Marcin Sikorski

INTERACTION DESIGN IN AGILE IT PROJECTS

Gdańsk Tech
Publishing House

GDAŃSK UNIVERSITY OF TECHNOLOGY PUBLISHERS
CHAIRMAN OF EDITORIAL BOARD
Dariusz Mikielewicz

EDITOR OF SCIENTIFIC PUBLICATIONS
Michał Szydłowski

REVIEWERS
Witold Chmielarz
Jerzy Grobelny

COVER DESIGN
Wioleta Lipska-Kamińska

Published under the permission
of the Rector of Gdańsk University of Technology

Gdańsk University of Technology publications may be purchased at
https://www.sklep.pg.edu.pl

ISBN 978-83-7348-840-3

Gdańsk University of Technology Publishing House

Edition I. Ark. ed. 8,8, ark. print 10,5, 247/1134

Spis treści

Preface

Interactive systems, such as various types of software, online services or mobile applications, in recent years have become an integral part of everyday life. These systems are becoming increasingly complex from a technical viewpoint, especially in their “back-end” part, including necessary IT infrastructure, databases, web services and architectures that remain invisible for end-users. Despite engineering complexity of the back-end, from user’s perspective the operation of an interactive system should be as easy as possible. The user interface, often referred as “front-end”, should be designed to be simple to use, visually attractive, providing a positive User Experience (UX) and – above all – granting functionality and usability for end-users or customers.

For this reason, interaction design has recently emerged as a distinct professional area of information technology (IT). Interaction design is taking its roots from the scientific discipline of Human-Computer Interaction (HCI), which is located on the crossroads of social sciences (mainly management and cognitive sciences) and engineering sciences (mostly computer science and software engineering).

Quality of interaction and quality of user experience (UX) now are indispensable elements of IT product quality. Consequently, the User-Centred Design (UCD) approach, being a part of HCI, has been successfully applied for improving usability of IT products and adding a “customer’s voice” to IT projects.

IT projects have undergone radical changes in recent years. Nowadays due to market pressures, most of IT solutions, such as online services, websites, and mobile applications, are designed and developed using the agile approach. The agile approach in IT project management declares readiness for rapid changes in requirements, customer focus and quality assurance based on two pillars: excellent communication in the development team and intensive cooperation with customers. Agile approach introduced “sprints“ – short, dynamic design cycles, frequent prototyping and regular evaluation of the developing product by prospective users and customers. Focusing on users’ needs, contemporary IT projects attempt

to combine techniques inherited from classical software engineering with novel techniques borrowed from the agile approach.

Therefore, the main goal of this book is to present the impact of agile approach on User-Centred Design, that resulted in gradual adaptation of interaction design methods to agile IT projects.

The first part of this book (chapters 1–4) provides an overview of interaction design principles for graphical, web and mobile user interfaces. All three types of user interfaces are now popular as typical points of access to applications and services users need for their daily activities.

The second part (chapters 5–10) present a critical review of user-centred techniques useful for improving usability of interactive products, primarily addressed to agile IT projects. A number of user-centred techniques useful at different stages of an agile IT project were presented, with focus on optimizing their positive impact to users, customers and project clients.

Regarding quality management terminology, the first part of the book represents a product-oriented perspective, while the second part is highlighting a project-oriented view, spanning all main stages of a typical IT project: the Strategy, Analysis, Design, Development (prototyping), Evaluation and Testing, and the Retrospective.

The author hopes that this book will be a source of valuable theoretical and practical knowledge for all researchers and practitioners (especially IT managers) involved in cooperation with users and customers in IT projects. Furthermore, readers interested in new trends in interaction design should also find here an inspiration for creating software-based solutions developed with adequate balancing engineering excellence with human needs and values.

The author would like to thank the Reviewers of this book, Prof. Witold Chmielarz, and Prof. Jerzy Grobelny, for their valuable comments which helped to bring the book to the final shape.

1. Quality of interactive products

1.1. Interactive IT products

Interactive systems built with Information-Technology (IT) play an important role in supporting human activities in business and in everyday life. Examples of interactive systems include products and solutions such as:

- software systems for office, engineering, financial or commercial purposes, etc.;
- application software for personal use like games, education, family finances, etc.;
- websites for public information, e-commerce and services, as well as intranet portals for internal use in companies;
- web applications for calculations, navigation, reservations, etc., (often embedded into specific websites for shopping, travel or insurance;
- mobile applications, including mobile websites (m-pages), providing users with access to online services from mobile devices.

Additionally, various electronic devices like built-in car audio and navigation systems, remote controls, etc., can be also treated as interactive systems, if only they enable users to access some type of user interface with display and control functions.

Interactive systems are used for work and for private life, including learning and entertainment. They can be used in following modes (Figure 1.1):

- desktop mode: user usually sitting at the desk with a stationary computer, operating a specific software to complete a specific task;
- web-based mode: user is using a computer with a web browser with to explore the internet or to use a specific web application for a specific task;
- mobile mode: user is using a mobile device (a smartphone or tablet) with mobile web browser or mobile applications dedicated for specific tasks.

In desktop mode users are usually put in the role system operators because for specific work-related tasks a specialistic software has to be used, often licenced to an institution or a company. Otherwise, in web-based and mobile modes usually users are consumers having a broad choice of information, applications and online services, some of them can be fee-based. As a result, expectations of consumers are often higher than those of system operators.

Figure 1.1. IT-based interactive systems and services
(Credits: https://wdrfree.com)

Interactive systems usually have two important design areas:

- front-end: the user interface, including the screen and other devices helpful in control actions;
- back-end: any other components invisible for the user, like servers, cloud services or network infrastructure;

Interaction design is limited to designing the user interface (the front-end), but also factors located in the back-end (like hardware performance or network speed) may have a serious influence on user's satisfaction from using a specific IT products.

A well-designed user interface improves the performance of users' tasks, provides a smooth interaction and pleasure of use, shaping users' positive attitude to the system and to the manufacturer's brand. Task completion time, tolerance for human errors, understandability, ease of learn and use, are frequently mentioned users' expectations regarding IT products.

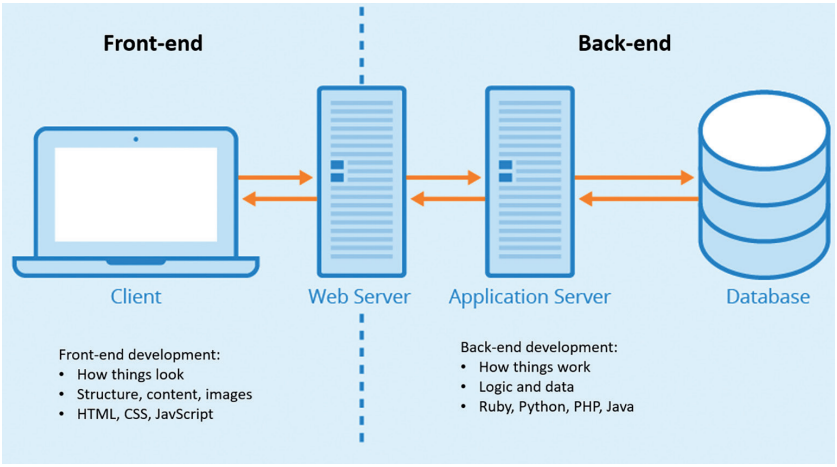

Figure 1.2. Front-end and back-end of a web-based service
(Credits: https://teamquest.pl)

In general, the quality of an interactive IT product can be viewed in three dimensions:

- technical quality ("engineering excellence"), mainly described in terms of technical requirements, represents the degree of technical excellence of a specific solution and refers mainly to measurable design characteristics (e.g. performance, reliability, complexity of the system architecture, and code attributes);
- ergonomic quality ("user-friendliness"), describing system's compliance with ergonomic requirements regarding the user interface, the ease of use and feeling of comfort when using the system; sometimes the ergonomic quality is also described using rather vague terms "intuitiveness", "friendliness", or "ease of use";
- usability ("quality in use") expressed as by the level of user satisfaction resulting from the use of the product in real tasks by specific users; achieving high usability depends on providing both technical quality and ergonomics quality during designing and manufacturing of an interactive product.

Because the technical excellence is no longer a guarantee for the market success, in contemporary IT projects design efforts are largely directed towards building a competitive advantage in areas such as usability, user experience, enriching user's lifestyle or other factors building the perceived quality of a specific IT product. For this reason, knowledge accumulated by research conducted within the discipline Human-Computer Interaction (HCI) has been broadly used for designing software systems, websites, online services or mobile applications (Hartson and Pyla, 2012; Pinhanez, 2009).

1.2. Perceived quality of IT products

Basic terminology

Perceived quality of interactive products (hereafter limited to the front-end only) is shaped by following main factors (Shneiderman et. al., 2017; Rogers et al., 2015; Hartson and Pyla, 2012; Nielsen, 1993):

1. Functionality: the range of functions available in a specific interactive product, such as a software application, website or online service; additionally, the degree to which available set of function is matching users' needs is considered, too.
2. Usability: the degree to which a user can achieve specific objectives in a specific context of use. According to ISO 9241-11 standard usability is defined as the outcome of tasks efficiency, task effectiveness and user satisfaction.
3. User Experience: the whole of user's emotions resulting from the outcome of IT product operation. According to Hassenzahl (2008), User Experience is combination of pragmatic (task-related) and hedonistic (pleasure-related) emotions, which jointly shape user's attitude to a specific IT product, service website or mobile app and to the service provider.

The term "usability" for IT products is defined in different ways in literature and in ISO standards. It can have different meanings for instance in the case of software operated through a computer screen and another meaning for on-line services operated from a mobile device like a smartphone or a tablet. For this reason, a classical, yet very universal definition proposed by Jakob Nielsen (Nielsen 1993) can be still useful. Nielsen defined that the usability of an IT product:

- is a quality attribute expressing the ease of use of an interactive product and it can also mean a set of methods used to improve the usability of a product in the design process (called "usability engineering");
- is defined by such characteristics as: ease of learning, effectiveness in achieving task objectives, ease in remembering accomplished skills, tolerance for human errors, and user satisfaction, understood as subjectively perceived enjoyment from using the system.

Nielsen's definition sometimes interpreted in a way expressing usability as the outcome of all quality characteristics which make the product supportive in completing user's tasks, easy to learn and operate, and pleasant to use.

While the term "quality" describes a generic excellence of an IT product, the term "usability" describes quality in use, experienced by specific users in specific tasks executed in a specific environment. For instance, a validated, high-quality website may represents different usability levels for young users and for seniors with possible visual impairments. Usability is therefore a subjective matter, and can

be rather described than measured, considering a specific context of use: users' characteristics, their tasks, and a local environment.

The term "User Experience" (UX) represents all emotions, feelings, impressions resulting from the operation of a specific interactive system, like a software product or an internet shop. According to Hassenzahl (2008), User Experience is a combination of two principal components: (1) pragmatic (task-related) and (2) hedonistic (pleasure-related) emotions. They jointly shape user's attitude to a specific interactive product, website or mobile app, and also to the brand of a service provider.

Individual episodes of UX get cumulated across subsequent interactions, like operating a software or visiting a website. Even quick memories "easy-to-use", "intuitive", "complicated", "slow" or "demanding" can instantly and significantly shape customers' willingness to return, and whether a specific website is likely to become a favourite one.

If UX is positive, satisfied and loyal customers often convert their positive user experiences into opinions, comments and encouraging recommendations. As they are published online, they are important for attracting new customers, hence optimizing UX is a critical issue in web design.

In case of operating a software product, UX also shapes the attitude to the system regarding whether an operator likes it or not. Nevertheless, even if UX is negative, an operator has no choice just to use it, but probably with frustration affecting quality of work and with negative attitude to a specific system, software house or a specific company.

The ISO-based model of perceived quality

Considering the impact of a local context of use (specific users, tasks and environment), perceived quality of an interactive product results from two factors: (1) quality of design (product characteristics), and (2) quality in use (how these characterises actually preformed in a specific context of use).

International standards ISO/IEC 9126 and ISO 9241-11 specify required quality characteristics for a software application – and more generally for software-based interactive product. The standard ISO/IEC 9126 specifies six key characteristics of software product quality: functionality, performance, reliability, maintainability, portability, and usability. According to ISO/IEC 9126, usability is composed of the following sub-characteristics:

- understandability: ease at which a user understands system functionalities;
- learnability: ease at which a user learns to operate the system;
- operability: ease at which a user uses the system in a given context;
- attractiveness: stimulating user interest;
- compliance: compliance with standard solutions regarding usability.

Paradoxically, another standard ISO 9241-11 defines usability as a result of three principal components:

- effectiveness: the degree to which the product performs its objectives in a specific context of use;
- efficiency: the relationship between the task outcomes and expenditures incurred for their accomplishments;
- satisfaction: the degree of user satisfaction resulting from the actual use of a product in specific tasks, in a specific context of use.

Apparent discrepancy between the two standards can be mitigated by adequate interpreting usability-related characteristics to a specific interactive product located in a specific context of use, as suggested in both standards.

Figure 1.3 presents the components affecting perceived quality of an interactive software product as a combined view of (1) quality characteristics derived from standards and (2) user-perceived quality factors originating from a specific context of use.

User's characteristics, goals, and tasks to be executed in local environment are coupled with a specific IT product and its characteristics, as shown in the left-hand part of Figure 1.3.

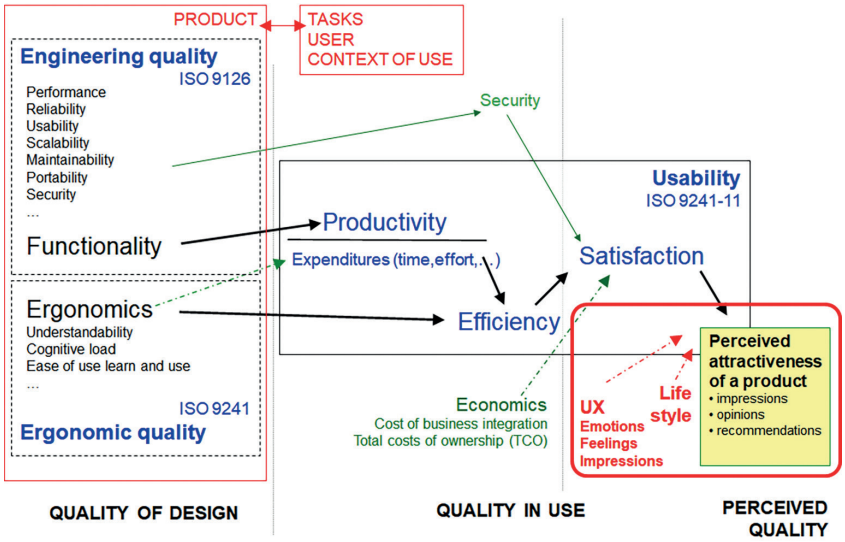

Figure 1.3. Perceived quality of an interactive software product

From the user's viewpoint, among engineering quality characteristics defined by ISO/IEC 9126 the functionality is essential, and it should be adequate to the user's task. The task is executed by a specific user described by demographic profile, education, attitude, skill, current goals etc. Achieving user's goal can be easier or difficult, depending by an interplay between components shaping the engineering

and the ergonomic quality of the IT product. As a result, actual performance of user's tasks, related to all expenditures required to produce a specific outcome, expresses the efficiency of a computer-supported task. In case of a software product, efficiency is the key parameter which shapes user's satisfaction from the task outcome, and from the usage of a specific software product.

Interpretation of usability expressed in the standard ISO 9241-11 for almost two decades was adequate to all contexts where the user was put in a role of system operator. However, for many interactive products today users are no longer merely trained system operators, acting on behalf of a specific employer or an institution. Currently users often are online customers who have specific requirements; they pay attention to the specific content (like products, prices etc.) and usability of a selected website (i.e. how easily the task can be completed), but also if the task execution (a shopping process) was pleasant, convenient and compliant to their subjective expectations.

The FUVUX model

ISO-based quality concepts have been suitable primarily for software products and users employed as system operators in companies. Nowadays, while majority of IT products are available as online services, competing for customers aware of broad choice and expecting outstanding quality in all dimensions.

Obviously, functionality, usability, and User Experience, relevant to a specific interactive product, also affect the trust users/customers have towards a specific service and its vendor. In a short-term perspective, momentary user experiences shape user's willingness to return and use the service website or app again. In a long-term perspective, cumulated user experiences affect user-perceived quality of relationship with a specific service, its vendor and user's attitude to its brand (e.g. a specific on-line bank).

Functionality, usability, user experience and value constitute subsequent layers, affecting user-perceived perceived quality (Figure 1.4). This mechanism, described as the FUVUX model (Sikorski, 2012), describes a general framework for designing contemporary interactive systems, applicable not only for e-business or e-commerce areas. The four components sequentially upgrading the user-perceived quality are the following:

- **F**unctionality: the system should have all the functions needed by the user to perform all tasks;
- **U**sability: the system should enable achieving intended result with the lowest possible inputs from the user;
- **U**ser e**X**perience: the system should provide the user with positive emotions, feelings, and experiences, encouraging the user return to the site;
- **V**alue: the system should support developing of valuable relationships between the supplier and user (client) and should provide a feel of the benefits from system use.

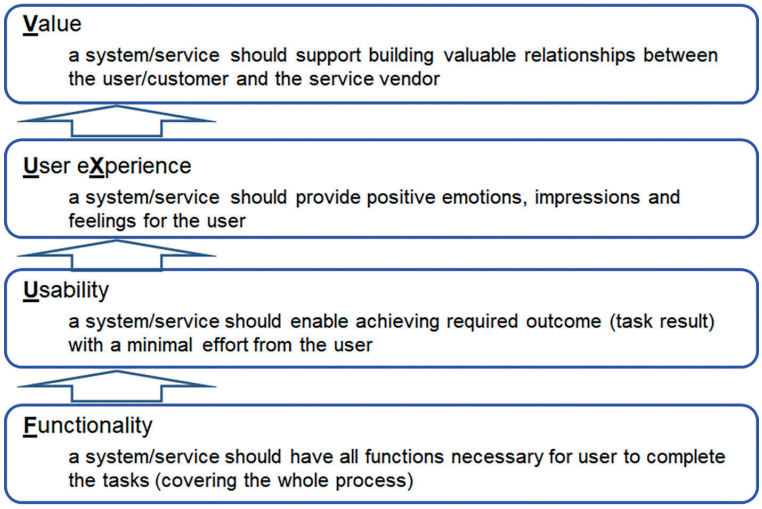

Figure 1.4. The FUVUX model for digital product

The FUVUX model is based on following assumptions, stacked bottom-up:

1. Providing users with adequate functionality enables users to complete theirs tasks and evaluate usability of a specific system.
2. Only achieving high usability leads to positive UX, resulting in user's willingness to reuse the system or service again.
3. Only cumulated positive UX makes users revisit and believe that the system or delivers benefits (value) and builds win-win relationships with the vendor or brand.

The FUVUX model defines the basic features of an interactive system, and the four layers composing customer-perceived quality. The FUVUX model also synthetically describes the chain of expectations for interactive systems (including web and mobile apps), viewed from the user (online customer) perspective.

Service development layer model

The development layer model describes how the maturity of an online service affects user/customer engagement in online interactions.

This model, proposed by Sikorski (2008) and presented in Figure 1.5 defines:

- the design focus (left-hand side), composed of five layers of increasing service maturity: Functionality, Usability, Experience, Relationship and Value;
- the interaction outcome (right-hand side), observed by user/consumer: Performance, Satisfaction, Delight, Loyalty and Lifestyle.

The design focus presents a slight extension of FUVUX, reflecting relevant outcome for the consumer, as a component of perceived quality.

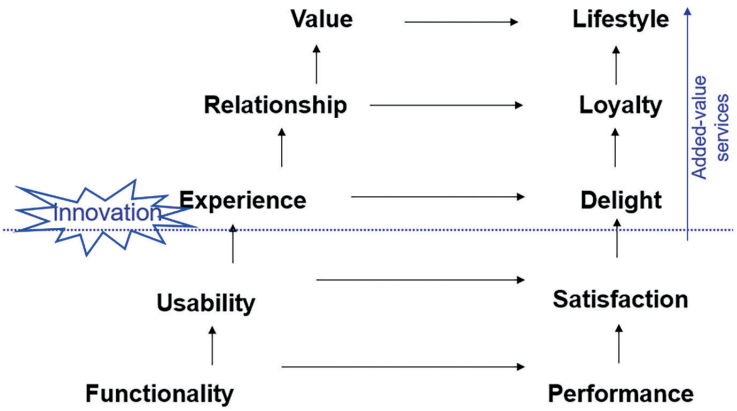

Figure 1.5. The layer development model for online services (adapted from Sikorski, 2008)

In a gradual evolution of online services an UX-related innovation is a breaking point, opening a gateway to added-value services, which are able to convert consumer's delight to a permanent loyalty. Furthermore, if frequent interactions create a value-based relationship, an online service gets integrated with a customer's lifestyle. Added-value services are the lucrative part of online services because they are frequently used as a part of a specific lifestyle, they often improve quality of life or deliver a specific value (like safety, in online insurances) and generate stable revenues from commissions or subscription fees.

1.3. The interplay between HCI and interaction design

Interaction design is the practice of designing interactive products and digital services. Interaction with user/consumer is an essential part of the product, enabling transfer of value between service vendor and user, and developing customer loyalty.

Interaction design methods largely originate from Human-Computer Interaction (HCI) as a research discipline which created basic terminology, design models and evaluation methods. Most importantly, HCI proposed methods for involving users into design activities, as the most-cost-effective way for early detecting of design flaws.

HCI encompasses the key term "usability" as a subjective measure, situated in a local "context of use" composed of the triad: user-tasks-environment (Figure 1.6). As a result, usability of an interactive system must be planned for a combination of factors creating a typical set of conditions in which users will be executing their tasks.

While HCI builds largely on a combination of cognitive psychology and software engineering, interaction design incorporates practice-oriented disciplines such as visual design, information architecture, user research, prototyping and consumer research.

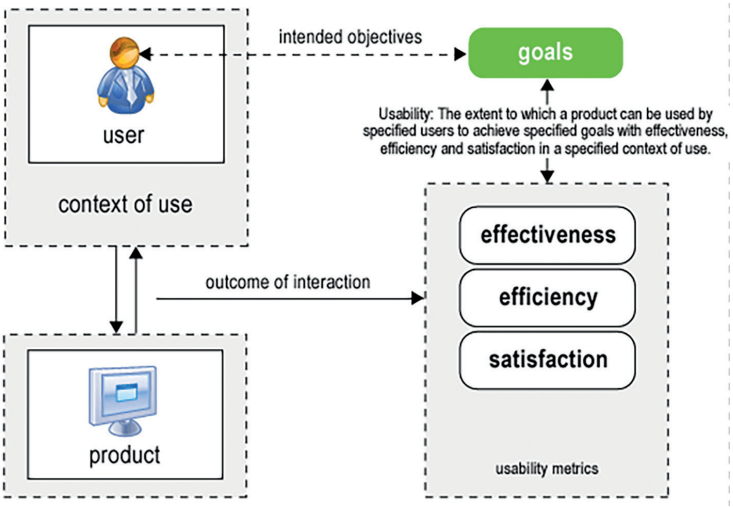

Figure 1.6. The usability framework relevant to ISO 9241-11
(Credits: http://ui-designer.net/usability/usersgoals.htm)

Both HCI and interaction design strongly promote user-centred design, assuming that high quality and usability of interactive systems can be delivered largely by appropriate managerial actions rather than by merely refining desired quality characteristic of a product. This implies frequent contacts between users/consumers and the design team, frequent testing and evaluation and making design decisions considering feedback received from prospective users/customers.

HCI research has introduced novel methods and techniques, nowadays widely adopted for interaction design, such as

- context of use analysis for understanding users' needs;
- persona for describing a profile of a typical user;
- tasks scenarios and use cases for identifying prospective users' activities;
- screen templates, wireframes and design patterns for providing consistency in visual design;
- storyboards and user flow for planning user navigation though a prospective system;
- different types of prototypes for evaluating validity of design concepts and collecting user feedback.

HCI and interaction design have many overlapping topics, combining a research perspective with a practical viewpoint, useful for designers, developers and

managers working in IT projects. In this approach all activities are user-centred, what means that user satisfaction is placed as a vital factor for defining quality of each interactive system. Furthermore, user involvement in an IT project is a key method for delivering high usability and UX for prospective users and customers.

1.4. IT products and IT projects

Quality assurance and quality management in contemporary IT projects are generally based on two approaches:

- product-oriented approach: precise specification of required quality characteristics, emphasis on control, measurements and verification of specified requirements through regular inspections and testing in the specific points of product development;
- process-oriented approach: involving users to collaborate with designers throughout the project lifecycle and delivering quality product by team-based iterative design, testing and evaluation, performed with real users.

Quality of interaction is a natural part of interactive product quality. Because there are no theoretical models which could be used for predicting prospective reactions of users, frequent testing and evaluation are vital points of user-centred design activities.

Informed managerial decisions in IT projects, as to how often and in what roles users should be involved, is even more important for delivering usability in the final product. Suboptimal usability of an interactive product is more often caused by the lack or inadequate user involvement than faulty specification of required quality/usability characteristics.

For instance, required characteristics of a user interface can be (more or less correctly) specified as visual clarity, consistency, aesthetics, user guidance, user control and compliance with standards or design patterns. However, only after these characteristics are conveyed on specific design activities and validation checkpoints, they can be implemented and tested by prospective users.

Again, this is the project management responsibility to create conditions that predefined quality characteristics be actually delivered. Project managers thus need to orchestrate the team members and their skills, organization, cooperation and coordination within the team, working with users in the project, communicating with project stakeholders, and providing assurance that all important requirements were appropriately handled during the project. No need to mention that solving various problems arising during the project without compromising product quality is a challenge, especially when project time and costs are fixed.

Therefore, a skilful interaction design in an IT project involves a lot of evangelizing, negotiating, communication - from systematic process of casting users into

appropriate roles, through design, testing and collecting user feedback. Usually prospective users are not available on-site, so they need to be represented by someone who can communicate their needs to the design team. Whether it would be a delegated customer representative, or a team member employed as an UX specialist, UX manager or an UX consultant – each choice may seriously affect the quality and usability of a final product.

Undoubtfully, involving users in appropriate stages of design is still the most cost-effective way of providing usability and UX, and it requires user-centred managerial viewpoint from the very beginning of an IT project. This process however starts from specifying the product concept, its design goals and guidelines for designing specific user interfaces. They will be presented in subsequent chapters, for GUI, web, and mobile user interfaces, respectively.

2. Graphical user interfaces

2.1. Specific features of graphical user interfaces

Early computers, since 1960s present in business, administrative and military applications, were mainly intended to perform numerical calculations. Then computers displayed only alphanumeric characters and commands needed to be typed in from the keyboard by a system operator. Such interaction mode – the only one then available – was called the Command Line Interface (CLI).

At that time for software designers, system performance and reliability were the top priorities. Ease of use or software usability were not required by computer owners, because systems were operated only by trained specialists employed as console operators or programmers.

Since the 1990s, personal computers (PCs) became available also beyond the professional work domain, and new application areas soon emerged. Computer games, education, and creative activities were supported with multimedia and communication capabilities, clearly indicating that computers can also be useful in work-unrelated activities. The users were no longer trained operators; thus, software applications had to get easier to use.

