booked for <date>/<time>.

The designer analyses the user requirements for the mobile app and website (irrespective if they are based on a context of use analysis or are being anticipated in lack of empirical information).

UR1: With the system, the user shall be able to select a restaurant of choice.

UR2: With the system the user shall be able to select a date and time for an intended reservation.

UR3: With the system, the user shall be able to know when the tables are available at the intended date.

UR4: With the system, the user shall be able to know the shape of the table (round or rectangular).

UR 5: With the system, the user shall be able to know the maximum number of people servable at the table.

UR6: With the system, the user shall be able to select the channel through which they can receive their confirmation.

UR7: With the system, the user shall be able to know whether the reservation was successful.

UR 8: With the system, the user shall be able to select whether the reservation confirmation should be stored in their personal calendar.

The designer considers how the reservation process should work with the interactive system. Considering the user requirements, the designer identifies that the user does not need to call the restaurant and explain their request. They can pick a restaurant of choice and check for the availability of tables at the desired date and time. Once the guest selects a table and books it, the confirmation is received either by e-mail or by text message, and documentation happens at the same time.

Copyright 2025, UXQB e. V.

As a result, the designer adapts the task models for design to five subtasks accordingly

1. Decide on the restaurant of choice

   UR1: With the system, the user shall be able to select a restaurant of choice.
2. Check for the availability of tables at the desired date and time

   UR2: With the system, the user shall be able to select a date and time for an intended reservation.

   UR3: With the system, the user shall be able to know when tables are available at the intended date.
3. Decide on one available table and book it

   UR4: With the system, the user shall be able to know the shape of the table (round or rectangular).

   UR5: With the system, the user shall be able to know the maximum number of people servable at the table.

   UR6: With the system, the user shall be able to select the channel through which they can receive their confirmation.
4. Secure the reservation details

   UR7: With the system, the user shall be able to recognise that the reservation was successful.
5. Inform fellow guests about the reservation

   UR: None – Outside the scope of the interactive system to be designed

Designers usually prefer an open solution space. If there are restrictions to this solution space, it must be checked whether this has any influence on the subtasks of the user, which results in adapting the task models for design.

While working iteratively on the design activities, designers might uncover further changes required in the task models for design. These changes result in adapting the task model for design continuously throughout the entire design process.

The following are considered while deciding if a task model should be adapted for design:

- Identifying the constraints for design, considering business objectives and technological decisions (for example, mobile app and website) from the design project's stakeholders
- Checking whether there are any limitations for the intended context of use (regarding user groups, tasks, user goals, environment, or resources) for the interactive system
- Determining whether working on an already existing technical system creates constraints for the design, as subtasks (for example, storing the confirmation) will be supported outside the scope of the interactive system (for example, by the personal calendar app of the user).

Further changes can be caused by feedback from users and other stakeholders or by a better understanding regarding task objects, their attributes, and executable functions that are necessary for users to interact successfully with the system. Changes can also affect work and business processes and must be discussed with all parties involved.

2.2.2 Creating interaction specifications based on task models

When designers have developed an initial idea for future use by adapting the task model for design, they specify how the interaction between the user and the system should take place. At the same time, they define what support must be provided by the technical system.

Copyright 2025, UXQB e. V.

The starting point for modelling interaction can be a variety of activities, for example, drawing simple scribbles, conducting a role play, or coming up with a story. This way, designers model a dialogue for each supported task and their subtasks. The designer must ensure that the user requirements are met for each task. To describe the interaction between the user and the system and support the task at hand as well as possible, the dialogue steps between the user and the system throughout all the subtasks for each task model are determined.

Dialogue step

Each individual action of the user (making a choice, making an input) and the resulting reaction at the user interface.

Each dialogue step includes

- the action of the user (selecting an available table)
- the reaction of the interactive system (showing the shape of the table and the maximum number of people that can sit at the table)
- whether the dialogue step is initiated by the user (for example, the restaurant visitor) or automatically by the system (for example, informing the user that they are running late for their booked table).

A table that maps the task model for design, the dialogue steps for each subtask, and the user requirements to be implemented, serves as a structured notation to specify the interaction.

Interaction specification

A specification of all dialogue steps to be enabled between the user and the intended interactive system adhering to the task model for design and the user requirements.

The dialogue steps consist of each user's action and the corresponding reaction of the user interface. They are embedded in the task model for design and user requirements so that the dialogue becomes an explicit, step-by-step guide within each subtask of the task to be supported and clearly points to the user requirements.

This way it is possible to specify dialogue steps in terms of user actions and reactions of the interactive system, which clearly adhere to the task model for design and truly implement the user requirements.

The term interaction specification can be misleading. It is not about specifying a concrete interaction of the user by selecting concrete user interface elements. It is about the general description of this interaction in terms of a dialogue (actions of the user and reactions of the interactive system across the subtasks) whilst the user is achieving their goal. Since this specification prescribes the intended behaviour of the user when performing tasks with the system, it can also be interpreted as a use scenario in a structured form (in contrast to a narrative use scenario in semi-structured form).

An interaction specification has the structure shown in Figure 6. Such an interaction specification should be drafted for each task supported by the system.

- Column 1 contains the task model for design including all subtasks for the task.
- Columns 2 and 3 contains the dialogue steps within each subtask.
  Typically, each subtask contains one or more dialogue steps.
- Column 4 contains the user requirements to be implemented by the dialogue steps.

Copyright 2025, UXQB e. V.

Subtasks	Action of the user	Reaction of the user interface	User requirements
		Initial action guiding information: <information supplied by the system that enables the start of subtask 1>	
Subtask 1	Action 1.1	Reaction 1.1	User requirement 1.1 User requirement 1.2
	Action 1.2	Reaction 1.2	User requirement 1.3
	…	…	…
Subtask 2	Action 2.1	Reaction 2.1	User requirement 2.1 User requirement 2.2
	Action 2.2	Reaction 2.2	User requirement 2.3
Subtask 3	Action 3.1	Reaction 3.1	User requirement 3.1
…	…	…	…

Figure 6: Structure of an interaction specification

The order of specifying content items within each interaction specification is as follows:

1. Add all subtasks to column 1.
2. Assign each user requirement to at least one subtask and add each user requirement to column 4.
3. Specify the “initial action-guiding information” in the top cell of column 3. This is the information needed by the user, so they can take the first action as part of the first subtask.
4. For each subtask, specify the sequence of necessary actions by the user and the reactions of the user interface (dialogue). Actions are always choices or inputs and reactions are the information expected by the user after each action. For each reaction, whenever necessary, additional action guiding information can be added.

   Example: Interaction specification containing the task model for design, the user requirements, and the dialogue steps to be completed by the user.

Table 5 Exemplary interaction specification

Subtasks	Action of the user	Reaction of the user interface	User requirements
		Initial action-guiding information: • Current location of the user • Restaurants nearby	
1. Select the restaurant of choice	Input: (Parts of the) name of the restaurant	Show: Restaurants matching the (parts of the) name	UR1: With the system, the user shall be able to select a restaurant of choice.
2. Check for availability of tables at the desired date and time	Select: • Date • Time • Number of people	Show: • Time slots the tables are available for the intended number of people • Status information in case no tables are available	UR2: With the system, the user shall be able to select a date and time for an intended reservation. UR3: With the system, the user shall be able to recognise when tables are available at the intended date.
3. Decide on one available table and book	Select: (One) available table	Show: Details of available table	UR4: With the system, the user shall be able to know the shape of the table (round or rectangular). UR5: With the system, the user shall be able to know the maximum number of people servable at the table.
	Select: Reserve	Show: • First name • Last name • Mobile number • E-mail address	UR6: With the system, the user shall be able to select, the channel through which they can receive their confirmation.

Subtasks	Action of the user	Reaction of the user interface	User requirements
	Input: • First name • Last name • Mobile number and/or • E-mail address	Show: • First name • Last name • Mobile number and/or • Email-address • Additional action-guiding information: “Confirmation-E-mail can be stored in calendar app.”	
4. Secure reservation details	Select: Confirm reservation	Show: Status information that reservation confirmation has been sent	UR7: With the system, the user shall be able to recognise whether the reservation was successful.
5. Inform other participants about reservation	Dialogue continues on e-mail app or text message app (out of scope)	-/-	

In addition to regular use, which is covered by the example of the interaction specification above, the designer must also consider irregular use. Irregular use refers to undesired situations that may occur during use, such as user errors. The designer also creates interaction specifications for irregular use (for example, error handling), as shown in Table 6.

Example of an interaction specification supporting error robustness

Table 6 Interaction specification of irregular use (error handling)

Action of the user	Reaction of the user interface
Input: Number of people as a word, for example, “three”.	Show: Next to the input field for the number of people, it is indicated that only numbers can be entered (use error recovery).
Input: Number of people as number “3”	Show: Input is correct.
Input: Name of the person making the reservation	The name of the person making the reservation contains a zero instead of an O, the system interprets this typing error unambiguously, does not output an error message and changes the number to a letter by itself (use error tolerance)

When developing interaction specifications, quality criteria must be considered in order to avoid methodological mistakes.

Quality criteria for interaction specifications

- Each interaction specification must be referenced to a specific task and (one or more) user group(s).
- Subtasks and user requirements must be part of the interaction specification and considered when identifying task objects, attributes, and executable functions.
- Subtasks must always be formulated as object-verb constructions oriented towards the goal to be achieved and not towards the details of the solution in mind.
- The interaction specification must be complete regarding subtasks and dialogue steps from the perspective of the respective task model.
- Each dialogue step must be described from the point of view of the task to be completed, not from the perspective of a required solution.
- Interaction specifications must be formulated specifically to ensure that dialogue steps, executable functions, and task objects can be clearly identified.
- The same level of granularity must be applied to all dialogue steps. Dialogue steps must always be phrased as inputs, choices, or “locate / recognise / understand” as part of a subtasks.

Common mistakes include

- adding new user requirements that are not based on the context of use but are based on the solution in mind
- describing the implementation of interaction with the interactive system by stating specific user interface elements (having a particular interactive system in mind)
- restricting the supported dialogue steps too early, for example, based on technical framework constraints
- neglecting dependencies between the individual tasks because individual tasks are considered in isolation.

2.2.3 Identifying task objects, attributes, and executable functions in interaction specifications

When the designer has specified the interactions between the user and the system, specific task objects, their attributes, required executable functions, and the necessary explicit action-guiding information (user assistance) become apparent. The designer derives task objects and executable functions from the dialogue steps.

- Each reaction (column 3 of the interaction specification) involves a task object or an attribute of a task object (or additional, explicit action-guiding information needed by the user).
- Each action (column 2 of the interaction specification) naturally represents an action on a task object. If individual actions always directly follow each other within a subtask, this set of actions can be specified as a coherent executable function. For example, the action “reserve” is always followed by the action “input personal details” and then “confirm reservation”. Therefore, “reserve” may represent an executable function containing all these actions.

The example below shows the derived task objects and executable functions from the interaction specification stated above. It is helpful to first have an overview of task objects and executable functions and then have a table that specifies the attributes for each task object.

An example of an overview of task objects and executable functions on each task object:

Copyright 2025, UXQB e. V.

Table 7 Overview on all task objects and executable functions

Task objects	Executable functions on each task object

Restaurant	• Show available tables
	  • Select room
	  • …

All available tables	• Specify a table
	  • Select shape
	  • Select location (at window, in centre)
	  • …

Available Table	• Reserve:
	  • Input personal details
	  • Select “Confirm reservation”
	  • …

Reservation	• View • …

Reservation confirmation	• Save in calendar
	  • select calendar (private, work…)
	  • …
	• Cancel reservation
	(This executable function has been added from another use scenario for the task “cancel a reservation”)

System developers and designers thereby receive a structured and detailed overview of the task objects, their attributes (see Table 8) and executable functions (see Table 7) which must be provided by the interactive system to be developed.

For each subtask, the designer must determine the task objects the user created and/or modified and/or gathered information about.

Example: If the reservation was successful, the task object “reservation confirmation” must be created by the interactive system. The visitor views (gathers information about) the confirmation to determine whether the reservation was successful.

Example for an overview of task objects and their attributes in detail:

Table 8 Specification of all task objects and their attributes

Task objects	Attributes	Details for each attribute
Restaurant		
	Name	A-Z
	Address	Street and number Postcode City Country
	Telephone	+ Country code Phone number
	E-Mail	<name>@<domain>.<domain suffix>
List of available tables for a specific date and time	Available table 1 … Available table n	Time 1…n Shape (round or rectangular)
Available Table	Availability	hh. mm
	Shape	Round Rectangular
	Max number of people	2, 4, 6, 8
Reservation	Person who has made a reservation	A-Z
	Date and Time	dd.mm.yyyy, hh.mm
	Table	Round Rectangular Indoors Outdoors
	Number of guests	1, 2, 3, 4, 5, 6, 7, 8
Reservation confirmation	Name of sender	A-Z <name>@<domain>.<domain suffix>
	E-mail address of the sender (in case of text message, only name of sender)	
	Name of restaurant	A-Z
	Address of restaurant	Street and number Postcode City Country
	Date and time of reservation	

Copyright 2025, UXQB e. V.

Task objects	Attributes	Details for each attribute
	Name of the person who made the reservation	
	Number of guests	
	Type of table	Round Rectangular

It is also possible to derive the task objects and executable functions directly from the user requirements.

For example, the user requirement “UR3: With the system, the user shall be able to recognise when the tables are available at the intended date.” allows us to derive the task object “Available table” with its attributes “time 1…n”.

However, this approach requires rigor and discipline in formulating user requirements, which is typically only the case if an explicit context of use analysis with subsequent analysis of user requirements has preceded the design project (see CPUX-UR). It is advisable to use the approach of analysing interaction specifications for conceptual modelling, particularly with team effort, where it needs to be ensured that every team member is included.

2.2.4 Considering dependencies and creating variations

The sub-activities described in conceptual modelling (Creating Task Models for the Design, Creating Interaction Specifications, Identifying Task objects, Attributes, and executable functions) are performed for each individual task to be supported by the interactive system.

> Example: In addition to the task “reserve a table”, the tasks “cancel a reservation”, “postpone a reservation” and “write a guestbook entry” should also be supported. This is why the designer has created task models and interaction specifications for all three tasks. They have also identified task objects, attributes, and executable functions that are not just related to the reservation.

Task models for the design must always be created from the perspective of all tasks to be supported. All existing dependencies between the tasks must be considered. This enables the user to process tasks flexibly. When iterating through the various tasks to be supported and their dependencies, the need for adjustments to the individual task models for design is identified. The identified task objects, their attributes, and the executable functions are further refined.

> Example: The user group “restaurant staff” may have the task “cancel reservations" in case of an emergency. A subtask of the restaurant staff’s task model for “cancel reservations” could be “inform guests about their cancelled reservation”. In order for this subtask to be completed successfully, the staff must be able to reach the restaurant visitors who reserved a table (for example, by phone or mail). Therefore, the designer has added a subtask "Enter personal data for notification purposes" to the restaurant visitor’s task model for the task "Reserve a table". If the user requirements of the new subtask “Enter personal data for notification purposes” are unknown, this subtask would need to be discussed with users in order to get feedback on user needs and derive user requirements.

The designer not only iterates through dependencies of different tasks, but also different variants of task models for design and interaction specifications for the same task.

Copyright 2025, UXQB e. V.

Example: The designer creates several task models for the task "Reserve a table". In a more innovative variant, restaurant visitors are able to select the desired table from a set of photos that show different views of the table's position. This creates an additional sub-task for the users (guessing the location of the table).

The designer switches the focus between single task models and dependencies between task models. The dependencies between task models 1 and 2 and 2 and 3, as shown in Figure 7, along with the dependencies between task models 1 and 3 have to be considered. Designers work with different variants of task models for design and interaction specifications to validate the interaction, task objects, attributes, and executable functions with users and further stakeholders.

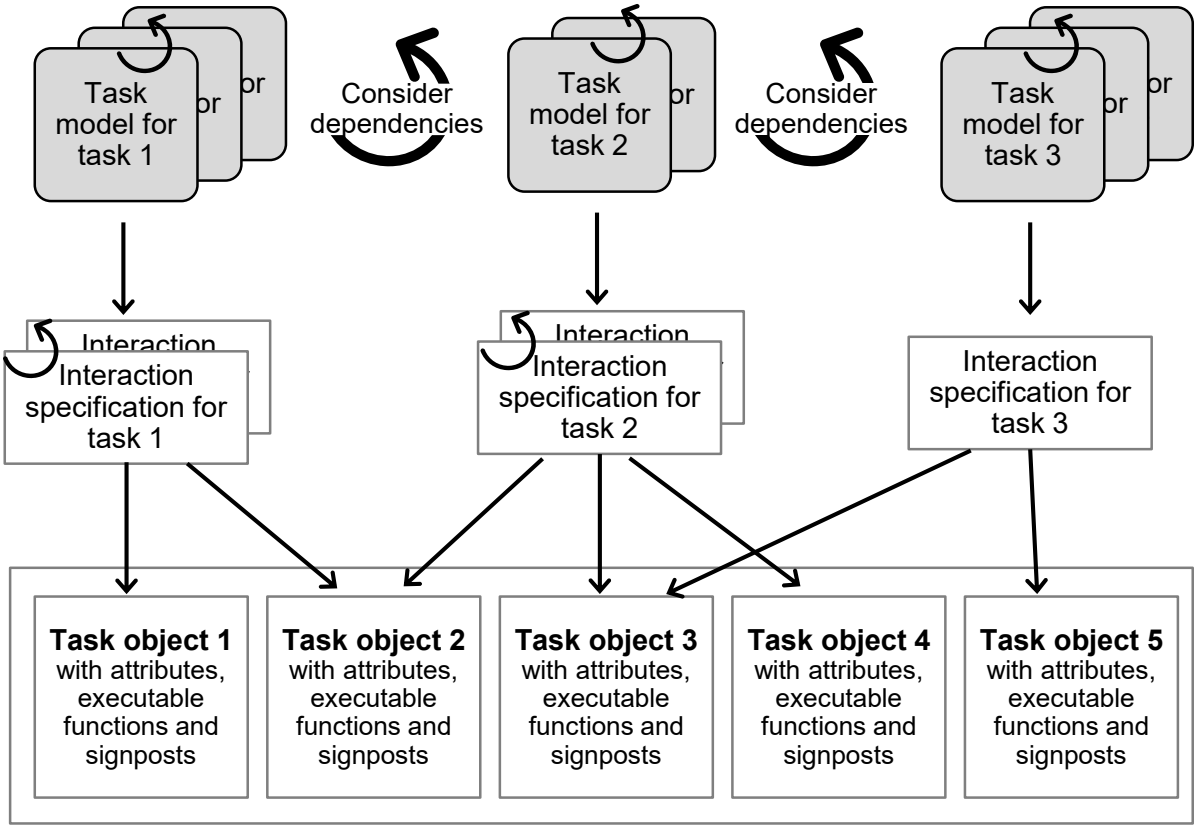


Figure 7 Consideration of dependencies between tasks and creation of variations of the same task

2.2.5 Communicating use scenarios to users and stakeholders

The specifications of the designer must be documented in a way that enables users and other stakeholders to understand and evaluate the intended interaction with the system.

Use scenario

CPUX-F Definition

Narrative text description that describes an intended usage situation with the interactive system under development.

- The purpose of use scenarios is to provide a very early, tangible basis for discussions about what the intended interactive system could be like for the user, before prototypes are constructed. Use scenarios are based on a deep understanding of the context of use, user needs, user requirements as well as discussions with users and stakeholders.
- The specific user in the use scenario is often a persona.
- Use scenarios illustrate the use of the interactive system in a real context. They can be viewed as textual representations of the initial prototypes of a new interactive system. They enable developers to understand processes and context.
- A use scenario should avoid placing unnecessary constraints on the design by referencing specific objects, such as command buttons, in the user interface.

Interaction specifications can be used as a basis to illustrate use scenarios in various forms, for example, narrative text, story board, user journey map.

In Early design, interaction specifications help to specify dialogue steps that support the user's tasks, effectively and efficiently meeting the user requirements. Use scenarios in their different forms are particularly suitable for communicating the specified dialogue steps to users and other stakeholders.

Use scenarios in the narrative form as well as in the storyboard form help underline the comprehensibility of decisions on the use of an intended interactive system through the means of storytelling. User journey maps illustrate use scenarios that refer to several touchpoints or several user tasks along a broader process.

User journey map

CPUX-F Definition

A graphical or tabular description of all encounters users have with the interactive system, covering all touchpoints that influence the user experience, making the overall user experience tangible for others.

Designers must be able to convert an interaction specification into different forms of use scenarios. Even if the designer has not created an interaction specification, they at least assume the dialogue steps before describing the user behaviour in a use scenario.

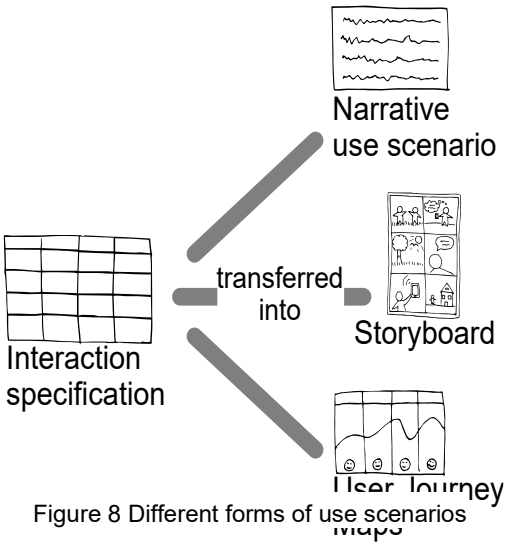


Figure 8 Different forms of use scenarios

Different roles in a design project require different ways of communicating. This is related to the way they work with this information.

- For designers, the development of interaction specifications for each task greatly simplifies the development of the information architecture and the task-related interaction sequences in interaction design. The table form of the interaction specification is necessary to portray the dialogue as accurately as possible. Other roles involved in design not only receive an understanding of the intended behaviour of the interactive system, but also get insights into the task objects, their attributes, and executable functions that must be provided to the user by the intended interactive system.
- For the validation of interaction specifications with users and the collection of feedback from stakeholders, use scenarios in the narrative form, storyboards, or user journey maps are more suitable. Users and other stakeholders gain an understanding of how they can achieve their goals in the future when interacting with the interactive system and what task objects and executable functions can be provided to them for that purpose.

  Example: The interaction specification for regular use can be easily transferred into a simple narrative use scenario:
  “Lisa wants to make a reservation for her and four colleagues in her preferred restaurant. She uses the “enjoy your evening” app where she picks the restaurant of her choice and checks for the availability of tables on the desired date and time. Once she selects a table (she prefers round tables for groups) and makes the booking, the confirmation is sent by e-mail to her, along with a calendar entry. She then simply sends the calendar entry to her colleagues and everybody is instantly informed that the booking has been completed.”

Copyright 2025, UXQB e. V.

3 First drafts

Using the results of conceptual modelling, the designer develops the information architecture as the foundation for creating task-oriented interaction sequences, the structure of the user interface, and its required views. To get feedback for optimising the design iteratively, the designer makes their design decisions tangible for users and other stakeholders.

3.1 Design activity: information architecture

When developing the information architecture, it is the designer's task to elaborate on the conceptual model. They identify signposts which will be used to navigate to task objects and executable functions, and connection paths to navigate between them, and visualise the information architecture for evaluation with users and other stakeholders. There are several ways to present structured task objects, signposts, and navigation paths, including arranged overviews of task objects, tabular overviews of all task objects, hierarchical trees, and schematic representations.

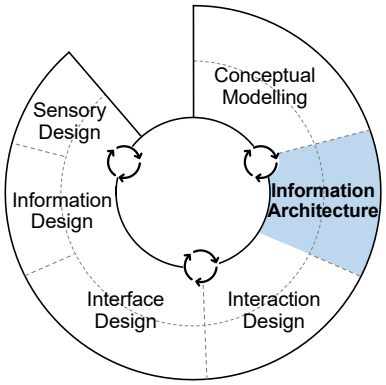


Figure 9 Design activity: Information architecture

The designer has to create a **navigation system**, based on following the **navigation structure**, realised by the inclusion of **navigation elements**. Signposts are used within the navigation structure to make task objects and executable functions accessible to the user.

Card-sorting and **tree-testing** are methods for validating the information architecture and making conformity with the mental models of the users visible.