In the meantime, in available software applications their user interfaces were much improved with novel windows-based visual environments. Subsequent developments in graphical user interfaces have shown that popular software applications can be much easier to use. They can also be the source of user satisfaction, engagement, and positive motivation to use computers also for domestic use. Subsequently, with growing market demands, computer manufacturers initiated systematic research to improve software usability. Further progress in this area resulted in developing a novel concept of user-system interaction, namely **Graphical User Interface (GUI)**.

The concept of GUI was based on several principles that enabled users with easier operation of software applications (Olsen, 2003):

1. The whole surface of the screen was used for the user interface. Graphical objects located on the screen (icons, buttons, menus etc.) could be pointed and clicked by a user operating a mouse, joystick or graphical tablet.
2. The graphical objects (like icons) were designed in a way resembling actual objects users knew from real life, and used across the whole application in a consistent manner.
3. User's task could be completed step-by-step in an interactive mode, with possibility of customization individual paths according to user's preferences.
4. After users learned how to operate a specific software application, newly acquired skills could be easily reused in another application in a similar manner.

The success of GUI was crafted by a handy combination of novel components described as a WIMP – Windows, Icons, Menu and Pointer (Figure 2.1):

- Windows: a framed fragment of the screen surface, which may contain an editable workspace, message box, form to fill in or any other content. The windows allowed viewing an object from different perspectives, switching among multiple documents and could be opened, scrolled, stretched, overlapped, closed, and moved around the screen using the mouse.
- Icons: clickable graphical objects that open when clicked on, such as symbols, buttons, labels or widgets, representing applications, objects, tools, and commands (like starting a program).
- Menu: an ordered, logical composition of available options (labels, icons, images etc.) that could be scrolled through, selected and activated after clicked on by the user. In GUI there can be various types of menu, like drop-down, vertical, horizontal, and toolbars or palettes.
- Pointer: a distinct symbol on the screen (the cursor), changing current position with the mouse's movements and enabling specific operation on objects that were pointed and clicked on by the user. in GUI the pointer is a mouse-controlled point of entry to the windows, menus, and icons on the screen.

For making the GUI-based system operational and intuitive, **visual metaphor** is essential. A metaphor (Figure 2.2) is a characteristic element of GUI, serving as a visual scenery of the user interface. A metaphor is usually built on an analogy, making use of objects already familiar to the user from real life, like a calendar, clock, pen, dustbin or shopping cart. If the screen objects can be used in a way similar to analogical objects know from real life, users learn the system operation faster and find it more intuitive. A visual metaphor, apart from its aesthetical value, suggests the "mechanics" of system operation, including the look, layout and behaviour of objects, as well as constraints in moving objects across the screen.

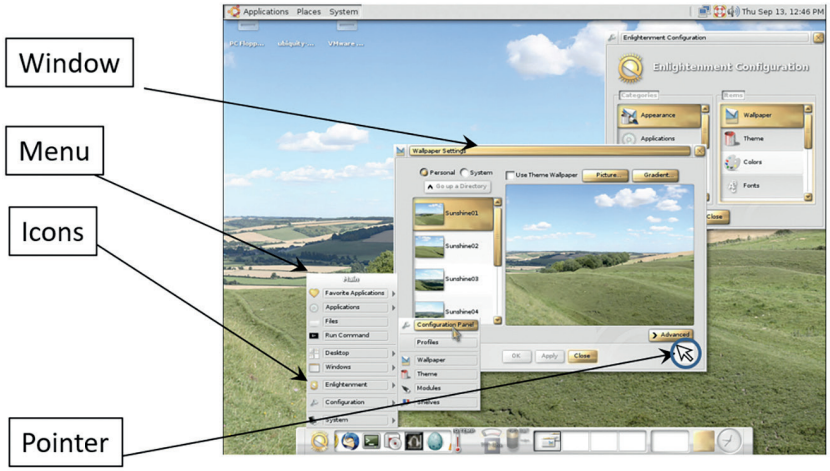

Figure 2.1. WIMP elements of Graphical User Interface (GUI)
(Credits: https://commons.wikimedia.org/)

Figure 2.2. A GUI-based metaphor of a weather station
(Credits: https://freepik.com)

A metaphor is only an illusion of workspace on a screen surface, but during the system operation it is a significant cognitive support for the user. Visual metaphors in GUI should refer to actual environment where the user's activity takes place, for instance an office, store, library, open terrain, etc. In engineering software applications, the metaphors often resemble the look of electronic equipment, machinery or

measurement tools, while in computer games the metaphor is usually the scenery where the action takes place, like a battlefield or construction site.

The GUI concept, combining WIMP with a visual metaphor, enables software users to perform intuitive, direct operations on screen objects, such as drag-and-drop, scroll, or resize. Moreover, with GUI non-vocational users were finally able to easily operate complex software with the mouse, and to explore using computers beyond their professional domains.

However, in software engineering projects developing user-friendly software requires an easy-to-use GUI to be developed in a procedural, repetitive manner. As a result, software designers and developers needed to acquire new design skills while GUI design guidelines gradually were getting available. Initially, software manufacturers built their GUI solutions by copying successful patterns and adopting best practices. Subsequently, industry- and academia-based systematic research produced design guidelines for newly emerging GUI solutions.

2.2. Design guidelines for GUI

The most universal, classical principles for designing human-computer interaction, known as usability heuristics, were developed by Jacob Nielsen (1993) as a result of extensive user-based research on usability shaping factors for then-existing computer systems. The Nielsen's usability heuristics are as follows:

H1. Visibility of system status. Keep users informed of system status with constant feedback.
The system should provide the user with regular feedback on what is currently going on by confirmation of actions, progress bar, a clock/hourglass, a text message, etc.

H2. Match between system and the real world. Set information in a logical, natural order.
The system should use simple and natural dialogue (language, phrases, concepts, symbols familiar to the user) in a natural and logical order of information appearance. Whenever possible, visual analogies (metaphors) should be used based on easily understood objects and operations.

H3. User control and freedom. Ensure users can easily undo/redo actions.
The system should guide the user through consecutive steps of the task. The functions "undo", "repeat" and "cancel" should be available for each operation. The user should be able to pause the task at any time and to continue from the same point after a break.

H4. Consistency and standards. Maintain consistent standards, so users know what to do next without having to learn new toolsets.
The system should use the same words, symbols, situations, and actions in the same way across all activities. Common standards and habits should be

respected, using symbols, operations and patterns which users already know from other systems.

H5. Error prevention. Prevent errors if possible; wherever not, warn users before they commit to actions.
Situations prone to human errors should be detected early and typical errors possibly avoided with a variety of aids such as checking spelling, grammar, correctness of names, validity of data format etc. User's work should be saved, protected and recoverable in a controllable, well-understood and trusted procedure, known to the user when the work starts.

H6. Recognition rather than recall. Do not make users remember information; keep options, etc. visible.
Minimize 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. Aids for minimizing user's cognitive load should be used, like patters, galleries, autocorrection of word phrases etc. All available navigation options should be visible at once (no hidden elements) and dialogue windows should not include elements that are unnecessary or very rarely used.

H7. Flexibility and efficiency of use. Make systems flexible so novices and experts can choose to do more or less on them.
Allow users to tailor frequent actions for both inexperienced and experienced users, who should be allowed to complete their actions in multiple ways. Provide intuitive keyboard shortcuts, quick access to recently used objects, frequently used tools, or other "favourite" components. Automating routine tasks should be provided by saving user profiles, preferences, data, history, or by self-development of macros for tedious, multi-step operations.

H8. Aesthetic and minimalist design. Design with aesthetics and minimalism in mind – do not clutter with unnecessary items.
Screen design free from unnecessary objects will allow the user to focus on the task and finish it faster. Elegant design is often based on minimalistic, but carefully orchestrated composition of visual elements.

H9. Help users recognize, diagnose, and recover from errors. Provide plain-language error messages to pinpoint problems and likely solutions.
Error messages should be expressed in a plain language, in a friendly tone, suggesting a constructive action or other solution, like using links to on-line help and support. Full reversibility of actions should be provided – usually it is users' preferred way to return and continue work after an error. Possibility for recovering from errors not only improves tasks performance, but it builds user's mood and confidence to the system, affecting general UX.

H10. Help and documentation. Offer easy-to-search troubleshooting resources, if needed.
Even though it is better if the system can be used without documentation, usually it will be necessary to provide some form of help or support for the users. Various forms of support should be available, like on-screen hints, message windows, direct prompts, or access to online chats staffed by a competent agent.

Since then, Nielsen's usability heuristics have been used as universal guidelines for interaction design, and they also serve as criteria for evaluating usability of interactive products. Figure 2.3 presents a symbolic, lightweight explanation of Nielsen's usability heuristics.

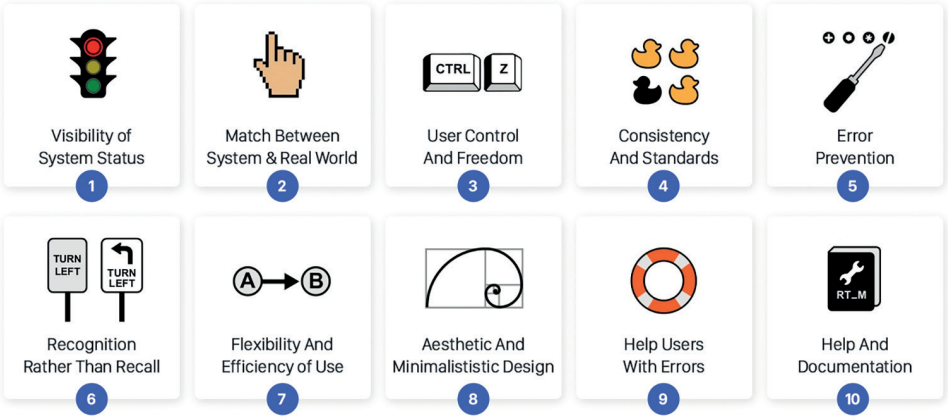

Figure 2.3. A symbolic explanation of Nielsen's usability heuristics
(Credits: https://uxdesign.cc)

In the HCI literature there are also many other design guidelines presented by other authors, who expanded the original scope of Nielsen's usability heuristics. Among many, following design recommendations based on Jacobsen and Meyer (2019), Malewicz and Malewicz (2019); Hartson and Pyla (2012), Norman, (1999); Tognazzini, (1995); Shneiderman et al. (2017), Dix (2004) and Krug (2005), deserve a particular attention.

1. **Visibility of key objects**
   - All important functions and object should be visible to the user at once and there should be a clearly visible on-screen prompt on how to get started.
   - Navigation controls should always be visible and should not disappear from the screen.
   - The structure of the whole system should be transparent, with a visible division into distinctive sections or modules.

2. **Mapping objects to actions**
   - An accurate translation of user intentions into the look and behaviour of user interface elements is necessary for a smooth interaction.
   - Mapping user's expectations into system behaviour should be as natural as possible by using easy understandable visual analogies, symbols, and labelling.
3. **User guidance**
   - The user should know in advance how many steps need to be performed and what data may be needed.
   - Steps and operations should be guided step-by-step, supported by on-line validation and reversible "undo-redo" options.
4. **Intuitiveness**
   - The system should advance and "put forward" objects and data the user needs in subsequent steps, making impression of "reading user's mind" during the system operation.
   - In order to pre-program these actions and behaviours, users' work with the system should be observed and actual task-related activities accordingly reflected in user's workflow.
5. **User's autonomy**
   - Users should be able to perform their tasks in multiple ways, depending on available skills, time, constraints, environment, or other local conditions.
   - Explorable interfaces should be provided not only for "open" tasks, but for all situations where users are encouraged to self-learning, gaining news skills or getting familiar with other areas of work process, beyond their local, own routines.
6. **Support for the visually impaired users**
   - All visual aids should complement each other: scaling the size of letters and graphics, the ability to zoom the workspace, change colours or contrast, hiding graphics, magnifying selected objects, etc.
   - Additional aids should be available for improving the visibility of objects in varying lighting conditions (sunlight, darkness), what would provide smooth interaction by reducing a chance for human errors.
7. **Consistency**
   - Consistent user interfaces decrease error rate, reduce users' annoyance, and minimize the load on human memory and mental effort.
   - For providing consistency, appropriate GUI styleguides, patterns and templates provided by the manufacturers of software platforms and environments should be followed whenever possible.

The above general design guidelines are supported by hundreds of tips regarding screen design, icons, menus, dialogue boxes, navigation planning, and other interaction design details, easily available in interaction design handbooks

(e.g. Sharp et al., 2019; Malewicz and Malewicz, 2019; Galitz 2013) and in numerous sources online, for instance:

- User Interface Design Handbook: https://designcode.io/ui-design-handbook;
- Usability.gov: https://www.usability.gov/what-and-why/user-interface-design.html;
- Interaction Design Foundation: https://www.interaction-design.org/ebook.

2.3. Standardization for consistency

Among all usability guidelines, achieving user interface consistency is definitely the critical factor for reducing users' effort in learning and operating any interactive system. Standardization and reuse of user interface elements are the main methods for achieving user interface consistency (Sharp et al., 2019; Shneiderman et al., 2017). Following aspects of standardization for consistency are especially important: using screen templates, user interface styleguides, and "de facto" standards, shortly described below.

Screen templates and wireframes

User interface consistency can be easily achieved by unification of visual elements and by designing a user interface on a series or standardized layouts where location, look and behaviour of control elements (buttons, windows etc.) is similar across the whole system. Such templates, often called "wireframes", form a reusable skeleton for all screens, and designate specific areas where users can find various types of information or objects (Figure 2.4).

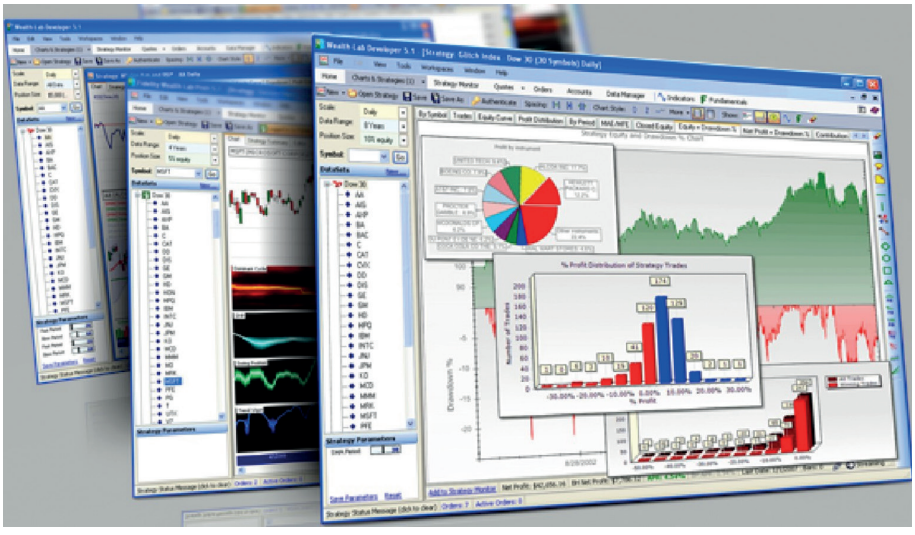

Figure 2.4. Template-based screen consistency
(Credits: https://www.pngwing.com)

Styleguides

With years of expertise in development of IT systems, software companies attempt to standardize user interface elements, so user interface consistency could be provided at low cost. The styleguides are documents that present recommended patterns for typical design issues like appearance, location and behaviour of user interface components (controls or widgets) such as windows, buttons, toolbars, dialog boxes, menus, navigation tools and other visual elements.

Promoting the use of styleguides to designers assumes that an easy-to-use GUI can be achieved not by reinventing but by using consistent graphical layouts across the whole system and by reuse of standardized elements the user are already familiar with.

User interface styleguides are available from software companies such as (Figure 2.5):

- Apple: https://developer.apple.com/design/human-interface-guidelines/;
- Microsoft: https://docs.microsoft.com/en-us/windows/win32/appuistart/-user-interface-principles;
- Google: https://developers.google.com/style/;
- SAP: https://experience.sap.com/guidelines/.

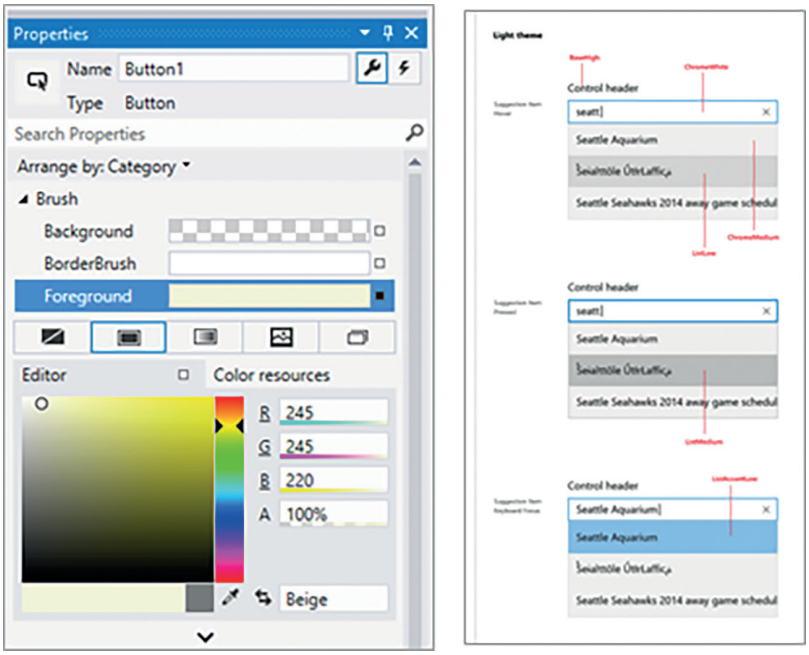

Figure 2.5. An excerpt from the Microsoft Windows styleguide (Credits: https://docs.microsoft.com/en-us/windows/uwp/)

“De facto” standards

The term “de facto” standards describe those user interface elements which became well-known to users and are widely used in software applications, but they had not gone through any formal standardization process. “De facto” standards include examples such as: a hover panel showing folders the left-hand part of the screen and their contents in the right-hand panel, a ribbon menu, a +/- zoom button, or local menu hidden under right-button of a mouse.

In user interface design, including “de facto” standards makes systems easier to use and provides the users with familiar experience already known from other applications. Many “de facto” standards which turned to be successful and popular usually get included into styleguides and they eventually become recommended design patterns.

2.4. Evaluation of user interfaces

There are three main types of user interface evaluation for IT projects: heuristic evaluation, checklist-based inspection, and user-based usability testing.

Heuristic evaluation

Heuristic evaluation (Nielsen and Molich 1990) is performed by an external expert - interaction designer, user interface or UX specialist. This method is aimed to assess the degree of compliance of specific user interface (IT product) with regard to Nielsen’s usability heuristics, presented in section 2.2. In heuristic evaluation an evaluator needs to assess to what extend each usability heuristic is satisfied in an evaluated system. An expert is interpreting each aspect of the user interface against each heuristic with relevance to specific tasks performed by users.

Heuristic evaluation obviously is a subjective process, but at the same time it opens room for discovering new opportunities for improving the product beyond existing usability problems. For instance, in addition to evaluating the user interface, heuristic evaluation can also refer to other aspects of the product, like factors affecting consumer trust or other business aspects. Practical guides to conducting heuristic evaluation were provided by Nielsen (1994) and Wong (2020).

Checklist-based inspection

Checklist-based inspection is usually conducted by a qualified specialist or tester, working for a specific IT development team. Evaluation process is based on systematic inspection of software product (GUI), following all items (questions) included a specific evaluation checklist.

Evaluation is usually performed by marking a score on a specific scale, expressing several possible grades of conformance with a specific question. Many

checklists include also open questions of blank fields for adding comments or notes relevant to items being evaluated.

Contrary to heuristic evaluation, a checklist-based evaluation is less prone to subjectivity, less flexible and not aimed at discovering new problems beyond the scope of the checklist. After evaluation is completed, the coverage percentage ratio is calculated to produce reports and graphs showing the current state of product usability and a list of required corrections.

There are many checklists available for evaluating user interfaces, accessible both from the classical HCI literature and from on-line sources such as:

- https://www.stickyminds.com/sites/default/files/article/file/2014/GraphicalUITestingChecklist.pdf;
- https://www.methodsandtools.com/archive/archive.php?id=37;
- https://lvivity.com/checklist-for-ui-testing.

User-based usability evaluation

Usability evaluation is a type of user-based testing, performed with a sample of prospective users invited to test a specific software product during the **usability testing** session. Users perform a series of pre-specified tasks with a given software product, or a prototype. Their actions are usually video recorded in order to collect quantitative and qualitative data useful in detecting where errors occur, and which user interface elements make the software product difficult to understand and operate.

After the testing, users fill in a questionnaire with questions about their satisfaction and product usability and share their opinions in an interview. Usability evaluation with users my be supplemented with expert reviews or with laboratory experiments.

The above evaluation methods are applicable for the GUI, but also for web and mobile user interfaces, after some modifications, presented in Chapters 3 and 4.

2.5. Trends and innovations

Towards a more natural interaction

Despite GUI was a great step forward from the CLI (Command Line Interface), there is still an ongoing search for even more intuitive interaction techniques, called NUI (Natural User Interfaces). The main limitation of GUI is necessity to use devices intermediating between the system and user, such as a keyboard, mouse of joystick. No wonder that novel interaction techniques are continuously developed, attempting to expand the concept of GUI towards more natural interaction NUI – Natural User Interfaces (Figure 2.6).

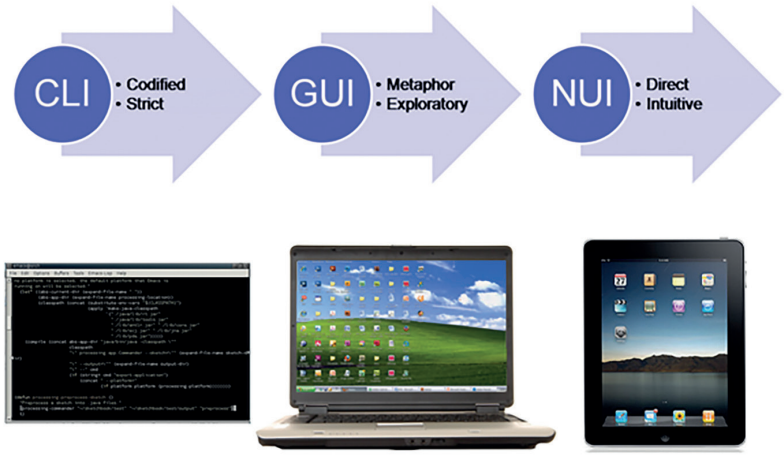

Figure 2.6. The concepts of CLI, GUI and NUI
(Credits: https://interaction-design.org)

The NUI concept aims at building user-system interaction as reflecting natural human communication activities, such as speech, gaze, touch or gestures. In NUI, now-dominating intermediary devices such as mouse or keyboard could be eliminated, and user-system interaction would much resemble a direct human-human communication.

Although the concept of NUI is still largely in a laboratory phase, some interaction techniques presented below open interesting opportunities towards NUI. At least, they are significantly expanding GUI and improving User Experience. For instance, voice interfaces, or Augmented Reality (AR) do not require complex computing and can much improve UX, so they are getting quite popular in many application areas.

Despite of recent developments in user-system interaction methods, now dominating GUI is not likely to be rapidly replaced by NUI in the near future. Instead, a gradual evolution towards natural user interfaces (NUI) will rather take place, based on NUI-like extensions, such as speech recognition, gestures and touch control, and new types of input devices. This evolution is already taking place, supported by novel trends and innovative forms of interaction, shortly presented below.

On-screen direct manipulation

Direct manipulation is an interaction technique allowing users make direct changes to on-screen objects without using dialog boxes. This method is usually executed by right-clicking the mouse over an object, highlighting the object, and next changing its geometrical parameters (shape, size, location, etc.) directly by dragging or rotating an object with the mouse or by selecting additional options from the context menu (e.g. changing the colour). A popular form of direct manip-

ulation is for example the “drag and drop” or other similar mouse-controlled operations performed directly on geometric objects (Figure 2.7).

Direct manipulation was the first extension of GUI towards more natural interaction and gained high adoption in engineering, creative, or gaming applications. Direct manipulation produces an immediate effect, it is pleasant to use, easy to learn and encourages experimentation, thus making it a method very much liked by all categories of users. Despite obvious advantages, applying direct manipulation may be problematic to objects of operations which do not have a direct visual representation.

Nevertheless, although used mostly with mouse, direct manipulation has become even more popular with the spread of touchscreens-equipped devices like tablets or smartphones.

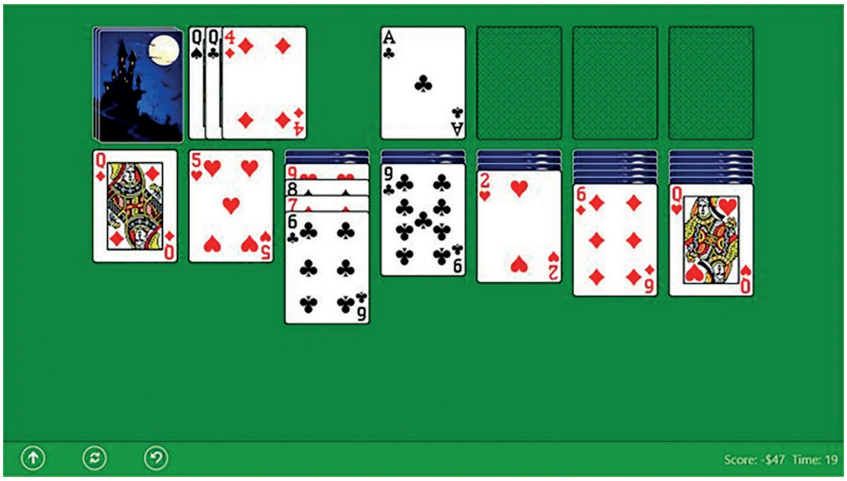

Figure 2.7. Direct manipulation in a solitaire card-game
(Credits: https://www.microsoft.com)

Touch and haptic user interfaces

The touch screen interface is an interface allowing users to interact with software simply by touching screen objects by a finger. It has become pleasant and engaging and because users no longer have to use buttons or a mouse to hover over the GUI elements.

Although for some time touch screens no longer seem to be a novelty, they are an important step toward more natural interfaces, simplifying manual operation on handheld devices. Small touch screen devices are appropriate for tasks involving manual selection and pointing, but they are not much suitable for precise operations such as drawing, movement, operations on objects, or for entering a great amount of data (such as writing text). For this reason, apart from mobile devices, touch screens are often used for situations where space is limited (e.g. cars, ma-

chines), and for operations which do not require extensive data entry or high-precision movements.

Haptic user interfaces also use touch for operating a software app or device, but they can also transmit force and vibrations from and to user. For this reason, haptic interfaces are often used for remote control of devices, or in computer simulators where manual operations (such as assembly, repair, surgery, etc.) are the subject of training.

Voice interfaces

Voice interfaces, often available as an add-on to GUI, open for users the possibility for operating software (or devices) with voice commands. Although yet far from perfect, voice interfaces have undergone a big progress in recent years. Current technology for generating output voice messages for the user (one way communication) is quite well developed, synthetically generated messages are useful as an additional channel to inform users, but for the blind or manually disabled users but they are the main form of interaction.

Voice interfaces based on human speech recognition include chatbots, call routing, speech to text, and handsfree control of computer and mobile operations. Speech recognition is one of the most common examples of a natural modalities being already implemented.

However, in a two-way communication voice interfaces still face problems with correct understanding if users speak with dialect, with errors or in a noisy surrounding. Users often complain that generated voice responses are still too “robotic”, and significantly less realistic than those between humans conversating with emotions, tone, and sentiment.

Virtual Reality

Virtual reality (VR) Interfaces apply computer-generated visualizations that provide users with the illusion of participation within the artificially created environment. Images, animations, or videos are projected on a screen or inside user’s headset, providing a sense of presence (“immersion”) is a certain thematic environment. VR makes use of user’s natural gestures for moving objects or issuing control commands, but it requires uncomfortable equipment (headset, gloves, cables) to be attached to the body of a user. In addition to this, VR require cameras, motion sensors, supporting software and projection systems to be unstilled in user’s space for recognizing specific human gestures and then translating them into actions. Simple VR systems do not require expensive projection equipment, but an affordable headset, nevertheless numerous motion sensors are necessary for using gestures for manipulating an object (Figure 2.8).

A VR interface provides users with a new kind of experience and emotional engagement, enabling them to navigate in a 3D space and to interact with 3D objects,

and encouraging active participation in a specific action by creating the psychological state of “presence” or “immersion” within a certain reality.

Because VR interfaces are costly, so far, they have been used mostly in complex systems like training simulators or visualization environments for architects, automotive designers, and other construction-related professionals. Most recently, growing availability of low-cost VR solutions (headsets and software) opens new opportunities for broadening VR application areas also to personal therapeutic, educational or entertainment purposes.

Figure 2.8. Virtual reality in an engineering design application
(Credits: https://robodk.com)

Augmented Reality

Augmented reality (AR) enriches the real physical space display with images generated by a computer. Differently than in VR, an expensive headset is not required for AR, using special glasses instead. It is also possible to display the image directly on a screen, whether it is a computer or a handheld device, like a tablet. It is much simpler and cheaper, often even a low-cost solution.

For this reasons AR finds primary applications in maintenance and servicing of industrial machinery, in museums for expanding visitor’s experience when viewing historical objects and in gaming. In a coming future more advanced applications of AR are expected to revolutionize e-learning (conducting educational experiments), maintenance of industrial equipment (Figure 2.9) and occupational training for workers in robotized manufacturing plants.

Figure 2.9. Augmented reality in industrial applications
(Credits: https://i-scoop.eu)

Multimodal interfaces

Multimodal user interfaces use simultaneously many input and many output modalities (channels) for user-system interaction. It means that for instance voice, touch and gesture can be used in parallel, without necessity to switch the channels in a manual way. This offers the user a very natural interaction, high autonomy, and potentially hight tolerance to errors, which automatically are handled by the system, possibly using AI algorithms, too.

Because the system automatically recognizes user's commands and their modality, such systems are very complex from an engineering point of view. For instance, gesture recognition allows the controllers to have very precise accelerometers and gyroscopes for sensing the rotation, acceleration, and tilting. As a result, multimodal systems are very costly, and unfortunately yet not as reliable as required for industrial or military applications. For this reason, most of multimodal interfaces thus very costly, and yet not as reliable as required for industrial or military applications can be found in the area of entertainment and gaming, including Microsoft's Kinect that allows the gamers to interact through their gestures, body motions, and speech commands.

Brain-Computer Interface

The Brain-Computer Interfaces (BCI) may seem to be mysterious, as they are able to read user's neural signals and make use of them with adequate software that translates the neural signals into actions. Nevertheless, there are still many

challenges regarding user control and reliability of BCI-base systems, especially adequacy of intention recognitions, accuracy of control actions, or calibrating each system (device) to individual characteristics of a specific user.

Although BCI offers a huge potential, due to their complexity, BCI-based devices are still in a laboratory phase. However, in a not very distant future they may have many applications, particularly in the health sector. It may allow the paralyzed patients to operate their wheelchair or even an ergoskeleton-supported limbs merely by “the power of the thought”.

Future developments

In recent years user interfaces have been significantly expanded with innovative components, enriching user-system interaction. User interfaces have undergone following stages:

- CLI (Command Line Interface) – text-based interface based on user-typed commands, now used only in professional software applications;
- GUI (Graphical User Interface) – now dominating user interface, based on WIMP and direct manipulation;
- NUI (Natural User Interface) – a yet experimental user interface, aimed at enabling users to control computers with voice, gestures or body movements, without devices such as mouse or keyboard.

The ultimate goal of NUI is to create a smooth and seamless interaction between the user and machine, performed in the most natural way possibly resembling an actual human-human communication — it is as if the user interface does not even exist.

Unfortunately, NUI requires using many sensors in user’s environment and making use of complex optimization models for compensating human variability and imprecision. On the upsides, it is important that NUI it is not very difficult to learn for the novices, who step-by-step are getting more and more advanced. However, although NUI is a very attractive concept, due to its complexity its full implementation to everyday life is still very far from completed.

As a result, despite of recent developments is not likely that in the foreseeable future GUI would be replaced by NUI. Instead, GUI will be rather evolving by gradually with extensions such as embedded online content (like chatbots or online help systems) or connections to online collaboration platforms. Incremental changes make GUI gradually undergoing some sort of hybridization with online contents and integrated with web applications. With advancements in computer graphics, current GUI interfaces enable new forms of information presentation, access to multimedia (video with sound), and visual communication with other individuals. Moreover, GUI-based applications are essential for stimulating human creativity and exploration, which deepen the understanding of the task situation by the user.

However, a good GUI is still difficult to design and implement, and the complexity of interaction techniques requires regular user-based evaluations in an IT project. Moreover, in some tasks the user productivity in GUI paradoxically can be much lower than the in CLI-based systems due to the fact that manual mouse operations are often much slower than skilled typing.

Collaboration and communication require users should have access to a computer connected the internet or to a corporate network. As a result, a new networking computing paradigm emerged, based on user interaction with websites and with web applications, which offer functionality formerly not available in GUI-based software.

These changes gave rise to web-based and mobile user interfaces which will be presented in the next chapter.

3. Web user interfaces

3.1. Specific features of Web user interfaces

Since the 1990s businesses and PC users gained access to the Internet, which enabled them to explore online content, e-shops, and online services. For users Internet-related benefits were the consequence of two major factors:

- availability of web browsers enabling users to access the online content, although at first only “from the desk”, using a using a cable connection to the internet;
- availability of websites with online content, at first purely informative, then followed by e-commerce.

Currently, following types of web user interfaces are in use:

1. Informational websites. These websites primarily provide textual and visual information, and due to rather poor interactivity the user usually remains a passive consumer (reader) of the content.
2. Service websites. These websites primarily provide specific services, like making a reservation, buying an item, or travel planning, and they require some activities to be performed by the user. All e-commerce, e-business, and e-government services available usually fall into this category.
3. Web applications: Known also as internet applications, they perform specific functions like calculations, bookings, payments, etc. They have similar appearance to typical software applications, but they are developed as typical components to be embedded as a part of a specific website. In order to improve task performance and UX, some web applications may be standalone applications (without a browser menu), often installed as a “thin client” in a local system.

Comparing to GUI, the design of web user interfaces has a broader scope because a website is expected to provide not only usability (task-related), but also pleasant emotions for the user (positive UX). Moreover, due to specificity of

web-based interaction, websites and web services can be accessed by users from a global market, so they need to be designed for an international audience.

While the GUI is strongly tied to the paradigm of work-related desktop computing, web users expect enjoyment and pleasant experiences, apart from task performance. With the emergence of various e-business websites, per analogy to market situations from real life, users gradually accepted their position as online consumers who have quality-related requirements. Subsequently, web designers and website owners had to put usability as precondition for smooth completing self-service transactions, and the positive UX as a second precondition that consumers would return and shop again.

Web-based interaction introduced new user interface elements previously not present in GUI-based systems:

- attractive content (text, images, and video) capturing attention of users;
- intensive use of graphics for creating a unique website style and atmosphere;
- cursor turning into a “hand” over a hyperlink on a webpage;
- free navigation through the website structure, supported by a variety of guidance aids such as a site map, or marking places already visited links;
- novel functions such as search window, shopping basket, forward-backward buttons, access to history, favourites, etc.

As a result, the web-based interaction is more intensive, engaging, and enabling users to feel pleasure in online explorations, which were not available in GUI-based systems. As web design was progressing, it soon became clear that not a web browser, but the content of each web page, its usability and UX, largely constitute UX for online customers. Furthermore, web design had to follow changing trends and latest developments in internet technologies (Figure 3.1).

For websites and web-based interaction, main design objectives can be specified as follows:

1. Providing adequate functionality, understandability and ease of use (usability).
2. Providing attractive, engaging content with good readability and accessibility.
3. Delivering positive UX for memorable emotional impressions for the users.
4. Developing valuable business relationships with users-customers.
5. Providing respectful interaction, customer privacy and security protection.

The prevalence of web user interfaces introduced a new computing paradigm: free access to rich online content, using computers for services and enjoyment, and turning users into online consumers. Subsequently, design guidelines for web user interfaces needed to be developed so the users to their best could exploit unlimited opportunities for exploring the web.

Figure 3.1. Changing trends in web design (2004–2008–2015–2020)
(Credits: https://bbc.co.uk)

3.2. Design guidelines for Web user interfaces

Guidelines for Web user interface design largely originate from design recommendations relevant to GUI. For However, for web design classical Nielsen's usability heuristics had to be considerably adapted to specific characteristics of web-based interactions (Nielsen 2000, Nielsen and Loranger 2006), as follows:

H1. Visibility of system status. Keep users informed of system status with constant feedback.

The system (the website) should always provide information about what is currently happening in the system through confirmations and messages. For instance, the user should be informed with a progress bar when the data are still in transfer and how long it may take. Also, user's current location in the structure of the website should be clearly marked.

H2. Match between system and the real world. Set information in a logical, natural order.

The system should inform the user with a plain language, with words, phrases and graphics familiar to the user. Information should appear in a natural and logical order, especially when filing in online forms or checking out to the payment. Language versions should be provided, or at least translations of

important content, instructions, or labels. Navigation tools, maps, and location plans should reflect the actual layout of objects.

H3. User control and freedom. Ensure users can easily undo/redo actions.
The system should guide the user through subsequent steps of the task. Functions like “cancel”, “back”, “forward” “undo” should be available for each operation. Variable navigation aids should be provided: navigation path (“breadcrumbs”), site maps, easy return to the home page and clear, hierarchical view of objects. Additionally, there should be provided an easy correction of errors in online forms, preview the actual appearance of objects.

H4. Consistency and standards. Maintain consistent standards, so users know what to do next without having to learn new toolsets.
The system should use consistent page layouts, and permanent positioning of clickable objects such as logo, headers, menus etc., which should operate in the way known to users from other systems. Names and labels should be short and communicative, following popular conventions. Acronyms and abbreviations usually are not obvious for a larger audience, so they should not be used.

H5. Error prevention. Prevent errors if possible; wherever not, warn users before they commit to actions.
Interaction, layout and content of the website should be designed in a way minimizing possibility for human error and with opportunity of easy correction, like pre-set default data values, automatic spell checking, autocompletion, and forms which do not lose data when “back” button is clicked. Error-preventing visual aids for improving visibility should be always available, such as font enlargement, appropriate scaling of webpage (including online forms), and clear preview of data/documents to be submitted.

H6. Recognition rather than recall. Do not make users remember information; keep all available options visible.
Minimizing the user’s memory load should be achieved by making all objects, actions, and options clearly visible, as well as by simple, minimalistic design. All navigation options should be visible at once, clear marking of visited vs. unvisited links. For a temporary storage of items, a clipboard, wish list or a handy personal repository should be available.

H7. Flexibility and efficiency of use. Make systems flexible so novices and experts can choose to do more or less on them.
The website should accelerate the user in reaching the goal by multiple shortcuts, aids like “skip intro”, “jump to” and a local search engine complementary to hierarchy menu. Also, a preview of recently selected items, autocompleting data or phrases, and always visible list of most commonly used options (frequently purchase items, for example) should be available.

H8. Aesthetic and minimalistic design. Design with aesthetics and minimalism in mind – do not clutter with unnecessary items.

In website design visual elegance should be achieved through simplicity and minimalism: smart use of white space, adequate level of detail, clear layout of pages without any unnecessary ornamentation, providing transparent choices and careful using of only purposeful graphics.

H9. Help users recognize, diagnose, and recover from errors. Provide plain-language error messages to pinpoint problems and likely solutions.

In websites errors usually occur when users fill in online forms or select specific parameters from a dialogue box. Validation of user input data should take place yet in the data field, default formats of data, hints and tips should be proposed before the user makes a mistake and frequent confirmation messages for the websites. Also, for new or infrequent users/customers, a supportive and user-friendly system for login and password reminders will be necessary.

H10. Help and documentation. Offer easy-to-search troubleshooting resources, if needed.

Although websites are intended to be entirely self-service systems, users sometimes may need help in case of problems or questions. Adequate, complementary forms of support should be provided, like on-screen help, live chat, active phone line or answers to typical questions. For websites offering specific services or products, like travel or medicines, supplementary explanations, procedures, or instructions may be necessary.

In the HCI literature there are numerous design guidelines for web user interface (e.g. Beaird and George, 2014; Phyo, 2003; Pearrow, 2000), which highlight web-specific issues such as:

1. **Consistency of presentation and navigation**
   - Consistent navigation is essential for positive UX, but it strongly depends on specific screen layouts, and on placement of navigation tools.
   - Designing screen layouts with regular grid helps to maintain consistency in visualization and in navigation.
2. **Readability of text, images, and icons**
   - Texts structured into short paragraphs, with adequate level of detail and highlighted keywords. Only sans-serif fonts should be used, as they are easier to read on the screen.
   - Text should be adequately balanced with images and white space.
   - Icons and other graphical objects should clearly communicate their meaning.
3. **A variety of navigation tools**
   - Global and local navigation should be complementary, supported by site map and internal search window. For fast navigation traditional links (navy-magenta) are most intuitive, and hidden menus or other disappearing navigation elements should not be used.
   - Breadcrumbs (University > Faculty > Department > Staff > John Brown) can be used to help users in quick browsing across the website information hierarchy.

- A quick return to the homepage should be provided by a click on header or logo located in top-left corner of each subpage.

4. **Perfectionism in online forms**
   - Because online forms are a frequent source of users' frustration, they should be carefully designed and tested for possible human errors.
   - In designing online forms a logical and clear layout for data field should be used, with prompted default formats of the data, and real time validation of data in the field.
   - After the online form was submitted, a user should promptly receive a confirmation.
5. **Adequate use of graphics**
   - Moderate use of graphics is required for accomplishing a minimalistic, elegant design.
   - Adequate use of graphics and colour is essential for creating the website mood suitable to the brand image.
   - The use of videos and animation should be purposeful and minimal.
6. **Trust and confidence**
   - Multiple ways of creating confidence to website brand should be used, and not only in e-commerce websites.
   - Excellence and perfectionism in design and verified, engaging content are essential for building user's trust to the website and its brand.
   - Extensive contacts data, information about people behind the company and references to established institutions enhance online users' trust to the company and its brand.
   - Clear declarations about customer care and support, and visible information about security of payment process and protecting user privacy are also important trust-building elements.

In addition to the above guidelines, providing convenience and comfort for web users is also essential. Websites are gateways to services, so avoiding customer frustration and disappointment is essential to minimize the dropout of customers (Nielsen, 2000; Nielsen and Loranger, 2006). Amenities such as easy registration, password recovery, user guidance, storing the shopping history and the contents of unpaid shopping baskets, are simple but essential components making a website not only usable, but also service-oriented as to customer expectations.

3.3. User Experience factors

As presented in Chapter 1, a basic model of UX proposed by Hassenzahl (2008) specifies its two main components: the pragmatic (task-related) and hedon-

istic (pleasure-related) quality. For the web-based interaction this definition needs to be expanded, because:

- in activities carried out online there are many more factors contributing to UX than described by the Hassenzahl model;
- in e-business and other areas UX instances get cumulated in subsequent interactions, shaping individual attitude to system re-use and to the specific online brand (Hassenzahl and Tractinsky, 2006).

A "honeycomb" model of UX (Morville, 2004) is therefore much more suitable for web-related activities, and it specifies required UX-related features of an interactive product (Figure 3.2):

- valuable (the core of positive UX): does the website deliver value for end users?
- useful: is the website useful in practical situations?
- usable: is the website easy to use?
- desirable: is the website and its offer attractive?
- accessible: is the website accessible for users with disabilities and for any type of system/device users may have?
- credible: is the website trusted and believable?
- findable: is the website easy to find and is it easy to navigate?

According to the honeycomb model, for gaining positive cumulated UX, user's subjective assessment from all the above components should be clearly positive. Steadily positive UX, cumulated during some period of time, should result in revisiting the website and – if further interactions produces positive UX – in developing valuable business relationships and customer loyalty.

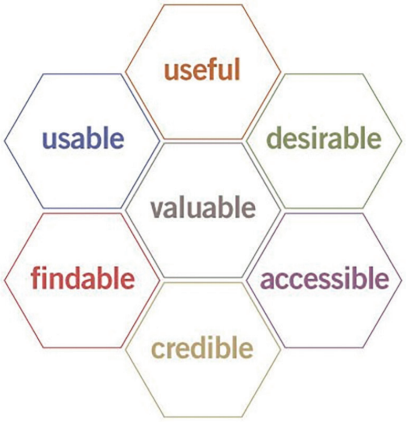

Figure 3.2. A "honeycomb" model of User Experience (Morville, 2004)

Selected UX-shaping factors, which are important in website user interfaces design, are presented below.

Predictability

Users expect each website would behave in a predictable way, with a consistent appearance and reliable feedback responses. Predictable behaviour is especially important in websites with a very complex structure, where conventional navigation is difficult for users. Instead, users often expect to use a local search engine, which – if effective – further reduces necessity to wade in complicated navigation structures. In such cases, after quick landing on the final product page, user has no further contact with website navigation. Return to the home page or skipping among the thematic sections should be made in an easy, predictable manner valid across the whole website. When users can faultlessly predict in what location to look for specific elements, subjectively perceived speed of task completion will contribute to increasing UX.

Speed

Users expect websites to work fast. Despite the speed is a subjective assessment, it is a very important UX-shaping factor. Apart from the time consumed for correcting errors, speed-related UX is the outcome of the subjectively perceived pace of data transfer. For this reason, optimizing web pages for fast data loading is crucial, especially graphics and scripts which are prone to causing delays.

Preventing user errors saves time needed for corrections, what also affects perceived time of task completion. To make an impression of a speedy operation, user‘s cursor movements should be smoothly guided by buttons with standard colours (red for alert, “Stop” or “Cancel” and green for “OK”, “Next” etc.), or by simply enlarging the size of default buttons such like “Next”, “OK”, “Accept” or “Submit”. The effect of perceived speed may be even better, if default buttons are green per analogy to “big green button” effect known from the copier machines, where brightly green button stands out from other buttons in the control panel. This simple trick makes finding and clicking an object faster and it shows how a *de facto* standard can be used for accelerating user’s activities and improving UX.

Navigation

Although in websites hierarchic structures of content are still present, users often navigate in a non-hierarchic manner, taking advantage of flexible and reversible navigation paths. Since local search engines are present in most websites, the significance of classical navigation tools (like hierarchic menus, site maps, „breadcrumbs”, etc.) has diminished. For optimizing navigation paths, web analytics is often used for tracking pathways most frequently attended by users. This knowledge can be successfully utilized for planning the layout of the website, to provide users with a quick access to most popular products and for building a smooth checkout process.

In website design a positive UX resulting from a smooth navigation, following elements are helpful: choosing screens templates for navigation (Figure 3.3) distinct marking of clickable elements, highlighting subsequent steps of a process, opening a new link in a new card, and placing on each subpage a clickable logo making an easy return to the home page.

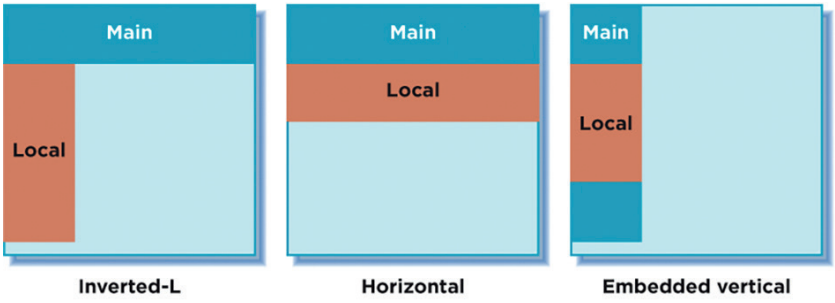

Figure 3.3. Typical navigation templates for web user interface
(Credits: https://www.oreilly.com)

User guidance

Contemporary websites usually have a complex structure, combining hierarchical information architecture with network-like user flows. Navigation is no longer simply top-down – it can be subject-by-subject, allowing user skip among different sections of the website. In websites for procedural tasks, like buying tickets, submitting reports, applying for funds etc., navigation paths should guide the user smoothly through subsequent steps towards intended task outcome. At the same time the user should feel a solid sense of control with reasonable amount of flexibility and freedom as needed. A clear indication what part of the service the user is currently in, strengthens the sense of control and facilitates easy return to the home page. For positive UX, it is important that users would be guided especially in more difficult parts of the procedure or in infrequently visited websites with lengthy procedural tasks.

Supporting exploration

For positive UX, especially regarding its hedonic component, providing liberty of exploration, pleasant astonishments and interesting discoveries are essential. Many websites are intended for open, exploratory tasks whose outcome can vary a lot, for instance searching for a memorable gift, or for a romantic holiday destination. Such websites should encourage users to free exploration of sections which would meet their potential interests. In traditional websites with a static content, advertisements and other visual suggestions should be directly prompting users to explore specific topics. In systems with a dynamically generated content, especially where recommender systems are used, a panel with clearly visible list of the

recommended objects (“Most popular”, “Bestsellers”, etc.) is often used to attract customers’ attention, to make them explore contents, and to stimulate cross-selling whenever possible.

3.4. Web accessibility and interoperability

Currently a large part of web users have various visual impairments, making for them difficult seeing some elements of website contents. In many countries web designers are obliged to provide adequate accessibility for visually impaired users, in accordance with W3C[1] guidelines, national standards and legal regulations. For instance, it is mandatory to validate the webpage code and modify it a way which guarantees accessibility for the visually impaired users, and to provide compatibility with “screen readers” used by blind users. Easy to operate control aids should be available for enlarging font and graphics, for setting the contrast, and for hiding graphics and animations as needed. Whenever possible, using new web technologies such as HTML5 and CSS3 is recommended.

Providing accessibility currently is not only a legal issue, but also an important UX-related factor for each “healthy” user who may not be able to improve text readability by increasing the font, contrast, or to rescale the webpage when needed. Such situations may occur for anyone, causing annoyance, frustration and resulting in negative UX. Therefore, even if not required by law, accessibility aids should be always provided such as:

- zooming, appropriate scaling, contrast adjustment;
- reducing dynamic content (animations, video);
- presenting important data in a simple, alphanumeric format;
- optimizing online forms for accessibility, jointly with the content.

Accessibility may be also limited when a website does not open quickly or loading images gets stuck due to a slow data transfer. Therefore, possibility of adjusting the website display for low-speed networks should be provided, especially when technological deficiencies at users’ side may be expected.

The technical aspect of accessibility, called interoperability (not yet required by law), describes providing users with access to websites’ functionality and contents no matter which browser and what device is used. Subsequently, all content available online should “technologically neutral” – accessible from standard platforms like Windows, iOS, Android, and diverse devices like PCs, laptops, tablets, smartphones, TV sets etc.

All aspects of accessibility can be successfully solved using the Responsive Web Design or ARIA technologies (Kearney et al., 2020) to make website automatically resize to any display type (desktops, tablets, and phones) and to any

[1] W3C – World Wide Web Consortium

orientation (vertical or horizontal). Automatic reformatting of the contents to display correctly goes at the cost of increased code complexity but it provides much enhanced UX for users.

3.5. Evaluation of web user interfaces

Similarly to GUI, for evaluating web user interfaces following methods are available:

- expert-based methods (reviews): heuristic evaluation and checklist-based inspections
- user-based evaluations: usability testing and satisfaction surveys.

Heuristic evaluation

Heuristic evaluation is aimed at evaluating conformance of a website to Nielsen's usability heuristics adequately adapted to specificity of web-based interaction. Expert evaluation of a Web interface can be performed using appropriately adapted heuristic evaluation (described in Section 2.4). Performed in a similar manner like for GUI, heuristic evaluation is subjective, flexible, and exploratory, but additionally it often includes references to similar or competitive websites.

Checklist-based inspection

Web site inspections are conducted using web usability checklists, which identify the coverage ratio for specified requirements. A typical web usability checklist has usually 30-50 items (questions), grouped into several thematic sections corresponding to the main criteria for the evaluation.

Web usability checklists are less subjective than heuristic evaluation and they allow at relatively quick identification of usability/accessibility deficiencies. A minor limitation the checklists have is that they do not suggest how to improve a website, either they do not address website's emotional impact on user (UX).

In the literature (e.g. Rubin and Chisnell, 2008; Wong, 2002; Dumas and Redish, 1999). there are many checklists available for evaluating Websites. They can be divided into two categories:

- usability checklists, for which a typical evaluation scope is website usability and user satisfaction;
- accessibility checklists aimed at evaluating website's compliance with requirements regarding visually impaired users.

In addition to manually-operated accessibility checklists there are also numerous software applications (e.g. ANDI, see Figure 3.4) for automated accessibility testing which validate the code for compliance with W3G guidelines. After analysing the website at specific URL address, a checker shows the fragments of code

which need to be reworked because they reduce content accessibility for users (Csontos and Heckl, 2020).

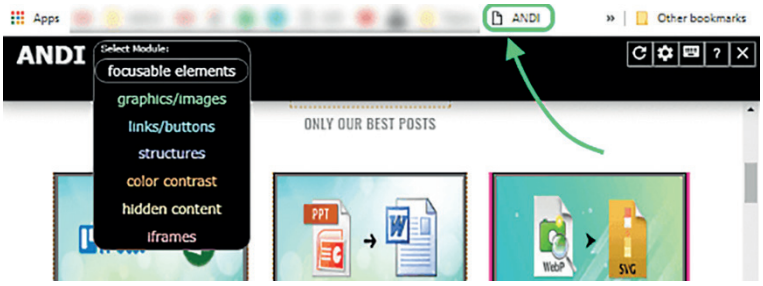

Figure 3.4. Web accessibility testing tools ANDI
(Credits: https://www.ssa.gov/accessibility/andi/help/install.html)

User-based evaluation and satisfaction surveys

Usability testing for websites can be performed with users in conventional manner in laboratory conditions. Because it is focused on collecting possibly accurate measurements with and adequate research apparatus, nowadays in agile IT projects low-cost usability testing methods are getting increasingly popular. They utilize less resources ("testing with 5 users" slogan), so cheap and quick in delivering practical recommendations instead of high precision results.