Learning Objectives	
3.1.a	Understand the different elements of information architecture including task objects, executable functions, content, terminology, and navigation structure
3.1.b	Know the different information needed to develop the information architecture
3.1.c	Understand how to create the information architecture
3.1.d	Be able to enhance and structure task objects as an important part of the information architecture
3.1.e	Understand the difference between navigation systems, types of navigation structures, and navigation elements
3.1.f	Be able to identify deviations between the information architecture and the user's mental model represented in card sorting results, and initiate corrective action
3.1.g	Understand the difference between card sorting and tree testing

3.1.1 Development of the information architecture

In conceptual modelling, the designer will have identified task objects, attributes, and executable functions that are relevant to the user across all tasks to be supported by the system. This is the basis for developing an information architecture that will be support all the intended tasks of the users.

The information architecture must fulfil two aspects:

- It must ensure that the task objects, attributes, and executable functions match the expectations of the users and are learnable from a semantic point of view (what task objects and executable functions are presented and how do they interrelate?)
- It must help users locate task objects and executable functions from the point of view of interaction (Where to go next, within the navigation structure?). For this purpose, task objects must be “signposted”.

  Example: The task object “science fiction book” includes a signpost labelled “add to shopping cart” which leads the user to the executable function “order book” and allows them to enter the purchase process of the book.

  Example: A user wants to buy a science fiction book (task object) online. They first need to locate the overview of all science fiction books. To be able to decide on a book, they need to find out how to gather more information about the different books: authors, short descriptions, and prices (attributes). If the user finally decides on one book, they need to understand how to select it and enter the purchase process (executable function).

A good information architecture supports flexible task handling by the user in terms of the sequence and interruptibility of the tasks. Both the enhancement of task objects and the structuring of task objects must contribute to this.

Example: If a user is interrupted while writing an email, it is saved as a draft. To ensure that the user can continue writing it later, the designer added a signpost “continue writing” to the executable function "open draft mail".

The elements of an information architecture are as follows:

- Titles of all task objects, (for example, “book”) and titles of all attributes of each task objects, (for example, “author of book”)
- Titles of executable functions, for example, “proceed to checkout” and titles of all individual actions enabled within each executable function (for example, “select number of copies to be ordered”)
- Content which includes attributes of task objects, for example, “author of book” and parameters for setting executable functions, for example, “filter categories”
- Terminology: All terms used in the user interface, for all contents, and also all labels of signposts like “add to shopping cart”
- Navigation structure to
  - navigate towards task objects and executable functions using titled signposts
  - navigate within task objects and executable functions (for example, “table of contents” of a book is presented to the user, which allows them to navigate within the book to see parts of the contents)
  - navigate across task objects and executable functions using connection paths (for example, other books with similar contents)

When developing an information architecture, designers can use various bits of information as input:

- Information from the context of use analysis – User group profiles, personas, task models, and as-is scenarios provide information about the natural superordination or subordination of task objects.
- Mental models of users – If the mental models of the users are not considered, users may not be able to navigate through the interactive system.
- Currently used systems – It is often helpful to identify which task objects, attributes, and executable functions can be found in existing systems and how they are structured.

Copyright 2025, UXQB e. V.

Based on this, designers can perform a critical analysis on whether task objects need to be amended, customised, or structured differently.
- Content to be presented – An essential input for the information architecture can be derived from an analysis of the content to be presented (data/facts, text, images, audio/video), which varies from area to area. Contents to be presented are attributes of task objects (for example, a short description of a movie on a streaming platform).

Quality criteria for developing an information architecture

There are quality criteria that an information architecture must adhere to and methodical mistakes that must be avoided:
- Along with the context of use information, the contents of the information exchange between the user and the system must also be used as input for the resulting information architecture.
- When developing an information architecture the focus must be on representing and structuring task objects and executable functions and not on the use of specific user interface elements.
- Links between the task objects must be included in the form of connection paths.
- User goals, business-centred objectives, as well as the resulting task models for design and user requirements must be considered when developing the information architecture.
- For evaluating the developed information architecture, an appropriate type of representation must be chosen depending on the purpose of evaluation.

Common methodical mistakes include
- focusing on a particular task by developing a structure of task objects in isolation while neglecting relationships and dependencies between task objects
- focusing on the design of the user interface by selecting, configuring, and combining user interface elements instead of focussing on the development of access to task objects and executable functions and their structure
- specifying task objects that cannot be traced back to the context of use information.

The following subchapters outline a possible procedure for developing an information architecture:
1. Enhance task objects with signposts
2. Structure task objects by determining connection paths
3. Make the information architecture visible for evaluation
4. Create the navigation structure using connection paths and signposts
5. Evaluate the information architecture.

3.1.2 Enhance task objects with signposts

The first sub-activity of developing an information architecture focusses on the individual task objects. Starting from a specific task object, the designer identifies other task objects that are related to it.
- Which other task objects could be relevant in the context of a specific task object?
- Which attributes and signposts must be added as a result?

The interaction specifications and the underlying context of use information help with enhancing task objects from the user's point of view. Compared to the task object the user knows from their known (analogue and digital) world, a task object represented in the user interface can contain greater or fewer attributes and signposts.

Example:

In a restaurant, the visitor gets a lot of information about the attributes of the task object "table" if they take a closer look at it: the shape and height of the table, its location in the room, whether there are decorations on the table, the number of chairs, whether the chairs are upholstered, its availability at the moment, etc.
Considering the results of conceptual modelling, the designer decides to display the task object "table" in less detail on the restaurant's website to focus on what's important from the perspective of the user's goal (have a table reserved): maximum number of people, its location, and its availability throughout the whole week.

During the process of identifying signposts, the fact that task objects are created for different users with varying goals and tasks must be taken into account. A single task object may therefore be used in a variety of tasks.

Example of different tasks on one task object: A ticket inspector and a passenger use the same task object: the ticket. These two user groups have different goals. The ticket inspector wants to check the validity of tickets, while the passenger needs the ticket in order to find their reserved seat. Consequently, the inspector searches for different attributes of the task object "ticket" (information on the validity of the ticket) to the passenger (who requires information about their seat number).

When enhancing a specific task object with signposts, the designer must consider the relationship with other task objects in order to specify necessary signposts to them. They specify

- signposts that lead from other task objects to the current task object to another task object (trigger words)
- signposts that lead from the current task object to one or more executable functions or other task objects (calls to action)

This way, the designer provides a basis for the subsequent structuring of all task objects and executable functions. A task object is fully specified once its attributes, the executable functions, the action they enable, and the signposts are determined.

Table 9 Fully specified task object "reservation confirmation"

Task object: reservation	**Details**
Attributes	• The person who made the reservation • Date and Time • Table • Number of guests

Executable functions
(and the actions they support)
	• Save in calendar
	  • select calendar (private, work, …)
	  • …
	• Cancel reservation
	  • input reason
	  • …

Signposts to executable functions and other task objects (calls to action):	• Save reservation confirmation • Cancel this reservation • …
Signposts from other navigational points (trigger words):	• My reservations • …

3.1.3 Structure task objects by determining connection paths

If task objects are well understood and detailed, the designer can provide the user with a meaningful structure based on the interaction specification and the context of use information. They decide on how users get to a task object and where they go from there by creating connection paths that lead to and from task objects.

Task objects are structured from two different perspectives:

- The super- and subordination and the relationship of task objects that must correspond to the user's expectations or are easily learnable from a semantic point of view.

  Example: The task object "overview of all books" is superior to the task object "single book". The task object "apple" is subordinated to the task object "fruits".
- The signposts between task objects that indicate which task objects or executable functions must be accessible via connection paths from other task objects during navigation.

  Example: The executable function "add to shopping cart" is directly accessible from the task object "single book".

The goal of this sub-activity is to develop a structure of task objects that can be understood by users. Task objects must be structured logically so that the users can use their mental models to find their way around the user interface.

If users recognise task objects structured in a similar way to those in their known analogue and digital world, they can use their task-related knowledge while performing and initiating actions on task objects without having to explicitly learn a new way of completing specific subtasks.

Example: A user opening their mail program expects to see an overview of all mails. They wouldn't expect to see a single mail, for example, the last read mail first. The task object "e-mail" is subordinated to the task object "All e-mails".

Users navigate from a general term to a sub-term and from there to the task object, executable function, or user assistance that they are looking for.

Copyright 2025, UXQB e. V.

Example: task object “All e-mails” → task object “e-mail” → executable function “delete e-mail”.

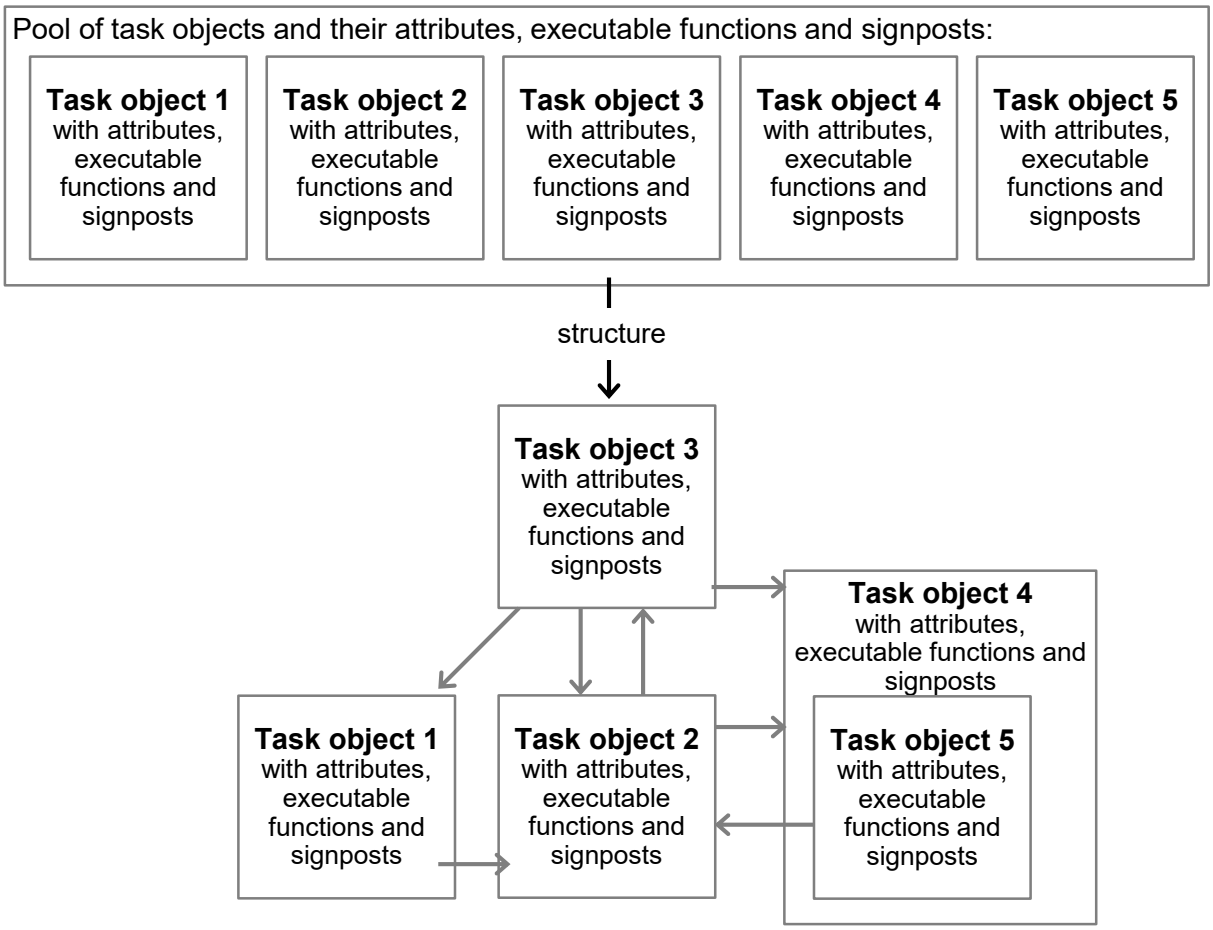


Figure 10 Creation of an information architecture that corresponds to the user's mental models and ensures navigation between task objects

3.1.4 Make the information architecture visible for evaluation

To evaluate different variants of the information architecture with the internal team, users, and other stakeholders, it must be made visible. There are various representations to make structured task objects visible and to present it. Some exemplary representations (that are illustrated in the following sections) are

- schematic representation
- arranged fully specified task objects
- hierarchical tree
- tabular overview of all task objects.

All types of presentations present the detailed and structured task objects in conjunction with each other as well as the signposts to executable functions on them. Each type of presentation takes a different perspective on the task objects and their attributes, and signposts to executable functions and other task objects. Some types put more emphasis on the details, others more on the relationship and super- and subordination of the task objects. Designers must decide which presentation type is most suitable for their individual evaluation or in the current project state.

All the above types of presentations are especially useful for designers, system architects, and developers. Based on this, designers create a structure of the user interface, appropriate

views, and interaction sequences. System architects and developers check if the attributes and executable functions are available in the required level of detail, whether the connection paths can be realised, and whether the intended information architecture can be implemented on the technical platform.

In the following sections, the characteristics of the types listed above are explained:

Schematic representation

A schematic representation consists of several boxes representing task objects that are set in relation to each other. The hierarchy of boxes shows which task objects are superior or subordinate to other task objects.

Using the schematic representation, the designer can demonstrate that the task object can contain further task objects (for example, the task object “deleted mails” belongs to the task object “folders”). Connection paths (calls to action) are indicated by arrows pointing from one task object to another.

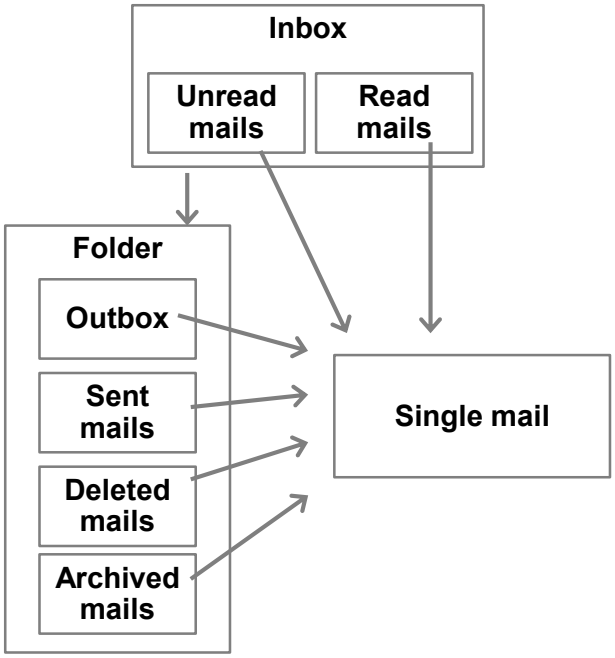


Figure 11 Example for a schematic representation

Depending on the purpose of evaluation, the designer can decide to add more or less detail to the task objects (for example). Also, the designer can use different colours to distinguish different levels of the interactive system in the scheme. The schematic representation helps to capture all task objects and their relationships immediately.

Arranged fully-specified task objects

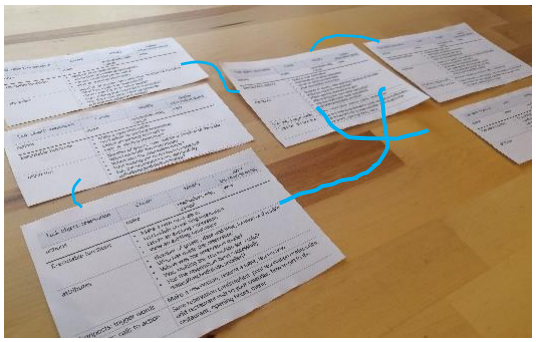


Figure 12 Arranged task object overviews

Figure 12 exemplary shows an overview of all the task objects and their attributes as derived from the interaction specification. This overview can be represented by a set of tables detailing each task object. These tables can then be arranged so that the structure and connection paths between the individual task objects become visible, as shown in Figure 12. The advantage of this type of presentation is that the designer and their stakeholders can get both a detailed overview of each task object and its relationship to other task objects. This easily allows the designer to identify and draw connection paths. The arrangement of the tables can also be adapted quickly. This presentation type places a strong focus on the detailed task objects and their relationships.

An arranged task object overview in table form can also be converted into a template form. For example, the cores and paths method provides templates (see Figure 13) for describing each derived task object in detail [Kalb2012]. The designer determines the structure by following the inward and outward paths. Just like the overviews in table form, a stack of

completed cores and paths templates can be arranged easily to make the structure and the connection paths visible.

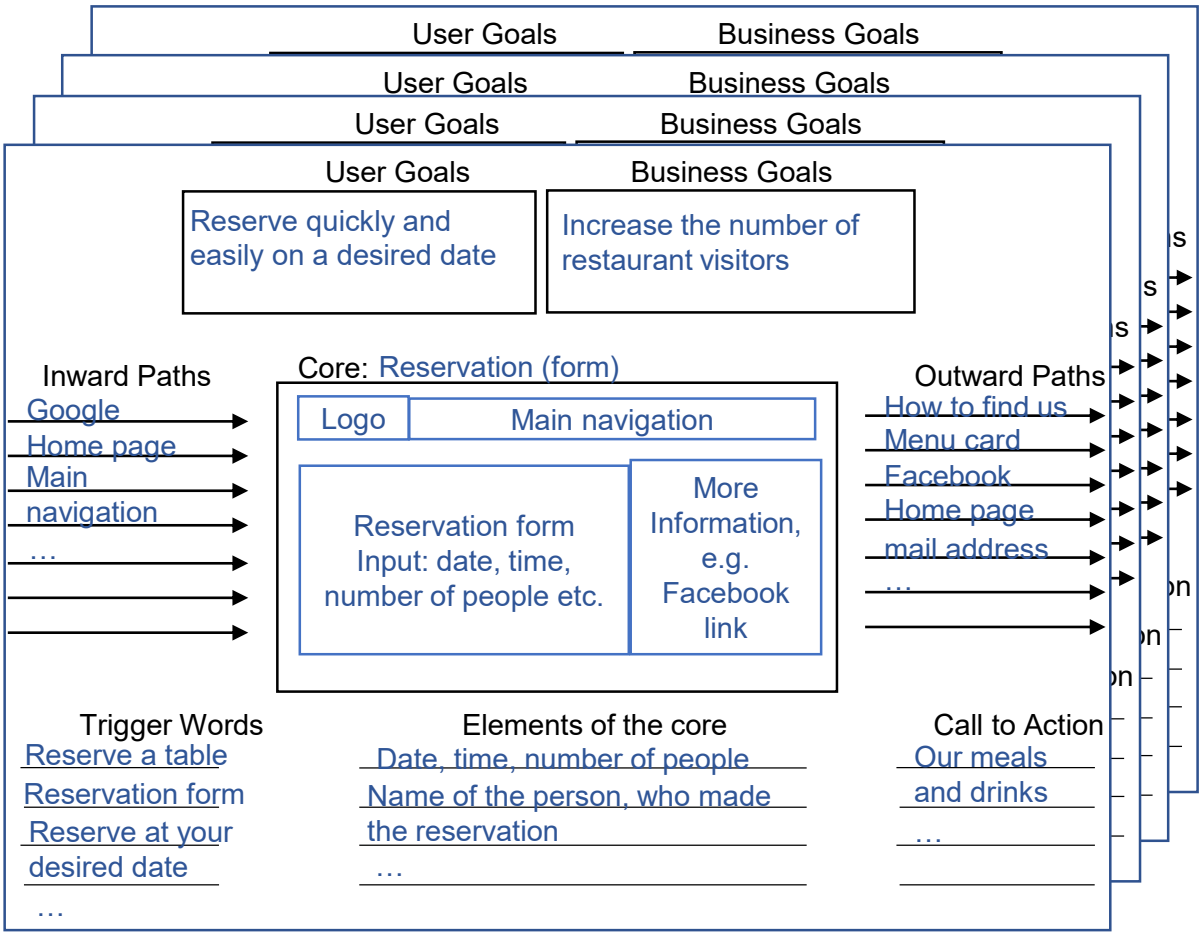


Figure 13 Example for a stack of filled out cores and paths templates

Hierarchical tree

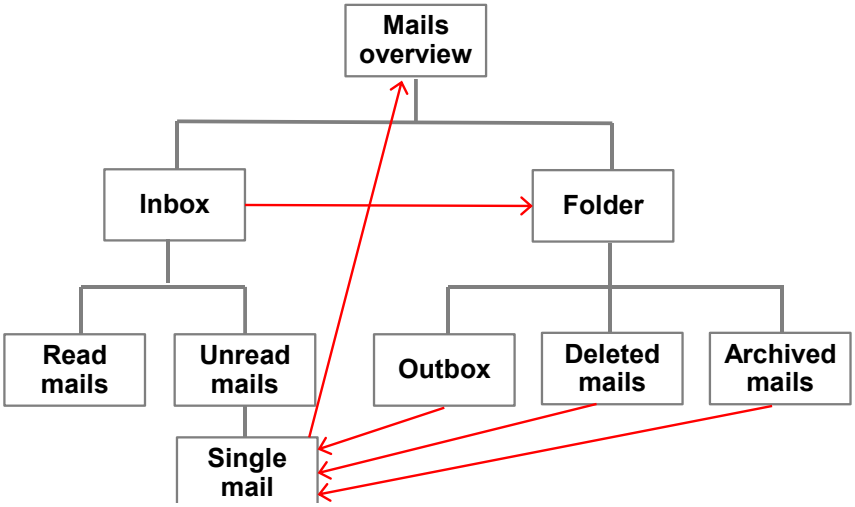


Figure 14 Example for a hierarchical tree

In a hierarchical tree, the task objects are arranged hierarchically. In contrast to the overviews in the tabular and template form, the hierarchical tree emphasises the structure of task objects and how they depend on each other. A super- and subordination of tasks becomes visible in a compact way. In comparison to the overviews described above, the hierarchical tree focuses less on visualising attributes, executable functions, and the labels of signposts. Calls to action can also be added to a hierarchical tree. Figure 14 shows an example of a hierarchical tree that contains calls to action (red arrows).

Tabular overview of all task objects and executable functions

- The individual tables can be transferred to one large table containing all detailed task objects and executable functions (Table 10). The advantage of this type of presentation is that all the task objects are in one place. Additionally, the relationships between individual task objects become visible. Any details missing from task objects can be identified and added easily. The connection paths between task objects can be understood by reading the columns “trigger words” and “calls to action” or illustrating them by adding arrows as you can see in Table 10.

Copyright 2025, UXQB e. V.

Table 10 Tabular overview of all task objects

Task object	Attributes	Executable functions	Signposts to executable functions and other task objects (calls to action):	Signposts from other navigational points (trigger words):

Restaurant	• Name
	• Address
	• Telephone
	• E-Mail
		• Show available tables
		  • Select room
		  • …
		• Reserve now
		• ..
			• Restaurants around you
			• …	

Reservation	• Person who has made a reservation • Date and Time • Table • Number of guests	• View an existing reservation • Make a new reservation • Reschedule an existing reservation • Cancel an existing reservation • …	• Show all available tables •	• My reservations • …

All available tables	• Available table 1 - Available table n
	• Date and time of availability
		• Specify a table
		  • Select shape
		  • Select location (at the window, in the centre)
		  • …
			• Tables at the window
			• …
				• Show available tables
				• …

Reservation confirmation	• Name of sender
	• E-mail address of sender
	• Name of restaurant
	• Address of restaurant
	• Date and time of reservation
	• Name of person who reserved
	• Number of guests
	• Type of table
		• View
		• Save in calendar
		  • select calendar (private, work, …)
		  • …
		• Cancel reservation
			• Add reservation to calendar
			• …
				• View confirmation
				• …

Copyright 2025, UXQB e. V.

3.1.5 Create the navigation structure using connection paths and signposts

3.1.5.1 Develop the navigation structure

The navigation structure is part of the information architecture, but not identical to it.

Navigation structure

CPUX-F Definition
The logical organisation of the units of displayed information that comprise the user interface.

Navigation structure is a part of the information architecture and forms the structure within which task objects and executable functions can be found specifically through signposts.

Navigation structure comprises all the navigation paths and shows how to get from one unit to another within the user interface structure.

It uses the actual signposts that will be presented to the user rather than the terms used in the information architecture. These signposts can be represented by icon/text combinations, hyperlinks, etc. The design of the navigation will already have been prepared during the development of the information architecture by specifying signposts and connection paths. [RoMA2015]. The connection path's signposts and labels can be found in the enhanced task objects.

Example:

- The signpost “recycle bin symbol” on an icon placed on each book in the “list of books selected for purchase” (labelled “shopping cart”) leads to the executable function “delete”, which results in the deletion of the task object “book”. In this example, the icon serves as a navigation element and is a part of the contextual navigation system.
- The link “This might also interest you” is the label for a connection path that takes the user in an online bookstore from the current page with the task object “book” to a page with the task object “suggested books” (that might be of interest to the user).