Remote usability testing describes the testing when users remain at home, like typically during internet shopping (Figure 3.5). Their actions are recorded using their computers, but the test tasks, interviews and surveys are managed from external locations by the test organizers. During the test, users perform specific tasks scenarios using the evaluated website, like buying specific products online.

Satisfaction surveys usually follow usability testing and can be performed as a paper questionnaire (only on-site), an online form or an audio-video interview. Direct interviews with users allow to discover also factors affecting emotional state (UX) of website users, which are otherwise difficult to be captured in conventional user satisfaction questionnaires.

Ad-hoc satisfaction ratings are used for rapid collecting user feedback on a popup window about the rate of user satisfaction just after completing a task. Ad-hoc satisfaction surveys contain only a few short questions, and they are frequently used during a regular operation of a website, and rarely during the design process. Ad-hoc collected data is rough and not sufficient for in-depth explanatory analysis, so users are usually prompted to open questions where they can explain their rating, if it was lower than the maximum value.

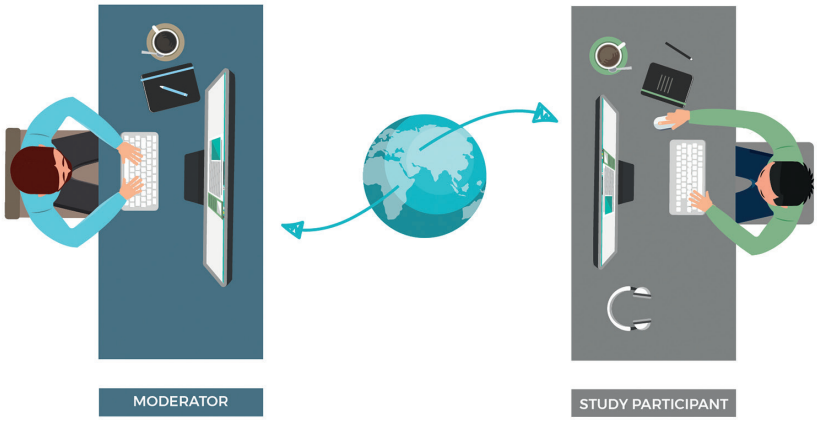

Figure 3.5. Remote usability testing
(Credits: https://www.packtpub.com)

3.6. Trends and innovations

Servicizing the Web

Servicizing (Rothenburg, 2007; Toffel, 2008) is a business transaction practice in which satisfying customer needs is achieved (1) by selling the specific functionality rather than the product itself, or (2) by increasing the service component of a product offer. The servicizing concept is based on the idea that customers want to purchase a solution to their problem, not necessarily the ownership of a problem-solving tool (like a software). Therefore, they are rather willing to pay the function (the value) the product can release, than the product itself.

SaaS (Software as a Service) is a distribution model in which individuals and businesses can get access to cloud-based web applications for a reasonable subscription fee instead of paying for a lifetime ownership of a software licence. Similarly, Spotify and other streaming services provide subscription fee-based access to digital content (music, videos, etc.) instead of selling files or CDs to be owned by a consumer.

Servicizing is now quite popular for several reasons:

- users are consumers who treat web-based solutions as useful services, not as typical software products or systems;
- consumption of value from digital products is often based on a subscription fee rather than on owning a device;
- users expect the possibility of terminating their subscriptions whenever their needs change (which is impossible when owning a software which is no longer needed).

Servicizing also changes the definition of value for customer, as customers are now rather willing to reward the value of a direct solution for their problem, than owning the tool. The servicizing concept thus allows to convert an IT product (usually a website or mobile application) into a solution-delivering service or a subscription-based personal assistant. As a result, online services are pervasive and penetrating all areas of professional and private life.

There are three main statements resulting from the servicizing concept:

1. In online services adequate functionality, usability and UX are necessary to cumulate positive user experiences and to create trust-based customer-vendor relationships.
2. In agile IT projects designers are primarily developing services for customers, not merely software products.
3. The customers want to pay benefits they get from a specific service, not its engineering excellence or an ownership of the IT tool.

For designing web-based interactions, servicizing means that all types of websites and web applications deliver value only by activation user-service interaction, when customer is operating a website or an app. It is similar to the Service Dominant Logic (SDL) concept, which states that value is created by mutual exchange of activities among actors involved in a business relationship (Vargo and Lusch, 2008).

Both servicizing and SDL are perfectly suited as conceptual frameworks for web-based applications and online services, no matter if they provide rapid solutions to common problems, or have a long-term effect in improving quality of consumer's life.

The Web is going business

In the timespan of recent several years, online services have become a natural part of everyday life. In e-business to be successful, a broadened design perspective must be applied to designing websites and web applications.

In this broader view, designing user interface, caring for usability and UX is only a part on the job. Beyond visual aspects, the website should be trustworthy, accessible, and supportive in strengthening the brand image. A website should also clearly inform that this site safe and credible. A solid assurance about protecting customer data, protecting privacy and guaranteed security of payment, confirmed by certificates and authorizations should be provided. Nevertheless, perfectness in each detail of web design, outstanding usability and UX are the fundaments for building trust and credibility for designing economical interactions online.

In efforts to keep an online service resilient to market competition, during its lifecycle it is important not only to keep it growing financially, but also strengthening its maturity by raising subsequent maturity levels. While the FUVUX model (Sikorski, 2012) identifies crucial design areas (functionality, usability, user expe-

rience and value) which should be subsequently developed to gain acceptance with customers, the layer-based model for online services describes the process of gradual growth and development. Both models, presented in Chapter 1, can be used to conceptualize the systematic upgrade of online services. The upgrade development process may take place a series of small, incremental changes, as well as major revitalizations taking place in dedicated projects aimed at improving Customer Experience.

E-business websites and online services are usually integrated with Customer Relationship Management (CRM) systems, which allow preparing customized offerings and campaigns. As a result, nowadays the scope of web design covers issues beyond pure user interface design: strengthening the online vendors' image and brand, as trusted and caring for customers, and designing economic interactions, especially online relationships which build customer loyalty (Figure 3.6).

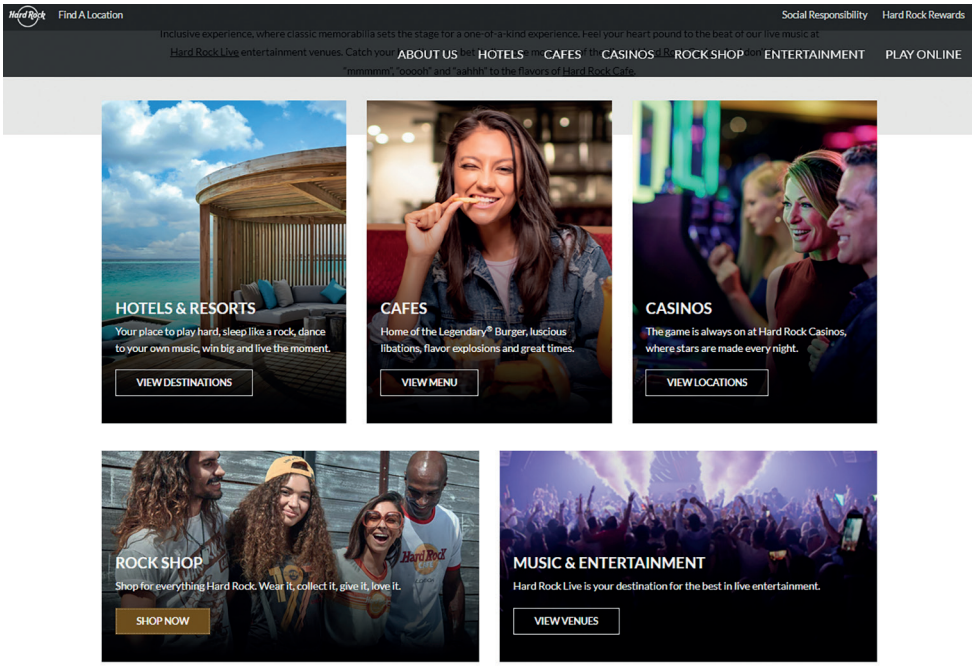

Figure 3.6. An example of a website creating relationship with a brand (Credits: https://hardrock.com)

Expanded interactivity

In today's digital business customers expect more interactivity, regarding following areas:

1. Interactivity with a product or service
   User's motivation to buy a product is much affected by visual stimuli. For this reason, multiple modes of visual interactivity with the product must be provided,

using high-quality interactive graphics, ability to zoom or flip the product view, with quick loading of visual contents and immediate feedback to user actions. Excellent visual search is also very important, not only by finding product images from an internal search engine, but also by gallery views, history (with product images) and a purposeful use of suggestions generated by recommender systems. For instance, Rich Internet Applications (RIA) with AJAX technology can improve UX by enabling smooth zooming, viewing, rotating objects etc., during an online configuration of a product or when browsing a big information space.

2. Interactivity with the service vendor
   Users/customers often have questions regarding the product, delivery, guarantee, etc. They may be sending a question by e-mail or using an online form or a chat available in a website. Excellent responsiveness is a crucial factor for building positive Customer Experience. Customers frequently complain that in many websites an online chat is not always staffed, or it is operated by a chatbot often distrusted by users. Users also complain that getting response to a written enquiry often takes long (or sometimes it is never received); no wonder that a vendor who responds quickly has a higher chance for receiving the customer's order.
3. Interactivity with other customers
   Customers' decisions are often much driven by recommendations, opinions and comments from other users. For this reason it is essential that online vendors provide users the possibility for social interactions directly on their website, or on vendor's fanpage. Even if comments are sometimes critical, customers usually appreciate vendor's openness if they are politely responded. Otherwise, if vendor had not provided an area for customer's comments, the critical ones will be probably published online elsewhere, leaving the vendor with little possibility to respond.

The Web is going social

In 1990s, early videoconferencing apps (like Skype) prepared users to new opportunities social interacting online can offer. Subsequently, rapid emergence of Facebook and Twitter empowered users to communicate and to publish comments, videos, or podcasts.

In recent years social media became a powerful tool for creating online communities, marketing communication, social and political campaigning and publishing own content (User Generated Content - UGC). UGC and largely spontaneous social activity online significantly benefited in democratizing the web. In many places it also much contributed to strengthening civic society, to social reinforcement, rapid information spread and mobilizing communities for actions when needed.

Also, businesses have successfully combined websites with social media. Websites present data, facts, and periodically updated information, which remain rather static and official. Instead, social media (like company fanpages or video

channels) conduct dynamic interaction with target communities and often are complementary to official websites.

Social media created new business opportunities for campaigning, crowdsourcing, and fundraising. They also contributed to creating a variety of new jobs, like social marketers, influencers, bloggers, and other types of online content creators.

On the other hand, users' liberty in social media created significant problems regarding credibility, validity and accurateness of user-published news or opinions. Subsequently social media turned out to be prone to abuse, online violence, hate speech and propagating various extremisms. Nevertheless, despite of some regulating efforts, they are still associated with high risk for misinformation, fake news, scam, and aggression. This raises concerns about institutions and regulators unable to efficiently protect the most vulnerable members of the society, like children and youth.

Actually, in social media UX results largely from interacting with content almost entirely generated by other users, which is often controversial and much beyond the control of webmasters and moderators. In web design using social media, all these concerns should be adequately handled in designing web-based interaction. Nowadays credibility and trust are essential to build relationships, and they are an essential part of interaction design.

Web-based conversational interfaces

In 1990s early web-based videoconferencing apps (like Skype) gave rise to online interactions with remotely located persons. Today conversational interfaces are an essential part of social media, but in addition to connecting people, they can be also used for conducting dialogues between human user and a software robot, such as chatbot of a virtual agent.

Both chatbots and virtual agents are conversational interfaces powered by AI-based speech engines. They are able to "understand" questions users' put in a natural language and to prepare an answer to be presented by a text message or by a synthesized voice.

Chatbots and virtual agents are often embedded into websites of utility companies, banks and other institutions daily serving frequent enquiries from thousands of customers. The use of chatbots is often motivated by expectations that call centre's costs will be reduced, because most of typical questions could be possibly answered by a software robot.

Chatbots are chat-like conversational interfaces, where a user can type a question to a text-based dialogue window, usually marked by an icon symbolizing an automated chatbot. If equipped with suitable language engine, chatbots are able to conduct text-based conversations in a way much similar to human-human dialogue (Figure 3.7).

Virtual agents are on-screen human-like characters, often animated, that respond to user questions using synthetized voice or by a text message (Figure 3.8). They usually represent assistant personnel from specific institution that owns the website, now "staffed" a virtual person. Virtual agents are of complex engineering, because they need various AI-based mechanisms to dynamically adjust replies to the context, to customer's expectations and to a specific dialogue style. The animation of the agent (mimics and gestures) also needs to be synchronized with the content of the dialogue.

Figure 3.7. Conceptualization of a chatbot-based conversational web user interface
(Credits: https://chatbotsmagazine.com)

If a conversational interface (a chatbot or a virtual agent) works smoothly, users should be impressed, and their positive user experience should result in confidence to a virtual agent and in willingness to reuse. Unfortunately, despite of significant advancements in speech and language engineering, conversations with a chatbot or virtual agent yet do not resemble a real human-human dialogue for following reasons:

- comparing to chatbots, users have much higher – and difficult to meet – expectations for virtual agents as to the quality of dialogue and their behaviour to be realistic;
- very few virtual agents look and behave in a way similar to a real human; users often criticize that domain knowledge and dialogue flexibility is much inferior to a living company representative;

- virtual agents and chatbots lack the ability to understand context, humour, irony, jargon or situational issues, so the dialogue is prone to jams, errors and misunderstandings.

As a result, in many cases the UX with chatbots and virtual agents is still far from expected, and users are reluctant to use them again. Certainly, imperfect AI and speech engineering can be blamed, but also the lack of precise and universal guidelines how to design virtual agents and chatbots to be competent, and able to resemble the style of human-human dialogues.

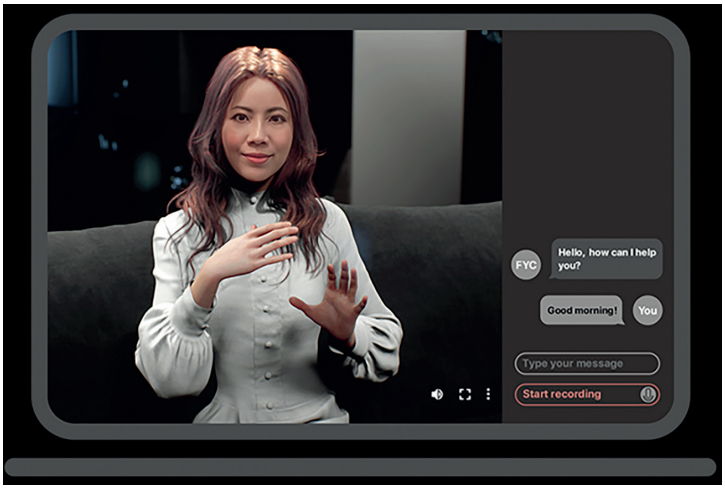

Figure 3.8. A virtual assistant – an example
(Credits: https://www.pinscreen.com/vitualassistant)

Collaborative online environments

Web-based collaborative online environments have recently become popular in Computer-Supported Cooperative Work (CSCW). Collaborative environments revolutionized the teamwork among remotely located personnel in offices, business and industry. Now users can take part in virtual teams using a web browser, a locally installed application (a thin client) or a mobile application on a handheld device (Figure 3.9).

In addition to chat and videoconferencing, collaborative environments offer a set of synchronized functionalities, for instance including:

- scheduling virtual meetings and inviting participants to virtual teams;
- sharing screens, documents, folders, and other resources;
- shared workspace (whiteboard) for collaborative brainstorming, drawing, sketching etc.;
- applications for collaborative engineering design, modelling, prototyping or decision making;
- control panel for customizing the screen layout for individual needs.

Nevertheless, currently available online collaboration platforms often leave users with mixed UX after participation in virtual teams. Following complaints are frequently heard:

- technical quality of connections: quality of sound, image or video, poor visual contact, time lags desynchronizing video and sound, or insufficient fluency of communication among team members;
- high load of simultaneous multitasking: conducting work process, dynamically assigning roles to team members, sharing access to files, conversating with participants, coping with unexpected problems and operating control panel of collaboration platform;
- meagre communication in virtual team: some participants remain passive while others are overworked, the balance between autonomy and supervision remains problematic, and tasks and deadlines assigned online need to be later confirmed in a written form as e-mails, memos or minutes.

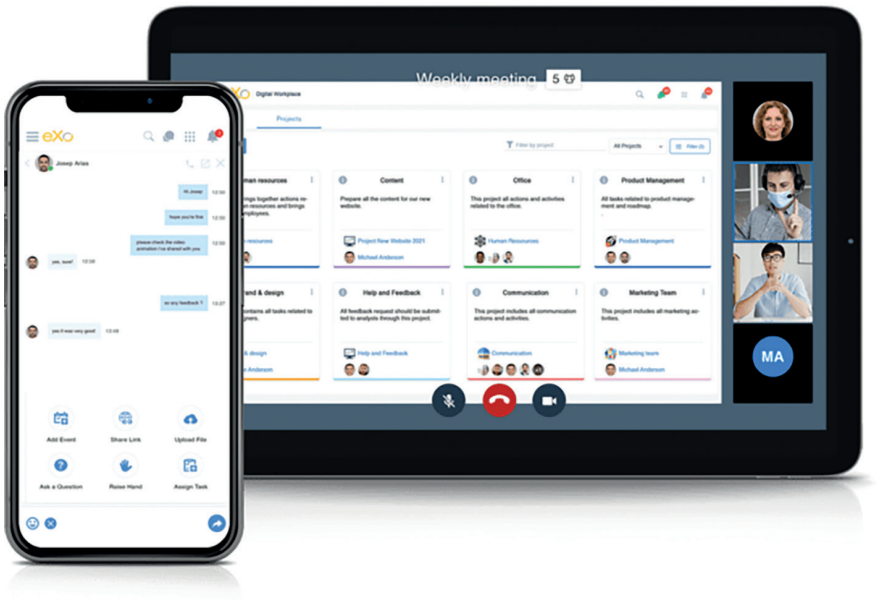

Figure 3.9. A collaborative web user interface - an example
(Credits: https://www.exoplatform.com)

Ongoing strive for competitive advantage

Website owners in e-commerce operate in a very competitive environment, so they constantly look for novelties aimed at attracting online customers and retaining their loyalty. To achieve this goal, online businesses eagerly use social marketing, Search Engine Optimization (SEO), and web analytics for identifying patterns of online consumer behaviour.

Online businesses constantly search for new functionalities, for new business models and for new methods of building customer relationships. As all online businesses now use CRM systems, which are often combined with AI-based recommender systems and customer profiling. This requires collecting and mining data collected on customer online behaviours, so protecting data privacy and security are here especially relevant. Search for new functionalities is aimed at providing new experiences also with newly emerging technological opportunities, like virtual agents. E-commerce websites may also use an AR or VR add-ons to set the product in a new context, for instance enabling user to see oneself onscreen, dressed in clothing considered for purchase.

Ongoing search for competitive advantage is continuous process of frequent implementing small changes. Classical, GUI based software applications needed periodical, time-consuming upgrades, reducing availability of the system for users. Instead, web-based online services undergo numerous, systematic improvements in User Experience as well as technical and business aspects. Only from time to time websites undergo bigger revitalization projects, covering major improvements in both front-end interactions and in back-end processes. Their radical effect will be easily noticed by customers, but not always enthusiastically, if changes seem too radical for some customers.

While in web design a typical IT project usually takes weeks, up to several months, further maintenance and improvement process will be hopefully spread over many years. As a result, a website and online service must be also designed for easy maintenance, upgrades and revitalisation, so thoughtful selecting the right technology and system architecture are of the utmost importance.

The Web is going mobile

Nowadays users operate a broad spectrum of Web user interfaces: from simple websites to advanced online services and digital platforms, connecting business process participants. In everyday use, in a global scale, web user interface is now probably more frequently used than GUI. While GUI remains mostly for desktop work-related use, economic and social interactions largely take place online, using web user interfaces.

In recent years availability of wi-fi networks and handheld devices (smartphones and tablets) increased popularity of both m-websites and mobile apps. As a result of further servicizing, all online services are now available with mobile devices from any location, any time, using mobile user interfaces presented in the next chapter.

4. Mobile user interfaces

4.1. Specific features of mobile user interface

Since about 2000 dynamic development of online services, accompanied by the wide availability of smartphones and tablets, enabled computer users to access the internet not only in static, desktop-based mode. Per analogy to web-based experience, mobile users expected that mobile services and mobile applications would provide performance, efficiency and positive UX with rewarding relationships.

Subsequently, existing websites were soon adapted to small screens. As a result, mobile websites (m-pages, m-sites) and mobile applications became extremely popular. Nowadays mobile applications are frequently used literally for everything, ranging from direct communication and solving ad-hoc problems, to long-term supporting improving quality of live. Typical activities users perform with mobile applications include for instance:

- solving daily problems: shopping, cooking, calculations, navigation, or travel planning;
- news and information “on the go”: e-newspapers, e-books, reading news, or e-books;
- staying connected: text, chats, video communication, e-mails, communities and social networks;
- entertainment: games, puzzles, riddles, music, video, or social interactions;
- education and personal development: learning in commute time, viewing tutorials and videos, educational games;
- improving own lifestyle: persuasive applications (Figure 4.1) for wellbeing, fitness, e-health, saving, or ecology;
- professional work: performing work-related tasks out of office, teleconferencing, collaborating, remote diagnostics, etc.

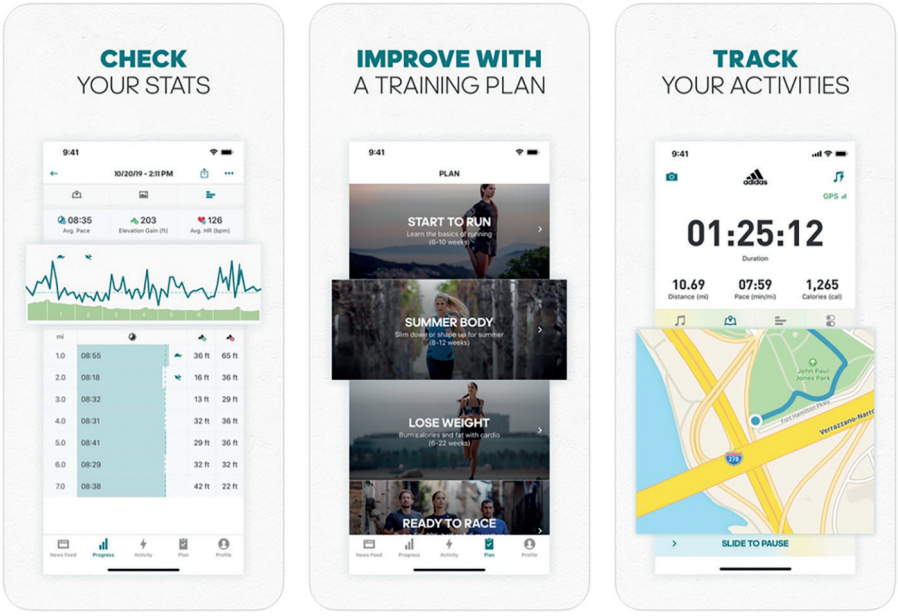

Figure 4.1. A lifestyle-related mobile application – a workout assistant (Credits: https://www.runtastic.com)

In mobile interaction social aspect is especially important. Ability to stay connected all the time with family and friends makes users prefer to use mobile applications for ad-hoc social activities.

Relying on mobile devices and mobile apps in almost any aspect of everyday life, and habit of having a mobile device always nearby, led to so-called mobile user experience (Adobe, 2015) which is an important factor for today's online consumers. When using the same service in different interaction modes (mobile, web, desktop) online customers expect consistent UX when accessing the same service using a conventional website, m-page and mobile app.

Design objectives for mobile interaction are shaped by two forces: firstly, conforming to limitations imposed by specificity of mobile devices, and secondly, by exploring new business opportunities offered by mobile business and marketing.

Regarding limitations, following factors largely affect the design of mobile interactions:

- small screen size, limited input capacities, no right click, small buttons, small labels;
- poor reactivity of a touchscreen, uncomfortable scrolling and typing;
- negative environmental impacts, like sunlight, dust or noisy environment;
- limited timespan of user's attention, frequent multitasking with external interruptions;

- limited bandwidth and computing resources, affecting speed, performance and task efficiency.

Regarding new business opportunities, design objectives for mobile interaction may be driven by such factors as constant access to an online consumer, availability of novel mobile marketing tools, and the need to mitigate user's concerns about privacy protection in public networks.

After compromising limitations and new opportunities following design goals emerge for mobile applications:

- adequate functionality and practical usefulness in solving a specific problem;
- time on task and task performance;
- usability - ease of use;
- positive UX and willingness to reuse the application;
- long-term values - supporting individual lifestyle, improving quality of life.

From the business viewpoint, design objectives for mobile interaction design do not differ much from web-based design, including all components of the FUVUX model: functionality, usability, user experience and value for customer. Also the layer model of online service development applies for mobile apps and services, too. Both models were presented in Chapter 1.

4.2. Design guidelines

Guidelines for mobile user interface design largely originate from design recommendations relevant to GUI and web user interfaces. However, for mobile user interfaces Nielsen's usability heuristics need to be considerably adapted, regarding limitations of a small screen and variable environmental context (Nielsen and Budiu, 2013).

H1. Visibility of system status. Keep users informed of system status with constant feedback.

The mobile application should always provide information about what is currently application in the system through confirmations and messages. For instance, the user should be informed with a progress bar when the data are still in transfer and how long it may take.

H2. Match between system and the real world. Set information in a logical, natural order.

The application should inform the user with a plain language, with words, phrases and graphics familiar to the user. Information should application in a natural and logical order, especially when a user is filing in online forms or checking out to the payment. Abbreviations and acronyms should be avoided, using instead short labels with a common language.

H3. User control and freedom. Ensure users can easily undo/redo actions.
The application should smoothly guide the user through subsequent steps of the task. Functions like “cancel”, “back”, “forward” “undo” should be easily visible for each operation. Additionally, there should be provided an easy correction of errors in online forms, preview of the data before they are sent, preview the actual appearance of objects, preferably with zooming or scaling the view.

H4. Consistency and standards. Maintain consistent standards, so users know what to do next without having to learn new toolsets.
The application should use consistent page layouts, and permanent positioning of clickable objects such as buttons, headers, labels, menus etc., which should operate in the way known to users from other systems. Names and labels should be short and communicative, following popular conventions. Be consistent within the use of interaction gestures, controls, functions, and other elements of user interface. Use clear, intuitive graphical symbols, commonly known to the users. In design, for symbols to be clear, adequate use of simplification and abstraction are essential.

H5. Error prevention. Prevent errors if possible; wherever not, warn users before they commit to actions.
Interaction, layout and content of the application should be designed in a way minimizing possibility for human error. Prevent the user from getting lost. Other hints for preventing errors include pre-set default data values, automatic spell checking, autocompletion, forms which do not lose data when the “back” button is clicked.

H6. Recognition rather than recall. Do not make users remember information; keep important objects always visible.
Make sure that the main functions of the application are easily accessible. Minimize the user’s memory load. For a temporary storage of items, a clipboard, wish list or a handy personal repository should be available. Use short menu paths for the main functions or keep the main functions visible all the time.

H7. Flexibility and efficiency of use. Make systems flexible so novices and experts can choose to do more or less on them.
The application should accelerate the user in reaching the goal by using multiple types of shortcuts, as well as by using a local search complementary to hierarchy menu. Also, a preview of recently selected items, autocompleting data or phrases, and always visible list of most commonly used options (recently selected items, for example) should be available. For increasing task performance in repetitive tasks, allow the user to switch off or hide unnecessary screen elements.

H8. Aesthetic and minimalistic design. Design with aesthetics and minimalism in mind – do not clutter with unnecessary items.
Make all objects, actions, and options clearly visible through visual elegance, simplicity and minimalism. Visual elements should guide users gaze to important elements, like using the "big green button" pattern. All elements should work well together and complement each other (Grobelny and Michalski, 2020). Balance, clarity and adequate contrast are important for users. Clarity of the screen is mainly achieved through contrast, which can be created with opposites, such as dark and light objects. The distinction between insignificant and significant visual elements needs to be made clear in order to guide attention to specific details.

H9. Help users recognize, diagnose, and recover from errors. Provide plain-language error messages to pinpoint problems and likely solutions.
In mobile applications users usually commit errors when they perform activities in unfavourable conditions, like operating controls in space constraints or in poor visibility due to sunlight. Validation of user input data should take place real time in the data field, and default formats of data, hints and tips should be proposed before important action is to be completed. Error messages should be expressed in plain language, should precisely indicate the problem, and constructively suggest a solution. When errors occur, recovering from them should be straightforward, preferably using reversible actions.

H10. Help and documentation. Offer easy-to-search troubleshooting resources, if needed.
Although mobile applications are intended to be entirely self-service systems, users still may need help in case of problems or questions. Adequate forms of support should be provided, like on-screen help, answers to typical questions, or a live chat or phoneline. Provide both a quick guidance focused on the user's task and more detailed documentation to read, preferably with search functions. Pay attention to the understandability of the user support information.

Other guidelines important for mobile user interfaces include (Nielsen and Budiu, 2013; Hartson and Pyla, 2012):

1. **Respecting diversity of mobile usage patterns**
   - In mobile interactions there is a very limited focus and timespan of user's visual attention with quick instances of manual operations. Users are accessing the device frequently but for very short periods of time. Limitations of finger-operated touch combined with poor reactivity of a touchscreen can make interaction more difficult.
   - There can be changing light and weather conditions, and operation in a noisy surrounding making sound signalization unreliable. Scrolling and typing can be uncomfortable when affected by cold, humidity, dirt, fat, etc.

- Multitasking operations are often “on the go” while walking, eating, talking, or inside a vehicle (vibrations etc.), lacking the comfort of device operation indoors. Multi-access is common – users often access the same online service switching among several mobile devices they own. There are variable usage patterns, for instance ad-hoc problem solving, staying connected or just killing the time by entertainment, or socializing.

2. **Providing excellent visibility of important objects**
   - For mobile user interface design the surface of a small screen should be used sparingly, focusing the content on the main functionality of the application. Available space should be used in most useful purpose by placing essential information, control elements, important data fields in central location, with the most important elements in the upper 50% of the screen.
   - Text must be easy to read: only Arial front or similar, size min. 11pt, preferably scalable and flowable, excellent contrast with the background. Graphics should be minimalistic and informative, and important elements must be visible all the time (no hiding).
   - Limitations of objects’ visibility due to: poor weather, sunlight, location, as well as visual impairments of users, should be included as a critical design factor.
3. **Providing efficient manual control by fingers**
   - The need of typing with the finger should be minimized, by using default values, words from vocabulary, auto-complete, etc.
   - Clickable objects must be big, at least 9x9mm, the bigger the better, default object (like “OK”, “Next” or “Submit”) should be distinctive, wherever possible use the green colour for default (“the big green button” pattern). The time and effort of manual navigation should be reduced: minimal number of steps, immediate feedback, manual selection better than typing, eliminate scrolling instead rather using buttons [Next >] or [>] than sliders.
   - Online forms in small screens are especially disliked by users: eliminate long scrolling, provide big, scalable data fields, readable labels, real-time data validation, an instant confirmation on successful submitting and a smooth transfer to the next activity.
4. **Providing reliable guidance**
   - In multi-step procedural tasks, users expect guidance: a visual path (like “Step 3 of 5” or a progress bar) should be provided.
   - If there are questions related to the displayed content, there should be possibility to explain unclarities (like missing parameter in product description) directly from the screen where the questionable item is located (for instance calling a hotline).
   - All guidance hints should be designed in such a way that they are clearly visible also in the sunlight.

5. **Providing excellent feedback**
   - For actions performed on a small screen excellent confirmations for the status of operation should be provided. Visual confirmation of operations using objects (sending, receiving, refreshing, reformatting, saving etc.) should be easily visible also in the sunlight.
   - Alerts for the users should be easily received also in unfavourable environmental conditions. Preferably, users should be able to select what types of confirmations (visual, sound, vibration) are preferred.
6. **Solving user's dilemma: an application or m-page**
   - Mobile users can access preferable services using or a mobile version of a website (m-page, m-website) or a dedicated mobile application. Because users are often sceptic as to downloading an application they do not know, they need to be convinced that a specific application offers more benefits than an m-site, for instance access to exclusive offers.
   - User's hesitation whether to download a mobile application or rather not, may be rooted in numerous uncertainties, for instance: will the application have access to user's data in a device or in the cloud? will the speed of an application be better than of m-page? will the application affect the battery life?
   - For convincing users to download a mobile application tangible benefits (missing in m-pages) should be presented, for example ability to work offline, improved ease of use, usability and UX, or functionalities not available in m-page, like user profiles, history, ability of sharing data. The process of communicating the value of a mobile application versus the m-page includes three phases: informing, encouraging, and retaining by rewarding user's loyalty with increasing benefits.

4.3. Design patterns for standardization and consistency

In mobile user interface design, following design patterns reduces users' workload and makes manual operations easier. In mobile user interfaces three types of designs patterns (Mendoza, 2013) can be used:

- interaction elements, like typical controls or other visual objects;
- interactions solutions, like typical widgets consisting of several objects;
- relationships among typical objects, like a location of default buttons in a dialogue window.

Typical design patterns, used in both iOS and Android, according to Neil (2015), include for instance:

- Springboard. It is a starting point to dive into an application, an equivalent of home screen. A regular grid should be used to arrange objects of equal importance and an irregular layout to highlight certain objects.
- Expanding list. It allows user to go through 2-3 levels of information (text or graphics) with Touching the [+], [>] or [-] symbol will expand / collapse the list through subsequent levels of the description hierarchy.
- Tip. A small object with a text bubble. It can be used anywhere to suggest the purpose of the object or how to perform the next operation.
- Page carousel. Used for fast browsing or a quick transition between several pages by moving screens (Figure 4.2).
- Metaphor. Start screen with a metaphor of a real environment, already known to the user. Due to poor visibility using a wrong metaphor is especially misleading in mobile user interface. The quality of metaphor has a very strong impact on willingness to use the application and to understanding the purpose and the way of operating the application.

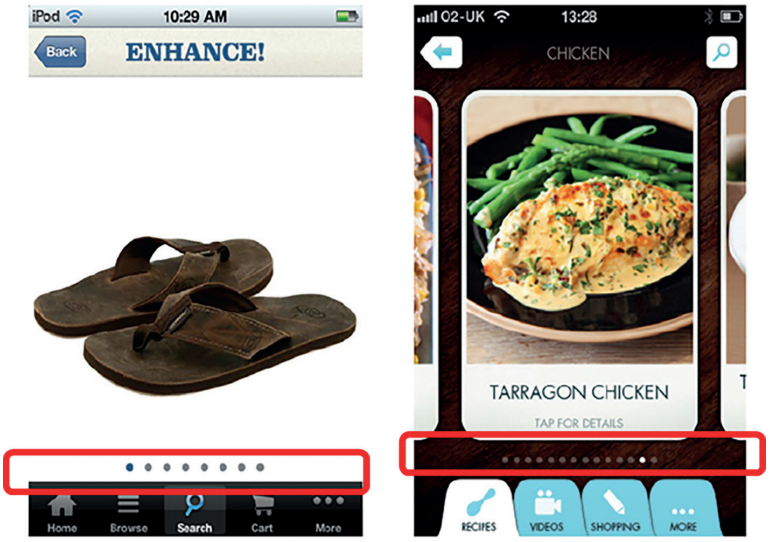

Figure 4.2. A carousel – design pattern for mobile user interface (Credits: https://www.androidpatterns.com)

4.4. Mobile accessibility

In contrary to websites, which are developed mainly with HTML-based technologies, mobile applications developed in different programming techniques conform rather poorly to accessibility guidelines. Similarly to web user interfaces the World

Wide Web Consortium specified general guidelines for providing accessibility for impaired users (W3C, 2020):

- colours should satisfy adequate contract requirements;
- visibility of all important objects should be improved with appropriate user controls;
- all active elements must be focusable, such as links, buttons, and form fields;
- text equivalent must be provided for every image or a non-text element within the app;
- avoid scrolling of screens, using a switch “next” or “back” instead;
- touch targets, like buttons, must be large enough for the user to interact with;
- displayed content should not be restricted to a single orientation, such as portrait or landscape, unless essential.

Mobile applications should provide users with ability to increase the font, contrast and adequate scalability of the screen layout in any position, which are the minimal features regarding accessibility. As an alternative, redirecting visually impaired users from a mobile app to a fully accessible mobile page is recommended option, if required functionality is the same.

4.5. Mobile UX factors

Mobile UX

Mobile user experience can be described in terms proposed by Hassenzahl (2008) as a composite hedonistic and pragmatic components, but it is too synthetic for mobile interactions. Due to business-related specificity of mobile applications, the “honeycomb” model by Morville (2004), should be used, with UX components can defined as follows:

- useful: functionality and content should satisfy specific needs of the user;
- usable: the application/m-website must be easy to use;
- desirable: the application/ m-website is the preferred option for performing the task;
- findable: the application/m-website is easy to find on the internet or in the app stores;
- accessible: relevant content needs to be accessible to people with disabilities;
- credible: users believe that the application or a website can be trusted;
- valuable: users believe that they benefit for using the application/m-website.

Similarly to web-based interactions, subjective instances of UX get cumulated, shaping user-consumer’s attitude to the reusing the application or service and to its brand (Mendoza, 2013).

Personalized User Experience

A handheld device is a very personal equipment, because it offers many possibilities for service personalization:

- the application would know user's name, and can use it in greetings and headers;
- user's default data and settings may be shared among applications or devices for cross-selling;
- web analytics can be used for inferencing from user's "favourites" and "frequently used";
- analysis of user's behaviour and preferences is rather easy because:
  - geolocation data show user's frequent routes and daily habits;
  - search phrases, history of activity informs on user's preferences;
  - recommender systems can be used for projecting user's expected needs and prospective behaviours.

Users are aware that mobile websites and applications can store and share user data with other entities. Therefore it may create concerns as to privacy protection, and eventually reluctance to download and use an application. News spreading about consequences of user profiling and tracking access to community and friends increase users' anxiety. For this reason mobile users should be informed that the application was granted access to user data and that these data will not be shared with anyone. This is the crucial element of trust, essential for shaping a positive personalized UX and keeping users loyal to a specific brand.

Successful UX though product design

An IT project aimed at developing a mobile application bears usual managerial risks, plus the one that application fails to gain popularity, despite its technical correctness. The market for mobile applications seems to be saturated now: users have a vast availability of mobile applications of any type. Downloading and trying them is easy, so users often uninstall new applications shortly after they found them unfit to individual needs.

For this reason designing a mobile application, which will attract a large number of users and retain them for a long time, is not easy. From the analysis of successful mobile applications for individual users (Adobe, 2015; Adobe, 2013) some common success factors can be identified:

- narrow scope, perfectly focusing the functionality, usability and UX of an application around its main purpose (value) essential for users:
- user feedback, constructed in way delivering users' complains to the design team and responding to them in an appropriate manner;
- analysing competitive solutions, and delivering competitive advantage for customers;
- attractive loyalty programmes, helpful in retaining customers and in promoting cross-sales interesting for users-consumers.

Successful UX though understanding the mobile context

Considering the costs and efforts needed to develop a dedicated mobile application, it is clear that accurate strategy and precise concept of the application are crucial. Developing product strategy for a mobile app starts from answering several preliminary questions:

- What particular problem an application is expected to solve?
- Why this application would be better than other alternative solutions?
- Why would users use this application, and not any competitive one?
- Where are the users located when the problem occurs? Why would they turn to the application to find a solution?
- What is the ideal solution of the problem, expected by users? How it differs from exiting solutions?
- Why should the users-consumers return? Why should they abandon using competitive applications? Which loyalty program would support their long-term loyalty?
- What different payment schemes will be offered for users-consumers?
- Which is the most adequate technology for the application? What prospective IT infrastructure (cloud services, business partners) will be needed?

Finding out the right answers well before design and development begin, reduces potential risk that the application would not gain sufficient number of users and business goals would not be met.

Successful UX through project management

Market success of the application is not only the outcome of a good concept, but also the result of thoughtful organization of the whole product lifecycle, which includes:

- project strategy: identifying the market niche and target users, preparing a good concept for the application and infrastructure, preparing a business case for convincing business partners and sponsors;
- organization of the project: recruiting competent people, providing technology and software development infrastructure, establishing cooperation with the business customer, applying an adequate project management model;
- managing operational activities: teamwork-based design, development and testing;
- deploying and promoting the application/service: promoting the application/service to users-customers, building the market and community for the application/service, retaining users by attractive loyalty programs;
- maintenance and revitalization: change management, scalable IT infrastructure, maintenance, modifying business models for growth, securing cooperation networks, financial background and business alliances.

Considering availability of development frameworks for mobile applications, development process may seem easy. Indeed, it may be relatively easy to deliver an application that is technically correct, but it fails short in gaining popularity due to UX-related deficiencies or business-related flaws.

4.6. Evaluation of web user interfaces

Heuristic evaluation

Heuristic evaluation is aimed at evaluating conformance of a website to Nielsen's usability heuristics adequately adapted to specificity of mobile interaction. Expert evaluation of a mobile interface can be performed using appropriately adapted heuristic evaluation (described in Section 2.4). Performed in a similar manner like for GUI or web user interface, heuristic evaluation is subjective, flexible and exploratory, and additionally it often includes references to websites or mobile applications serving the same purpose.

Checklist-based inspection

Mobile user interface inspections and reviews are conducted using standard a usability check-list identify the coverage ratio for requirements specified in the checklist. Questions-checkpoints are usually grouped into several thematic sections corresponding to the main criteria for the evaluation.

In the literature and online there are many checklists available for evaluating mobile websites and applications. They can be divided into two categories:

- usability checklists, with evaluate usability and sometimes mobile UX;
- accessibility checklists evaluating m-website of mobile app's compliance with requirements regarding accessibility for visually impaired users specified by W3C Consortium (W3C, 2020).

User-based evaluation and satisfaction surveys

Following types of usability testing are applied for mobile user interface (Albert et.al., 2010):

- laboratory usability testing: testing indoors, feasible only for validating general concept in for early prototypes;
- outdoor usability testing: testing "in the field", in a variety of contexts to which the application is addressed (like user's home, open space, public transport or other locations).

Usability testing of mobile user interface is usually less formal than in laboratory conditions, and aimed at pointing out specific improvements preferred by users. In addition to usability testing, following types of user satisfaction surveys are used:

- traditional online user satisfaction questionnaires: for a mobile application and a website respectively, usually as extensive post-test surveys with users who have just completed usability testing;
- ad-hoc satisfaction checks – a quick sampling of user satisfaction as an instant quick survey for users who have just completed testing mobile a m-website or a mobile app;
- post-test interviews: because filling in a survey questionnaire on a small screen is difficult, an informal interview with user in all cases is often used for collecting feedback and exploring improvement opportunities.

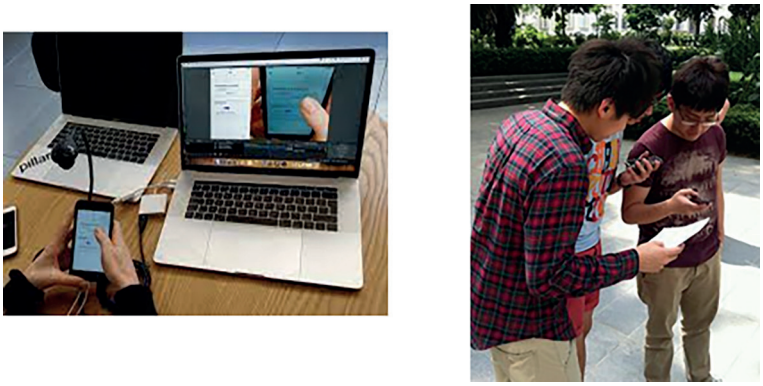

Figure 4.3. Mobile usability testing in a laboratory and outdoors
(Credits: https://www.justinmind.com/ and https://wiki.smu.edu.sg/is480/)

4.7. Trends and innovations

Mobile lifestyle

Users of mobile devices usually stay connected online all the time, because during the day they use mobile apps many times for various purposes. The often treat their device as a personal assistant and a preferred channel for accessing online services and especially connecting with friends.

Mobile lifestyle (Adobe, 2015; Adobe, 2013) is characterized by users' belief that many everyday problems can be solved ad-hoc using a specific app. Mobile apps are essential for users when at home (for activities such as cooking, shopping, planning), also at travel (music, navigation or finding places to visit) and at work, usually as a supplementary channel of communication. Mobile apps are treated as useful services, used for convenience and improving quality of life.

Permanent using mobile apps, and keeping the mobile device nearby, creates a sort of digital addiction, especially dangerous for young users. Users take their device with them also when moving around indoors, and they may feel the "separation anxiety" when the device is away, down or off-line. They almost "live" in the

online space, feel the need to be permanently connected, and may have more "friends" online than in real life. They often show reduced ability for sensible conversating and discussing meaningful issues, and instead they prefer communicating by brief "tweets" and memes. Therefore for hygienic and safety reasons, children and youth need a supervision regarding how they are affected by prevalent use of mobile apps. Mobile lifestyle is inevitable, but it should be kept balanced, reasonably taking benefits for work and everyday life, and positive outcomes of social interaction.

Mobile lifestyle of consumers, and frequent interactions with online contents, contributed much to creation of new business opportunities online. Professional bloggers, influencers, youtubers take advantage of broad ability of creating, distributing and commercializing User-Generated Content (UGC). When consumers are permanently connected and available online, business communication strategies must be also digital. When users have ad-hoc wish to shop, they should not be forced to get seated at a computer screen. Mobile apps should be available at the finger's reach, and their functionality, usability and UX should be orchestrated to streamline conversion from product watching to buying.

Combination with AR, VR and voice

E-business owners operate in a very competitive environment. Therefore they do their best to attract online customers and retain their loyalty for a possibly long time. Most recently, new developments in software technology enable transferring innovations and new functionalities form service websites to mobile applications.

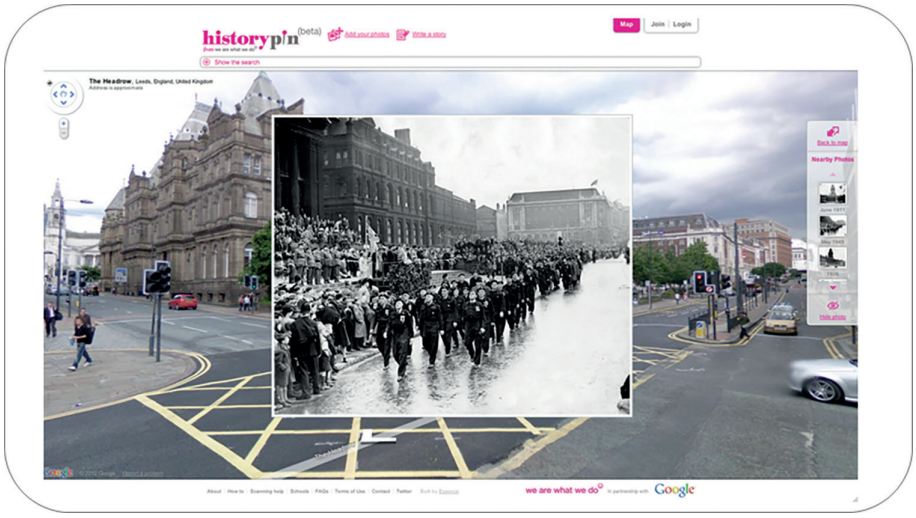

Figure 4.4. A mobile application with AR
(Credits: https://historypin.org)

For instance, such technologies as AR, VR and voice interfaces, now are getting popular also with mobile applications. Low-cost headsets for VR are now available, creating demand for VR-based mobile applications, primarily for entertainment and education. AR-based mobile applications (Figure 4.4) now are widely used for supporting industrial machinery maintenance, for expanding user experience in museums, gaming, entertainment, and probably they fill be widely used also in e-health mobile applications in a not very distant future. Voice interfaces and chatbots are used in the same way as in their WWW counterparts – websites and web applications, considering additional constraints how the service is operated with a handheld in real settings, often outdoors.

New mobile computing paradigms

With wide availability of mobile technologies, new computing paradigms have recently emerged, such as ubiquitous computing, Internet of Things, wearable computing and persuasive computing.

Persuasive technologies are mobile applications designed to help people monitor their behaviour and to persuade them a positive change in their lifestyle, attitude and behaviour, e.g. caring more for fitness, sleeping, weight, eating habits etc. Persuasive applications often use sensors installed in wearable devices, like smartwatches, wristbands etc. Persuasive apps promote novel functionality and novel experiences such as feedback tools encouraging self-reflection or visual-social tools, based on boards and charts benchmarking how the user performs in specific activities in relation to peers and friends (Figure 4.5).

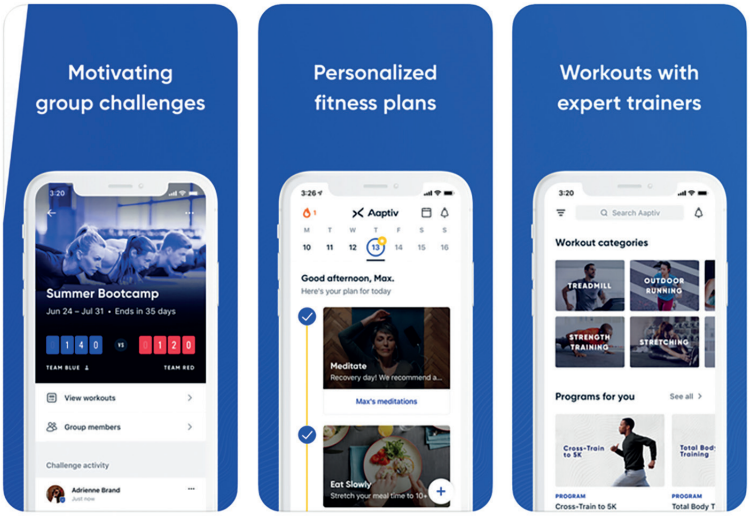

Figure 4.5. A persuasive mobile application – an example
(Credits: https://aaptiv.com)

Voice control is often used not only for providing drivers with hands-free operation and activating commands by voice. Voice-controlled household devices or systems are no longer a novelty, either. Most recently “smart speakers” got increasingly popular as an interaction devices for “smart home”, leading towards application of Internet of Things (IoT) computing paradigm in everyday life.

Servicizing of mobile interaction

Per analogy to web-based interaction, mobile applications can be used for a variety of purposes, from ad-hoc problem solving to long-term improving of quality of life. It is really hard to find an area of human activity not supported by some mobile applications – no matter what their quality and usability may be.

Integration of mobile applications with individual lifestyles, resulted that smartphones now are used mostly for accessing the online services, and not so much for phone conversations. Because users tailor their smartphones by downloading mobile applications matching individual needs, nowadays mobile devices serve usually as multifunctional personal assistants. Moreover, their value for users comes from a personal set mobile applications, not from a mobile device by itself. As a result, when users/consumers are constantly connected to Wi-Fi, it opens new opportunities for targeting users with novel lifestyle-related applications and services.

As a result, current IT projects are largely focused on delivering services for customers, and software is no longer the final product. Mobile lifestyle, and servicizing of mobile applications, affect the scope of IT projects and the way how they are managed. While for back-end solutions traditional project management methodologies are often used, mobile apps and other front-end solutions are usually designed in agile projects.

User-centred methods for interaction design and cooperating with customers in IT projects will be presented and discussed in remaining part of this book.

5. IT projects – cooperation with users

5.1. IT projects and software development lifecycles

Contemporary IT projects are based on three main approaches (Sommerville, 2016):

1. Classical approach (also known as sequential, linear or traditional) identifies the main phases of the software project with the assumption that they are located in a roughly linear, easily predictable process. The sequential approach is slow and costly as to coping with unexpected changes in requirements. For software projects performed in stable conditions, typical for institutional clients, it is assumed that almost all design activities can be performed without the active participation of prospective users.
2. Iterative approach (also known as spiral, cyclic or incremental) establishes more contact points with users (customers) and provides better responsiveness to sudden changes occurring during the project. In iterative approach an IT project is conducted as a series of cyclic iterations which deliver subsequent software components in a spiral, incremental manner. For achieving high usability of the product, the iterative approach highlights that involving prospective users into the project is vital, especially in identifying requirements, evaluation of prototypes and usability testing.
3. Agile approach operates on quick, interactive software development cycles and intensive teamwork. Agile approach leaves much freedom to the team how to optimize their work process and make the project “slim” by reducing project documentation to a reasonable minimum. The agile approach declares high responsiveness to sudden changes, typical for projects conducted for business customers. Most importantly, high quality and usability of the product is achieved at low cost, largely by frequent prototyping and intensive communication with users (customers).

All three approaches are widely present in contemporary software engineering and IT project management (Sommerville 2016), although the agile approach has recently gained extremely high popularity.

All types of IT projects are built upon synchronization of three main processes:

- workflow: conceptual and engineering activities that directly increase value of the product through all tasks performed across all stages of the project;
- communication: information flows among team members that support the workflow and other relevant activities in the project;
- management: all activities (typically assigned to the project leaders) aimed at coordinating the team towards delivering an expected product; all classical management functions apply hereby: planning, organizing, motivating, monitoring and continuous improvement.

The project manager and the development team play the crucial role in carrying out project activities, spanned by the three abovementioned processes. Excellent communication with the customer and with other stakeholders is the key factor for delivering a product with high quality and usability. The term “project stakeholders” describes all parties interested in the outcome of a project: primarily, the project team, the client, end-users, the board managers and optionally external bodies such as regulatory institutions or media. For better understanding of teamwork taking place in IT projects, and perspectives and motivations of participants, it is essential to clarify the most important roles:

- The user (end-user) is the person actually operating the system. Sometimes, users are customers if they own the rights to use a specific software application or an online service. Ease of use and practical usefulness of are the product characteristics most appreciated by the user.
- The customer (retail customer or user, see above) is the person actually interacting with a specific software or online service, which is the source of income for the software vendor or online service owner. For instance the user of a banking application certainly is a consumer, generating profit to the bank. Ease of use, practical utility and ability to collect some benefits represent product characteristics most appreciated by the user.
- The client (also interchangeably called business customer) is usually a representative of an organization which places an order for delivery of a specific IT system or service. A client can be external (from outside) or internal – like another department from the same organization where the project team is located.
- The sponsor is the person (usually a representative of an organization) who authorizes and controls the funding used to develop the software product. The project sponsor cooperates strictly with the project leader in making decisions for optimal allocation of available resources such as budget, time, staff, technology, etc., attempting to complete the project within given constraints.