First, designers make a complete list of connection paths and signposts needed to visualise the navigation structure. The labels of all connection paths are extracted and arranged hierarchically; for example, in a navigation tree. Each entry in a navigation tree represents a cue for the user that helps them locate a needed task object or executable function or a particular form of user assistance.

The prioritisation and selection of entries within the navigation must be based on the results of the context of use analysis; for example, based on the prioritisation of user tasks by identifying the top tasks [McGo2018].

3.1.5.2 Navigation systems to implement a navigation structure

Navigation systems are means to implement a navigation structure and consist of navigation elements and subsystems.

Navigation system

An arrangement of information that provides users with their location in the interactive system. It serves the user to determine their position and helps them find their way through the interactive system (Where am I? What is nearby? What is related to what here?).

There are three major types of navigation systems (global, local, and contextual navigation systems) and several supplemental navigation systems (sitemaps, indexes, and guides) that

are typically used in desktop-oriented websites, but which also exist in mobile environments [RoMA2015].

- Global navigation is always available and offers direct access to the most important key areas and functions of the interactive system.
- Local navigation complements the global navigation system and allows the user to explore a selected area more deeply.
- Contextual navigation links to pages, documents, or task objects.

Example: The typical placement of user interface elements for the different navigation systems in a wireframe of a website or a web-based application:

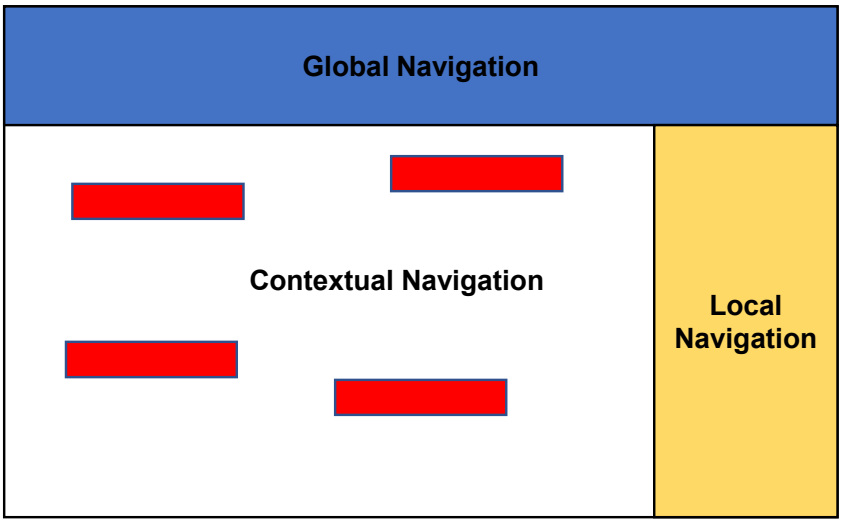


Figure 15 An example of the placements of different navigation systems

In addition, there are supplemental navigation systems that exist additionally to aid the user: sitemaps, indexes, and guides:

- Sitemaps provide an overview of the information environment.
- Indexes provide direct access to specific content.
- Guides provide a linear navigation tailored to a specific audience, topic, or task.

  Examples:
  - Sitemap: a table of contents in a book that represents the hierarchy of the book
  - Index: a back-of-book index that presents the keywords alphabetically without representing the hierarchy of the book
  - Guide: a walk-through that introduces new users to the system's functionality.

3.1.5.3 Structure types for navigation within content

When designing the navigation, the designer must first find out what content needs to be provided and how the user expects to navigate through it. Depending on this, the designer chooses a structure type for navigation within the content. Frequently used structure types are [Kalb2007]:

1. Linear structure: The linear structure is the simplest form of a navigation model. In the case of websites, for example, the concept means that every single page is linked to the previous one.

Figure 16 Linear structure

   Examples of a linear structure: Step-by-Step user guidance through the setup process of a smartphone. The payment process of an online shop.

2. Hierarchical structure: Hierarchical navigation is similar to the hierarchical tree. The home screen is at the top; downwards the form divides into categories, each of which branch out into further subcategories.

   Example of a hierarchical structure: Navigation on the Amazon website.

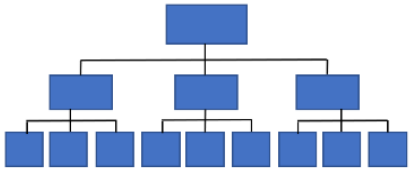


Figure 17 Hierarchical structure

3. Net structure: The net structure is more complex, but it does offer the option of extensive cross-linking of content. It can, however, also cause confusion if the user loses orientation.

   An example of a net structure: Navigation on Wikipedia website.

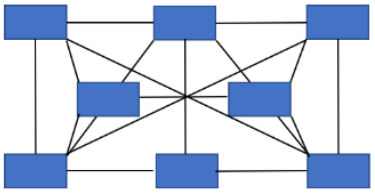


Figure 18 Net structure

4. Hub and spoke navigation: Hub and spoke navigation offers an efficient way to 'reset' a search and return to the starting point of the search (such as the home page).

   Example of a Hub and spoke navigation: Dashboard functionality as access to the apps installed on the smartphone. By pressing the Home button, the user can always return to the Home Screen.

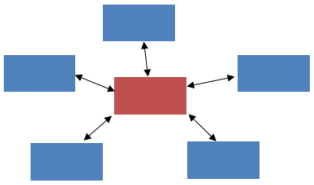


Figure 19 Hub and spoke navigation

Navigation structures are typically static, but can also be dynamic in various ways, meaning that navigation paths are made available at run time depending on the system state.

Examples:

- A cross-link is only made available on a page if a certain value is entered in a certain form field.
- A set of pages is placed in a hub structure. The hub is a search page and the only navigation elements are a search field and the button "search". The accessibility of the pages on the search page depends on the search terms.

3.1.5.4 Navigation elements as part of a navigation system

Navigation elements

User interface elements that serve as signposts, such as navigation bars (vertical or horizontal), functional navigation, breadcrumbs, contextual links or contextual buttons, or in-page links are used to navigate through the interactive system. Some user interface elements are typically used to create navigation systems. Frequently used user interface elements can appear as design patterns for navigation.

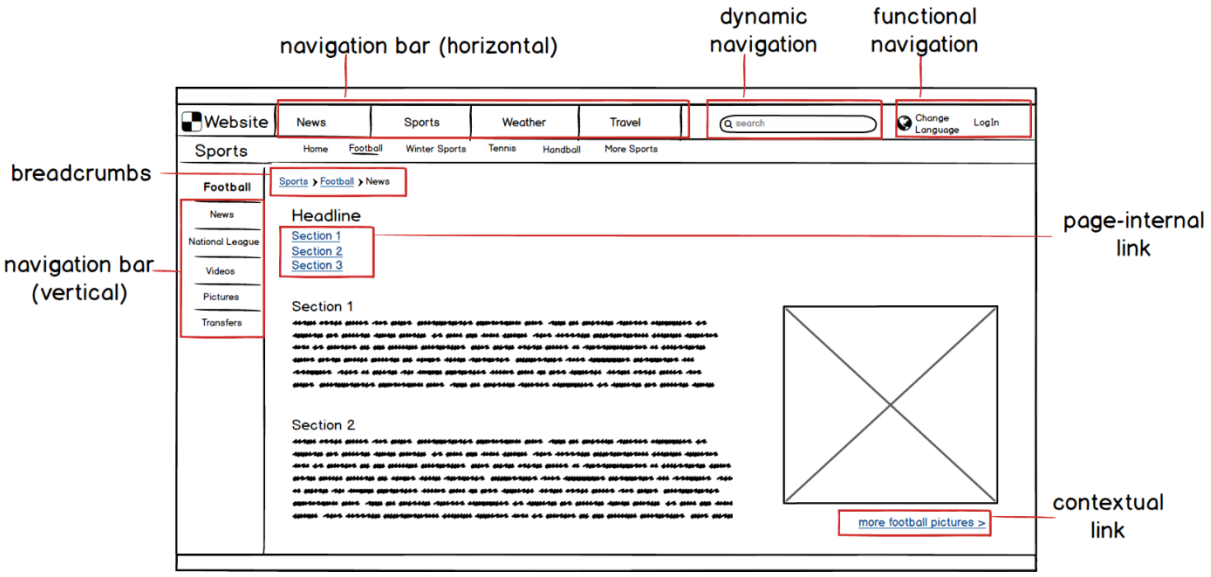


Figure 20: Navigation elements

Table 11 Navigation elements

Navigation element	Description
Navigation bar	Often called “menu”, this can be vertical or horizontal
Functional navigation	Provides access to the application itself by linking to pages that support specific functions such as “login” or “change language”
Breadcrumbs	Show users which hierarchy level they are currently located in and allow them to navigate to any higher level.
Contextual link or button	Often called “cross link”, this is a direct navigation to a specific page irrespective of its location in the navigation structure; a button used when the navigation action also involves the execution of a system function
In-page link	A link that navigates to a different place within the current page.

Example: Navigation elements

- Global navigation is displayed via the navigation bar (horizontal) and contextual navigation is displayed via contextual, in-page links.
- The arrangement of the contents follows a hierarchical navigation structure.
- The breadcrumb navigation element allows users to move systematically through the content.

Many other user interface elements or user interface patterns might serve as navigation elements, especially when implementing dynamic navigation. Dynamic navigation means that navigation paths and the associated navigation elements only become available to the user as they are needed.

The frequently used user interface elements for navigation can emerge as a design pattern for navigation.

3.1.6 Evaluate the information architecture

A common difficulty experienced when developing an information architecture is classifying the task objects into a clear hierarchy, especially when terms used for attributes or signposts may have different meanings to different people. To prevent this, the mental models of the users must be understood, to validate the navigation paths, the titles of task objects, and the labels of signposts to executable functions and other task objects. This will ensure that the information architecture matches the mental models of user groups. Two different methods can be used for this purpose: Card sorting and tree testing.

3.1.6.1 Card sorting

During card sorting, participants are asked to sort given terms into groups (open card sorting) or to classify them into prescribed structures (closed card sorting). Sorting, prioritising, and renaming the terms and categories reveals structures and relationships, and verifies users' understanding of the terms. Card sorting can be an analogue or a digital (remote card sorting) procedure.

Card sorting

CPUX-F Definition
A method for structuring information – such as menus in a **navigation structure** – that involves writing key concepts onto different cards and asking users to sort these cards into groups.

Card sorting can be used to identify existing mental models of the user and validate the information architecture. A card sorting activity can be either open (where users create terms and categories themselves) or closed (where users sort terms according to prescribed categories).

The results of a card sorting session (analogue) consist of the note-taker's documentation:

- photos of the structures of the categories (prescribed or named by the users themselves) with assigned terms
- notes made during thinking aloud
- the documentation of the interview that takes place during the debriefing, where the supervisor discusses the result, any new terms, and the categories, as well as the remaining categories or terms that are left over with the participant.

If there is no existing information architecture, an information architecture can be developed using the results of card sorting. The structure of the categories and assigned terms should then be used as the basis for navigation paths and the hierarchy of the information architecture. All terms, therefore, should be located within their assigned category and the categories should be easy to find. Any term that came from or was easily understood by the users during card sorting, should retain its name.

Copyright 2025, UXQB e. V.

Evaluating against an existing information architecture

If an information architecture already exists, it should be evaluated for its fit with the results of card sorting. In what cases were other categories formed by the users? Where are terms assigned differently? Where is a different wording used for the categories (in case of a round of open card sorting)? These deviations must then be evaluated, and a decision made regarding whether the information architecture needs to be adapted as a whole or the navigation structure needs to be adapted according to the terminology. The results of card sorting can be very informative for the design of the information architecture (notes from thinking aloud and documented interviews), as they provide explanations for deviations from the mental model based on the differing interpretations by the users.

When aligning the information architecture with card sorting results, quality criteria must be considered, and methodical errors must be avoided.

Quality criteria include

- the results of card sorting must be evaluated for validity (i.e. you usually run several card-sorting sessions).
- the changes resulting from the results of a round of card sorting must be communicated to the stakeholders (other than the users).
- the categories and terms must be named within the information architecture in line with the outcomes of the card sorting (i.e. it is about the exact wording in the card sorting).

A common methodical mistake is:

- arbitrarily renaming categories after testing when adapting the information architecture.

3.1.6.2 Tree testing

Tree testing

A method for evaluating the retrievability of information during task completion within a given navigation structure. The user moves through the navigation structure in order to perform a given test task successfully.

During tree testing, users search for information using clearly formulated tasks within a given hierarchical tree/menu structure. To conduct a tree test, all the main categories and subcategories of the intended navigation structure (the tree) must be prepared. The moderator gives the participant a piece of information to be found or a task, and lets them make navigation decisions while navigating through the navigation structure (tree) to get to the place where the task can be completed, or the information found.

The method makes possible detours visible, which the user uncovers when navigating through the menu structure. Different places users search for the information can be tested to determine the variant that appears most logical to the user. Tree Testing is performed analogously or digitally (for example, remote tree testing).

Copyright 2025, UXQB e. V.

3.2 Design activity: Interaction design

In Interaction design, design decisions are made based on the interaction specification and the information architecture.

The most important tasks for the designer are defining and visualising the task-related **interaction sequences** and **views** and, thereby, building the **user interface structure** of the perceptible part of the user interface.

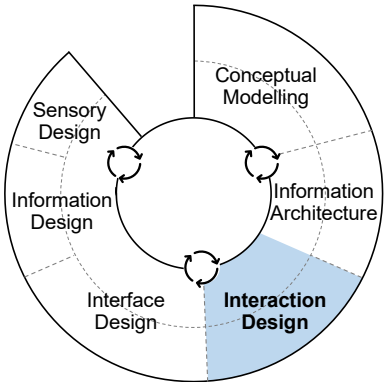


Figure 21 Design activity: Interaction design

Learning Objectives	
3.2.a	Know which design decisions are part of Interaction design.
3.2.b	Understand how to define interaction sequences and create the user interface structure.

3.2.1 Define task-related interaction sequences

Interaction sequence

The translation of dialogue steps specified in the interaction specification for each task into a sequence of interactions in a logical order in the user interface. If necessary, an interaction sequence will have multiple views which together represent the user interface structure.

The interaction sequence enables users to complete a task effectively and efficiently across all dialogue steps specified in the interaction specification.

For the translation of dialogue steps into interactions, the designer must decide which use scenarios to implement. For each supported task, the designer needs to

- sort the task objects structured in the information architecture along the sequence of dialogue steps, considering task objects and their interrelationships
- create the interactions and action sequences needed in the user interface for the user to effectively and efficiently work along the sequence of dialogue steps
- visualise the resulting interaction sequence.

3.2.2 Visualise interaction sequences

The chosen interaction sequences must be visualised for evaluation with users and stakeholders. The designer must make design decisions about the views that need to be provided (as well as the connection paths between views) and any forms of user assistance, for each interaction sequence to be visualised.

View

A composition of all the user interface elements perceived by the user in one specific temporal context (for example, at one point in time), representing one or more task objects. A view can also be a screen or a page.

The resulting views for each interaction sequence must enable the user to complete a task effectively and efficiently.

For the visualisation of each interaction sequence, the designer will

- use placeholders that represent task objects including information and signposts to executable functions
- arrange placeholders into a view or a sequence of views and determine how users will interact with them in order to work through the sequence of dialogue steps
- if possible, create multiple versions and variants of a visualisation for an interaction sequence
- test views for task-related interaction sequences with users and subsequently refine the interaction sequences
- repeat the procedure for every single interaction specification and its corresponding interaction sequence, aiming to re-use as many views as possible
- if necessary, adjust views or add new views in order to support the completion of all tasks.

Sometimes, it is possible to fit an interaction sequence into a single view.

Example: one view which represents two task objects

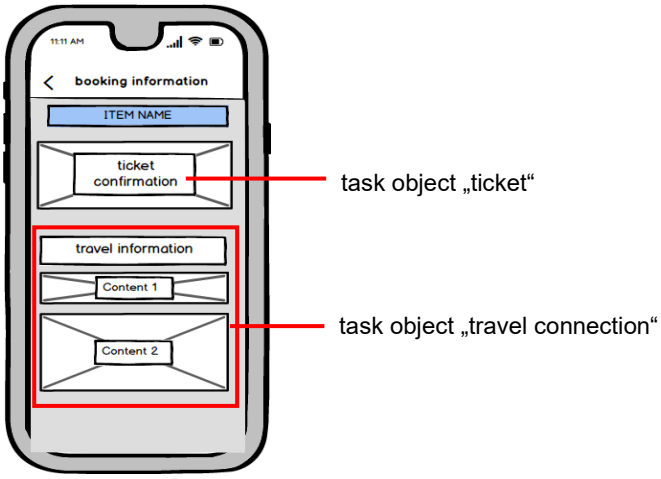


Figure 22: An interaction sequence in one view

If an interaction sequence does not fit in one view, the interaction has to be designed across multiple views.

Example:

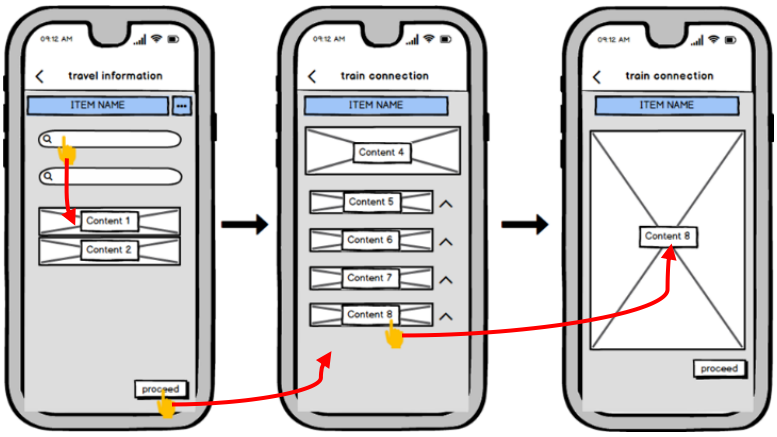


Figure 23: Interaction across a row of (three) views

It is helpful to use simple visualisations (such as the above sketches) for each task-related interaction sequence. The interaction design does not yet determine how the system looks. Instead, it focuses on the logical sequence of views for an effective and efficient interaction

through the necessary dialogue steps. Using visualisations can lead to modified design decisions for interaction sequences.

3.2.3 Structure the user interface based on all required views

By visualising task-related interaction sequences for all interaction specifications or use scenarios, the designer creates the views necessary in the interactive system for the user to work on their tasks and achieve their goals. The sum of all views and the connection paths between them makes up the user interface structure.

User interface structure

The arrangement of all views of the interactive system that enable users to perform their tasks.

An iterative refinement and adaptation of the user interface structure may be necessary, especially with the increasing number and complexity of interaction specifications or use scenarios involved. The designer has to make design decisions and, by considering the results of the context of use analysis, the trade-off between the number of views and the complexity of each view.

Walk-up-and-use-systems require views with low complexity so that users can find their way to the next step of an interaction without explicit learning effort. These systems prefer a greater number of views with less complexity over fewer views with more complexity, and sometimes they even provide just one interaction sequence for the whole sequence of views. Systems for experts on the other hand very often include interaction sequences for a larger number of use scenarios in one or a small number of views. This leads to views of higher complexity, reducing the necessity for navigation through views and minimising a “loss of context”.

Copyright 2025, UXQB e. V.

3.3 Make design decisions tangible to get feedback

The creation of design solutions is a process, where visualisations of drafted solutions are iteratively improved through feedback from evaluation with internal and external stakeholders, including users. Visualisations are work products that can be described in a differentiated way based on various characteristics.

While iterating design solutions, designers create visual work products with different levels of detail: Simple **sketches**, **wireframes**, **wireflows**, and **low-** and **high-fidelity prototypes**. They all help to make design decisions visible for communication and evaluation. Scope and complexity can be tailored to the current design phase.

Guidelines for creating low-fidelity prototypes as well as criteria for choosing appropriate prototyping tools are given.

Learning Objectives	
3.3.a	Understand the differences between sketches, wireframes, wireflows, and the different types of prototypes.
3.3.b	Understand the benefits of visualising design decisions in sketches, wireframes, wireflows, and prototypes, early and continuously throughout all design activities.
3.3.c	Be able to visualise and describe task-related interaction sequences based on a set of wireframes.
3.3.d	Know the criteria for selecting prototyping tools.

3.3.1 Characteristics of visualisations across design phases

A variety of visualisations can be created throughout the design process. Depending on the size of the project and the people involved, designers may decide to iterate using anything from one to many kinds of visualisations.

Visualisations have different characteristics that are suitable within a specific design phase or for a certain purpose. The following dimensions help to decide which visualisation will best fit the intended use.

The characteristics that differentiate visualisations are,

- The phase in the design process that the product is created and used in
  - First drafts
  - Refined design.
- Appearance and detail: What does the visualisation look like at first glance? During the design process, the appearance becomes more detailed and real, resulting in more time and effort needed for creation.
  - line drawings
  - structurally-arranged line drawings
  - a handcrafted dummy based to an extent on detailed line drawings
  - a very detailed screen view with interaction.
- Closeness to final product: How close is the experience with the work product to the experience with the final product? In earlier phases of the design process, the experience of work products often cannot be compared to the real product at all. Further into the process, the experience becomes closer and might be hard to distinguish from the real thing.
- Typical purpose: What is the main reason for creating the visualisation?
  - For rapid internal iteration: Draft multiple ideas per view

    - For internal iteration: Draft a rough layout of a view, draft a navigation structure using the rough layouts
    - For internal and external evaluation: Draft a screen design (placement of elements), visualise the detailed screen design.
- Recipient: Who is the visualisation addressing? During the design process, the recipients may be the UX team, the interdisciplinary project team, business stakeholders, or developers, but they can also be external: for example, potential users or client representatives.
- The number of views represented in the visualisation, i.e., how many views are included in one visualisation? In earlier phases, often just a single view is included, but later in the process, work products often comprise more than one view.
- The number of variants of the visualisation: How many different versions of the visualisation are created? In earlier design phases, often a lot of variants are drawn, evaluated, and discarded. As the design process proceeds, typically the number of variants goes down.
- Interactivity: Does the work allow simulation of use scenarios or include interactivity? If more than one state of a view is included, and the transition from one state to the next is visualised or if the visualisation is even “clickable”, it should allow a simulation of its use.
- With or without navigation structure: If more than one view is included and the transition from one to the next view is visualised, the work product should include a navigation structure.

The following section describes the typical types of visualisations. Table 12 on page 75 summarises the characteristics that differentiate them from one another.

3.3.2 Typical types of visualisations

Sketch

A short-lived work product for the designer’s use or for rapid team-internal iterations of first solution ideas in Early design that is used in preparation for other work products. It is a quick line drawing, usually created by hand. It comprises placeholders for content areas of just one view but is drawn in many variants.

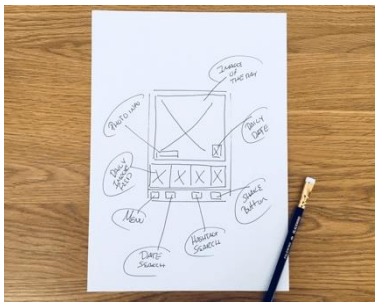


Figure 24 Sketch

Designers might produce a great variety of sketches to quickly try out ideas and in this process, learn about and understand user needs, user requirements, and the intended use of the system. Most of the sketches are quickly discarded and replaced by new ideas.

Wireframe

CPUX-F Definition
A screen or page in a low-fidelity prototype for a graphical user interface, comprised of lines, rectangular boxes, and text that represent the intended interaction design.

A work product for internal validation of a single view in the First drafts phase. It's a line drawing, either created by hand or using software.

It shows a rough layout of just one view, possibly in several variants, and helps the internal team and stakeholders decide on the main direction for further development of the solution.

Wireframes comprise placeholders for user interface elements as needed.

A wireframe shows a rough layout of one view, possibly in several variants to help the internal team and stakeholders to decide on the main direction for the further development of the solution.

Wireframes are most often the product of some thought and understanding of user needs for a single view. Sometimes a few variants are created that represent different approaches for the solution.

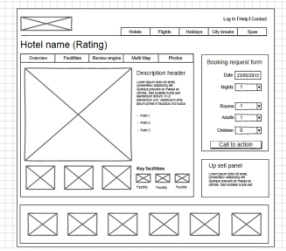


Figure 25 Wireframe

They are more elaborated in terms of layout and details than sketches and are often used as the basis for low-fidelity prototypes [JaMe2017].

Wireflow

A work product for internal validation of the system's navigation structure in the First drafts phase. It consists of several wireframes that show the interaction with the intended design solution and helps to iterate the navigation structure of a drafted solution.