The most popular IT project methodologies will be briefly presented hereafter, based on relevant software lifecycle models available in IT management literature, e.g. Chmielarz (2016); Sommerville (2016); Cobb (2011); Pressman (2000).

5.2. Classical methodologies for IT projects

The waterfall

Classical methodologies for IT projects management assume that project activities can be described as a sequence of predictable activities, performed in a predictable manner.

The classical, sequential model ("the waterfall") identifies subsequent stages in the software lifecycle, placed in a cascading, waterfall-like sequential order, shown in Figure 5.1:

- Analysis – requirements identification and specification,
- Design – system architecture design,
- Implementation – software coding,
- Testing – software tests and evaluation feedback,
- Maintenance – deployment, operation and modifications for changes.

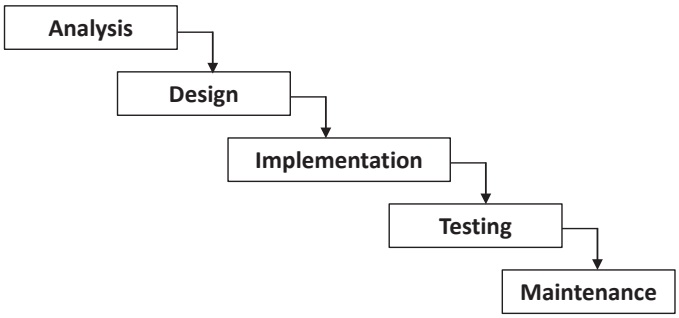

Figure 5.1. The waterfall model of software development (adapted from Pressman, 2000)

The waterfall model is based on assumption that each phase always begins after successful completion of the previous phase, what if often incorrect in real life. Moreover, in the waterfall model the contact with the client (rarely end user) is taking place only at the beginning and at the end of the project, what is frequently blamed for insufficient usability of the final product. Nevertheless, despite these limitations, the waterfall model still may be useful for procedural software projects, assuming the stability of specified requirements.

The spiral model

The spiral model portrays an IT project as an expanding spiral, unfolding from its centre (Sommerville 2016; Jayaswal and Patton 2009). Each iteration includes the cyclic passage through four activity areas, shown in Figure 5.2:

1. Action planning,
2. Risk analysis and selection of design variants,
3. Software engineering development,
4. Evaluation, testing and acceptance by the client representative.

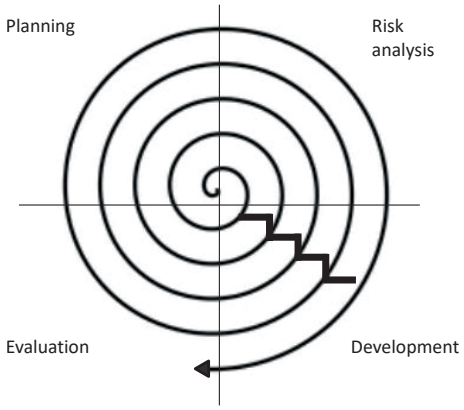

Figure 5.2. The spiral model of software development (adapted from Jayaswal and Patton, 2009)

The spiral model (Figure 5.2) includes the waterfall software engineering model in the Development area, but it strongly emphasizes the impact of other non-engineering areas, especially planning, risk analysis for decision-making and regular evaluation contact points with the client (customer).

Comparing to the waterfall model, the spiral model highlights the iterative nature of real-life design activities, where results are usually achieved in an iterative rather than in sequential manner. The main advantage of the spiral model is that it emphasizes the need for regular contact with the client (customer or user) because all deliverables must be evaluated in specified touchpoints in each iteration cycle. However, the main limitation of the spiral model is that it remains slow in producing software deliverables, especially if the system functionality is very complex. Nevertheless, experiences from IT projects based on the waterfall and spiral models, after numerous refinements served as a basis for the development of widely used process-oriented, classical methodologies such as RUP, PRINCE2 or PMI, respectively (Chmielarz, 2016).

Incremental model

Incremental model is a process of software development where requirements are broken down into standalone, parallel "streams" of software development cycle (Figure 5.3). Each stream is planned for delivering a specific software-based component (such as module, functionality, or feature) called "increment". The Incremental development is done in subsequent steps such as analysis, design, implementation and coding, testing and verification. Delivered increments are subsequently integrated and deployed to the client's site for operation and maintenance.

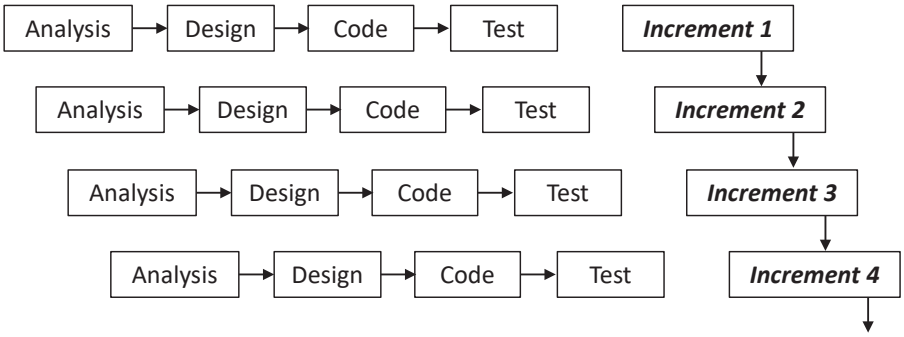

Figure 5.3. The incremental model of software development.
(adapted from Hartson and Pyla, 2012)

The first increment is usually the core of the product, in which the essential requirements are included, while supplementary features will be added in next increments. Advantages of the incremental model include benefits such as that system development is broken down into many mini development projects which are easier to manage, and that the client (customer) may start gaining business value before the whole system is completed.

Rapid prototyping

The process of rapid prototyping is also known under the name Rapid Application Development (RAD), shown in Figure 5.4. It includes the following cyclic steps (Sommerville, 2016):

1. Requirements planning: identify only the most essential functional requirements, temporarily ignoring other non-functional details.
2. User design:
   – a quick, initial prototype of user interfaces is developed;
   – end-users test the prototype and provide feedback on additions or changes;
   – the prototype is refined using the feedback from evaluation.
3. Construction: the software is coded and implemented, based on the refined prototype.

4. Release: deployment of a ready-to-use component to the client, with minor improvements yet to be made.

The "User design" is repeated until the product gains end users' acceptance. As shown in Figure 5.4 all these steps of RAD are performed in a sequential manner, only the User Design cycle is iterative, with user-based prototype testing and evaluation. Rapid prototyping enables delivering a software product quickly, iteratively and with the participation of domain experts (end-users or their representatives) as prototype testers.

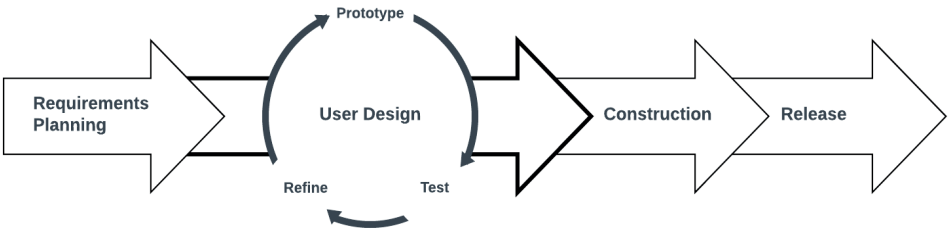

Figure 5.4. The rapid prototyping model (RAD) for software development (adapted from Jayaswal and Patton, 2009)

The RAD assumes participation of users only in testing and evaluating the mock-up of the user interface, without all other elements essential for making the system really operational. This limitation poses a significant difficulty in integrating rapid prototyping with software engineering practices, because the quick prototype usually will not be coded in the same technology as the real system. Despite introducing an iterative element of user-based testing, the rapid prototyping model still remains technology oriented, focused rather on increasing speed of delivery than on actual, process-based developments regarding usability and ergonomics of user interface.

5.3. Iterative methodologies of IT projects

User-Centred Design (UCD)

The iterative User-Centred Design (UCD) approach was developed following the idea of a spiral development cycle. The UCD identifies IT project as a spiral cycle of development activities leading to achieving product usability in an iterative, collaborative manner with direct participation of prospective users. The UCD approach was validated in IT projects and eventually was included in the standard ISO 13407 as a recommended, iterative design process for IT products, shown in Figure 5.5.

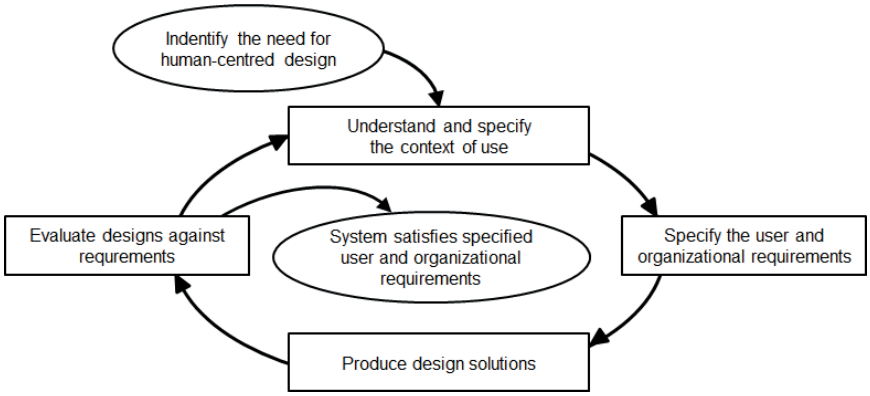

Figure 5.5. Iterative user-centred design (UCD) process
(Adapted from ISO 13407:1999)

As shown in Figure 5.5, after identifying the need for user-centred design as a starting point, the UCD process is performed as an iterative repetition of the following steps:

- understand and specify the context of use of the product: the purpose of the product, who are the users, what tasks they perform and in what environment;
- specify the user and organizational requirements: identifying user's needs and requirements as well as organizational constraints;
- produce design solutions: develop a specific design concept or a prototype;
- evaluate designs against requirements: evaluate with users the concepts, prototypes and solutions and collect user feedback.

In the UCD approach, active participation of users and frequent testing are aimed as the main usability assurance tools in IT projects. The iterative UCD approach requires participation of users in specified points of an IT project, particularly in identifying requirements of user interfaces and in testing prototypes.

The ISO 13407 standard recommends that appropriate identification of user needs and appropriate collaboration with prospective users are fundamental for a successful usability assurance of any interactive system. Particularly, the UCD process points out that the key element of a successful product is that the design team appropriately understands the context of use for a prospective product.

The UCD-STAR model

The UCD model proposed in ISO 13407 has established a fundamental framework for collaboration between designers and users, nevertheless it imposes a fixed sequence of activities, which might be not suitable for all types of IT projects.

Therefore the UCD-STAR model was proposed by Hix and Hartson (1993), to be more iterative by providing designers with more flexibility. The UCD-STAR mod-

el still relies on the iterative approach, but it identifies required activity areas, not a fixed sequence of activities expected from the design team (Figure 5.6):

- Task/Functional analysis – understanding user's needs and task-related activities;
- Requirements specification – finding out what people need from the system;
- Conceptual/Formal design – creating the overall idea for the new system and details how the system will work;
- Implementation – developing the code;
- Prototyping – bringing ideas to life, envisioning further developments;
- Evaluation – checking up how far the design solution is acceptable.

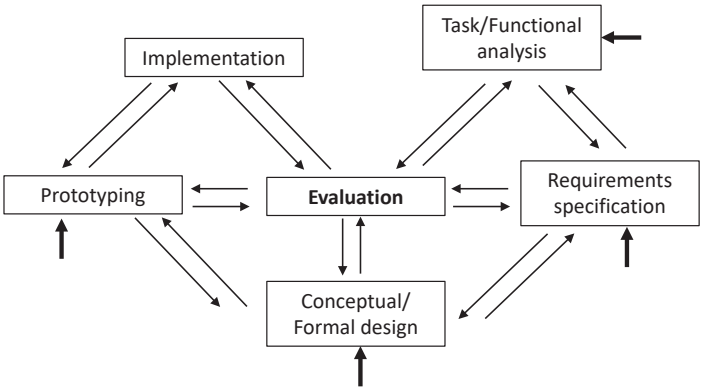

Figure 5.6. The UCD-STAR model for developing interactive products. (adapted from Hix and Hartson, 1993)

Comparing to the ISO-based UCD model, in the UCD-STAR model activities can be performed in "non-orderly" manner subject to the specific design situation. Most importantly, the transfers between any activities can be made only via evaluation, preferably user-based. The UCD-STAR model offers more flexibility, as the designers may start the process at any point, moving freely among relevant design areas, and customizing the UCD process to their actual needs.

The contribution of UCD to IT projects

The UCD models introduced the iterative design approach to IT projects, placing the user in an active role very close to designers. Most importantly, UCD has defined usability as an essential component of IT systems quality. The UCD approach has also introduced many innovative methods and techniques, such as:

- participatory observation – techniques based on ethnographic approach;
- context of use – an analysis of user characteristics, tasks and environment;
- user workshops – moderated teamwork with users as a part of an IT project;
- task analysis – developing precise task scenarios and use cases for projecting desired functionality of a system;

- persona – an imagined profile of a representative user;
- card sorting – projecting the information structure by users with a set of sticky cards;
- prototyping and testing – building user interface prototypes and testing them with users;
- regular collecting user feedback – interviews, surveys and questionnaires.

Because UCD methods and techniques are focused on establishing strong collaboration with prospective users across the whole duration of the project, they are often classified as "soft" design methods, in contrary to traditional software engineering.

Main advantages of the UCD approach include:

- involving users to IT projects from the start: early focus on users, tasks and actual context of use;
- iterative design, development, and improvements, and fixing problems quickly, based on frequent feedback from prospective users;
- bringing users' knowledge on the task and the context, and utilising their real-life expertise which might be otherwise inaccessible to design team;
- delivering product usability in a cost-effective way, based on immediate users feedback from evaluating proposed design solutions.

Despite of the above benefits, there are also some downsides of UCD:

- it is sometimes hard to get users involved in a project (cost, reluctance etc.);
- users are not expert designers so expecting them to produce design ideas may not work;
- users are not always right, as they often know things "as-they-are" but not what they may really need;
- the user (customer) is central to the design process but might not be present all the time, being available only in specific points of the project.

5.4. Agile methodologies for IT projects

Although the UCD approach is iterative, it still bears the burden of a slow progress and slow delivery, much below expectations of managers in current IT projects. To face this problem and to make IT projects faster and more reactive to changes, a set of quick and "light" project management methodologies was developed, collectively called "agile" in contrary to classic, slow and "heavy" design approaches.

Main agile methodologies will be briefly presented below (based on Stellman and Greene, 2013; Shore and Warden, 2008) starting from the Kanban and Lean, which laid the foundations for agile approaches for IT projects.

Kanban

Originating from industrial applications in Toyota, the Kanban is a simple technique useful in agile project management, providing rapid visual control of tasks performed by the team. A typical Kanban board has three columns with post-it cards which depict specific tasks and their current status: „to do", „in progress" or „done" (Figure 5.7).

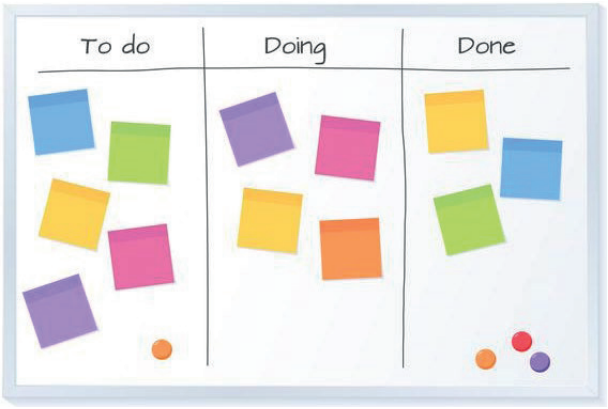

Figure 5.7. The Kanban board
(Credits: https://www.freepik.com)

Post-it sticky cards represent tasks directly resulting from specific requirements to be implemented and delivered as software functionalities. While the work process advancing, the cards are moved among the columns according to the progress of a specific task. The columns of the Kanban board can be modified according to the needs of the team. The number of cards per column can be also limited depending on the current throughput of the team. Kanban board improves transparency over a work process because it provides rapid visual monitoring and control for all team members, hence the Kanban board should be placed centrally in a shared workspace.

The Kanban board is one of the simplest and cheapest techniques useful for improving the process of software development, making it faster and more transparent. Although the Kanban board does not visualize any direct iterations, it depicts the dynamics of the project and the progress achieved in specific tasks by the development team. The Kanban board also records how ideas and requirements gradually transform into new functionalities and into value to be delivered for the customer.

Lean

Lean is an industry-based management approach aimed at economization of work processes by getting them "slimmer". This optimization is primarily made by

eliminating all unnecessary elements – inputs, structures and actions which do not add value to the end result. In principle, these activities should make a specific process more effective, more reactive and more engaging for its participants in the team.

Stellman and Greene (2013) list the seven main values of Lean: "eliminating waste," "supporting learning," "making decisions as late as possible," "delivering as early as possible," "giving powers to the team," "building integrity into the product," "recognizing the whole." Implementing these values in practice is based on sharing information within the team, on individual engagement of each member in looking for improvements and on collective decision making. As the Lean approach is supported with a variety of team management techniques and tools, its implementation in practice is relatively easy.

Despite its industrial origin, the Lean values are applicable also for software development processes in IT projects. Similarly to industrial settings, the most important issue is educating the team so they understand the Lean values and are able to use the Lean techniques and tools in an adequate manner. In software engineering the Lean approach gained high popularity especially in small companies and start-ups, which wanted to deliver a successful product despite operating with very limited resources (Cohn, 2013).

Extreme Programming (XP)

The eXtreme Programming (XP) is a set or engineering and teamwork management techniques known for its significant contribution to the development of agile project management methodologies, as we know them today. The XP has been successful in iterative delivering software products faster and in more cost-effective way than it used to be in classical or in iterative approaches (Jayaswal and Patton, 2009; Cohn, 2013). The XP software development model has following specific features:

- intensive teamwork, streamlined by an energetic team leader;
- short, dynamic development cycles;
- continuous, incremental delivery – delivering single functionalities or ready-to-use components even before the whole system is completed.

The XP has introduced to IT projects novel activities such as:

- standups – daily short meetings for work planning and a short discussion of current issues;
- pair programming – working in pairs helps developers in detecting errors, in mutual reviews of the code and in reaching joint responsibility for its quality;
- test-driven development – test scenarios are prepared before relevant functionality was created;
- refactoring – simplification of code and continuous improving its clarity event after testing was completed.

For speeding up the development the XP introduced specific principles of communication in the team. Preferably all members of the team should be located in one big room: shared workspace should be equipped with whiteboards depicting project-related diagrams, architectures, charts, drawings, etc. Secondly, a good visual contact among the team members, combined with easy access to visual tools, intensify the teamwork, collaboration, and discussions, so beneficial for the speed of software development and the quality of deliverables.

Since the emergence of XP, its values and principles have been successfully validated in software development practice. As a result, they substantially facilitated the emergence of other agile approaches, in particular the Scrum, which is now the most popular agile methodology in IT projects.

Scrum

The Scrum methodology (The Scrum Guide, 2020) has successfully incorporated key elements of XP, Lean and Kanban, and established several characteristic elements, now quite popular in agile projects management:

- sprints – short, dynamic, iterative development cycles based on an intensive teamwork;
- readiness for sudden changes in requirements, occurring during the project;
- quick software delivery based on intensive communication with the client (customer) and within the team.

Selected components of the Scrum methodology are presented in Figure 5.8. The dynamics of agile teamwork is shaped mostly by rapid development cycles – short sprints fuelled by intensive interactions taking place inside the team and with the customer. Agile teams usually work in self-organization mode, in a shared workspace (one big room or open space office), equipped with whiteboards and other visual tools.

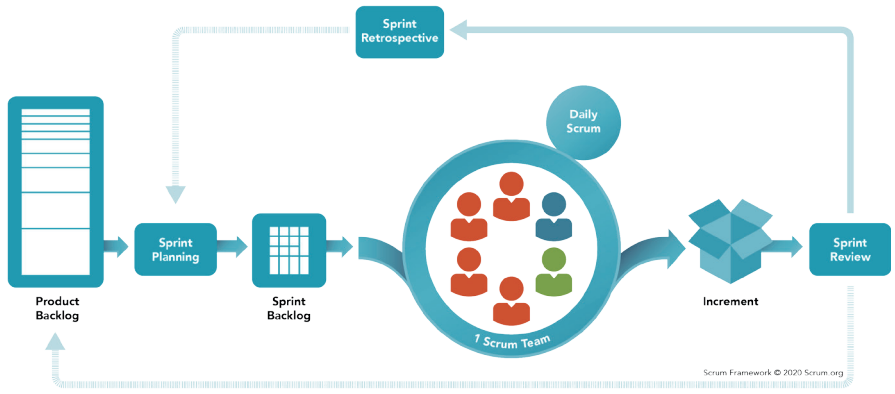

Figure 5.8. The main elements of Scrum
(Credits: https://www.scrum.org)

The Scrum methodology identifies following basic roles in an IT team:

- Scrum Master, who acts as a team leader, keeps guiding the team members towards self-organization and collective decision making, coordinates their daily activities and removes various barriers occurring in their everyday work.
- Product Owner, who acts in the project as a customers' representative, ensuring communication between the client or customer and the development team. Key responsibilities include deciding on the scope of system functionality, allocating the tasks to the development team, prioritizing prospective functionalities regarding their business value, and accepting (or rejecting) the delivered work outcome. In addition, Product Owner should be always available to the team and have ability to clearly communicate the vision of the emerging product.
- The development team (agile team) are remaining team members, mainly software developers, tasked with delivering an increment – a finished, ready-to-use part of the product. The team has the authority to organize and manage their work on their own for achieving the best possible efficiency of work. The optimal Scrum team should consist of members with multidisciplinary background and good interpersonal skills, because in agile projects the source code is usually owned by the entire development team, not by individuals.

In agile teams it is the responsibility of the Scrum Master to optimize working conditions for the team members and to run all required meetings. Instead, the Product Owner should focus merely on product quality and usability, to be ensured by intensive communication with the client (customer) and with the rest of the team.

The most important activities in Scrum projects include:

- Sprints are intensive, quick, product development iterations, taking form one to several weeks, ending with the delivery an increment – a specific, ready-to-use part of the product. Each sprint starts from the sprint planning meeting and ends with the sprint review after the client accepted a finished increment.
- Sprint planning meeting gathers the entire development team, the Scrum Master, the Product Owner and the client (business customer). Basing on the list of functions to be performed by the product, they define the functionalities to be developed in the current Sprint. The Sprint usually needs to be divided into the sequence of smaller and shorter iterations called Scrums.
- Scrum is a short development cycle devoted to a single component of the product. A complete series of subsequent Scrums, covering the timespan of the entire sprint, should result in delivering a ready-to-use component (an increment) to be demonstrated to the client for validation and feedback.
- Standup (called also "daily Scrum", or "daily") is a brief, 15-minute standing meeting taking place every at the start of each workday. The purpose of this meeting is to synchronize the work of all developers and organize their tasks for the current day. The Scrum Master should respond with quick decisions how to cope with reported issues.

- Sprint review is a meeting taking place at the end of each sprint, aimed at demonstrating and validating the specific outcome (increment, deliverable). In addition to mandatory participants such as the development team, the Product Owner and the Scrum Master, the client and other stakeholders are also invited. An execution of complete user's task should be performed on a prototype to validate usability and User Experience. The developers should be able to answer all questions about newly implemented features.
- Sprint retrospective is a meeting organized between the last Sprint review and the Sprint planning for the next iteration. This meeting – with participation limited only to the Scrum team members – is aimed at identifying improvements to be made for the next iteration, basing on a critical retrospective of all activities performed during the previous sprint. The Scrum Master is responsible for create an open, positive, and stimulating atmosphere for making the retrospective meeting as constructive as possible.

The Scrum methodology introduced following artefacts essential for driving the work progress in agile development teams:

- Product backlog – the catalogue of functional and non-functional requirements, comprising the complete product. The Product Owner is responsible for the backlog content, its updating, distribution and availability to all team members. Product backlog is always divided into smaller parts (called sprint backlogs) covering selected requirements, scheduled for processing in next sprints.
- Sprint backlog – contains the subset of requirements for a specific deliverable declared as the outcome of a current sprint. It is the collection of all items (like user stories of requirements) to be implemented in the current sprint. The team members select specific items to be processed in the current sprint, following their priorities determined by the Product Owner. The content of a sprint backlog reflects all the work the team has to do to deliver a specific, ready-to-use part of the product (an increment).
- User story – a short, structured projection of task to be performed by the user and its expected outcome. User story represents a requirement for a specific functionality to be implemented by the agile team.
- Increment – a set of items (tasks, jobs, activities etc.) from the product backlog which need to be completed during the current sprint. Handing over an increment (like a component, functionality or feature) from the current sprint to the client means that the component is ready to produce business value for the client or for customers, even if the entire system is not yet completed.
- Definition of "done" (DoD) is a set of acceptance criteria for the outcome (increment, component etc.) resulting from each user story. Among other issues, the outcome should have had a code review, have been tested, be immediately deployable and approved by the client. Finally, the item card on Kanban board should be moved to "done".

For keeping the Scrum team members informed in real time specific visual control tools are used such as:

- Scrum board – a Kanban board showing allocations of tasks and their progress, accordingly modified to the needs of a specific team;
- Product burnout chart – a chart which presents the amount of remaining work in all planned iterations (sprints), necessary for implementing all pending requirements till the product is complete.
- Sprint burnout chart – a chart analogous to the product burnout chart but used for tracking work progress in a single sprint.

The Scrum methodology works well for relatively small projects, taking up to several months and typically engaging the team up to about 20 individuals. However, for bigger projects multiplied and combined efforts often need to be developed with agile methodologies. For this purpose extensions of agile methodology can be used, such as:

- LeSS (Large-Scale-Scrum) is addressed to IT projects with multiple Scrum teams collaborating upon one product for one customer. LeSS follows fundamental Scrum principles, naming them as Transparency, Empirical Process Control, Iterative Development, and Self-Managing Teams. LeSS is mainly focused on efficient communicating among representatives of several agile teams, and with the customer (Cohn, 2013).
- SAFe (Scaled Agile Framework) is a set of workflow patterns intended to guide organizations in expanding lean and agile practices across large numbers of synchronized agile teams. SAFe was developed by integrating three primary bodies of knowledge: agile software development, lean product development, and systems thinking. SAFe addresses the need of big companies willing to transfer their conventional practices of their design teams closer to agile approach (Cohn, 2013)
- Nexus is an organizational framework which introduces a Nexus Integration Team, composed by Product Owner, Scrum master and Team Members. The Integration Team is entirely responsible for synchronizing and integrating the increments delivered by numerous Scrum teams in a specific project (Bittner et al., 2017).

Design Thinking

The tendency to make IT projects more cost-effective and more reactive to changes has resulted in development of other agile approaches, going beyond the Scrum or XP. Among them, especially interesting is the Design Thinking, primarily used for designing various services, both traditional and online.

Design Thinking identifies five closed-loop, highly interactive spheres of design activity, (Stickdorn and Schneider, 2010; Meroni and Sangiorgi, 2011), presented in Figure 5.9:

- Empathize. Understanding – taking the viewpoint of a user/customer, including physical, psychological and emotional needs when attempting to use a specific service.
- Define. Redefining – reformulating the problem to be solved from the viewpoint of a user/customer.
- Ideate. Creating – generating numerous ideas for possible solutions with help of collaborative creativity tools and under supervision of a skilled moderator who helps to overcome mental barriers in creativity process.
- Prototype. Developing – the team members (including users) spontaneously build low-cost prototypes, inject new ideas and collaboratively expand initial concepts.
- Test. Validating – the prototype is evaluated by prospective users/customers who provide the team with feedback regarding possible improvements.

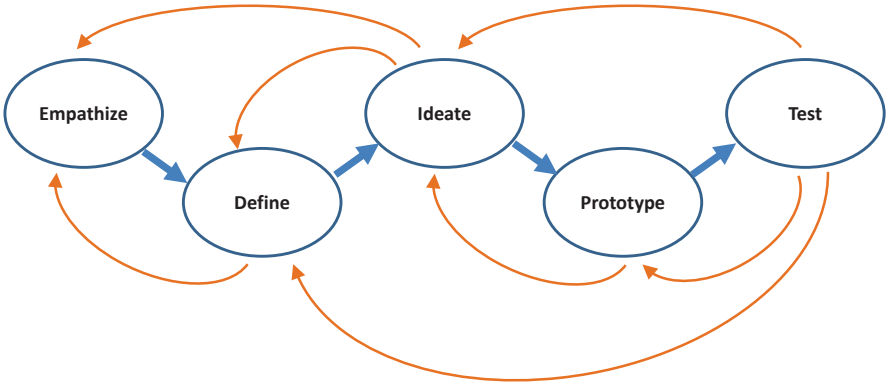

Figure 5.9. Five iterative phases of Design Thinking
(adapted from Stickdorn and Schneider, 2010)

Design Thinking approach is iterative, assuming that numerous feedback loops will be created as a result of user-based workshops and testing, ultimately improving quality of the final product or service.

Design Thinking proved to be especially useful in IT projects expected to deliver digital innovations or novel online services. In Design Thinking projects it is expected that real users (not the proxies or representatives like business customers or clients) will be involved, and in all activities, not only in selected contact points.

According to Thallmaier (2015), and Humphreys and Grayson (2008), this significant extension leads to two new forms of user activity during and after the project:

- Co-Design: active participation in creating value during the project, working together with designers;

- Co-Production: active participation in creating value during the service consumption (after the service was deployed to the market) – configuring, sharing information, publishing user-created content, strengthening relationships etc.

In the area of designing interactive services, where providing positive User Experience plays a crucial role for user-perceived quality, Co-Design and Co-Creation approaches bring an important contribution. Nevertheless, in all user-centred approaches, active participation of prospective users (customers) provides a cost-effective quality assurance which for usability works usually much more efficiently than formal measurement methods dominating in traditional software engineering practice (Sikorski, 2013).

5.5. Collaboration with users in IT projects

In recent years the methods of collaboration between the IT team and clients, users and customers have undergone significant changes.

In classical IT projects the contacts of IT teams with users ranged from non-existing to sporadic, located only in selected points of the project. It was generally assumed that the IT team was able to correctly recognize the end users' needs, or they were clearly communicated by users' representative such as the client or business customer.

In UCD and STAR models users need to be involved in selected points of the project, especially when specifying requirements, evaluation prototypes, testing usability and collecting user feedback.

In Scrum and other agile methodologies the user (customer) sometimes is present with the team, but more often is represented by a designated team member (Product Owner, UX manager or another stakeholder). Except the development phase, the user (or representative) is involved in almost all activities of the project, ranging from initial interviews, problem exploration, to user workshops and elaborating user stories and use cases, to design and evaluation of prototypes. Product Owner needs to balance business requirements expressed by the client with user-based requirements, captured during user workshops, interviews or collecting user stories.

In Design Thinking the users (customers) are present with the team in all activities and fully involved in many roles, also as a co-designers, in addition to testing and evaluation. Nevertheless, users' active participation is especially important in the first parts of the project: problem definition, analysis of user's needs, specifying requirements and preparing product concept.

Individual users may directly participate in IT projects, or – more frequently – indirectly through their representatives such as business customers or authorized team members (Product Owner, UX lead, usability manager, etc.).

In the idealised case the user is almost full-fledged member of the team, present all the time like in Design Thinking, but in most software, development projects such availability is not realistic, despite highly recommended. Anyway, in any IT projects involving even a small group of representative users full-time is better than the relying merely on communication with the client (business customer).

There are many ways in which IT projects can be conducted in business settings with methodologies presented in the previous chapter. They may differ in defining specific stages, their content and recommended transitions. Nevertheless, they all originate from an engineering roots where logical reasoning and pragmatic management coined subsequent phases of developing usable products for customers. Each phase is contributing to the project overall success and feeds into next stages of product development process. For the purpose of this book, following stages were defined:

- **Strategy**. This phase includes building the rationale for the product and for the IT project, **ideating** the product, **planning** its value for target users (consumers) and outlining the scope of a prospective IT project.
- **Analysis**. This phase includes systematic **analysis** of users' problems and tasks, **understanding** users' needs located in a specific context of use, and **specifying** users' requirements.
- **Design**. This phase includes developing task scenarios, **envisioning** product functionalities and conceptual foundations of a specific user interface.
- **Development**. This phase includes **prototyping** – preparing concepts, templates and prototypes to be evaluated by prospective users, as well as **coding**, which includes developing the architecture and the code of subsequent components creating the product.
- **Validation**. This phase includes **evaluation** and **testing** as well as collecting users' feedback for the acceptance of specific components; additionally, a **retrospective** upon possible improvements of the design process is often included here.
- **Deployment**. This phase includes deploying the product to its target environment, training end-users, and handing over to the customer all responsibilities relevant to the system **operation**. This phase often includes also preparing the contractual solution for the system **maintenance**, **improvements** and – especially in case of online services – running **promotion** and **marketing** campaigns for attracting new users (consumers) and retaining the current ones.

The next chapters of this book cover the phases listed above (except Deployment), and relevant methods of interaction design applicable mostly in agile projects. In further chapters in focus are mostly projects developing online services and mobile applications for individual users (consumers). Nevertheless, presented methods and techniques may be also useful for classical projects, and for "hybrid" projects which combine elements of classical, iterative, and agile projects. An vice versa, some methods originating from the classical and iterative approaches are still useful also in an agile design context, so they will be briefly discussed as well.

6. Strategy – envisioning the product

6.1. The outline of Strategy

The Strategy phase is aimed to:

- identify the need for the product (application, service) and its market goals;
- identify the problem to be solved and potential users (customers);
- prepare a solid rationale for launching a project which would deliver this product to the market.

The Strategy phase is to project a roadmap of a process, that would make possible achieving these goals, and would cover design, implementation, deployment and operation of a prospective product. The decisions made while shaping a product strategy will have a significant impact on the remaining phases of prospective project.

The Strategy phase is not aimed at presenting specific proposals or solutions: it is rather expected to deliver a vision of the product, realistic enough to convince decision makers and team members to engage. The decision about launching a project should be based on facts, data and observations suggesting that there are users (consumers) awaiting such a product and eventually ready to pay for a specific solution.

From the users-centred perspective these goals (activities) can be redefined to envisioning a product which should perfectly respond users' needs resulting form a problem situation the product (application or service) is to solve. Therefore prospective users (customers) should be reached out and invited to participate not only in the Strategy, but also in next phases of the prospective project.

User research activities planned in the Strategy phase are limited to collecting only basic information, necessary for initial identification of users' needs, preparing the product concept and rationale its design and development.

Continuous focus on customer's needs is the element critical for avoiding bad design. Collaboration with user should be established in the Strategy phase as a set of principles:

- users should be involved throughout the process;
- identify user needs – establish requirements;
- develop alternative designs to meet these;
- build early prototypes that a can be communicated and assessed by users;
- frequently test and evaluate what is being built throughout the process (not only at the end);
- iteration is a part of the process – often by abandoning previous concepts or prototypes.

As a result of the Strategy phase, the outline of a prospective IT projects should be delivered, to be accepted by relevant decision makers (sponsors, board, etc.).

6.2. Identifying the problem

The key issue when working on the Strategy is correct understanding the problem situation, and what exactly prospective users need. The best way to cope with this issue it is taking a sample of observations from real life:

- observing users in real-life situations exactly when the problem shows up;
- studying users' tasks when they attempt to solve the problem with existing apps and other methods;
- interviewing users and discovering why existing solutions are not satisfactory, thus opening a gap for innovation.

Because there are now plenty of IT solutions available for any purpose, finding a revolutionary idea for a completely novel product, service or app may be extremely difficult. More realistically, an idea for solving an existing problem in a better way may seem more feasible. It all starts from a quick glance around – locating the user or customer who faces a specific problem and needs a quick and efficient solution. However, this approach works well largely for IT products located out of professional software domain, where a more formal, systematic approach needs to be applied.

Nevertheless, if the case is finding a problem to solve with a mobile app or service, there are still plenty unresolved issues, in both individual and social dimensions. In the individual dimension everyday life brings many examples, such as:

- driving-related problems such as problems with finding a place for parking, or avoiding traffic jams;
- personal financial management, optimizing household consumption;
- personal health and dietary nutrition, rush and nervous lifestyle;
- work-life balance, social relations with family or friends;

- travelling, curiosity about the world, passions and hobbies;
- spiritual sphere, enriching personal lifestyle, helping those in need.

In the social dimension, potential ideas for a novel app or service may address areas such as:

- improving security for children or for the disadvantaged,
- collaborative education for gifted children;
- integrating local communities, public or social health, ecology, quality of life
- social relations, tensions, prejudices, racism, migrations, etc.

Social domain is obviously more complex and designing interaction will always include social media or other channels of community-based communication.

For both individual and social lives there is always area for endless innovations and improvements possibly enabled by online services and apps – brand new or redesigned from existing ones.

After identifying an interesting problem area, envisioning of the product (app or service) leading to problem solution should be done. It is usually a highly personal process guided by individual creativity skills. In this process adequate reprocessing of knowledge acquired from observation is essential. Before reaching an acceptable outcome, following steps take place:

- Preparation: preparatory activities to find a solution – reformulating the problem, collecting and classifying information, analysing the context;
- Incubation: the process of subconscious understanding of the core of a problem, based on subconscious and conscious processing of information collected during preparation;
- Enlightenment: the revelation – a more or less sudden appearance of a long-searched idea as a result of an unexpected association or “jump” of intuition;
- Verification: initial checking suitability of idea for solving a given problem, and its practical feasibility.

In this process the Incubation phase is crucial. It takes usually longer than expected, and it involves many subconscious cognitive activities, in which the information collected from external world gets combined with internal exploratory understanding of the problem.

Beyond individual, internal activities, the creative process of product envisioning is strongly fuelled by positive reinforcements from the external environments. Apart from information and stimuli incoming from readings and observations, communication with others is very helpful in this stage. Different forms of brainstorming, working in pairs, exchanging views, cognitive trips and simulations help to discover and explore new aspects relevant to prospective product.

6.3. Identifying users' needs

After an initial envisioning of a prospective product, it is time for collecting hard data which may convert the product idea based on visionary intuition to the product concept based on objective facts.

Now a more systematic search is required to gain a deeper understanding of user needs, based primarily on first-hand sources: evidence-based data, verified factual information, and credible references. We need especially facts and stories which describe users' habits, needs and motivations, helpful in prospective converting users into loyal consumers. This knowledge will be essential in finalizing the product concept is Strategy, but it will undergo many refinements in further stages of the prospective project.

In order to gain this knowledge it is necessary to reach out to a small group of real users, who are familiar with the problem and who preferably use already existing solutions (apps or services). The best starting point is a direct on-site observation when users face a specific problem and unsuccessfully try to solve with existing apps or other methods.

Conducting several in-depth interviews with representative individuals needs to address issues such as:

1. Which functionality of the app is critical for achieving the complete solution of the specific problem?
2. How would you describe problem solution as a complete one?
3. Which apps and services available online do you use now for solving this problem?
4. How far are you satisfied with these apps (or with your favourite app)?
5. What is still missing in these apps, what are the gaps in their functionality, usability and UX?
6. What are the difficulties, concerns or limitations you experience when using these apps?
7. Do you share your experiences on solving this problem or using this app with other users?
8. What advantages could convince you to start using an alternative app for solving this problem?

The outcome of user interviews should be extended by a thorough desk research: studying existing alternative solutions (apps and services online), further exploring the problem space using case studies published online, reading reports helpful in analysing potential markets or in identifying potential business goals.

Information found in internet, combined with information from real users (potential customers) may reveal possible type of competitive advantage and top value to be offered by the product for customers.

Optionally, the outcome of initial user research study can be extended with:

- Persona – a brief profile of a typical user (consumer);
- a description of context of use – the specific environment or situation where the product will be used;
- a short video presenting a prospective product in a simulated use by target users, in a way similar to seen in tv commercials.

Nevertheless, the scope of this study should be strictly limited to the goals of Strategy phase: identifying basic needs of prospective users (customers), specifying the problem to solve, and projecting a vison of the product (app, service, solution) and its potential market.

6.4. Presenting the product vision

Presentation of product vison to a specific audience (usually the decision makers) is usually delivered in as a slide show, with a text document to follow if the concept was accepted. Product vision is supported by three interrelated documents: product plan, project plan and business case.

Product plan

The objective of a product plan is to convince the decision makers (the board, prospective project sponsors or local sceptics):

- that the product will be successful because it comes to meet a vital need of real customers;
- that it is worth launching a project that would deliver this product and deploy it to on the market;
- that sufficient resources (people, technology, skills, etc.) are available to make this project successful.

A product plan presents the product concept in a structured manner, usually covering following sections:

1. Problem and solution statement
   - prepare a clear definition of the problem the app will solve for users (customers);
   - identify conditions which make the solution complete for the suer (customer).
2. Audience definition
   - identify who will download or buy your app, basing on the profile of representative customers who already use existing apps/services;
   - create user profiles (personas) of customers based on initial user research and user feedback;
   - describe key needs of your users (customers) relevant to expected solution and essential functionality of an app or service.

3. Gap definition
   – perform desk research: existing alternative solutions and analysis of competitive products;
   – perform user research: existing apps/services and their deficits, as unsatisfied needs declared by users;
   – identify unique functionality and competitive advantages of a prospective product.
4. Market definition
   – explain where to find customers for the new app, how many, and what are distinctive features of this market segment;
   – explain which features will attract the audience and which advantages of the product are driving potential revenues.
5. Technical feasibility
   – explain which technologies have been used by competitive products (apps/services);
   – point out which other technologies seem to have a promising growth potential for this product.
6. Business potential
   – highlight product uniqueness addressing the vital needs of the market (who, where, how many);
   – clarify why people would use this app (at least three strong reasons) and possibly pay a subscription fee;
   – explain main business goals relevant to the prospective product;
   – propose a simple business model depicting value flows, competitors, partners, communication channels, relevant costs, revenues (Osterwalder 2010), and explain how this business model could grow, including the CRM approach and community development.

If the product plan was accepted, further documents need to be prepared such as:

- product brief – a synthetic, textual version of the abovementioned product concept;
- project plan (or a project brief) – a document which presents an outline of a prospective IT project in which the product will be created;
- business case – a document presenting business rationale for developing the product and locating it on the market.

Project plan

A project plan is a document or a slide show which presents an outline of a prospective IT project in which the product will be created. A project plan should reveal strategic concepts, management frameworks and operational details regarding for instance:

- proposed organization of the project (the leader, the team, methodology of project management etc.);
- sources for acquiring necessary resources (team, technology, support, legal/admin, financial etc.)
- the sources for recruiting competent people for the project;
- the scope of analytic works: identifying end user needs and specifying main customer requirements;
- cooperation with the client (business customer) and other stakeholders during the project;
- risk management model for the project and its ability to cope with sudden changes;
- dealing with existing constraints (platform, infrastructure, business networks and alliances, etc.)

Before starting a project, following questions should be also answered:

- Is the purpose of the app to build revenue for the company through mobile sales?
- Is it to serve as the beginning, middle or conclusion of a series of apps?
- Will it drive traffic to the company website or physical retail sales by increasing brand awareness?
- Will the app exist to provide service also to existing customers?

In business practice often the **project brief** document is used as an easy-to-read, abbreviated version of product plan, presenting the general framework of the project, its goals, audience and the most essential data.

Last but not least, when planning the project, technical considerations should be not neglected. For instance, if the intended product is an online service or a mobile app, especially platform-related decisions should be made as soon as possible, for example, smartphone or tablet dominant; iOS or Android-centric; and single-platform or cross-platform development.

Strategizing the primary platform of an app will determine user interface design requirements. Not only the iOS and Android versions will require different software development environments and different organization of the project, but also a tablet-first app will have a different user interface than the one designed primarily for smartphones. The apps intended to “push” users to a physical location (a shop, a branch) will have functions and solutions different from those designed to sell goods or services directly online.

Technical considerations and constraints should be considered early, as they may critically affect the final shape of the product plan, equally as circumstances related to the market, budgeting or expected customer behaviour.

An optional supplement to the project plan may include the outline of post-project activities such as:

- deploying the app or service to the market and promoting it to end users (customers);
- retaining the users by attractive loyalty programs and building cooperation networks;
- conditions for long-term securing financial background, scalable IT infrastructure and other vital aspects of business logistics.

Business case

Business case is a document which should clarify why developing the product and undertaking this project is reasonable from a business point of view.

In addition to the product and project plans, business case will include information such as:

- high-level business goals for the product and measurable success criteria,
- understanding why the organization wants to create the product (app or service), regarding market expansion, building competitive advantage, expanding product portfolio or other reasons;
- assessing own development potential compared to competition, based on benchmarks from other apps in the competitive space;
- projecting the costs required to gain the competitive advantage; sometimes an effective improving of an existing solution (an app or service) that works can be preferred rather than reinventing the wheel;
- financial data, cash flows, revenue estimates etc. describing how the product can operate on the market; the costs incurred during the development (the project) should be separated from the cost incurred during operation (including maintenance);
- the business model outlined in product plan should be expanded to details, and its potential for long-term relationships with customers should be explained;
- relevance of financing the product development with technology and infrastructure should be included, preferable as variants to be considered before final decisions will be made.

After analysing the content of the business case, decision makers should be able to reflect on:

- determining which is the best business model for the product;
- composing a clear statement of what the business sees as the goals of the product;
- proposing a clear definition of product success goals and setting for them measurable metrics;
- making sure that the goals are not so numerous and diverse that they become contradictory or competitive.

Possible means which make the product vision presentation more convincing the decision makers include for example:

- short, dynamic slide show of the product concept (called also “design pitch”), illustrated by data collected from real users;
- additional items (“props”) such as: a mock-up prototype, a demo, a video prototype.

6.5. Deliverables from Strategy

The Strategy phase should deliver the outcomes such as: key problem identified, a rough idea of the product and approximate description of target users (customers). For decision makers, this presentation should combine author’s visionary intuition with results of user interviews and with facts and data which confirm the legitimacy of proposed concept.

The deliverables from the Strategy phase (the product plan, project plan and business case) are complementary (Figure 6.1). After the presentation and critical discussion, they should have ensured the decision makers and potential sponsors that:

- the vision of the product in based on actual needs of potential customers;
- there is a facts-based gap on the market to be filled in by the product with an attractive business potential;
- developing this product is feasible with available resources, as well as launching it to the market.

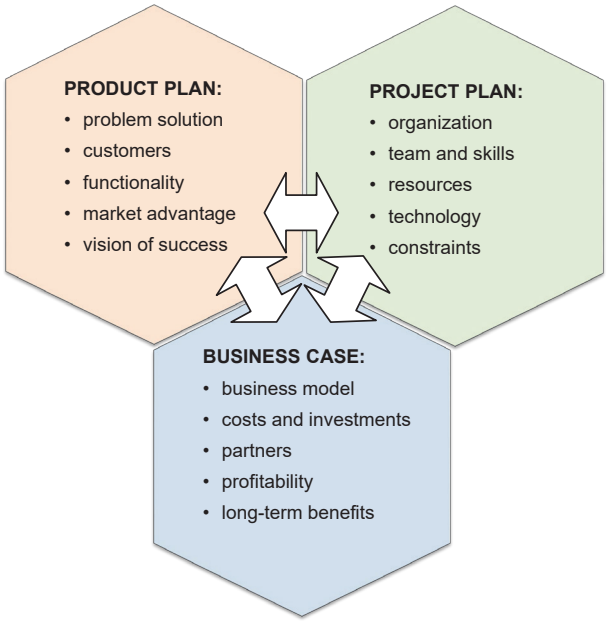

Figure 6.1. Interconnected product plan, project plan and business case

Product vision is complete with all their deliverables, nevertheless in practical setting preparing the detailed business plan if often delayed till the project unrolls, more information is collected, and the product is already in design and development. For this reason, a more systematic user research is required, covered in the Analysis phase, presented in the next chapter.

7. Analysis – understanding users' needs

7.1. The outline of Analysis

The Analysis phase is intended to conduct a thorough study of user's tasks, problem domain and the environment in which the prospective system will be operated. The Analysis builds on outcomes of Strategy, especially the product plan, but it explores the problem area in a more systematic, analytic manner.

In classical projects the Analysis phase is aimed to deliver specification of user requirements to drive next stages of the project. Requirements are usually acquired from the client, who is expected to express technical and business requirements, as well as to present requirements from users (system operators) who are not involved in the project.

In agile projects user requirements are collected largely as user stories directly by interviewing the users during user workshops. User stories are next converted to the product backlog, representing projected features of the product.

While in classical projects the focus of the Analysis is identification and specification of requirements, in agile projects the goals are twofold: firstly, understanding users' needs and expectations, and secondly, exploring new opportunities for innovation to make the product more attractive for customers. This difference is cause by the fact that in classical projects usually IT systems for business and industry are developed, while agile projects largely deliver web-based products, services and mobile apps for customers.

7.2. Identifying users' requirements

In classical projects collecting user requirements is particularly important, because the cost of usability errors caused by incorrectly specified requirements are especially high, due to communication difficulties between designers and users.

Poor interactivity with the user and lack of frequent feedback makes early detection of usability flaws virtually impossible unless the elements of iterative User-Centred Design process were applied.

Categories of requirements

In IT projects multiple categories of requirements must be considered (Sommerville, 2016; Pressman, 2000), briefly described below.

General requirements

- General requirements include issues such as business goals and context of the system, the scope of the system and plans for its evolution, controlling the impact of stakeholders of the system, or providing compliance with standards and legal regulations regarding financial security or other operational requirements.
- Examples: reduction in customer complaints by 50%; shortening account settlement time to 48 hours; achievement of full compliance with mandatory regulations.

Functional requirements

- Functional requirements directly relate to available system functionality, the types of processed information and exact descriptions of input to output state transitions. They have a direct impact on user task performance regarding the operation of the system (application or service), as well as on user (customer) satisfaction.

Examples: ability to manually update the prices of products; administration of registered users; creation of sales reports.

Non-functional requirements

- Non-functional requirements are not directly related to system functions, but to "invisible" system quality characteristics such as reliability, response time, safety, security, credibility, and system responses to user errors. Particularly, requirements for usability and the user interface fall into category of non-functional requirements. Non-functional requirements should be formulated in such a way to make them possible to validate.

Examples: ease of learning the system (validation: e.g. 90% of users are able to learn basic functions in 2-hours of training); visual attractiveness (validation: e.g. 90% of users declare they like the look of the system); reliability (e.g. max. one failure per month).

Limitations of the system environment

- Limitations of the system environment, which are a subset of non-functional requirements, define the system requirements (architecture, hardware, etc.), conditions of cooperation with other systems (devices) and requirements for standardization, documentation, operational and diagnostic procedures, etc.

Examples: the application will be running on MS Windows; application must be running on a specific type of database; minimal hardware requirements; ability to perform system maintenance remotely.

Implementation requirements

- Design and implementation requirements include issues such as constraints for time, budget and project team; available technology and existing standards; organization of work, design and testing methodology; preferred technologies and software development tools; methods of implementation of the system at the client's site.

Examples: project duration maximum six months; project will be implemented by internal staff only; project team will permanently work with five end-users delegated by the client.

Requirements acquisition and management

The main source of requirements are project stakeholders, for instance the clients, business customers, management staff of the client's company, the project team, experts, and subcontractors. End users are also important stakeholders, unfortunately too rarely contacted in classical projects.

Acquisition of requirements in classical software projects is usually the responsibility of an analyst, who is a member of the project team. The outcome of the analytic work should be delivered as a Requirements Specification document, which contains a general description of the system, and the description of requirements from relevant groups of stakeholders.

Managing requirements in an IT project is a difficult and responsible task, because of their strong impact on the quality of the resulting product. In acquisition and updating requirements frequently occur problems such as:

- limited experience of the analyst, lack of available analogies, difficulties in identifying and formalizing requirements for the system to be developed;
- knowledge related to user requirements is not normally available in the finished form, it is distributed, subjective and incomplete;
- lengthy discussions how to specify the requirements and how to determine the conditions for their acceptance in the finished system;
- non-functional requirements are often specified by imprecise terms, making their verification and validation very difficult;
- difficulties in communication between the team and the client who is often unable to articulate actual needs or does not know exactly what is feasible.

Persisting difficulties in defining requirements contributed to the emergence of a software engineering discipline called Requirements Engineering, applied often in classical projects, whose task is to develop effective methods of identifying and updating requirements across the whole project lifecycle.

Selected techniques for requirements identification

The most popular techniques used for requirements identification will be briefly presented hereafter (Sharp et. al., 2019; Jayaswal and Patton, 2009; Dix, 2004).

Context of use analysis

The context of use analysis is often regarded as part of a business analysis, because it is a "reconnaissance" before the start of the actual design work. Performed on site where users work, it not only facilitates identifying requirements and risk factors for the project, but also helps in establishing direct contact with prospective users.

Context of use analysis consists of three main components (ISO 13407):

1. Description of user characteristics: describing the profile of typical users, their task objectives and benefits expected from the use of the system.
2. Description of user's tasks: description of work processes, document workflow and organizational conditions affecting the use of the system.
3. Description of environmental conditions: identification of technical and environmental conditions affecting the way how users' work is actually performed with the system.