In a wireflow, multiple wireframes are arranged in a task-related order to demonstrate future transitions from one screen to another. An iterated and evaluated wireflow may serve as a basis for the subsequent creation of a low-fidelity prototype.

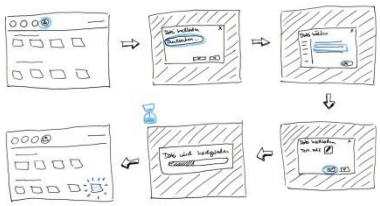


Figure 26 Wireflow

Prototype

CPUX-F Definition
A representation of all or part of an interactive system that, although limited in some way, can be used for analysis, design, and usability evaluation.

Prototypes fall into two types: low-fidelity prototypes and high-fidelity prototypes.

Copyright 2025, UXQB e. V.

Low-Fidelity prototype

CPUX-F Definition
A low-cost, simple illustration of a design or concept used to gather feedback from users and other stakeholders during the early stages of design.

It is based on preceding work products and advances them. In the First drafts phase its purpose is for internal and external evaluation of the draft system's screen design. It is a handcrafted or digital visualisation, often based on wireframes that resemble the product and include one or more views containing detailed content as required.

Low-fidelity prototypes are often the product of many iterations and a lot of internal feedback. They are typically based on evaluated wireframes and wireflows and show a comparatively advanced layout to wireframes. They are presented in a more detailed screen design and show some resemblance to the final product.

Figure 27 Low-Fidelity Prototype

Low-fidelity prototypes should precede the creation of high-fidelity prototypes. A variant of low-fidelity prototype is a paper prototype, which may be a creatively handcrafted version of the intended product for users to try out [Buxt2007].

The "Wizard of Oz" prototype variant is used in situations where user inputs are unpredictable. The highly flexible reactions of the system (prototype) are simulated by a human working as a hidden operator. "Wizard of Oz" prototypes are useful when design ideas are still wide open and they help to understand how users behave naturally and what they expect from the system [HaPy2019].

High-fidelity prototype

CPUX-F Definition
A software prototype of the user interface to the interactive system that is being designed. A high-fidelity prototype more closely resembles the finished interactive system.

A high-fidelity protoytpe is a sophisticated work product for the evaluation of the user experience of the system's screen design in the Refined design phase. It is a digital representation, usually created in only one version that resembles the product closely and may contain any number of detailed views, and includes interactivity and a navigational structure.

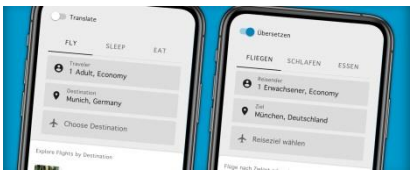


Figure 28 High-fidelity prototype

High-fidelity prototypes encapsulate the pre-final stage of the layout and sensory design decisions of a solution when the interface is very close to supporting all of the agreed user tasks in a design project.

There are "vertical" or "horizontal" high-fidelity prototypes, depending on whether they support drill-down interactivity (depth) or dip-in exploration of a menu or navigation structure (width).

For the work products introduced so far, various names might be used in different contexts. A frequently-used term for some or all of these work products is mock-up. The term mock-up is not defined in this curriculum.

Table 12 Types of visualisations and their characteristics (key characteristics are highlighted in bold)

	First Drafts				Refined Design
Type of visualisation	**Sketch**	**Wireframe**	**Wireflow**	**Low-fi prototype**	**High-fi prototype**
Typical purpose of work product	To draft, discuss and discard rapid iterations of first ideas, in preparation for other work products	To iterate rough layouts of a single view for internal validation	To evaluate interaction sequences and navigation structure, internally	To evaluate screen design with users and other stakeholders	To evaluate screen design and user experience with users and other stakeholders
Appearance and detail	Hand-drawn, line-based illustration, with placeholders for view areas	Hand- or digital-drawn, line-based illustration, with placeholders for user interface elements and content (as needed)	Structurally arranged wireframes	Handcrafted or digital representation of all, or parts of, the final product, with content as required	Digital representation of all, or parts of, the final product, with detailed content
Resemblance to the final product	No resemblance	No resemblance	No resemblance	Some resemblance	Close resemblance
Intended for use by	Designers and their team	Design team; stakeholders	Design team; stakeholders	Design team; stakeholders; users	Stakeholders; users
Number of views per work product	Number of views	One	One	**More than one**	**Any, depending on the interactive system and the purpose**
Number of variants per work product	**Many**	One or more	**Typically one or very few**	**Typically one or very few**	Typically one
Interactivity	No	No	No	Yes – whatever is necessary for evaluation	**Yes – whatever is necessary for evaluation**
Navigational structure	None	None	Yes	Yes – whatever is necessary for evaluation	**Yes – whatever is necessary for evaluation**

Copyright 2025, UXQB e. V.

3.3.3 Benefits of visualising design decisions early and continuously

The early and continuous use of visual work products in the design process is key for the iteration of first drafts, through refined designs to the eventual solution.

The different types of visualisations allow the designer to tailor their use according to the design phase the project is currently in.

The benefits of using visual work products are that

- in the First drafts phase, they are simple, fast, and cost-effective visualisations
- they allow for the quick use and disposal of work products
- they are a means to generate a great variety of ideas, even seemingly unpromising ones
- they are a means to gain a better understanding of the user needs and therefore the design project as a whole
- they are a means to work on early drafts in interdisciplinary teams
- they are a means to quickly present ideas for communication with different stakeholders and users
- they are a means for stakeholders to propose changes and add new ideas.

In the Refined design phase, narrowed down visualisations help

- facilitate effective stakeholder discussions and efficient user feedback (surveys, feedback gathering, and usability testing)
- efficiently uncover needs for adjustments
- facilitate quick re-testing of an improved visualisation
- facilitate the documentation of design decisions
- facilitate the specification of the eventual solution.

3.3.4 Iterating visualisations through evaluation

The primary purpose of creating visualisations for the intended solution is to improve them iteratively, through feedback. Depending on the design phase and the level of detail of the work product, feedback is gathered from different groups of people using different methods.

The first sketches may be iterated through informal feedback within the design team. The feedback at this stage centres around the most basic assessment, “can this work at all?” This way, the designer or the design team improves and narrows down ideas to a reasonable set of variants which they can follow up on. These might then be visualised in wireframes.

Wireframes and wireflows can be iterated through informal feedback from within the design team or from business stakeholders. At this stage, the leading question for wireframes is “which variant of solutions do we want to pursue further?”, whereas for wireflows, it is “does the navigation structure seem reasonable?” Users may be involved at this stage, especially if more elaborate work products will not be created in the design project.

Low- and high-fidelity prototypes are often iterated through formal feedback from evaluations with stakeholders and users.

3.3.5 Guidelines for creating low-fidelity prototypes

Low-fidelity prototypes are often based on wireframes and wireflows that are either hand-drawn or created digitally. In the creation of a low-fidelity prototype, the designer should follow a sequence of steps:

1. Gather all UX deliverables for the intended solution that were established so far in the design process. These may include,

Copyright 2025, UXQB e. V.

   a. an interaction specification
   b. a possibly adjusted task model for design
   c. derived task objects
   d. enhanced task objects
   e. structure across all detailed task objects.
2. Ensure the application of applicable user interface guidelines represented in, for example, ISO 9241-110, heuristics, design patterns, or style guides. Also see Chapter 6.2.
3. Depending on various factors like efficiency, required portability, personal ability, or company standard, decide whether to use hand drawings or digital tools for the creation of the prototype.
4. Create one wireframe per view, each including details and interaction elements as needed.
5. In the case of a paper prototype, use cut-outs and handcrafted materials to turn a hand-drawn prototype into a paper prototype, showing the interaction inside each view.
6. Discuss the low-fidelity prototype with team members and gather internal feedback in order to optimise the prototype by internal iteration.
7. Annotate the wireframes to explain the user's flow through the views. Also see Chapter 6.46.4.3.

Quality criteria for creating low-fidelity prototypes

- Work in a team: Internal discussion and iteration of the drafted solution will improve design decisions.
- Avoid concentrating on content or interaction details that are not necessary for the design task.
- Avoid showing any visual design – like colours or fonts – that is not necessary for the design task.
- Verify the created visualisation against all information available from the context of use.

Common methodical mistakes when creating visualisations of interaction sequences include,

- skipping internal iterations
- choosing an inappropriate level of fidelity for the visualisation
- concentrating on too much detail
- emphasising aspects of visual design
- skipping the comparison of the visualisation with information from the context of use.

3.3.6 Criteria for selecting prototyping tools

Even though visualisations can be drawn by hand or built using handcrafted materials, often digital prototyping tools are used. Prototyping tools in the form of software allow for fast creation of improved versions, as well as comprehensible documentation of design decisions.

With access to prototyping tools it can be tempting to create unnecessarily mature work products that include a lot of detail and unnecessary interactivity. As a general rule, the following can be said: The selection of the prototyping tool and the degree of maturity of the resulting work product should reflect the current design phase, the purpose of the visualisation, and the amount of time available for its creation.

The criteria for selecting the appropriate prototyping tools for a work product are as follows:

1. The tool must support the required level of fidelity of the work product.
2. The tool should ideally provide modes to create work products with varying degrees of fidelity, from hand-drawn screens to the appearance of a finished product. The tool should at least provide the level of fidelity needed for the design project at hand.

Copyright 2025, UXQB e. V.

3. The system must support the required experience of the solution for users and other stakeholders. In some cases, the recipient is supposed to have an interactive experience; sometimes the focus is on the appearance of a system, and in other cases, an extensive experience of task completion (in depth) is required.

   Example: Some tools allow the designer to link photographed sketches together for display on a stakeholder's mobile phone, and so provide a realistic use experience with minimal effort.
4. The system should support the required documentation of design decisions in the work product. Some tools provide the means to annotate or create a specification out of the work product. Some prototyping tools support both communication to stakeholders and documentation. For documentation, it is important to
   - annotate work products
   - document conventions (syntax)
   - ensure traceability to the UX deliverables that served as input – even at a later point in time
   - document considerations, variants, as well as the advantages and disadvantages of certain decisions.
5. The system may support the documentation of evaluated user interaction with the work product. Some prototyping tools provide screen and voice recording or mouse-click and key press tracking, in test sessions with users.

Copyright 2025, UXQB e. V.

4 Refined design

In interface design, user interface elements are selected, arranged, and combined, and their behaviour is defined. In information design, the designer ensures the comprehensibility and consistency of all meaningful information of the user interface. In sensory design, designers focus on the perception of relevant sensory channels.

4.1 Design activity: Interface design

Interface design deals with design decisions regarding the selection, arrangement, behaviour, and combination of **user interface elements** for all screens, views, or pages.

While creating interfaces for devices of different sizes, the designer must decide on the dynamic or static arrangement of user interface elements using development approaches such as r**esponsive design** or **adaptive design**. **Mobile-first** is an important design objective. The application of user interface guidelines contributes to effective, efficient, and satisfying user interfaces.

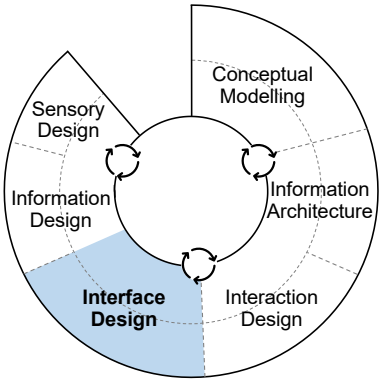


Figure 29 Design activity: Interface design

Learning Objectives	
4.1.a	Understand how to select and adapt user interface elements, their combination, arrangement, and behaviour on each individual view.
4.1.b	Know how to decide on the dynamic or static arrangement of user interface elements.
4.1.c	Understand that applying user interface guidelines for selecting, arranging, and combining user interface elements supports effective, efficient, and satisfactory interaction.

4.1.1 Create the interface design by selecting, arranging, combining, and defining the behaviour of user interface elements

User interface element
CPUX-F Definition
A basic component of a user interface that is presented to the user by the interactive system.

User interface elements have to be selected, arranged, defined regarding their behaviour, and combined depending on the user's tasks and with regard to meeting user requirements.

Before the designer selects user interface elements, they have to answer two questions:

- How must user interface elements be combined and arranged, so that the relevant task objects and executable functions are perceived and understood by the user?
- Which user interface elements correspond to the prior knowledge, experience, and expectations of the users?

Copyright 2025, UXQB e. V.

4.1.1.1 Select, arrange, and combine user interface elements

User interface elements must be selected, combined, and arranged according to the content and purpose so that the user can find them at the right moment within the interaction and use them to achieve their goal.

Example: In image processing, many user interface elements are used, such as the toolbar, individual buttons, or sliders.
The “slider” user interface element is suitable for adjusting the brightness, contrast, and saturation of an image.

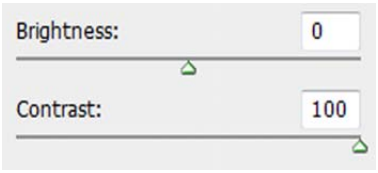


Figure 30 Typical user interface elements used for image processing

The combination of a slider, caption, and input field must allow the user to perceive these user interface elements as belonging together for the processing of the tasks “Adjust brightness” or “Adjust contrast”.

If suitable UI elements have been selected, combined, and arranged, the designer must also define the behaviour of the elements, for example, the duration of opening an element or the change of colours when changing the state.

4.1.1.2 Decide on the dynamic or static arrangement of user interface elements

Rather than creating separate websites or apps for wide-screen monitors, desktops, laptops, tablets, phones and smartwatches of all sizes, a single codebase can support users with different size viewports.

Responsive design

A development approach that results in dynamic changes to the appearance of a website or an app, depending on the screen size and the orientation of the device used to view it.

Examples of responsive design include

- page elements are reordered as the viewport grows or shrinks. A three-column desktop design may change to two columns for a tablet and a single column for a smartphone.
- horizontal lists of links are converted into drop-down menus.
- font sizes are adjusted.
- images are resized rather than cropped, or different images are displayed depending on the viewport size.
- secondary elements, such as supplementary navigation, are hidden on small viewports.

Rather than creating a single, fluid layout that adapts to changes in screen width, static layouts are created at key ‘breakpoints’ so that the layouts can be optimised for those specific widths.

Adaptive design

A development approach that employs several static, tailor-made layouts.
The appropriate layout is selected and displayed based on the screen width of the device used to view it.

It is common to design for the following screen widths in adaptive design: 320, 480, 760, 960, 1200 and 1600px.

Copyright 2025, UXQB e. V.

Adaptive design can be a useful approach for making existing desktop sites ‘mobile-friendly’, as the additional layouts can be created without affecting the existing site.

Adaptive design allows designers to incorporate functionality, data, interaction methods and features, native to the user’s device and its operating system.

Examples: swipe gestures, behavioural targeting, use of the camera and microphone, GPS, and biometric authentication on mobile.

Some key differences between adaptive and responsive design are:

- responsive design provides a uniform layout across all screen widths.
- adaptive design provides greater control for designers, allowing designers to optimise their design for each screen width.
- responsive design has only one core layout so takes less work to build and maintain.
- adaptive design is more labour-intensive, as designers need to build and maintain several layouts
- responsive design is fluid and so fills all the gaps that adaptive design would leave; however, responsive sites can behave in an unpredictable manner when resizing the screen, as elements move around the screen.
- adaptive design only accommodates the screen widths you have designed for
- responsive websites require sending all the code that generates the layout at all widths to the client, so they can be slower to download and respond, especially on low-bandwidth connections and less-capable devices.
- adaptive websites only require the code for that specific layout so they can be a lot quicker to download and respond – often 2 to 3 times faster than responsive websites.

To support an informed decision about static or dynamic design, ”Mobile first” is an important design approach:

Mobile-first

An approach in which the version that is optimised for mobile devices – in other words, smaller screens – is created first, and extensions for bigger screens (for example, desktop application) are designed later.

4.1.2 Appropriate use of user interface elements

As a decision-making aid for the use of suitable user interface elements, guiding questions help to clarify what the respective interaction task is about, which user interface elements are available, when they are used, which design guidelines are to be taken into account, and how the elements are arranged or combined appropriately.

Before the designer considers a new user interface element, they should research whether a suitable solution for the concrete case already exists. Only if no suitable user interface element for a problem exists must it be adapted or newly created.

1. Type of interaction: What is it about?
   - Data input
   - Data selection
   - Data display
   - Execution of commands
   - Grouping of contents.
2. Which user interface elements are available for this interaction?
   - Examples of data input: Input field, dialogue box, calendar picker

Copyright 2025, UXQB e. V.

- Examples of data selection: dropdown, radio buttons for single and check boxes for multiple selections.

3. In which cases do you use a certain user interface element? For example,
   - input field: The system requires information from the user that cannot be presented as a pre-defined option.
   - dialogue box: Explicit decision, action, or data entry is required.
   - calendar picker: A single date must be chosen by the user.
4. Which design guidelines are to be applied for the selected user interface element? Example for a calendar picker:
   - The month and year should be selectable independently from each other.
   - Local cultural date conventions should be observed during the implementation.
   - Users must be provided with the ability to set a preferred date format.
5. How should user interface elements be combined and arranged?
   - Arrange groups of user interface elements as expected
   - Group elements with similar content together
   - Make dependencies visible
   - Support the natural flow of reading
   - Display states correctly.

The consideration of these key questions and the application of corresponding user interface guidelines enable the designer to make appropriate design decisions that lead to the design of effective, efficient, and satisfying user interfaces.

[ISO 9241-161] contains guidelines for the use of user interface elements as well as an overview of typical user interface elements, including a description of possible states, components, and additional notes for the application.

Selecting user interface elements in accordance with user's expectations, previous knowledge, and experience is becoming increasingly important, as there are more and more people who (must) have regular access to a wide variety of interactive systems. The selection of terminology, visual language, symbols, and recurring user interface elements must be consistent with elements commonly used (within the interactive system, or within a product family or platform, such as an operating system).

Example: Typical mistakes when selecting user interface elements

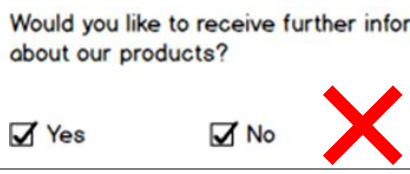
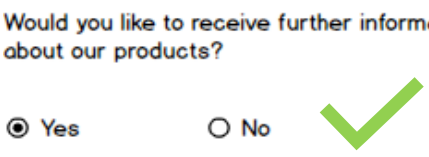
Incorrect Selection	Correct Selection
	
The use of checkboxes allows the user to select multiple options. The user interface element checkbox is not suitable in this case, because the user can answer both yes and no to the same question.	The use of radio buttons allows the user to select only one option. The radio button is suitable in this case because the user has to decide whether to answer yes or no to the question.

Figure 31 Typical mistakes when selecting user interface elements

Example: Typical mistakes when arranging user interface elements

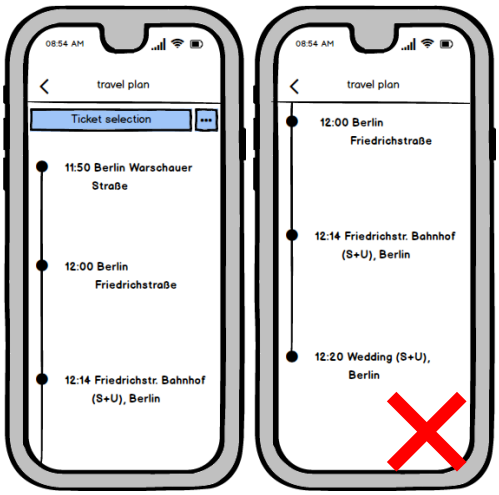
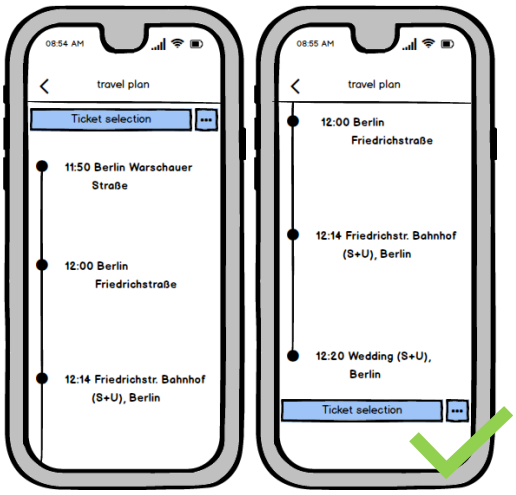
Incorrect Arrangement	Correct Arrangement
	
The “Ticket selection” command button is designed to only be visible at the top of the train connection. In order to select an offer, users have to scroll back up after they have scrolled down to read all route information.	The “Ticket selection” command button is configured so that it is always visible and can be selected regardless of the scroll position.

Figure 32 Typical mistakes when arranging user interface elements

Example: Typical mistakes when combining user interface elements:

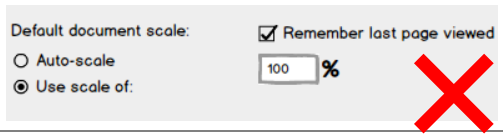
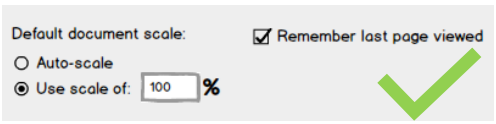
Incorrect Combination	Correct Combination
	
The input box for users to specify the scaling factor has been arranged in such a way that the relationship between the radio buttons and the caption is no longer clear.	The user interface elements are arranged correctly so that the relationship between the radio buttons, the caption, and the input box is clear.

Figure 33 Typical mistakes when combining user interface elements

Important characteristics of human perception must be considered in interface design to ensure that the user interface elements are perceived as a whole. Furthermore, attention must be paid both to internal consistency (uniform use of user interface elements within an application) and external consistency (uniform use of user interface elements across different applications). The use of style guides and design pattern supports these goals.

Example:

The “slider” element of the user interface is sufficient for an occasional user to adjust the brightness of an image during image processing. Sliders work best when the exact setting is not important. A professional photographer may need an input field rather than a slider, so they can set an exact brightness value.

Copyright 2025, UXQB e. V.

4.2 Design activity: Information design

The purpose of Information design is to make information in the user interface meaningful and comprehensible. In addition to conformance with the **principles for the presentation of information**, the process of **reading of information** must also be considered. Users often read digital content according to the **reading scheme** “F”. Frequently, this results in a loss of information that can be avoided by using the **inverted pyramid** approach. Other specific guidelines help the designer prepare and structure content in a clear and comprehensible way.

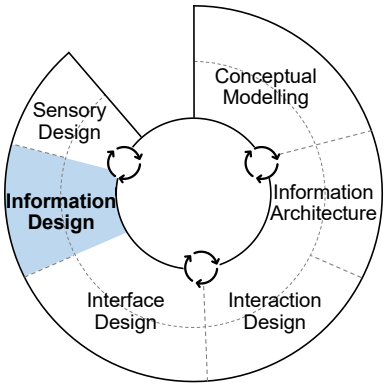


Figure 34 Design activity: Information design

Learning Objectives	
4.2.a	Understand the principles for the presentation of information
4.2.b	Understand how reading of information and comprehension of digital content can be supported using the inverted pyramid approach in order to consider the reading scheme “F”
4.2.c	Know the rules for simple language
4.2.d	Know the basic guidelines for the use of colours, graphics, images, and symbols in terms of comprehensibility

4.2.1 Principles for the presentation of information

Information design aims to facilitate perception and understanding of information by reducing the cognitive effort required to assimilate and process the information [RoCa2002].

The principles for the presentation of information serve to support the information’s self-descriptiveness and conformity with user expectations.

Principles for the presentation of information

Principles whose application in the development of user interfaces ensures good usability and a positive user experience from the viewpoint of presenting the information.

Considering these principles in the design of an interactive system ensures that users can handle content in an efficient, effective, and satisfactory way. [ISO 9241-112] “Ergonomics of human-system interaction – Part 112: Principles for the presentation of information” formulates six principles for the presentation of information that apply across the sensory channels (for example, visual or auditory). In addition, [ISO 9241-125] contains many user interface guidelines for the presentation of visual information.

Table 13 Principles for the presentation of information

Principle	Definition and Example (further examples can be found in ISO 9241-112)
Detectability	Presented information is detectable if the information is presented so that it will be recognised as present. Example: A prominent headline (large font and prominent placement in the upper centre) with a supporting image allows readers of a newsletter to quickly see what this short text is all about.
Freedom from distraction	Information is free from distractions if it is presented so that the required information will be perceived without other presented information interfering with its perception. Distractions from a user's point of view can result from both distracting events and information overload. Example: Playing a video, the website automatically adjusts the contrast of the background to avoid distracting users from the video.
Distinguishability	Information is distinguishable if it is presented such that discrete items or groups of items can be accurately differentiated, and the items of information are presented in a manner that supports their association with or differentiation from other items or groups of items. Example: On a news website, different menu levels are designed in different colours to clearly distinguish them from each other.
Interpretability	Information is interpretable if it is comprehended as intended. Example: Supporting images for each category name of an online shop clarify the content of the categories.
Conciseness	Information is concise if only the necessary information is presented. Example: On a website menu for booking tickets for events, various subcategories can be selected step by step as additional menu items. So, the user does not get too much information at once.
Consistency (internal and external)	Information is consistent if items of information with similar intent are presented similarly and items of information with different intent are presented in a different style and form, within and across the interactive systems and the user's environments. Example: A link named "Help" on a Web page takes the user to a page titled "Help" rather than to a page named "Information".

The principles detectability and freedom from distraction and distinguishability specify the design of information to support user's reading of information. The principles interpretability, conciseness, and consistency ensure the comprehensibility of information.

4.2.2 Information reading and comprehensibility of content

4.2.2.1 Step-by-step reading of information

Reading of information

An interplay of three ways for capturing content:

1. Scanning: User's attention during task completion is guided by the content
2. Skimming: Scanned contents are systematically passed over to get their gist
3. Reading: If a skimmed piece of content is recognised as helpful for task completion, the user reads its information systematically and completely.

Scanning phase: During the scanning phase, the user's attention is primarily directed towards the characteristics of the content. These are broadly scanned, and key information (headings, links, images) is noted and digested. A large amount of information needs to be dealt with. The user's attention is not systematically controlled, and information is processed at a relatively superficial level.

Skimming phase: The content that has been scanned is now skimmed to get a rough understanding of the content and arguments. The reading speed decreases and the amount of information to be systematically assimilated increases.

Reading phase: The third phase involves actual reading. The user decreases the information reading speed again and begins to systematically and comprehensively assimilate the presented information. Attention no longer hovers but becomes focused.

In order to process information, users need to rate the content according to how relevant they think it is for their task completion. Therefore, information needs to be structured and arranged in such a way that the user can quickly identify which information is of relevance during the hovering attention phase (scanning and skimming).

4.2.2.2 Supporting reading of information for digital content

Users assimilate digital content differently to printed text as they usually prefer to scan it. Reading schemes describes this process.

Reading scheme

A pattern describing the typical chronological order in which users place their attention on a page. The intended reading scheme for a page or set of pages depends on the basic layout, the design guidelines followed, and the extent to which single user interface elements get the user's attention on the page.

Examples of reading schemes are,

- F-reading scheme (starts top left, scans to the right side for keywords, starts again top left, scans the left down, scans the centre)
- Z-reading scheme (starts top left, scans to the right, scans across the mid-point to the bottom left, scans to the right)
- Central before peripheral (starts in the centre, moves to the peripheral)

The reading of digital text usually follows the "F" reading scheme. The user generally notices the right-hand side of the screen only if there are interesting stimuli.

The reading scheme "F" is neither optimal for users nor for conveying information, because information can easily be missed out. As content is displayed differently on various devices in responsive design, the content assimilated by the user using the "F" reading scheme differs according to the device they are using [Pern2017].

Copyright 2025, UXQB e. V.

When the main area of a page mainly consists of text, structuring information in accordance with the inverted pyramid approach improves the user's understanding of digital text. It directs the user's attention to the most important information more effectively and avoids information getting lost due to the "F" reading scheme.

Inverted pyramid approach

A guideline for the structuring of information within a text which states that the most important (often summarising) information should be placed at the beginning of the text, followed by supporting details and background information.

The inverted pyramid approach dictates the optimum way to structure information, to improve users' understanding of digital text:

1. Begin with the essential information
2. Follow with details and background information
3. Write in a concise way that is easy to understand
4. Identify your key messages and ensure they are clear
5. Sort your information according to relevance
6. Add summaries or highlight the most important points.

The inverted pyramid approach improves the comprehensibility of digital text, saves the user's resources (time and cognitive effort), helps the user to scan the text, and encourages them to keep reading [Scha2018].

4.2.2.3 Ensuring the comprehensibility of text with the rules of simple language

In particular, digital text must be prepared linguistically and stylistically according to the rules of simple language to make it easier to read [Mose2012, NiLo2006]:

Linguistic Level

- Use precise and meaningful headings
- Use short, simple sentences
- Choose objective, informative, and positive phrases
- Be consistent with the choice of words and sentence structure
- Gear the language towards the user
- Avoid incomprehensible rhetorical devices; for example, a metaphor unknown to the user
- Avoid abbreviations or in-house, technical terms, and specialist jargon
- Make the reader respond to different calls-to-action
- Use links with short and meaningful names.

Stylistic Level

- Ensure that recommended actions follow the correct sequential order
- Structure text in a logical and comprehensible way by choosing concise page headings that use keywords to describe the content
- Structure text in a visually appealing way using headings, subheadings, images, tables, lists, and key statements in order to make it easy to scan
- Only discuss one topic per paragraph
- Use teasers to draw attention to specific pieces of information
- Keep paragraphs short
- Never assume users' knowledge level.

Copyright 2025, UXQB e. V.

4.2.2.4 Ensuring the comprehensibility of information through colours

Figure 35 Example of colour coding

Designers must follow basic rules for the use of colours when designing information for interactive systems:

- Use colour sparingly and cautiously
- Match the colours used and choose no more than four main colours, as extreme colourfulness makes the choice of colour appear arbitrary
- Ensure a clear presentation through the high-contrast use of colours
- Be aware that the effect of a text may depend crucially on the colours used (For example, to draw attention to a use error and ensure its perception, the error text is displayed in red rather than in black or dark grey).
- If a colour has been assigned with a purpose or function within the interactive system, use it only for this purpose. It is possible to work with colour gradations instead.
- Be aware of colour conventions regarding colour coding, colour standards, or different cultural interpretations of meaning for colours.
- Never rely on colour alone to convey information. Users with colour blindness are unable to differentiate between certain colours. Based on culture, mental models, and interpretation, users may assign meaning to colours that was not intended by the designer. The most appropriate solution to conveying information is always to use text and support that information with colours and/or shapes.

  Example for colour coding: In an app, the user sees a train connection. There are two colours used in the connection details: red and green. In this combination, red could be interpreted as bad and green as good. Red indicates that the train will be very busy, while green indicates a low occupancy. Users with colour blindness may not be able to differentiate between the two colours.

4.2.2.5 Special features of language and images in terms of comprehensibility

In general terms, it is preferable to communicate information by linguistic means (text) rather than in a graphical way (images) because linguistic means are a particularly popular way to convey information and enable complex messages to be conveyed. Furthermore, linguistic means are less likely to be misunderstood compared to icons or other non-linguistic means. Therefore, the comprehensibility of icons can be supported using additional linguistic means [Ware2004].

Example: The floppy disk symbol represents the function “Save”. An additional label below the icon with the text “Save” supports its comprehensibility.

Graphical representations (images) offer advantages, too. They can be captured and understood at a glance, while text must be decoded (read).

Images can convey more emotionally-intense messages. Visualisations and graphical representations are well-suited to conveying metaphors visually. A combination of graphical representation and textual label is often advisable.

During the design process, it is important to determine which of the different ways of presenting information supports information transmission most effectively.

Table 14 Ways of presenting information

Images (static) are suitable for describing	Language is suitable for describing
• spatial structures • impressions of a place • specific details • structural relationships of information.	• extensive instructions relating to specific procedures • information based on logical relations • abstract verbal concepts • specified conditions.

4.2.3 Specific design recommendations for comprehensibility

4.2.3.1 Structuring information sets

Using information models (for example, information architecture) ensures that information is organised in an appropriate and comprehensible way. They represent information and the relationships between different types of information, which helps to communicate volumes of data and complex functionalities. Building information models relies on structuring information sets with the following basic rules:

- Make sure subordinate information can be understood by the way the superordinate information communicates it

  An example of not applying this rule: If the title does not explain the subordinate content appropriately, the user does not know what kind of content is included in this unit of information.
- Help the users orientate themselves by structuring the information to maintain a general overview of it

  An example of not applying this rule: A table with numerous sub-items in which the more detailed content is structured into very small units obstructs users from navigating through the information.
- Use the concepts that correspond with the user's mental model to avoid any confusion

  An example of applying this rule: Headings are generally placed above the text and are set apart using style and/or colour. Deviating from this mental model can cause the user to lose orientation.
- Ensure the user can get an initial overview by headings
- Make it easy for the user to filter information
- Group related information together.

4.2.3.2 Providing consistency

Consistency in terms of style, size, colour, and position across different applications reduces the cognitive load of the user during information reading.

Consistency improves the ability to recognise information and enables a more effective and efficient interaction. But it can potentially impede performance when processing tasks because presenting information consistently can prevent the optimal positioning of other essential items of information.

An example of a problem with consistency: The company logo is always placed on the top left-hand side of the screen, making it easy for the user to find it. However, this prevents other elements from being placed flexibly, for example, the positioning of menus.

Consistency can be differentiated into internal (within the application) and external (between applications) (also see 4.2.1).

Example of internal consistency: The colours used for design remain the same on every page of a website.

Example of external consistency: The colours used for design remain the same on both the desktop and the mobile versions of a website.

Every design decision that has been made for information design must be applied consistently to all content. The designer should use a content inventory for this purpose. It helps keeping track of all content to be designed and contributes to consistent design.

4.2.3.3 The Use of real-world metaphors to support comprehensibility

Metaphors are equivalents from the real world. The fact that metaphors relate to the real world can help the user understand the information contained on the user interface more effectively and efficiently, if they are utilised.

Example: The floppy disc symbol refers to a medium that was historically used to store data. The function of saving data can be communicated using this symbol.

The used metaphor must be understood by the user during interaction. Otherwise, it can lead to misunderstandings and input errors.

Example of metaphor changing over time: In order to understand the meaning of the floppy disc symbol, it is necessary to know that the floppy disc was formerly used to store data. This metaphor possibly will not be understood if this information is not known.

Useful metaphors for design combine consistency/familiarity and inconsistency/innovation. If they can be associated with the real world too easily, they hinder comprehensibility and the way the user deals with various elements of information.

Example of negative borrowing: An online pocket calculator that looks like a manual calculator helps the user understand the application. If the keypad in the digital application is designed the same way as a manual pocket calculator, it can hinder the use of this application because the user is used to a different kind of keypad in digital applications.

4.2.3.4 Using images that relate to the specific situation of use

Images can help the user understand complex data more easily. Depending on the ability to visualise task-related information, images can support the users in achieving their goals.

When selecting a suitable image form, the user requirements must always be considered. It is therefore necessary to identify the specific requirements in a particular situation and create an appropriate image.

Example: Displaying the location using a marker pin on a digital map for the user task “specify your location” is more appropriate than describing the location using text.

Copyright 2025, UXQB e. V.

4.3 Design activity: Sensory design

Before users can recognise the meaning of information, they must be able to perceive what is presented. If users miss perceiving information due to inattention or due to not being able to use the given sensory channels, the information needed for completing the task will be missing. **Gestalt laws**, the application of **white space,** and an appropriate layout help the designer to ensure perceptibility in visual design.

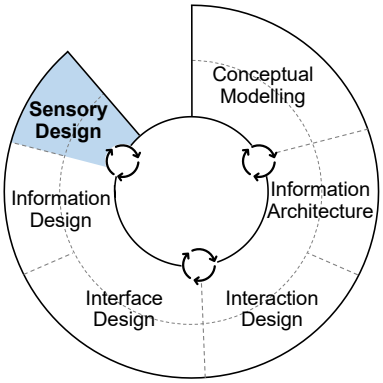


Figure 36 Design activity: Sensory design

Learning Objectives	
4.3.a	Understand how gestalt laws, colours, font sizes, and white space support perceptibility across all sensory channels.

4.3.1 Design the user interface regarding its perception through relevant sensory channels

In sensory design, the designer considers various sensory channels and ensures the interactive system is perceptible.

- The sensory design goes beyond a purely visual design because perceptibility can be influenced both positively and negatively through all sensory channels (visual, auditory, haptic, etc.).
- In sensory design, the design elements of different sensory modalities must be clearly perceptible and distinguishable from each other.
- Some aids support visual perception, such as font size, layout, distance, and colours.

4.3.2 Gestalt laws

Gestalt laws

Rules that describe how people perceive individual shapes – lines, surfaces, edges, colours – as a single unit, due to shared characteristics such as proximity and similarity.

Gestalt laws help while considering the perceptual aspects of sensory design:

Users may be assisted in their perception and interpretation of information when task objects and executable functions are configured together, similar in form, or given identical names. This enables connections of meaning to be established. Conversely, connections of meaning can be avoided by giving differing forms to objects with differing functions. A selection of gestalt laws, which should be applied when designing user interfaces is described below.

Law of proximity

Objects that are close to each other are more likely to be perceived as belonging together than objects that are distant from one another.

Example:

On a route planning overview, the user can perform two tasks: Entering the departure and destination of the trip as well as the date and the time of the trip. The labels “From”

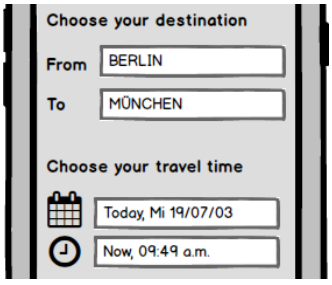


Figure 37 Example: law of proximity

and “To” and the symbols for the calendar and the clock are arranged close to the corresponding input fields, making it easier for the user to recognise which field requires what data.

Law of similarity

Elements that are similar to each other (for example, in shape, size, or colour) are perceived as belonging together.

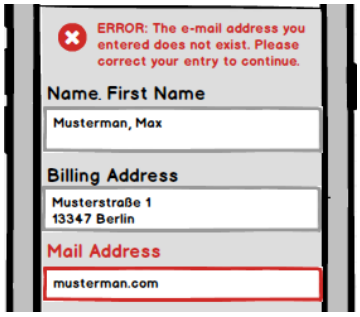


Figure 38 Example: law of similarity

Example:

During the input on a login page, a use error occurs in one of the fields. The affected field and the textual explanation of the error are marked in the same colour, making it easier for the user to understand the relationship of the field and text, allowing them to address the error more easily.

Law of good form

This is also known as the law of Prägnanz, which roughly translates to “the law of orderliness and simplicity”. The brain divides objects into structures that are as simple as possible. Even though complicated shapes may have more than one possible interpretation, our vision breaks them up into the simplest shapes, as this interpretation requires the least cognitive effort. It prevents us from becoming overwhelmed with information.

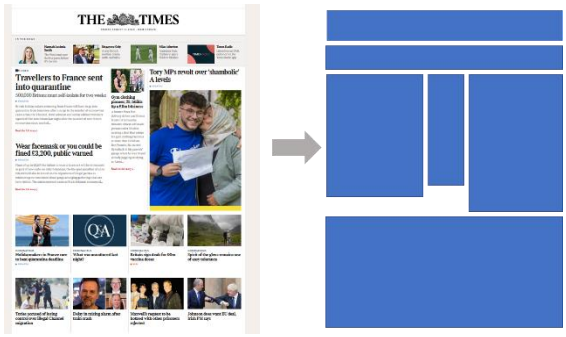


Figure 39 Example: law of good form

Example:

While designing a wireframe, elements can be arranged in blocks, rather than detailed pictures of the actual content. This makes it easier to organise the layout of the wireframe. Another example is the logo of the olympic games. Instead of seeing the logo as a complex cloud-like shape, it is perceived as individual rings overlapping.

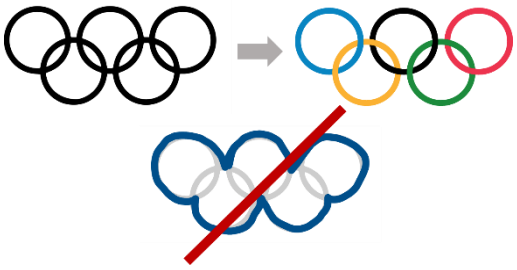


Figure 40 Example: law of good form

Law of continuation

The human eye will prefer to see continuous, unbroken paths and curves, trying to follow it even if another object might try to interrupt it.

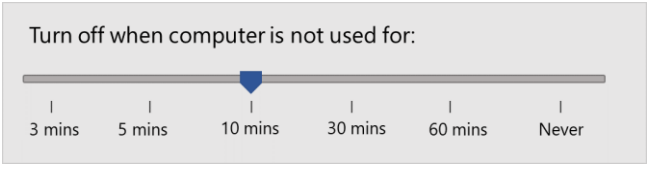


Figure 41 Example: law of continuation

Example:

Within system options, there is an option to determine when the computer should turn itself off after being inactive. This option is realised through a slider. Even though the slider is theoretically divided by the handle, the brain perceives the slider as a single line instead of two lines separated by the handle.

Law of closure

Incomplete, simple elements are perceived as forming familiar patterns, shapes, and groups. This law is often employed on UI elements such as loading indicators, progress bars, or sliders. Many icons and logos use this law as well.

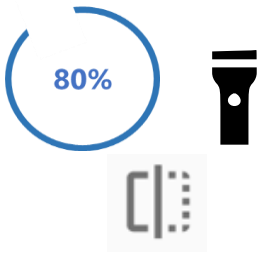


Figure 42 Example: law of closure

Example:

Closure is often employed when designing loading indicators, progress bars, or icons. Even though the shape representing the progress is incomplete, the user perceives it as a circle. The same applies to many commonly used icons. Despite the use of whitespace, the icon on the right is recognised as a flashlight.

Law of common fate

Figures with similar movement characteristics (for example, moving in the same direction, along the same path) are perceived as one unit because they share a common fate.

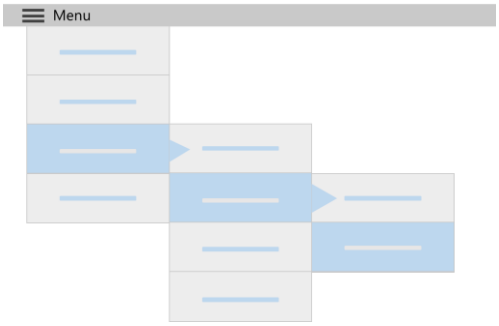


Figure 43 Example: law of common fate

Example:

All items of a slide-out menu's submenu open in the same direction. Because they share this movement, the elements appear to be related. This is the case for each level of the submenu, leading to each submenu's items being perceived as a group.

4.3.3 Colours, font sizes, and white space

Designers must follow basic rules for the use of colours and the consideration of colour combinations to ensure perceptibility.

The basic rules for the use of colours are as follows:

- The foreground colour should stand out sufficiently from the background. W3C Web Content Accessibility Guidelines mandate a contrast ratio of at least 4.5:1 for text against its background.
- The use of good contrast ensures legibility. Especially when displays are used outdoors, environmental light can impair contrast.
- Be aware that the use of certain RGB colours in conjunction can cause a flickering effect as their wavelengths are close together.

Additionally: The following recommendations for usage of colours are stated in [ISO9241-171]:

- Information should not be conveyed by colour alone.
- Colour schemes for users with visual impairments should be considered.
- Depending on the context of use, it should be checked whether the customising of colour coding and colour schemes is necessary.
- Sufficient contrast between the foreground and the background should be provided, preferably within basic settings.

Copyright 2025, UXQB e. V.

Example: In dark mode, bright pixels are displayed as dark; dark pixels as bright. The mode serves to protect the user's eyes when in a dark environment (for example, at night or in dark rooms).

Figure 44 An example of the light vs. dark mode

For increased legibility, font sizes should be customised to the user's requirements, or should be customisable by the user.

The following recommendations are stated in ISO 9241-171 "Ergonomics of human-system interaction – Part 171: Guidance on software accessibility" [ISO9241-171]:

- Information should not be conveyed solely through visual font attributes.
- Users should be able to set a minimum font size.
- If the font size changes, the scale, and layout of user interface elements should be adjusted proportionally.

Example:
When designing a train connection overview, elements that are difficult to distinguish from each other are used, and therefore, they are difficult to be perceived of visually (low-contrast colours, small font size). Instead, the designer should have taken care to use easily distinguishable elements (easily distinguishable colours, different font sizes) when designing the train connection overview.

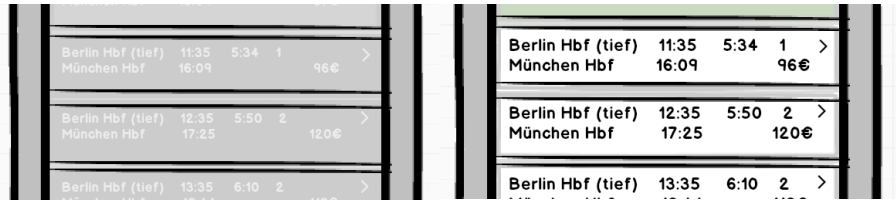


Figure 45 Elements that are difficult to distinguish

Another aspect that needs to be considered in sensory design, especially in visual perception design, is the inclusion of white space.

White space

The free and undescribed areas on a display (for example, margins, gutters, and space between columns and graphics). Designers can use white space to separate unrelated user interface elements and add aesthetic and effective value to the design.

The appropriate use of white space supports the recognition of the grouping and structure of user interface elements and thus the faster comprehension and readability - the user needs less concentration. The appropriate use of white space is also helpful from an aesthetic point of view, especially in contrast to separator lines or borders.

Example:

The homepage of a well-known search engine is completely white, with only the logo and the search bar in the middle. The use of the large empty area allows the elements in the middle to stand out. This clarifies the purpose of the page: to search for something.

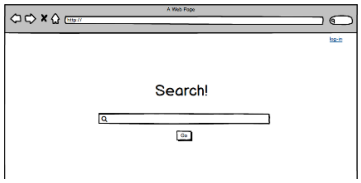


Figure 46 Example of a search engine site

5 Specific human needs

When designing the behaviour of an interactive system, accessibility and ethical aspects must be considered, along with the fact that human needs vary depending on culture.

5.1 Accessibility

Interactive systems must be designed in terms of **accessibility** to support temporary impairments or long-term **disabilities** and accommodate the effective application of **assistive technologies**. Designers should know the importance of accessible design and relevant laws, standards, and guidelines.

Learning Objectives	
5.1.a	Know which human disabilities make accessible design necessary and how often they occur.
5.1.b	Know examples of assistive technologies and how they support users.
5.1.c	Know which important standards and guidelines for accessible design can be considered.

Accessibility is a component of human-centred quality that must be considered during design to make user interfaces accessible to people with disabilities.

Accessibility

CPUX-F Definition
The extent to which an interactive system enables users to interact with it effectively, efficiently, and with satisfaction, regardless of their level of vision, hearing, dexterity, cognition, physical mobility, etc.

About 15% of the world's population have some form of disability, impacting their ability to carry out normal day-to-day activities. Some people have significant, long-term disabilities, whilst others are only temporarily impaired.

Disability

A physical or mental impairment which has a substantial and long-term adverse effect on that person's ability to carry out normal day-to-day activities.

Barriers in physical and digital environments can prevent people with disabilities from participating equally in society, for example, through a lack of access to digital services.

5.1.1 The importance of accessible design

More than one billion people live with some form of disability. Many have more than one impairment. About 20% of people with disabilities were born with their impairment, others have had to adapt to their impairments later in life. "Disability" covers a wide range of impairments – including motor, sight, hearing, and cognition – which manifest themselves in many ways.

> For example, users might struggle to recognise icons, understand words or idioms, use a keyboard or mouse, concentrate for long periods of time, differentiate between colours, or hear, see, or move at all.

Impairments are not necessarily permanent. They can affect us temporarily.

For example, breaking a bone, being deprived of sleep, suffering from a heavy cold – all impair our ability to perform tasks effectively, efficiently, and with satisfaction.

Impairments are not static. Our physical, sensory, cognitive and emotional abilities change based on our situation, and our environment can introduce or amplify existing impairments in a way that makes it more difficult to perform tasks with an interactive system.

For example, having attention deficit disorder or just the use of one hand and travelling by public transport, being in the cold when you have arthritis, or when at the beach on a sunny day and having low vision.

Good accessibility can never be achieved by working in isolation from others in the project team – design, content, and execution are equally important and depend on each other. Three important rules to adopt for websites or apps are,

- make sure all elements can be accessed by keyboard alone, and that focus is clearly visible
- plan the page structure and its headings to facilitate comprehension
- provide appropriate, descriptive alt text for images and captions or transcripts for videos.

For further design recommendations for implementing accessibility, see below.

5.1.2 Assistive technologies

Assistive technology

Software or hardware that can be added to an interactive system to support users with specific impairments in performing tasks.

Examples of assistive technologies: Screen reader, screen magnifier, speech input, customised keyboards.

- A user with a visual impairment might use screen magnification software to zoom in to specific areas on a page.
- A user with a motor impairment might use a switch to navigate and interact with a webpage.
- A blind user may use screen reader software to understand, navigate, and interact with an app.
- A user with a cognitive impairment who struggles with written text may use screen reader software to read the page out to them.

Assistive technologies rely on the interactive system's code to translate its nature to users.

For example,

- to tell users that there is a button or hyperlink that they can activate
- that there is a picture in the content, and to convey its description
- what a form field should contain, by referencing its label
- to convey the page structure and its headings.

5.1.3 Laws, standards and guidelines for accessible design

Laws exist to protect the rights of people with disabilities and guarantee that they have the same opportunities as people without disabilities:

- For the EU: European Accessibility Act
- For the UK: Equality Act 2010
- For the USA: Americans with Disability Act and, for government agencies, Section 508.

Rather than set their own guidelines and standards, many laws require products and services within their territories to also adhere to international guidelines. These guidelines are

often referenced in courts of law to determine whether or not an organisation is making sufficient adjustments to comply with accessibility law. Guidelines include:

- W3C's "Web Content Accessibility Guidelines (WCAG) 2.1
- ISO 9241-171: guidance on software accessibility

The UK's Government Digital Service team (GDS) has provided some very useful posters with "Dos and don'ts on designing for accessibility" https://accessibility.blog.gov.uk/2016/09/02/dos-and-donts-on-designing-for-accessibility/.

Be aware that designing according to accessibility guidelines or standards does *not* guarantee that you will produce an accessible interactive system. Always test with users who have disabilities to evaluate usability.

Copyright 2025, UXQB e. V.

5.2 Design ethics

In designing interactive systems designers influence the behaviour of users, therefore they have an ethical responsibility regarding their design decisions. Intentional influence on users' behaviour through **persuasive design** can be used to achieve behavioural adherence to socially desirable outcomes or to trick users into harmful behaviours to achieve business goals. Thus, user interfaces can be described in terms of the degree of honesty towards the user (**honest interface**) and in terms of the intentional manipulation into actions that run counter to the interests of users (**dark pattern**).

Designers influence users by activating **behaviour patterns** using different forms of **nudges** such as **interface metaphors** or by considering the user's **domain knowledge**. Furthermore, **illusions of control** or **default** settings are used to successfully persuade the user through design. Responsible designers are aware of their influence on user behaviour and therefore consider established ethical standards, rules, and guidelines when designing interactive systems.

Learning Objectives	
5.2.a	Understand that by designing an interactive system, designers influence human behaviour and therefore have an ethical responsibility.
5.2.b	Know the opportunities and risks of deliberately influencing human behaviour through design.
5.2.c	Understand the different forms of deliberate influence on user behaviour.
5.2.d	Know generally accepted ethical standards of professional associations that must be considered when designing an interactive system.

5.2.1 Influence by design and ethical consequences

Designers influence human behaviour (consciously and unconsciously) by the design choices they make when designing interactive systems and they must therefore act ethically and responsibly. In particular, when employing persuasive design, designers need to be aware of their ethical responsibility, as it can cause a change in users' attitudes.

Persuasive design

A way of designing an interactive system based on psychological and sociological theories about how to influence the user's attitudes or behaviours.

For designers, it can be difficult to maintain ethical responsibility in the context of business goals, but they must still be aware of this responsibility. Arguments concerning business goals should be made transparent and visible against ethics.

While making design decisions for interactive systems, designers develop user interfaces by moving back and forth on an ethical continuum from honest interface to dark pattern during a design project [Brig2011]. Moving on the continuum means deciding the extent to which there are opportunities and risks for the user when using the interactive system. Figure 47 shows this relation and Table 15 shows typical chances and risks.

Copyright 2025, UXQB e. V.

Honest interface

A user interface that considers the interests of users even over the (mostly economic) goals of the provider.

Example: The default settings of an application are set to the option that's safest for the user (even if that contradicts business goals).

Dark pattern

A method of persuasive design to manipulate users into performing actions that run counter to the interests of users and primarily serve the interests of system designers or providers.

The design of dark patterns often does not violate applicable laws. Dark patterns are ethically questionable due to the deception of users and the resulting restrictions on the user's decision making [Brig2013].

Example: Closing the window indicating the update of a new software version means agreeing to perform the update.

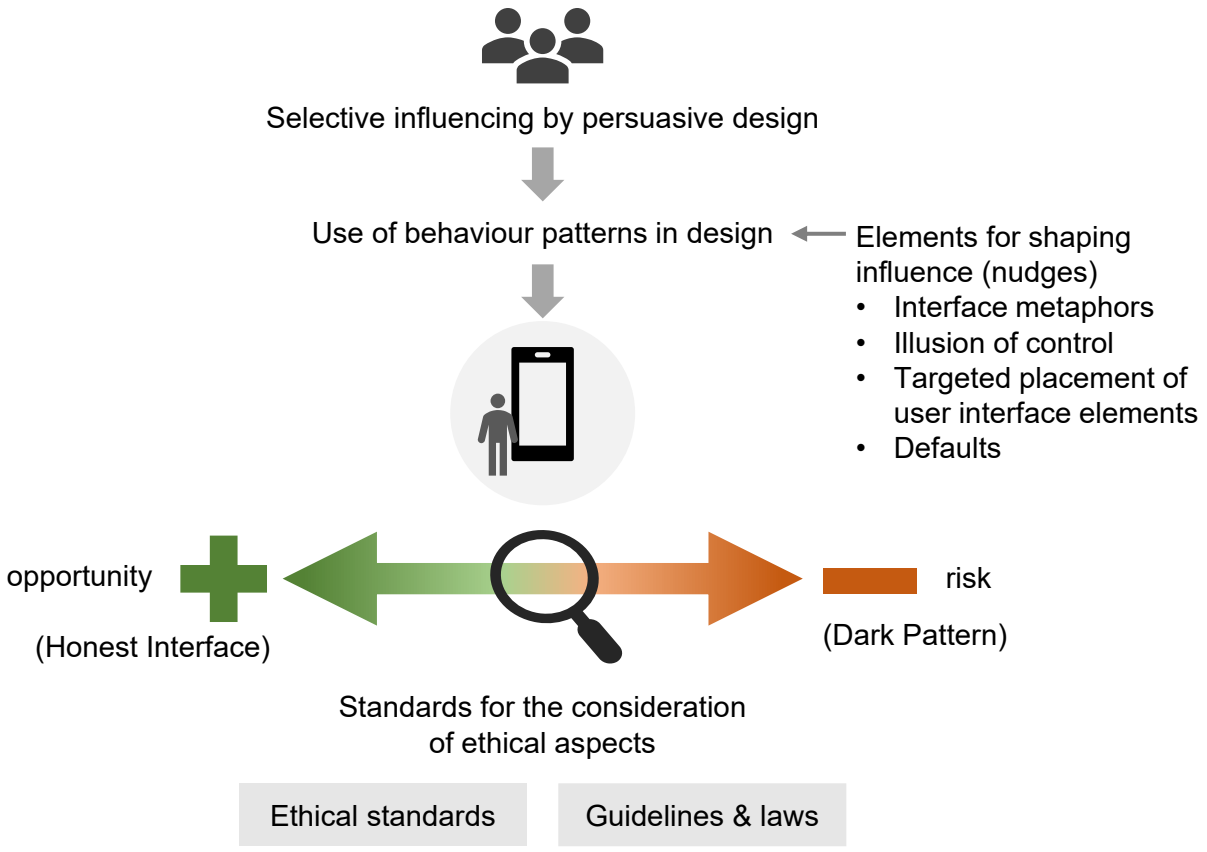


Figure 47 Aspects of selective influence in the design of interactive systems

User interfaces that lie somewhere between honest interface and dark pattern are often a result of the decision to use manipulative techniques in design, to ensure the system primarily supports the needs of the business.

Example: A streaming provider allows a one-month trial period, which is automatically converted into a paid subscription at the end of the trial period. This fact is clearly and comprehensibly pointed out on the website. Without employing this approach, the provider cannot afford to offer a trial period (which they hope will persuade the user to take up a paid subscription when the trial ends).

Copyright 2025, UXQB e. V.

User interfaces must be checked for unintentional influence and be designed with regard to intentional influence.

Table 15 Opportunities and risks of influencing human behaviour by design

Opportunities of influencing human behaviour by design	Risks of influencing human behaviour by design
• Encourage users to achieve their goals more efficiently with the interactive system • Users achieve goals they didn't know they could achieve • Trigger positive behavioural changes such as “Eco Design” (ecological design / sustainable design)	• Forcing goals on users that are not important to them • Conflict with ethical standards, guidelines, and laws • Hinder or completely suppress independent decision making • Deceiving users

Example: Eco Design promotes sustainable behaviour through persuasive design, for example, an email service provider providing a link to an “energy-saving login” that leads to a page that doesn't contain any items other than the login fields [JoMa2017].

5.2.2 Use of nudges as design elements of influence

Designers can influence the user’s behaviour by purposefully activating behaviour patterns using design elements of influence [Spre2018].

Behaviour pattern

A standardised recurring approach to completing tasks which guides a multitude of human decisions and stands in contrast to the concepts of rational decisions.

Example: A mobile payment app sets the option “tip” as default. Therefore, more users are tipping as they are always asked to tip when paying a bill and explicitly must set “no tip” if they do not want to tip.

The way in which the behaviour pattern is used in the example is also called a nudge [ThSu2008].

Nudge

Designing the user interface for a situation in which decisions have to be made with the aim of changing human behaviour in a predictable way, without prohibiting decision options or seriously changing their economic benefit.

5.2.3 Typical forms of nudges

Interface metaphors illustrate interaction by referencing everyday experiences which improve the user’s interaction with a technical system in terms of effectiveness and efficiency.

Interface metaphor

The attempt to match the user's prior knowledge with the functionality of the interactive system by activating specific domain knowledge directly through the user interface elements used and the type of interaction.

Copyright 2025, UXQB e. V.

Example:
The idea of the computer default screen being like a desktop and placing and arranging things (icons) on this desktop space is an interface metaphor.
Many desktop operating systems illustrate deleting a file by moving it to a recycle bin (recycle bin metaphor). This recycle bin in the interactive system must be emptied at regular intervals, as must the contents of a recycling bin in the real world. The recycle bin metaphor is an abstraction of the function “delete file”. There is a divergence between the intended function “temporarily remove a file by moving” and the meaning of the symbol “destroy”.

Existing domain knowledge can be used in the interaction with the system. Interface metaphors only work if they are understood and perceived by the user. Divergences lead to users being restricted in their use of the interactive system.

Domain knowledge

System independent knowledge about tasks, processes, and people involved within a certain application field.

The illusion of control over the interactive system results in users taking greater risks; for example, in relation to the processing of their personal data [BrAA2013].

Illusion of control

A feeling of control among users, even though they have little or no objective influence on the behaviour of the interactive system.

Example: In individual social networks, specific wording is used to lead users to believe they have a disproportionate sense of control over the data processed by the company. The request to subscribe to a supposedly free newsletter is: “Subscribe to our free newsletter”. The fact that the user exchanges personal data for this purpose is deliberately concealed.

The user must be informed about the consequences of their actions. The options offered on the user interface must have a technical implementation. If this is not the case, the illusion of control is used against the interest of the user and represents a dark pattern.

The targeted placement of user interface elements influences the perception of information. The examination of eye movements in the perception of advertisements allows conclusions to be drawn about the orientation of human attention.

The “hiding” of certain information and options is used to give less attention to information (dark pattern). In an honest interface, important information is presented in user interfaces in places that are more noticeable to the user.

Example: Recent news or other important notifications are placed in the top left corner of a page to get immediate attention from a user usually scanning the website in the “F” reading scheme.

Defaults go hand-in-hand with a great deal of responsibility. Defaults are rarely considered by the user and hence are not often changed.

Default

A setting pre-selected by the system unless the user explicitly changes it.

Defaults are defined by rules and laws in certain areas.

Examples:
When installing a computer programme, the additional installation of a browser extension is pre-selected.
The agreement to using data for promotional activities is not pre-selected and requires the explicit agreement of the user.

5.2.4 Responsible design by applying common ethical standards

When designing user interfaces, designers consider different interests and ethical aspects of design decisions. Responsible action, ethical standards, guidelines, and laws must be observed:

1. The General Data Protection Regulation (GDPR) is a legal regulation in the EU that (directly) determines the use of techniques for intentional influencing. It requires the development of systems according to the principles “Data Protection by Design” and “Data Protection by Default”. Data Protection by Default requires a high level of data protection for users, even if they do not explicitly object to the collection and use of personal data.

   Example: This requirement implies that the default settings of checkboxes by the designer must represent the options that offer the strongest data protection.
2. The Meta-Code of Ethics lays down transnational ethical principles which are specified by the ethics guidelines of the associations of the member countries, for example, the American Psychological Association and the Association for Computing Machinery.

   Example: The European Psychology Association [EFPA2005] provides professional ethical guidelines for psychological activities, which are also used for design solutions.
   - We need to be vigilant against personal, social, institutional, financial, and political influences which could lead to misuse or incorrect application of psychological insights.
   - The quality and results of services also depend on the extent to which people can use these services self-sufficiently and autonomously.
   - The impacts of our actions on third parties must be taken into account.

Usability tests help to verify and comply with ethical guidelines by providing evidence of ethically problematic design decisions and reducing their unintended use.

Copyright 2025, UXQB e. V.

5.3 Cultural diversity

Intercultural user interface design takes intercultural contexts of use into account through specific methods and concepts. For example, the strategies of **internationalisation** and **localisation** are applied. **Intercultural usability testing** helps to evaluate design decisions from an intercultural context of use.

Learning Objectives	
5.3.a	Understand methods for interculturalisation.
5.3.b	Know the concepts of internationalisation and localisation.

5.3.1 Intercultural user interface design (IUID)

Designing intercultural user interfaces for an interactive system considers all design decisions which are under cultural influences.

Intercultural user interface design

A process of adequately designing an interactive system considering the cultural aspects of the context of use.

The precondition for intercultural user interface design is knowledge about the cultural differences in human interaction with interactive systems and its considerations in designing solutions throughout the whole design process.

Interculturalising extends the adaption of the user interface beyond the translation into another language by considering all other user interface characteristics in the cultural context, such as adapting navigation structure, interaction frequency, or information density. Furthermore, it is fundamentally important to start with the intercultural design of user interfaces.

Therefore, it is necessary to create awareness for this topic at the beginning of the design activities. At the start of a project, the designer must establish if there is an intercultural context of use. If so, they must ensure that those with appropriate expertise in intercultural user interface design are involved in the project.

5.3.2 Methods for interculturalisation (IUID process)

The process of intercultural user interface design is based on human centred design, but considers specific aspects of intercultural communication, competence, and management, as well as intercultural software engineering, including internationalisation and localisation, and intercultural usability engineering [Heim2019].

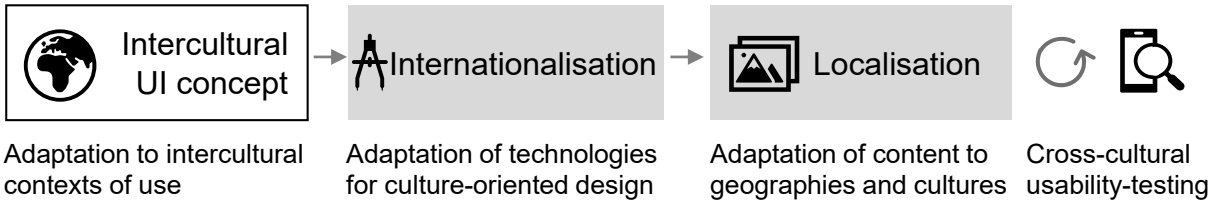


Figure 48 Principle procedure for the design of intercultural user interfaces

5.3.2.1 Intercultural user interface design according to cultural contexts of use

The following concepts are considered in intercultural user interface design:

- Cultural differences expressed by cultural dimensions (such as individualism vs. collectivism)
- Intercultural variables that represent different preferences of users from different cultures. These can be invisible (for example, way of thinking, decision premises, ideology, etc.) or visible (for example, layout, colours, font).
- User interface characteristics (for example, reading direction, symbols, layout, navigation structure, information density).

The procedure to adapt the user interface to cultural contexts of use involves,

1. selecting the subject of the design, the most important applications (for example, interculturally relevant usage situations), and the desired target cultures (for example, China, Western Europe, the Arab world, etc.)
2. determining the respective user interface elements (such as layout, buttons, text fields) and assigning them to the characteristics of the intercultural variables (for example, invisible, visible) as well as to the main property categories of user interfaces, such as presentation level, interaction level, navigation level, symbol level (for example, metaphors, icons), and thinking level (such as mental model)
3. determining the characteristics of space- and time-related variables, such as information density or interaction frequency
4. relating the intercultural variables to the properties of the user interfaces in order to derive cultural interaction indicators and design recommendations for the design of intercultural user interfaces.

5.3.2.2 Concept of internationalisation (I18N)

Internationalisation

The process of building up a platform for software application so that an interactive system can be adapted to various languages and regions without any change in the technical platform. Internationalisation is often abbreviated as I18N (18 letters between I and N).

The interactive system is designed and developed to ensure problem-free localisation; for example, by automatic adaptation or easy parameterisation of culturally dependent variables.

Localisation at a later date is possible without further alterations to the system architecture or other aspects of the technical platform – no further changes to the platform or source code are required during localisation.

An example of the internationalisation of an interactive system: A German company designs the entry fields of a website so that French words with longer word length can also be entered.

5.3.2.3 Concept of localisation (L10N)

Localisation

The process of adapting the internationalised interactive system for a specific region or language by translating text and adding or replacing locale-specific components by parameterisation. Localisation is often abbreviated as L10N (10 letters between L and N).

Adaptation is made with regards to geographical (local, regional, country-specific) and cultural (ethnic, linguistic, legal) requirements in such a way that the system is accepted by the user of the target market and can be used.

Copyright 2025, UXQB e. V.

Examples of the localisation of an interactive system:

- Reading direction can differ in different cultures, for example, the reading scheme “F” must be reversed (right to left) for the Arabic culture.
- Colours have different meanings in different cultures; for example, the meaning of “death” is associated with the colour “black” in Germany and with the colour “white” in China. Therefore, in the configuration of the interactive system, the colour is pre-set according to the target culture.

5.3.2.4 Intercultural evaluation and agreement

The agreement with stakeholders on intercultural user interface design projects and the evaluation with users and other stakeholders must take intercultural aspects into account.

Intercultural usability testing

A form of usability testing in which a product is tested in different cultures. Various teams with representatives from relevant user groups are involved.

Intercultural, therefore, refers to the inclusion of users from all relevant cultures during user participation (for example, in contextual interviews, usability tests). Usability tests should be performed in respective cultures or with different representatives from the user population in each culture (for example, from Japan, India, Europe). Evaluation can take place with users brought together in one place; it can take place in respective cultures or can be performed remotely.

Organising the required resources is complex business; therefore, planning must be carried out at an early stage. A shared commitment to an intercultural approach in the project team must be made against the background of these resources.

An example of intercultural usability testing: For the evaluation of an interculturally designed website, Chinese, Western European, and sub-Saharan African participants will be invited. The increased planning effort for the recruitment of participants from different countries was considered at the beginning of the project and the necessary resources have been provided. Without these resources, the tests could only have been carried out remotely.

Copyright 2025, UXQB e. V.

6 Aspects beyond the design activities

All participants in design projects have an idea of what a good solution should look like. They must be heard and respected. It is crucial for the designer to communicate with users and stakeholders as early as possible, involve them, and bring forward joint decisions as an agent for the users' perspective. Furthermore, the designer must skilfully employ design recommendations and understand the significance of documenting design decisions.

6.1 Managing stakeholders

A designer needs to manage stakeholders in the discussion of work products, to reach acceptance for the designed solution. They need to work towards **consensual agreement** about the quality criteria, as an agent for the user's perspective, and thereby emphasise **human-centred quality objectives**.

Designers need to adopt a mindset of continuous improvement of **human-centred quality**, which has several quality dimensions including usability, user experience, **avoidance of harm from use,** and accessibility.

Designers continuously improve human-centred quality through regular formative evaluation, by including user feedback into discussions with stakeholders, and by employing **participatory design**. Tools such as **customer journey maps** help visualise the work for others involved in the project.

Learning Objectives	
6.1.a	Understand how to achieve consensual agreement to balance the human-centred quality and the quality objectives of other stakeholders.
6.1.b	Know the relevance of human-centred quality objectives in achieving human-centred quality in the design process.
6.1.c	Know that stakeholders should be involved in a UX design project and that interdisciplinary work with stakeholders and team members contributes to a successful design solution.
6.1.d	Know user journey map as a method to visualise the intended interaction with the system.
6.1.e	Know how to distinguish between a customer journey map and a user journey map.

6.1.1 Setting quality objectives for the project

The main goals of a design project are the successful support of user goals and the satisfaction of additional stakeholders. Stakeholders in design projects may have different expectations about the solution and also different quality objectives. Their expectations should be collected and visibly documented in a kick-off workshop.

The possible quality objectives of stakeholder groups in a design project are as follows:

- Business: “In the context of a specific customer, the interactive system can be ready to be used within 5 workdays.”
- Users: “The PC operating system, installed in a standard configuration, needs at maximum 90 seconds from pressing the power button until the user interface provides the option to start an application.”

Copyright 2025, UXQB e. V.

- Technology: “It must be possible to maintain the interactive system remotely, for example, via a data connection over the internet.”

It is easy to mistake the project briefing for “all there is to know” about the design work at hand. Often important information is missing and needs to be worked out in consensual agreement with stakeholders.

Consensual agreement

A process for reaching a consensus decision with a party; for example, the stakeholders. Making a decision does not require the explicit approval of each individual, but it does depend on the there being no strong disapproval. It does not need any formal or symbolic acts to cement the obligation.

All the involved parties must agree, or be prepared, to give up or at least postpone any dissenting opinion or objections to the decision taken. All participants then support the decision regardless of any reservations they may have.

This can include any element of a design project. It is especially important for the designer to set up and lead discussions with stakeholders about quality criteria for the project and the relevance of user feedback.

It is the responsibility of the designer in the course of the project to negotiate a fair and consensual agreement about the quality objectives for the project with stakeholders, and for everyone to aim for these objectives in unison. When agreeing technical and business objectives for a successful design project, the designer, as the agent for the user, has to emphasise human-centred quality.

6.1.1.1 Human-centred quality objectives

From the point of view of a UX designer, as the agent for the user, the main goal of a design project is the successful support of user goals. The designer needs to adopt the mindset of continuous concept evaluation with users throughout the design process in order to learn from feedback and iteratively improve the human-centred quality of the designed solution.

Human-centred quality

The degree to which stakeholder requirements are met in an interactive system. They refer to the quality dimensions usability, accessibility, user experience, and avoidance of harm from use.

Human-centred quality is, in addition to technical quality, an important aspect of quality. It describes those dimensions of quality that users and other stakeholders actively perceive through interaction with the interactive system, while technology-centred quality provides the prerequisites for it. While human-centred quality is the result of implementing stakeholder requirements, technology-centred quality is the result of implementing system requirements.

Whether human-centred quality can be met in a design project, depends on the readiness for consensual agreement among the stakeholders and the usability maturity of the organisation. The more mature an organisation is and the better stakeholders work towards an agreement, the more likely it is that through iterative improvement the project will achieve the quality objectives that have been set.

A high usability maturity level refers to

- the open-mindedness of the management and stakeholders towards human-centred design
- a genuine wish to develop human-centred quality

- a commitment to actively participate in and support human-centred design activities
- the development of human-centred quality standards for projects
- ensuring the availability of necessary competencies and resources for integrating human-centred design into development activities.

To be an effective agent for users, the designer must know the quality objectives of all stakeholders and be present when work products are discussed. They are the stakeholder for the human-centred quality objectives.

Human-centred quality objectives

CPUX-F Definition
The goals that are to be achieved for the user of an interactive system when developing the interactive system. Human-centred quality objectives relate to one or more of the following components of human-centred quality: usability, accessibility, user experience, and avoidance of harm from use.

They are objectives of the stakeholders regarding the human-centred quality of the interactive system to be developed or improved. They are specified in the planning of human-centred design. They are not yet user requirements. Solutions are designed to fulfil them.

An example of a human-centred quality objective: The efficient operation of an application. The designer shouldn't integrate complex animations, which waste users' time.

They can be formulated as verifiable quantitative user requirements if they are to serve as acceptance criteria in the project.

An example of a quantitative user requirement: The user must be able to complete the task within 30 seconds.

A human-centred quality objective can also be formulated by any stakeholder.

An example: A management objective could be that users must be able to operate the intended system without explicit training.

Besides usability and user experience, avoidance of harm from use and accessibility are two further important dimensions of human-centred quality for which human-centred quality objectives might be relevant for a design project.

Demands related to safety or data privacy rules indicate the relevance of avoidance of harm from use.

Avoidance of harm from use

The extent to which the design of the interactive system minimises potential risks to acceptable levels in terms of stakeholder safety and health, economic considerations, or the environment.

An example: During use of an interactive system,

- avoid loss of unrecoverable data
- avoid unwanted access to personal data
- avoid financial risks
- avoid serious injuries.

6.1.2 Agreement on user feedback

For a designer in a design project, user feedback might be sufficient to question and iterate the solution. Stakeholders may have their own goals for the solution to be developed and

might object to changes based on user feedback. As an agent for the user, the designer needs to

- wholly understand the user perspective, as users are generally not involved in the discussion
- have strong arguments about how design decisions may affect users and their experience
- be able to explain the effects of evaluation results
- accept that a good compromise often is the best that can be done.

6.1.3 Involve stakeholders

The designer needs to iterate concepts with stakeholders and with the interdisciplinary team members, to include their feedback in order to meet their expectations. Participatory design has positive effects on the acceptance of the final product.

Participatory design

The engagement of users, members of the interdisciplinary team, and stakeholders in the conception and evaluation of the system to be designed.

Participation should start in the early phases of the design process, for example, through joint work on the design of solutions, informative evaluation, or continuous collection of user feedback.

Iteration should start with the presentation of use scenarios which can be complemented by the presentation of journey maps. Furthermore, any tangible version of the solution under development can be evaluated with stakeholders, just as it is done with users.

Iterating tangible work products with stakeholders:

- Make sure that during discussions with stakeholders, you relay the story behind your work product, for example, talk about a specific task or question the user has.
- Use scenarios, for example, in the form of a storyboard, can help stakeholders to put themselves in the situation of the user.
- The information architecture can be evaluated by visualisation, for example, with the cores and paths method.
- Design decisions can be experienced in initial prototypes.

Like use scenarios, user journey maps are another method to visualise the current completion of users' core tasks as well as the designer's optimised, intended vision of how core tasks will be completed, in a compact, clear, and structured way. User journey maps are used to communicate the current or intended interaction of a user persona with the interactive system [GePo2018].

Typical characteristics of user journey maps are that

- they focus on the core tasks of the user.
- they visualise the journey of a single user (persona) solving a task.
- they put phases, individual steps, and sequences in a chronological structure.
- they provide insight into the user's motivations and attitudes, for example, by mapping the detailed goals, thoughts, feelings, and pain points of a persona as well as opportunities for improvement.
- they leave out "internal" processes and actors that users don't interact with.

Maps must point out whether they show the as-is state or the intended state. The descriptions of the intended state make the vision of the future tangible for all stakeholders.

An example: User journey map (intended)

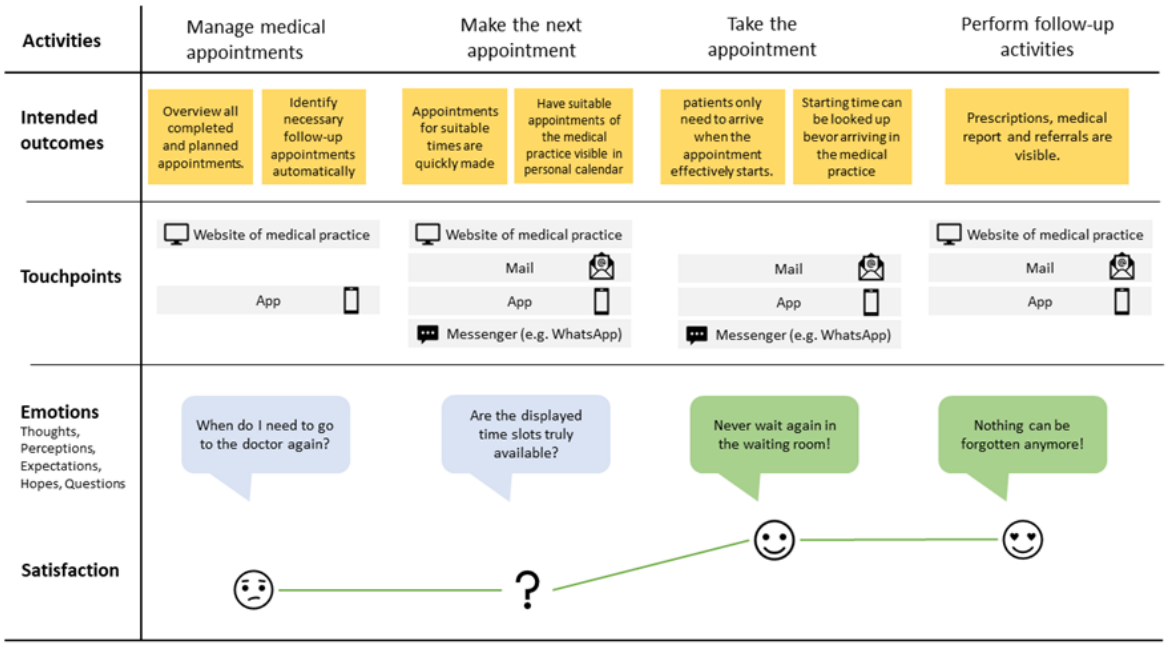


Figure 49 Example of a user journey map (intended use)

In contrast to user journey maps, customer journey maps are used to communicate the current or intended interaction of a person in the role of a customer rather than a user with the interactive system [GEPO2016].

Customer journey map

A graphical or tabular description of all encounters customers have with a product, service brand, or company, covering all touchpoints that influence the customer experience, making the over-all customer experience tangible for others.

Customer journey maps graphically or tabularly display all touchpoints on the “path” of a specific customer. They include all phases of the experience of discovering, considering, purchasing, and using an interactive system so that the entire customer experience can be understood by others. In contrast to user journey maps, they are based on the complete experience cycle of a customer instead of focusing on the “use phase” where the user’s core tasks are in focus.

Typical characteristics of customer journey maps are that

- they focus on all the touchpoints of a customer with the interactive system and its supplier
- going beyond the actual task completion, they visualise the journey of a single customer (persona) throughout the whole experience cycle of the customer relationship, from the purchase and procurement to the decommissioning and disposal.
- in addition to task-related goals, they focus on customer goals, like an appropriate price/performance ratio.

6.2 Setting the frame for design work

The designer has to set the boundaries for the design work by deciding on one or more **systems of design recommendations** to be applied, for example, deciding for **interaction principles, heuristics**, **design patterns**, or **user interface guidelines** in **style guides** to ensure a good human-centred quality.

Learning Objectives	
6.2.a	Be able to apply common systems of design recommendations.
6.2.b	Understand what design patterns are and where to get information about them.
6.2.c	Understand the meaning of a style guide for designing solutions and quality criteria for style guides.

6.2.1 Deciding on the design process and methods

Today a variety of design processes and design methods are in use. Each of these aims at different aspects of the development of a solution and may have an impact on the results. In the beginning of a project, the designer needs to clarify the process and methods to be used with stakeholders, in order to avoid misunderstandings and avoid risking the success of the design project.

Some popular design approaches and methods include

- Human-centred design: focuses on five activities to ensure human-centred products
- Lean UX: focuses on fast learning in agile teams by continuous UX validation of design decisions with users
- Design thinking: focuses on the creation of innovative and convincing ideas for known problems, which are not necessarily digital
- Ideation: focuses on the creation of a huge number of variants of a possible solution, involving users and stakeholders
- Design sprint: focuses on detailing a business model for a new digital product, which sometimes gets prototyped in the process.

It is important to pick an approach and method that are tailored to the goals of the design project while emphasising the user requirements and user involvement at the same time. The HCD approach comprises classical methods and is open to the introduction of other creative methods where needed.

6.2.2 Deciding on appropriate systems of design recommendations to be used

A design project might require the application of general or specialised systems of design recommendations.

System of design recommendations

A selection of principles, heuristics, rules, design patterns, and/or guidelines (often compiled in style guides) that serve as guidance and as a basis for designing the solution for an interactive system.

It is the responsibility of the designer to decide which recommendations are to be used and to apply them in a meaningful and methodically correct manner.

Systems of design recommendations range from rather general to very specific recommendations: interaction principles, heuristics, design patterns, and style guides.

Copyright 2025, UXQB e. V.

When deciding on systems of design recommendations,

- clarify whether mandatory standards exist (such as the Web Content Accessibility Guidelines).
- choose systems that suit the given design project (i.e. that provide a strong argumentative basis for or against design decisions).
- ensure the applicability, validity, and topicality of the chosen systems.
- study the chosen systems in detail.

When applying systems of design recommendations,

- use the chosen recommendations for design, for example, a requirements list can be hung up in the workspace in a visible position.
- use the chosen recommendations for evaluating design decisions: Document the results of the evaluation as a collection of findings from which proposals for changes can be derived.
- communicate with team members and stakeholders during design and evaluation so that individual recommendations and their implementation can be discussed.

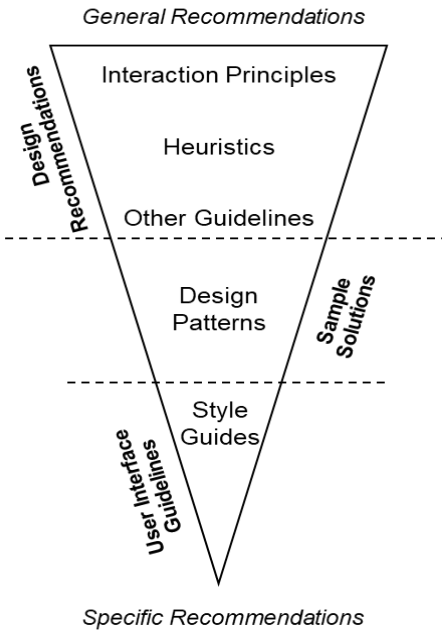


Figure 50 System of design recommendations from general to specific

Avoid these common mistakes:

- Applying unverified recommendations
- Working superficially with recommendations (based only on the “heading levels”)
- Skipping discussions within the team and with stakeholders
- Forgetting to document design decisions or suggested changes, adequately and visibly.

6.2.2.1 General design recommendations

There is a variety of general design recommendations, of which two are introduced here: The interaction principles of ISO 9241 [ISO9241-110] and Nielsen’s heuristics [Niel1993].

Standards of the ISO 9241 series

“Ergonomics of human-system interaction” is part of ISO 9241 and includes design recommendations for all types of interactive systems that must be met in human-centred design. ISO9241-110 names and describes seven interaction principles as technology-independent objectives for the design of interactions with interactive systems. They articulate goals with associated suggestions as to how these goals can be achieved.

Interaction principle (formerly dialogue principle)

CPUX-F Definition
A general goal for the design of useful and usable dialogues.

The designer can use these principles from ISO 9241-110 and the recommendations contained to make appropriate design decisions from the viewpoint of appropriate interaction design.

The seven interaction principles are suitability for the user’s task, self-descriptiveness, conformity with user expectations, learnability, controllability, use error robustness, as well as user engagement.

As of 2020, important changes regarding the interaction principles have been made which are of relevance to designers. The former principle of suitability for individualisation is now included as an aspect of controllability. Furthermore, error tolerance has been revised and renamed as use error robustness. Finally, the principle of user engagement has been added. Descriptions of the updated principles follow below.

Use error robustness: The interactive system assists the user in avoiding errors and in the case of identifiable errors, treats them tolerably and assists the user when recovering from errors.

“Use error robustness” involves guidance related to use-error avoidance, use-error tolerance, and use-error recovery, listing 12 recommendations in total. For example, it is recommended that you provide error messages that are precise, comprehensible, and polite.

> An example: Instead of saying, “An error occurred”, say “Passwords must consist of at least 10 characters. The password you entered contains 8 characters but is otherwise acceptable”.

User engagement: The interactive system presents functions and information in an inviting and motivating manner supporting continued interaction with the system.

“User engagement” includes 12 guiding recommendations related to motivating the user, trustworthiness of the system, and increasing user involvement with the system. For example, it is recommended that an interactive system should build trust in its use.

> An example: In addition to giving the user access to product ratings, an online shop displays a logo that each purchase has genuinely been made by the customer providing the feedback.

In total, 64 general recommendations accompany the interaction principles. While the interaction principles are relevant for every system, the approaches to their implementation may vary – the designer must decide which recommendation to follow depending on the context of use.

> An example: An online reservation system for restaurants does not inform the user that their reservation request was not processed due to incomplete input. The user expects to find a reserved table at the respective date but then learns that their hopes were in vain.
>
> If the designer of the restaurant’s website had followed the recommendation “The interactive system should assist the user in detecting, understanding, and correcting errors in input” of the interaction principle use error robustness, the visitor would have known how a valid reservation is made.

The other parts of ISO 9241 “Ergonomics of human-system interaction” focus on design recommendations for specific issues or domains, like the presentation of information (Part 112), design of forms (Part 143), user interface elements (Part 161), and software accessibility (Part 171).

Usability heuristics according to Nielsen

Heuristics (“rules of thumb”) describe the general guidelines for the design of usable systems with a focus on the user interface.

Heuristic

CPUX-F Definition
A generally recognised rule of thumb that helps to achieve usability.

Heuristics help in designing usable solutions and in usability evaluation.

Nielsen has formulated a widely accepted collection of ten usability heuristics with contents that represent essential success factors.

Table 16 Jakob Nielsen's 10 heuristics

Heuristic	Description
Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
Match between system and the real world	The system should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms.
User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
Recognition rather than recall	Adjust the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
Flexibility and efficiency of use	Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
Help user recognise, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Copyright 2025, UXQB e. V.

Heuristic	Description
Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list specific steps to be carried out, and not be too large.

Example: While using a website, the user suddenly receives a status code “502 Bad Gateway error” without any further description. The user is confused and closes the site. If the user does not understand the error message, then the information about the error is pointless. If the designer had followed the usability heuristic “Help user recognise, diagnose, and recover from errors”, the user would have known how to handle the error message.

Example: In order to better follow the heuristic from the last example, the designer revises the error message as follows: There is a problem with the connection to your server. Please check out these common causes and ways to fix the problem: 1. Try to reload the page by pressing F5 or use the reload-function in your browser, 2. Clear your browser’s cache (link to instruction: how to clear the cache), 3. Check the problem with your provider: Please give them the following information: “502 bad gateway error”.

Examples for all heuristics can be found in the CPUX-UT curriculum [UXQB2020].

6.2.2.2 Sample solutions for typical design problems

During the design of user interfaces, some design questions will always reoccur. At this more specific level of design recommendation, design patterns provide “sample solutions” for the designer to reuse as design decisions and ensure consistent design.

Design pattern

CPUX-F Definition
A solution to a commonly occurring design problem within a given context of use. The design pattern describes the design problem, a general solution, and examples of how to apply the solution.

Design patterns are described by a uniform schema and usually contain

- a description of the problem the user has
- the context of use for which the pattern is made
- related recommendations for design
- an explanation of how the solution works
- an example of an implementation of this design pattern in a real problem context.

This schema can also be used to define your own patterns [Tidw2011]

As a general rule, design patterns describe problems as well as their causes and represent a reusable solution approach as well as specific solution options.

Example of a design pattern for error messages:

- Problem description and context of use: For the design of an application form for an online store, it is known that users often enter invalid information or make typing errors. Therefore, in case of invalid entries, appropriate error messages should be displayed. For this, a known design solution will be implemented, so that users recognise the type of message and are supported in finding solutions for the problem.
- Recommendations for design and an explanation of the solution approach: Design input fields in a way that prevents erroneous input (for example, by format restrictions).

For errors that still occur, show a general error message stating that one or more entries are wrong. Additionally, mark the fields which caused the error and provide a detailed message for each field. In order to highlight this message, use colour, an alternative font style, or a graphic.

- Examples for the solution: Screenshots of this design pattern in other online shops.

The designer can use design patterns from design libraries or document them individually for re-use.

6.2.2.3 User interface guidelines

User interface guideline

CPUX-F Definition
A low-level specific rule or recommendation for user interface design that leaves little room for interpretation, allowing designers to implement it consistently.

This most specific level of design recommendations comprises manufacturer-specific user interface guidelines that are based on the extensive experience of practitioners or even empirical studies about the effect of certain design decisions. User interface guidelines are used to ensure the consistent design of products and product families, and ease design decisions.

Style guide

CPUX-F Definition
A collection of user interface guidelines used to ensure consistency in the appearance and behaviour of user interfaces across interactive systems produced by the same organisation.

A style guide might contain user interface guidelines for

- user interface elements
- view/page/screen layouts
- the behaviour of the user interface (not specific to single user interface elements)
- colours (the colour palette) to be used throughout the user interface
- typography
- the use of images, graphics, sound, and video

Style guides are more and more located in design systems.

An example of a user interface guideline from the iOS style guide:
“On devices running iOS 13 and later, people can use the touch and hold gesture to open a context menu, regardless of whether the device supports 3D Touch.”

A collection of user interface guidelines is a style guide, a document that provides the rules for the presentation of a specific interactive system according to a company’s brand and corporate design.

Style guides mainly refer to a class of interactive systems or to the interactive systems of a certain manufacturer. As such, they may be understood as local de facto standards with the goal of standardisation. A good style guide creates consistency in appearance and the behaviour of interactive systems while at the same time helping to communicate and document changes and new features [JaMe2017].

An example: A smartphone manufacturer has specific style guides for each of its smartphone models. These style guides are used for promoting the new developments of respective models. At the same time, there are overarching style guides for all smartphone models as well as a style guide that relates to all of the company’s products.

Copyright 2025, UXQB e. V.

Style guides may contribute to the development of design systems (see 6.4.3) for the implementation of interactive systems.

Typical components of a style guide are

- formulation of the corporate identity and corporate design
- concept and intentions of the product
- view/page/screen layout grid with exact details of proportions
- description and placement of essential functional components (for example, user interface elements with an explanation of their behaviour)
- description and placement of all essential content components (use of logo, headings, text, images, graphics, tables, sounds, videos, etc.)
- the colour palette with appropriate details and instructions for their use
- typography with specifications for fonts and font sizes, distances, types of use (heading, continuous text, captions, etc.).

Recommendations for good style guides:

- Make guidelines easy to apply and adopt (for example, indicate the colour code instead of just representing the colour)
- Make sure that the guidelines given in the style guide conform with general systems of design recommendations
- Include and explain examples for applying each guideline
- Include the references to appropriate design patterns with code snippets
- Follow the criteria for help and documentation (FLUID model) to ensure the usability of the style guide
- State the validity and the version of the style guide
- Include descriptions of the required behaviour of user interface elements since this is not directly visible in the style guide
- Decide on how compliance with the style guide will be checked
- Follow up on deviations from the style guide in a friendly manner
- Update the style guide regularly.

Copyright 2025, UXQB e. V.

6.3 Attending to implicit design tasks

If the scope of a design project covers a bigger part of a product or even the whole product, there are often **implicit design tasks** to be considered that are usually neglected by the initiator and do not show up in a project briefing. It is very important to attend to those implicit design tasks which often require their own context of use analysis.

Learning Objectives	
6.3.a	Understand what implicit design tasks are and know an example.
6.3.b	Know which basic guidelines must be followed for a well-designed search.
6.3.c	Understand the FLUID model for help and documentation.

Among the designer's responsibilities, include addressing implicit design tasks.

Implicit design task

A design task that often is not mentioned in a project briefing, and goes beyond the immediate task-related design. Coherent implementation of implicit design tasks is essential for users to be able to achieve their goals, and therefore, it is the responsibility of the designer to address them.

Among those implicit design tasks are the design of

- feedback, notifications, information, warnings, error messages (Your battery is running low. You might want to plug in your PC.)
- status information ("You have 7 new messages.")
- instructions (Separate email addresses using a space, comma, semicolon, or line break.)
- help: Support provided in the interactive system, which may deal with certain topics or procedures or may refer the user elsewhere
- user documentation: Information available to users in writing or otherwise about how the interactive system works, and how it is to be used
- search
- filtering and sorting lists.

Addressing implicit design tasks will improve the usability of and user experience with the user interface. As examples for the work on implicit design tasks, the curriculum will provide the subjects of search, help, and documentation.

6.3.1 Search

The option to search for content must be supported by almost every interactive system. To design a usable search, the designer must consider the common steps users take when the need for a search arises:

- Identify access to search
- Understand how to use the search function
- Identify the information to be searched for
- If necessary, decide on suggested search terms
- Get an overview of search results
- Reduce the set of search results, if necessary
- Process relevant search results for the intended purpose.

Copyright 2025, UXQB e. V.

Basic recommendations on how to design a search [Trav2009]:
- Provide a simple search as a default search (one input text field)
- Offer a default of search terms and criteria
- Provide a meaningful response to the search query, even when no results are returned.
- Do not mix the search result list with other tasks (for example, the editing of data within the search result list)
- Make the search error-tolerant, provide automatic spell checking, and allow the use of plurals and thematically similar search terms
- Use the most common searches, which often provide useful results for the user
- Define the scope of the search from the user's point of view
- Show matching search results in a results overview
- Explicitly specify the scope of the search results together with the search results
- Let users restrict the scope of search results (if relevant for the task)
- Provide useful meta information in the results overview, such as the document size, the creation date, or the file type
- Enable the search results to be compared using information that is relevant from the user's point of view
- Do not duplicate results
- Enable the further use of single results or result sets
- Finish the search so that all contents are accessible for further work.

6.3.2 Help and documentation

As with the design of a search, the designer needs to follow the steps users take when looking for help, to provide appropriate help and documentation. The FLUID model [Wrig1983] describes five steps that a user completes to solve a problem. Tips are given for each step:
- **F**ormulate: users formulate their problems in the form of search terms
  - Provide an overview of the tasks, for example, in tabular form, which supports the formulation of the problems
  - Provide an easily accessible list of typical errors.
- **L**ocate: users try to find the appropriate content
  - Provide an overview of step-by-step instructions for solving tasks or problems
  - Adapt the overview to the user's problems to be solved and clarify which problems cannot be solved this way
  - Integrate an easy-to-find index with relevant terms
  - Place the most important information at the beginning.
- **U**nderstand: users try to understand the content they have found
  - In addition to definitions and descriptions, use examples
  - Provide explanations of technical terms
  - Be consistent in word usage and use common terms
  - Speak the language of the user
  - Pay attention to comprehensibility, structure the text well, for example, into small text units, and provide the user with further information.
- **I**mplement: users act according to the received information
  - Make sure the information is up to date
  - Show examples of use
  - Explain the consequences of the application
  - Support the inclusion of information in the current task processing.
- **D**etermine: users check if their problems are solved

Copyright 2025, UXQB e. V.

- Explain how the system will behave
- Provide descriptions of possible errors and troubleshooting instructions.

6.4 Documenting design decisions

Throughout the process of Designing Solutions, the designer makes design decisions and defines the look and behaviour of the user interface. In the process, decisions are built on previous decisions, be it for a stand-alone product or a product in a product family. The designer has to decide whether or not it is necessary for the **documentation of design decisions** to take an **explicit approach** and to go for a **user interface specification** or the **annotation** of user interfaces.

Learning Objectives	
6.4.a	Know that it is important to be able to formulate design decisions for discussion.
6.4.b	Know forms of documentation.

6.4.1 The need for documentation

As the first step, design decisions are made in an implicit form: The designer makes design decisions and renders them tangible, for example, in a prototype. Thus, decisions are there for everybody to see in the prototype, but reasons, arguments, and possible trade-offs in this implicit form cannot be shared with or re-used by others.

Depending on the size of the design project, it might be necessary to document design decisions for others to work with or to build on to.

Documentation of design decisions

Written or other stored information that allows the consistent implementation and the re-use of successful design decisions or user interface elements.

Documentation of design decisions

- gives an overview of the existing design decisions and their contents (especially important for complex systems and/or platforms)
- comprehensively informs the team about the designed solution (especially important when the team composition changes)
- helps third parties to build on existing design decisions (particularly important if different teams work together or when agencies are commissioned

It is recommended to document design decisions, if

- it is legally required by standards (for example, for safety relevant systems according to ISO 26262)
- there is more than one designer working on a product or product family and design decisions are taken and reused by designers
- design decisions are supposed to be referred to for as long as the product life cycle, in case design personnel change before the end of the product life cycle
- the interactive system shows considerable complexity and cannot be overseen by one person
- the development model is waterfall-oriented (focuses on clear planning, strict processes, and documentation) rather than agile (focuses on trial and error and fast iteration without much documentation)

Copyright 2025, UXQB e. V.

6.4.2 Explicit approach

Explicit approach

An approach when designing user interfaces where important design decisions are explicitly formulated or documented.

Design decisions are documented in the form of a user interface specification or annotations of prototypes so that they can be traced, implemented, and reused by third parties

If documentation is required, an explicit approach has to be taken that makes design decisions transparent, comprehensible, and sustainable. For an explicit approach in the documentation of design decisions, on top of making them visible in wireframes or prototypes, the respective behaviour of the user interface needs to be explicitly formulated and explicitly documented in a comprehensible and reusable way.

Explicit formulation:

- The designer needs to be able to explicitly express reasons, arguments, and possible trade-offs that lead to design decisions in discussions with team members and stakeholders, in order to help acceptance and reuse of the decision.

  Example of an explicit formulation and its reasoning:

  “The send button for sending a message is always aligned on the right and shown twice, at the beginning and the end of the message”.

  This occupies precious screen real estate but it supports the user in instances where long messages push one or the other send buttons out of view.

Explicit documentation:

- Design decisions are documented in such a specific way that they can be implemented or applied repeatedly. One way to document design decisions is in the form of a user interface specification.

User interface specification

A precise description of relevant design decisions and of all attributes of the user interface of an interactive system. It can refer to design decisions for all design activities.

- As it provides the entirety of design decisions, it serves as a reference for the implementation of the user interface.

Another way to document design decisions is in the form of annotations (of prototypes).

Annotation

A note in the form of a commentary on a design decision, which is inserted directly, for example, in a prototype.

- It may contain both specific formulations of the design decisions and explanations or notes for implementation.

6.4.3 Types of documentation of design decisions

There are different ways to document design decisions; amongst them, is the annotation of prototypes and specification of the user interface. Design patterns and style guides can also be a form of documentation of design decisions. Designers should at least comment on the prototypes with annotations.

It should be documented with annotations if

- quick iterations are possible,
- the complexity of the interactive system is not too high

Copyright 2025, UXQB e. V.

- first solution approaches are developed in the early phases
- prototypes are created
- a continuous communication between design and development is possible.

Sometimes, however, it makes sense to document with specifications, especially if

- the interactive system has a complexity that cannot be overlooked by one person
- work products are handed over to service providers for further development
- an interdisciplinary, team-oriented participatory design is not possible.

The explicit documentation of design decisions may contribute to the development of design systems for the implementation of interactive systems. A design system is a comprehensive and complex documentation for the interactive systems of a company.

- It contains all the frontend components a team needs to design and develop a specific product. This ensures consistent design within a product family.
- A design system contains, for example, general design guidelines, one or more pattern libraries, as well as processes and code components.
- By explicitly documenting design decisions, the designer supports those who are responsible for developing a design system.

Copyright 2025, UXQB e. V.

Appendix 1: Overview of the CPUX-DS terms and activities

Learning units	Terms
1. Important perspectives for design activities	
1.1 The baseline for Designing Solutions	Designing Solutions, context of use analysis, mental model, conceptual model, system image
1.2 Overview on design activities	Design activity, design decision, conceptual modelling, information architecture, interaction design, interface design, information design, sensory design
1.3 Iterating as needed and as the project demands	Iterative process, Design Darwinism, validation, formative evaluation, user feedback
1.4 Considering the whole user experience across all touchpoints	Aesthetics of interaction, touchpoint, design thinking
2. Early design	
2.1 Design of user interfaces for the achievement of goals	Task object, executable function, signpost, task, action, user interface, task-related operation, user assistance
2.2 Design activity: conceptual modelling	Task model, user requirement, task model for design, dialogue step, interaction specification, use scenario, user journey map
3. First drafts	
3.1 Design activity: information architecture	Navigation structure, navigation system, navigation elements, card sorting, tree testing
3.2 Design activity: Interaction design	Interaction sequence, view, user interface structure
3.3 Make design decisions tangible to get feedback	Sketch, wireframe, wireflow, prototype, low-fidelity prototype, high-fidelity prototype
4. Refined Design	
4.1 Design activity: Interface design	User interface element, responsive design, adaptive design, mobile first
4.2 Design activity: Information design	Principles for the presentation of information, information reading, reading scheme, inverted pyramid approach
4.3 Design activity: Sensory design	Gestalt laws, white space
5. Specific human needs	
5.1 Accessibility	Accessibility, disability, assistive technologies
5.2 Design ethics	Persuasive design, honest interface, dark pattern, behaviour pattern, nudge, interface metaphor, domain knowledge, illusion of control, default,
5.3 Cultural diversity	Intercultural user interface design, internationalisation, localisation, intercultural usability testing
6. Aspects beyond the design activities	
6.1 Managing stakeholders	Consensual agreement, human-centred quality, human-centred quality objectives, avoidance of harm from user, participatory design, customer journey map
6.2 Setting the frame for design work	System of design recommendations, interaction principle (formerly dialogue principle), heuristic, design pattern, user interface guideline, style guide
6.3 Attending to implicit design tasks	Implicit design task
6.4 Documenting design decisions	Documentation of design decisions, explicit approach, user interface specification, annotation

Copyright 2025, UXQB e. V.

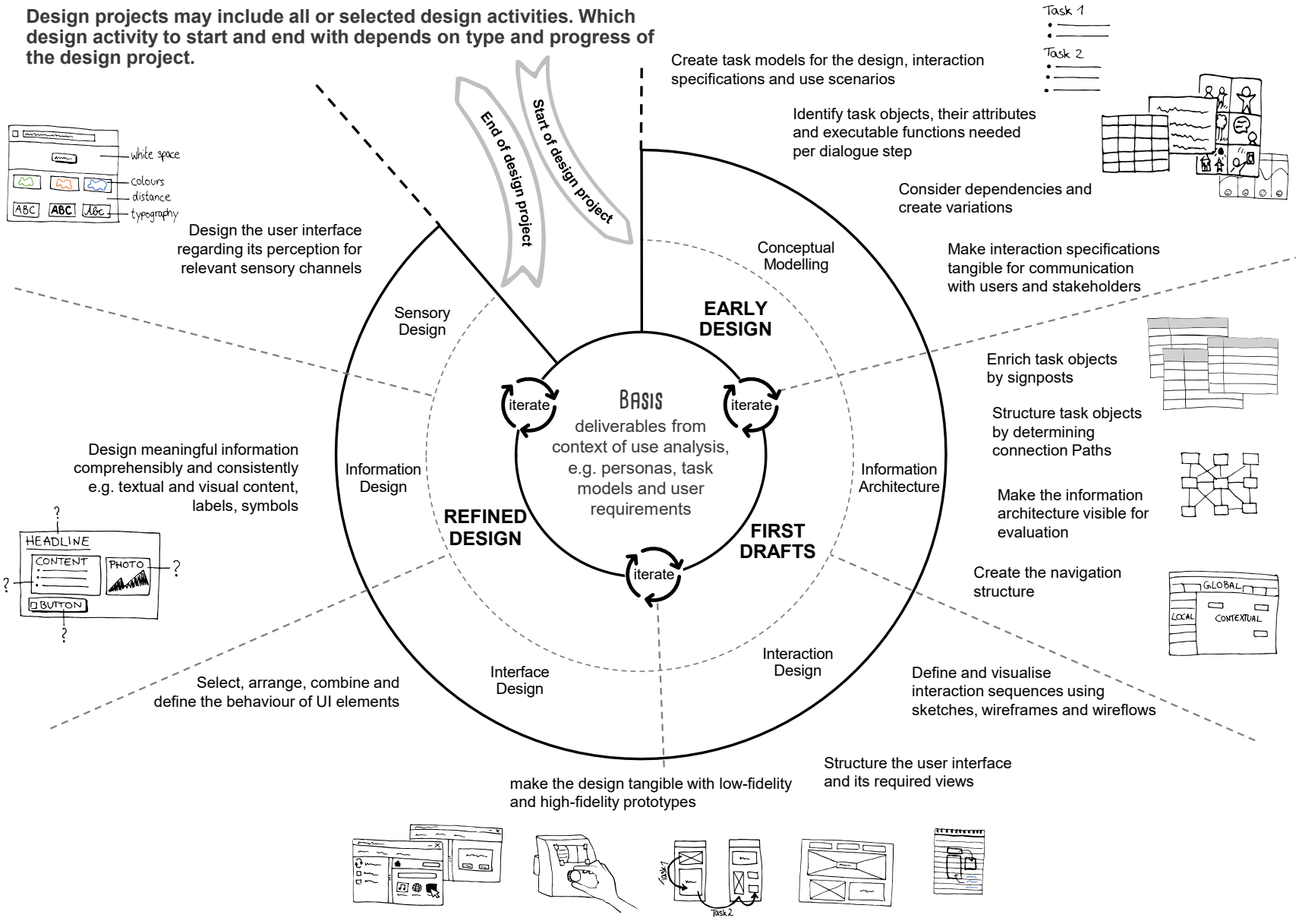


Figure 51 Design Process in Designing Solutions (Overview of CPUX-DS activities – alternative to Figure 2

Appendix 2: Model Seminar

Day 1 "Designing Solutions: Introduction, Process, Early Design" (360 min)

Timetable	Contents	Ideas for exercises (* = mandatory)
09:00 – 10:30	**Check-in and General Introduction to the course** **Introduction to Designing Solutions I** • Orientation: Designing Solutions regarding HCD (ISO 9241-210) • The baseline for Designing Solutions • Design activities in Designing Solutions (the process) • Iterating as needed and as the project demands	• Discuss examples for user problems based on unfulfilled user requirements (through design) • Classify deliverables in design activities • Discuss examples for iterations in different project situations
10:45 – 11:30	**Introduction to Designing Solutions II** • Considering the whole user experience	• Find examples of human needs relevant to design • Find (real) examples for aesthetic interactions
11:30 – 12:15	**Design of user interfaces for the achievement of goals I** • Task-related operation to achieve user goals	• How to understand the figure "Task-related Operation and components of an interactive system" • Identify task objects, attributes, executable functions, and signposts in a given screen flow.
13:15 – 14:00	**Design of user interfaces for the achievement of goals II** • User assistance: explicit action guiding information • Intended and undesirable consequences of user interface design	• Identify explicit action-guiding information in the given screen flow • Assess examples regarding intended and undesirable consequences of UI design
13:15 – 16:30 incl. 15 min break	**Design activity: Conceptual modelling** • Creating task models for design • Creating interaction specifications based on task models for design • Identifying task objects, attributes, and executable functions in interaction specifications • Considering dependencies and creating variations • Communicating use scenarios to users and stakeholders	• Create an interaction specification for one task in small working groups (step-by-step: adapt a task model for design, create an interaction specification, identify task objects, executable functions and attributes) * • Discuss participants' experiences in every step
16:30 – 17:00	**Introduction to the examination regulations** • Process, assessment, documents	• Homework: "five questions you should be able to answer"

Copyright 2025, UXQB e. V.

Day 2 “Designing Solutions: First Drafts” (360min)

Timetable	Contents	Ideas for exercises (* = mandatory)
09:00 – 09:45	**Looking back, open questions and agenda for today** • Discussing the homework • Question and answers • First Drafts: Overview	• Open discussion of answers and further questions of the participants
09:45 – 10:30	**Design activity: Information architecture I** • Development of an information architecture • Enhance task objects with signposts	• Discussing examples of different user interfaces regarding their information architecture and uncovering the elements of information architecture
10:45 – 12:15	**Design activity: Information architecture II** • Structure task objects by determining connection paths • Make the information architecture visible for evaluation • Create the navigation structure using connection paths and signposts	• Structure and visualise task objects with one selected method, using the example of the interaction specification *
13:15 – 14:15	**Design activity: Information architecture III** • Evaluate the information architecture	• Small group task: Card Sorting with prepared cards in accordance with the previous exercise • Discuss consequences of the results for the information architecture *
14:30 – 15:15	**Design activity: Interaction design** • Define task-related Interaction sequences • Visualise interaction sequences • Structure the user interface based on all required views	• Assess visualised interaction sequences from a task-oriented user perspective *
15:30 – 17:00	**Make design decisions tangible to get feedback** • Characteristics of visualisations across design phases and typical types of visualisations • Benefits of visualising design decisions and iterating visualisations • Iterating visualisations through evaluation • Guidelines for creating low-fidelity prototypes • Criteria for selecting prototyping tools	• Small group task: Prototyping with a number of sketches showing task-related interaction sequences (using the results of the prior example – the same example that was used for the interaction specification, the information architecture and card sorting) *

Copyright 2025, UXQB e. V.

Day 3 "Refined Design, Specific Human Needs & Design Projects and Aspects beyond Design Activities" (360min)

Timetable	Contents	Exercises
09:00 – 09:30	**Looking back, open questions and agenda for today**	
09:30 – 10:30	**Design activity: Interface design** • Create the interface design by selecting, arranging, combining, and defining the behaviour of user interface elements • Appropriate use of user interface elements	• Exercise about the correct use of UI elements
10:45 – 11:35	**Design activity: Information design** • Principles for the presentation of information • Information reading and comprehensibility of content • Specific design recommendations for comprehensibility	• Exercise about interpretation & misinterpretation of information
11:40 – 12:30	**Design activity: Sensory design** • Design the user interface regarding its perception through relevant sensory channels • Gestalt laws • Colours, font sizes, and white space	• Experiencing and discussing examples related to perception
13:30 – 15:00	**Specific human needs** • Accessibility • Design ethics • Cultural diversity	• Discussing positive and negative examples for the consequences of considering or not considering accessibility
15:15 – 16:45	**Aspects beyond the design activities** • Managing stakeholders • Setting the frame for design work • Attending to implicit design tasks • Documenting design decisions	• Discuss the quality objectives of stakeholders • Discuss experiences from small groups working in the course and their consequences for working with teams and stakeholders • Exercise about the application of heuristics * • Assess positive/negative examples for designing search and help • Discuss forms of documentation
16:45 – 17:00	Course feedback & evaluation	• UXQB feedback form

Copyright 2025, UXQB e. V.

Appendix 3: Important changes to this document

Date, Version	Changes compared to version
31 January 2021, version 1.01	• linguistic correction • implementation of editorial changes based on feedback from the German version rolled out in October
1 October 2025, version 1.02	• The requirement that a CPUX-F certificate is necessary to take the exam for this advanced-level certificate has been removed and relevant text passages have been amended.

Copyright 2025, UXQB e. V.

Appendix 4: References & Index

[BrAA2013] L. Brandimarte, I. Adjerid, A. Acquisti: Gone in 15 Seconds: The Limits of Privacy Transparency and Control. IEEE Security & Privacy, 11 (4), 2013.

[Brig2011] H. Brignull: Dark Patterns: Deception vs. Honesty in UI Design. https://alistapart.com/article/dark-patterns-deception-vs.-honesty-in-ui-design, 2011, last visited March 2019.

[Brig2013] H. Brignull: Dark Patterns: Inside the interfaces designed to trick you. http://www.theverge.com/2013/8/29/4640308/dark-patterns-inside-the-interfaces-designed-to-trick-you, 2013, last visited February 2019.

[Buxt2007] B. Buxton: Sketching User Experiences. Getting the design right and the right design. Morgan Kaufmann, Burlington, 2007.

[CaRo2014] J. M. Carroll, M. B. Rosson: Getting around the Task-Artiface Cycle: How to Make Claims and Design by Scenario. IBM Watson Research Center. ACM Transactions on Information Systems, Vol. 10. No. 2, 2014.

[EFPA2005] European Federation of Psychologists' Associations: Meta-Code of Ethics. http://ethics.efpa.eu/metaand-model-code/meta-code, 2005, last visited March 2019.

[Garr2010] J. Garrett: The Elements of User Experience: User-Centered Design for the Web and Beyond (Voices That Matter). 2nd Edition, New Riders, Indianapolis, 2010.

[GePo2016]] T. Geis, K. Polkehn: Customer Experience, User Experience – and the Business Analyst. UXQB (International Qualification Board for Usability and User Experience) & IIBA (International Institute of Business Analysis), https://www.gartner.com/imagesrv/media-products/pdf/iiba/customer-exp.pdf, 2016. last visited May 2020.

[GePo2018] T. Geis, K. Polkehn: Praxiswissen User Requirements: Nutzungsqualität systematisch, nachhaltig und agil in die Produktentwicklung integrieren. dpunkt.verlag, Heidelberg, 2018

[GeTe2018] T. Geis, G. Tesch: Basiswissen: Usability und User Experience. Systematisch und strukturiert vom Nutzungskontext zum gebrauchstauglichen Produkt. dpunkt.verlag, Heidelberg, 2018.

[HaPy2019] R. Hartson, P. Pyla: The UX Book. Process and guidelines for ensuring a quality user experience. Morgan Kaufmann, Burlington, 2019.

[HaDi2017] M. Hassenzahl, S. Diefenbach: Psychologie in der nutzerzentrierten Produktgestaltung: Mensch-Technik-Interaktion-Erlebnis. Springer, Berlin, 2017.

[Hall2018] E. Hall: Conversational Design. A Book Apart, New York, 2018.

[Heim2019] R. Heimgärtner: Intercultural User Interface Design. Springer International Publishing Springer, Basel, 2019.

[ISO9241-171] ISO 9241-171:2008. Ergonomics of human-system interaction – part 171: Guidance on software accessibility, Geneva, 2008.

[ISO9241-210] ISO 9241-210:2019. Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems, Geneva, 2019.

[ISO9241-110] ISO 9241-110:2020. Ergonomics of human-system interaction – Part 110: Interaction principles, Geneva, 2020.

Copyright 2025, UXQB e. V.

[ISO9241-112] ISO 9241-112:2017. Ergonomics of human-system interaction – Part 112: Principles for the presentation of information, Geneva, 2017.

[ISO9241-125] ISO 9241-125:2017. Ergonomics of human-system interaction – Part 125: Guidance on visual presentation of information, 2017.

[ISO9241-161] ISO 9241-1161:2016. Ergonomics of human-system interaction – Part 161: Guidance on visual user-interface elements, 2016.

[JaMe2017] J. Jacobsen, L. Meyer: Praxisbuch Usability und UX. Rheinwerk Verlag, Bonn, 2017.

[John2014] J. Johnson: Designing with the Mind in Mind: Simple Guideline to Understanding User Interface Design Guidelines. Morgan Kaufmann, Burlington, 2014.

[JoMa2017] T. Jokinen, M. Marwede: Use – User and environment friendly products: preserve and extend what is already made. https://sustainabilityguide.eu/ecodesign/use/, 2017, last visited February 2019.

[Kalb2007] J. Kalbach: Designing Web Navigation. O'Reilly Media, Sebastopol, 2007.

[KaLi2012] J. Kalbach, K. Lindemann (2012): Designing Screens Using Cores and Paths. https://boxesandarrows.com/designing-screens-using-cores-and-paths/, 2012, last visited June 2020.

[Knap2016] J. Knapp: Sprint – Wie man in nur fünf Tagen neue Ideen testet und Probleme löst. Redline Verlag, München, 2016.

[LeDH2014] E. Lenz, S. Diefenbach, M. Hassenzahl: Aesthetics of Interaction – A Literature Synthesis. https://www.researchgate.net/publication/267729174_Aesthetics_of_Interaction_-_A_Literature_Synthesis, 2014, last visited February 2019.

[Mayh1999] D. J. Mayhew: The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design. Morgan Kaufmann, San Francisco, 1999

[McGo2018] G. McGovern: Top Task: A How-to Guide. Silver Beach, Gormanston, Meath, 2018.

[Mose2012] C. Moser: User Experience Design: Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern. Springer Verlag, Berlin Heidelberg, 2012.

[Niel1993] J. Nielsen: Usability Engineering. Academic Press, London, 2013.

[NiLo2006] J. Nielsen, H. Loranger: Prioritizing Web Usability, New Riders Publishing, California, 2006.

[Norm2013] D. Norman: The Design of Everyday Things. Basic Books, New York, 2013.

[Pern2017] K. Pernice: F-Shaped Pattern of Reading on the Web: Misunderstood, But Still Relevant Even on Mobile, https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/, 2017, last visited April 2019.

[RoCa2002] M. Rosson, J. Carroll: Usability Engineering: Scenario-Based Development of Human-Computer Interaction, Morgan Kaufmann, Burlington, 2002.

[RoMA2015] L. Rosenfeld, P. Morville, J. Arango: Information Architecture for the World Wide Web: For the Web and Beyond, O'Reilly UK Ltd., Sebastopol, 2015.

[Scha2018] A. Schade: Inverted Pyramid: Writing for Comprehension. https://www.nngroup.com/articles/inverted-pyramid/, 2018, last visited April 2019.

[ShPl2009] B. Shneiderman, C. Plaisant: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson, Boston, 2009.

Copyright 2025, UXQB e. V.

[Spil2011] F. Spillers: Is your design Evolutionary or Revolutionary? https://www.experiencedynamics.com/blog/2011/03/your-design-evolutionary-or-revolutionary, 2011, last visited February 2019.

[Spre2018] P. Spreer: PsyConversion. 101 Behaviour Pattern für eine bessere User Experience und höhere Conversion-Rate im E-Commerce. 2018, Springer Gabler, Wiesbaden.

[Stre2018] R. Strebe: Aufschlussreiche Irrwege – Auswertung von Tree Testing https://www.usabilityblog.de/aufschlussreiche-irrwege-auswertung-von-tree-testing/, 2018, last visited March 2019.

[StJW2005] D. Stone, C. Jarrett, M. Woodroffe, S. Minocha: User Interface Design and Evaluation. Morgan Kaufmann, San Fracisco, 2005.

[ThSu2008] R. H. Thaler, C. R. Sunstein: Nudge. Improving Decisions About Health, Wealth, and Happiness. Yale University Press, New Haven & London, 2008.

[Tidw2011] J. Tidwell: Designing Interfaces. O'Reilly, Sebastopol, 2011.

[Trav2009] D. Travis: List of search usability guidelines. https://www.userfocus.co.uk/resources/searchchecklist.html, 2009, last visited March 2019.

[UXQB2020] UXQB e.V.: CPUX-UT Curriculum – Certified Professional for Usability and User Experience – Usability Testing and Evaluation. https://uxqb.org/wp-content/uploads/documents/CPUX-UT_EN_Curriculum.pdf, 2020, last visited August 2020.

[Ware2004] C. Ware: Information Visualization: Perception for Design, Morgan Kaufmann, Burlington, 2004.

[Wood2007] L. E. Wood: User Interface Design: Bridging the Gap from User Requirements to Design. Taylor and Francis Books, 2007.

[Wrig1983] P. Wright: Manual Dexterity — A User-Oriented Approach to Creating Computer Documentation, 1983.

Copyright 2025, UXQB e. V.

Index

Accessibility 95
Action 32
Adaptive design 80
Aesthetics of interaction 27
Annotation 121
Assistive technology 96
Avoidance of harm from use 108
Behaviour pattern 100
Card sorting 66
Conceptual model 10
Conceptual Modelling 13
Consensual agreement 107
Context of use analysis 9
Customer journey map 110
Dark pattern 99
Default 101
Design Activity 12
Design darwinism 21
Design decision 13
Design pattern 115
Design thinking 28
Designing Solutions 9
Dialogue step 41
Disability 95
Documentation of design decisions 120
Domain knowledge 101
Executable function 31
Explicit approach 121
Formative evaluation 22
Gestalt laws 91
Heuristic 114
High-Fidelity Prototype 74
Honest interface 99
Human-centred quality 107
Human-centred quality objectives 108
Illusion of control 101
Implicit design task 118
Information architecture 16
Information design 20
Interaction design 18
Interaction principle 112
Interaction sequence 68
Interaction specification 41
Intercultural usability testing 105
Intercultural user interface design 103
Interface design 19
Interface metaphor 100
Internationalisation 104
Inverted pyramid approach 87
Iterative process 21
Localisation 104
Low-Fidelity Prototype 74
Mental model 10
Mobile first 81
Navigation elements 65
Navigation structure 62
Navigation system 62
Nudge 100
Participatory design 109
Persuasive design 98
Principles for the presentation of information 84
Prototype 73
Reading scheme 86
Responsive design 80
Sensory design 20
Signpost 31
Sketch 72
Style guide 116
System image 10
System of design recommendations 111
Task 32
Task model 37
Task model for the design 38
Task object 31
Task-related operation 34
Touchpoint 28
Tree testing 67
Use scenario 50
User assistance 35
User feedback 23
User interface 34
User interface element 79
User interface guideline 116
User interface specification 121
User interface structure 70
User journey map 50
User requirements 38
Validation 22
View 68
White space 94
Wireflow 73
Wireframe 73

Copyright 2025, UXQB e. V.