Task analysis

The detailed description of user tasks is extracted from the context of use analysis and documented as a separate area of the project, including:

- detailed diagrams, flowcharts and other models describing users' tasks, their priority and criticality as well as the type of used data sources;
- approximate percentage of time users are working strictly with a computer;
- identification of quality requirements as to the task performance;
- identification of results achieved by the user in various situations, including disruptions and their impact on job performance.

The descriptions of user tasks should include not only on activities and facts directly associated with the system, but also other processes and work-related objects such as paper archives, folders, boards, meetings, phones, etc. Task analysis should also identify systems used currently and users' opinions on their usability. Identification of user's needs and preferences on how the work should be performed, expected division of labour, the order of tasks, etc., is also recommended.

Documentation analysis

Documentation analysis is often used for the description of the current forms of work organization and procedures. The following documents are analysed:

- process maps, work manuals, procedures and instructions how the task should be performed;
- historical documentation and archival data, which allow to estimate the throughput of work systems and their organizational characteristics;

- "best practices", as proven methods to perform tasks that might be considered for the proposed new system.

An additional benefit from the documentation analysis is that the employees (system operators) explain to the analyst how the process ideally should proceed and how the work tasks should be carried out. The discovery of new knowledge takes place, and the designers learn what obstacles may cause the expected result not delivered as planned.

Business process analysis

Business process analysis (often known as business analysis or business environment analysis) includes:

- description of technical conditions, such as: existing systems: hardware, software, telecommunications and network equipment, system integration plans, planned investments; tools used in data processing, documentation, maintenance of systems, timeliness and completeness;
- description of work organization, e.g. business logics, business process map, circulation of documents, diagrams of the process workflow, teamwork, communication; division of tasks, responsibilities, evaluation feedback about the work quality and performance; the impact of forced work pace, bottlenecks, disruptions, delays, their frequency and significance;
- description of the conditions related to management, e.g.: organizational structure and processes in the organization, management style, teamwork, personnel management, internal communication channels; relationships with business partners, cooperation networks, customer support and brand image;
- description of the physical work environment, e.g.: location: office space, lighting, noise, microclimate, space and furnishings, hazards, work outdoors, travel, protective clothing, personal protections etc.

Within business analysis for creating formal models of the process multiple diagramming notations are used, such as Data Flow Diagrams (DFD), Entity Relationship Diagrams (ERD), Unified Modelling Language (UML) or Business Process Modelling Notation (BPMN)

Analytical meetings

Analytical meetings are regularly held during preliminary stages of the project, but in classical projects they can take place also in later phases, such as design, development and testing. Analytical meetings are moderated by the project manager or the chief analyst. They usually take place at the client's premises and involve representatives of the client company, the project team, and sometimes also other invited persons.

The purpose of analytical meetings is to agree the stakeholders' viewpoints on the scope and functionality of the system, to develop a list of requirements and to agree on the final specification of requirements. The work with the client is based

largely on step-by step consenting on subsequent requirements, and team-based production of documents related to the actual project pragmatics.

Analytical meetings and business process analysis cover much broader area than the context of use. They go beyond the individual workplace level, to management-related issues at the organization level. Although business process analysis performed by a business analyst with the client can be sufficient for projecting the basic functionality of the system, for achieving high usability it is highly recommended to also use user-centred techniques described hereafter.

Participatory observation

A direct observation of users' work is the conventional technique for the context of use analysis. However, it is ineffective as a merely passive observation, even if the researcher (analyst) spent much time in the user's workplace. Participatory observation means that the researcher (analyst) actively participates in regular tasks performed by the user or in related auxiliary activities (Figure 7.1).

Participatory observation is often associated with the ethnographic approach, in which the researcher goes to the environment, where members of the studied community are living or working, and participates in their daily routines to make a better contact with them and to gain the most valuable research material. Similar approach works well also in the area of user-centred design for IT systems, because delivering usability without user observation and detailed user research is virtually impossible.

"Shadowing" is a particular version of participatory observation, while the researcher follows step-by-step all operations performed by the user (Figure 7.2). It can be used both for analysing user operations in an existing system, as well as for optimizing user operations in a newly designed prototype.

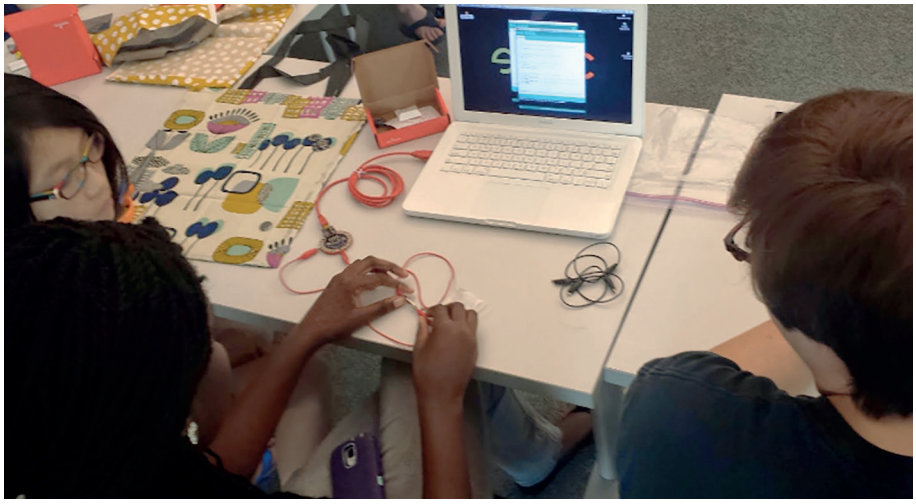

Figure 7.1. Participatory observation
(Credits: https://techknowtools.com)

Figure 7.2. Participatory observation – "shadowing"
(Credits: https://www.liveworkstudio.com)

Individual interviews

An interview with an individual user may be performed as formal, informal or semiformal, depending on how rigid the interview scenarios are applied. It also depends on the scope of intended questions, regarding for instance current methods of work, what disturbances may happen, how they are coped with etc.

Individual interviews are very useful for the exploration of specific topics or problem areas, but time-consuming for interviewees (users) who are usually distracted from their current activities. They are also time-consuming for the analyst, and with a larger number of respondents can be also costly due to the difficulty of compiling data collected from numerous interviews.

Group interviews

Interviews conducted with a group of several users are very valuable because users possess unique work-related knowledge which shows up only during the moderated group discussion and could not be normally acquired through individual interviews. Group interviews allow the analyst (moderator) to save much time, but they require adequate preparation beforehand to be efficient, and a skilled person for moderating, making handnotes and controlling audio recordings.

Processing results from a group interview is difficult, because collected data are purely qualitative (text, sentences, or emotions) and there can be significant differences in opinions expressed by the participants. Nevertheless, group interviews are widely applied in all situations where time is precious or opinions from various groups of shareholders must be obtained.

Survey questionnaires

Survey questionnaires are useful for collecting data from a larger number of respondents. Questionnaires distributed on paper or online allow to collect quan-

titative data as well as qualitative (descriptive) information using closed and open questions, accordingly. Questionnaires are generally tedious for users to fill in, so they should be short, containing only questions regarding essential requirements.

Survey questionnaires are useful for collecting data primarily on existing systems, because users have difficulty in expressing opinions speaking about the future or unknown solutions beyond their current experience. Otherwise, for projecting future solutions interviews and user workshops are more suitable.

Summary of classical requirements-related methods

The abovementioned methods for requirements identification mainly serve the projects devoted to design and development of IT systems for supporting work-related activities in offices, industry, services etc.

In real settings, for adequate identifying user requirements in classical projects usually a mix of techniques must be used, combining interviews, observations, documentation studies and – most importantly – user workshops. Knowledge gained from users is volatile and difficult to restore later even with possession of notes. For this reason, data analysis should begin as soon as possible after collection of data. The best practice is to review the collected materials, grouping them into thematic categories, and then formulate preliminary conclusions yet prior to the deeper, detailed analysis. They can be corrected later on, during user workshops (Hartson and Pyla, 2012; Gottesdiener, 2002), described in the next section.

7.3. Understanding users' needs

In contrary to classical projects, in agile projects collecting requirements is less formal. Moreover, it often goes much beyond requirements identification alone, reaching also understanding users' needs and expectations. This expansion may also reveal opportunities for novel features that make the product more attractive for users or buyers in the consumer market.

Selected techniques for understanding users' needs

Expanded context of use

Similarly to classical projects, expanded context of use also includes characteristic of users, their tasks, and the environment. However, it goes beyond a mere description what the user does, attempting to explore possible scenarios beyond routine activities.

Issues such as disturbances or obstacles from the environment are explored in order to propose novel functionalities and features which would mitigate the effect of annoyances to user experience. Simulating task scenarios which include abnormalities during the product operation help to design solutions that will be tolerant to human errors and to hardware or environment.

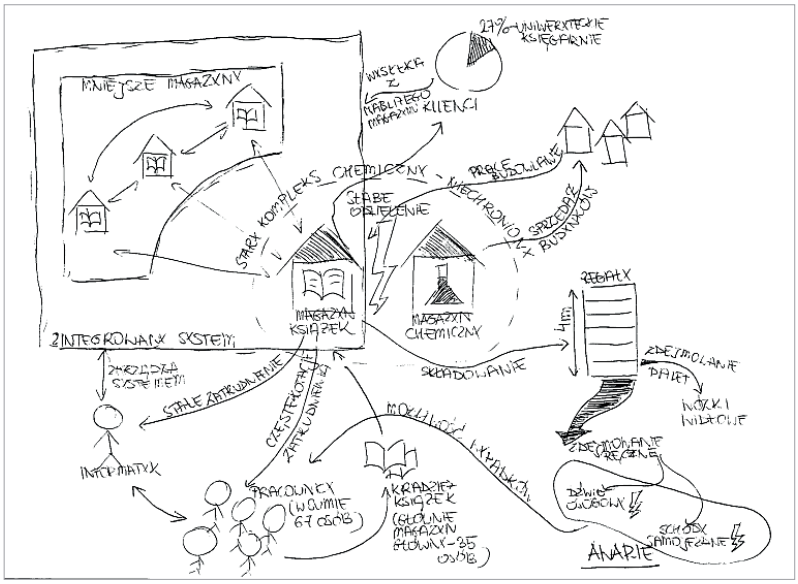

Figure 7.3. "Rich Picture" for expanding context of use of a business IT system (Credits: Agata Bojke)

Techniques such as storytelling, freehand sketched Rich Picture (Figure 7.3) or storyboards, are extremely efficient for team-based exploration of context of use. Techniques such as "Five-Why" or "How Might We…" can be also useful for discovering sources of anxiety or reluctance experienced by the user (consumer), as well as for discovering a new value to be offered for customers.

Ethnography and participatory observation

In interaction design ethnography-based observation means a direct focus on understanding users' habits and behaviours by active participation of researcher in users' daily activities. Comparing to participatory observations, in ethnography-based observation a researcher (analyst) is even more active, taking part in all users' activities and local "rituals" relevant to work of after-hours pursuits. The observer-ethnographer is "immersed" in the culture that they study, discovering regular habits and understanding users' behaviour when using a specific product, service, or app. As a result, ethnography-based observations usually take longer because of a broader scope of user research.

The benefits from ethnography for interaction design include drawings, explanations, and stories helpful in better understanding of an existing situation. Novel aspects regarding the context of use and user's habits can be thus revealed, such as new relationships discovered among users (customers), their tasks (problem to solve) and the environment in which the problem is located.

Interviews-insights

Insights are specific interviews aimed to reveal interesting and surprising observations ("insights") that can be used as a starting point for cognitive explorations resulting in the discovery of a novel solution. By getting to know users and their motivations, values and behaviours the analyst will find "insight" related to the key aspects of the problem.

The analyst prepares a list of expected information before starting the interview. During the interview, the interviewer must be open to newly emerging circumstances, rather avoiding direct questions, instead encouraging the subject to tell about own habits, behaviours or views on a specific topic. The interviewer may change the order of the topics, as well as their form and content of prompts, adapting to the subject's behaviour and the dynamics of the narrative.

Empathy map

Empathy map is a template-based interviewing technique originating from the Design Thinking, aimed at gaining a deep understanding user-customer' problem and "entering" into user's (customer's) problem situation to be solved.

The template shown in Figure 7.4. is used for collecting information helpful for identifying user's expectations, even if they are unspoken. Most importantly, motivations and barriers relevant to using (or rejecting) a specific online service or app are also discovered. Identification of expected benefits, but also limitations and anxieties hindering the user from using the specific app or service, offer guidelines leading to design of relaxed operation and more trust towards the product, service or app.

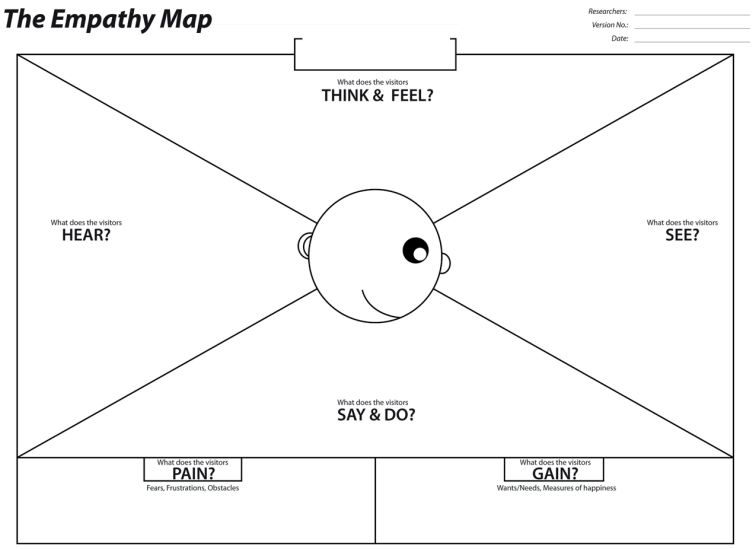

Figure 7.4. Example of a template used for Empathy map
(Credits: Osterwalder, 2010)

Persona

Services and apps developed in agile projects often aimed to support activities performed a way out of professional activity. If they are related to entertainment, education or lifestyle, more non-work-related information about target users will be needed.

Persona is often used also in classical projects, but in a format limited to describing basic characteristics of a typical user and work-related tasks the user wants to perform with the system.

Persona used in agile projects in an expanded version, apart from a brief description of occupational activities of a target user (consumer), usually also includes information about life goals, hypothetical personality traits, motivations, frustrations, and favourite brands (Figure 7.5). Expanded Persona may contain any information collected during observations, interviews, and Empathy maps.

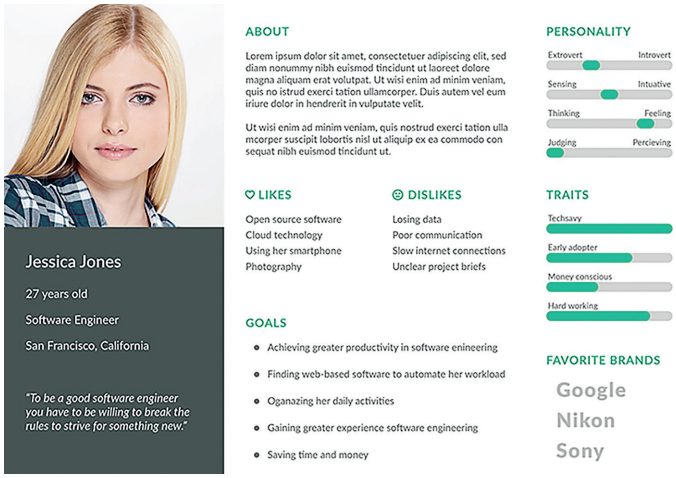

Figure 7.5. An example of expanded user profile – Persona (Credits: https://psdrepo.com)

Storytelling

Storytelling is a popular, fact-based, narrative technique often used during user workshops. Storytelling is focused on encouraging participants to personal reflection on actual experiences and observations regarding specific situations, products, or services. Storytelling session must be moderated (Figure 7.6) and is usually supported with visual add-ons or digital props such as video, prototype or a relevant object.

During the session users tell their stories regarding the topic (e.g. frustration with specific IT system or service) and thereafter they turn for collective projection of new experiences and solutions as expected from the project. Storytelling session often coverts into a workshop producing three projections: before-and-after, difference (roadmap) and call to action for change.

Figure 7.6. A storytelling session
(Credits: https://freepik.com)

User stories

User stories are typical task scenarios as told by actual end-users (consumers), and not the client. In the first round they can be verbally expressed stories how user preforms a specific task. Finally, using sticky cards, they should be converted to a synthetic format which includes: the stakeholder, representation of a specified need (intended action) and value to be delivered for user (customer) after the user action was completed (Figure 7.7).

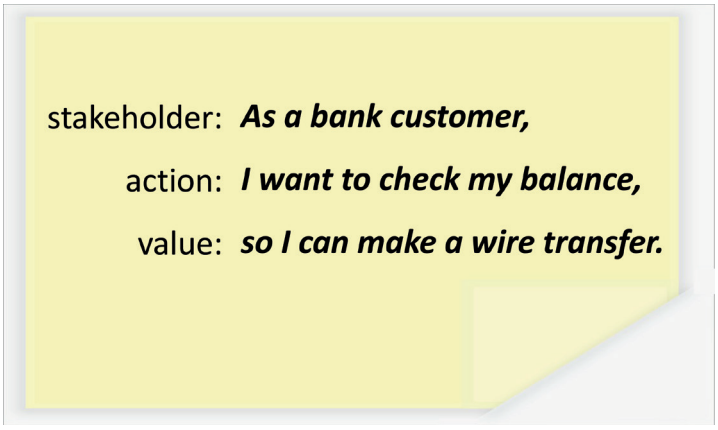

Figure 7.7. A user story – example

All collected user stories, build the projected functionality of a prospective product (system, service or app). Further exploring collected user stories within a specific context of use can reveal new user needs, new scenarios and novel features making the product more attractive and innovative.

User story mapping

User story mapping (Patton, 2014) is a popular technique for projecting users' tasks performed in a specific process. In Figure 7.8. the arrow depicts the flow direction of the main process, while the cards beyond (user stories) represent its main steps. The cards (user stories) placed in relevant columns below depict actions included in each main step, sorted according to their priority for the user. Specific selections of cards marked by horizontal lines represent functionalities to be implemented in subsequent versions (releases) of the product.

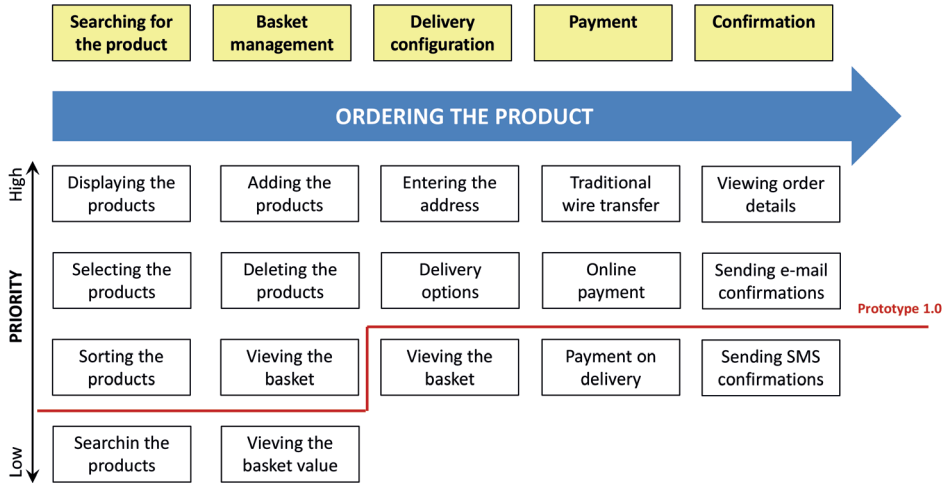

Figure 7.8. User story mapping – an e-commerce website example (Adapted from https://4ba.pl)

User story mapping is primarily used by the team to prioritize user stories for preparing product backlog and scheduling jobs (planned product features) to be located on the Kanban board.

User story mapping is often used also in meetings with the client for prioritizing key functionalities of the product. This technique is also helpful for common understanding of the product by the team and by the client, as well as for novel interpreting the functionality of the product in beyond current context of use. The main limitation of user story mapping is its inability to show possible relationships among relevant requirements (user stories), what may seriously affect the timing and complexity of development works to be done.

User workshops

In agile projects user workshops are regularly conducted for early validating outcomes resulting from current sprints. User workshops (Figure 7.9) may include activities such as moderated teamwork with informal interviews and discussions, identification of problem components and envisioning possible solutions and prioritizing design issues. User workshops may be facilitated by using "props" – specific

artefacts aimed at prompting participants, e.g. a prototype, video, scenario or device related to the topic of the workshop.

User workshops often use the teamwork with sticky notes on a whiteboard, which open unlimited opportunities for collective requirements elicitation, planning information architecture, user story mapping or using affinity diagrams for discovering innovative features of a prospective product.

Figure 7.9. User workshop
(Credits: https://xnsio.com/)

Affinity diagrams

Affinity diagrams is a flexible, workshop-based teamwork technique using sticky cards on a whiteboard to stimulate visual thinking of participants (Figure 7.10). The cards bear words, phrases or symbols depicting specific meanings, relevant to the problem under study. Team members collaborate in converting the contents of the whiteboard according to the session objective, by grouping, classifying, sequencing the cards, as well as finding out new connections and relationships. The main purpose of affinity diagram is to convert a chaotic collection of ideas (items, objects) into specific categories (clusters) which can be labelled with a meaningful keyword (topic, name).

Affinity diagrams are now extremely popular for design teams and for user workshops. It can be used at all stages of the projects for any conceptual and design activities. Furthermore, affinity diagrams do not need a skilled moderator, it is self-organizing technique for the team which quickly integrates for teamwork both users or clients and the design team. Affinity diagrams are helpful in redefining the problem and are essential for stimulating creative teamwork in concepts mapping, prioritization of items I brainstorming-related activities.

Figure 7.10. Affinity diagrams during a teamwork
(Credits: http://www.msrblog.com/)

7.4. Deliverables from Analysis

Comparing to Strategy, the techniques applied in the Analysis phase are aimed at:

- identification of basic requirements (functional and non-functional);
- understanding of users' needs, expectations, and feelings;
- identifying task scenarios and projecting expected functionality of the product;
- understanding context of use and local environmental factors that may disturb the smooth operation of a prospective system (product).

The Analysis results in significantly deeper understanding of users' needs comparing to what was discovered in the Strategy, when preparing the product plan (product concept) was the main goal.

Nevertheless, in the technological stream of the projects deliverables such as logical data models, DFD, ERD and UML diagrams or BPMN process maps form an indispensable fundament for software engineering works making the product ready-to-use.

In agile projects, the techniques presented in this chapter serve not only for collecting and recording information about users' needs. Moreover, in the product stream the focus is also on innovation, so collaboration with users (customers) takes place using more intensive interpersonal communication than in classical projects. User-based exploratory analysis helps to discover novel aspects for the product by "reframing" – redefining the problem, Persona and product functionality, as expected by prospective users (consumers).

The deliverables from the Analysis phase should provide a complete input to conceptual work to be performed in the Design phase.

8. Design – converting visions into concepts

8.1. The outline of Design

The Design phase is aimed at creating the concept of a prospective system, built upon information collected in the Analysis phase.

While the Analysis describes an existing situation and problem to be solved, Design is looking ahead – continues expanding current viewpoints, further redefining the problem, the context of use and value for the client and for the user (customer). Design draws on both: formal models related to software engineering, and on "soft" concepts resulting from user workshops and other activities related to interaction design.

In IT projects – in the Design and in other phases of the project – three parallel streams (workflows) can be distinguished:

- the product stream: all activities aimed at creating the concept and its refining till the final product – maximizing value for users (customers);
- the technological stream: preparing foundations for system architecture, software platforms, cloud services and other components of the IT infrastructure – maximizing engineering excellence and making the product operational;
- the business stream: preparing and refining the business model for the product or service, making it profitable –maximizing value for the client.

This chapter is devoted to design activities relevant only to the product stream, in which interaction design and UX-related issues are largely located. Design activities take the input from the deliverables of the Analysis: description of problem to be solved, user Persona, task scenarios, user stories with projected product functionality and features expected by prospective users (customers).

8.2. Conceptual design

The scope of conceptual design

Conceptual design for an interactive product involves several areas:

1. System architecture: software engineering and technological solutions making the prospective system feasible for implementation, coding, deployment and maintenance. This component of design is invisible for users (as largely located in the back-office) but it strongly affects system performance, speed and other UX-related factors.
2. Information architecture: the structure of content, menus and other components of the process users are going though when using the product. Information architecture can vary a lot, from strictly hierarchical to network-based. The type of information architecture and how the content was divided into modules or sections affects the complexity of navigation, system usability and UX related to a specific product.
3. Visual design: the elements of screen design such as layout, consistency, colours, icons and images affect product attractiveness and pleasure-related part of UX. Overloading users with bright colours or useless ornamentations deteriorate system ergonomics and usability, what leads to negative UX. Visual aesthetics based on minimalism and elegance should be present across the whole product: in screen layouts, visual metaphors, images and user controls.
4. Interaction design: includes the task flow (user flow, or user journey) guides the user through subsequent steps (screens) of the system operation, navigation and compliance with users' expectations. User interface design is a crucial part of interaction design, critically affecting product usability and UX.
5. User experience: all design activities aimed at creating pleasant emotions for the user (consumer), positive attitude to using the product again and creating valuable relationships with a specific brand.

In all the aforementioned areas conceptual design is searching for ideas and solutions preferably better that exiting ones, novel and innovative in building competitive advantage for the prospective product. Therefore in the product stream conceptual design is a particular activity not based on technologies and standards, but on creativity of designers and users into involved into various forms of workshops, described hereafter.

Selected techniques for conceptual design

User workshops

In the Analysis phase user workshops were using teamwork for better understanding of the problem to be solved "as it is" and balancing the viewpoints and expectations of various stakeholders. In the Design phase instead, user workshops gather the team and other stakeholders (preferably users and customers) to draw

ideas, concepts and solutions into a format suitable for further software implementation.

Collaborative activities performed during user workshops (figure 8.1) may include:

- developing an expanded Persona for a newly discovered segment of target users (customers);
- refining specific tasks scenarios, user stories and use cases;
- card sorting, affinity diagrams and user story mapping for refining customer journey through a specific process or service;
- redefining user's problem for introducing a novel functionality or a novel business model offering a better competitive advantage for the product and for the client.

Figure 8.1. User workshop – conceptual design
(Credits: https://uxdesign.cc)

For reprocessing and upgrading all information collected so far, some formerly used techniques can be used again, such as:

- group interviews: in the new iteration users are encouraged by the moderator to discuss emerging design ideas, stimulated by a demonstration of a particular product or concept, about which users express their opinions;
- requirements refinement workshops: user stories can be rearranged and reformatted to discover novel aspects or redefine user expectations (Figure 8.2);
- affinity diagrams: user story mapping again for discovering novel product features, or card sorting (figure 8.3) aimed to reveal the information architecture as imagined by users, for instance an intuitive menu structure preferred by users.

Figure 8.2. Requirements refinement workshop
(Credits: http://www.interaction-design.org/)

Figure 8.3. Card sorting – an example
(Credits: https://www.interaction-design.org)

Iterative refining and reprocessing earlier collected artefacts with users help to deepen understanding of the topic and add novel design aspects. As a result, inspirations for creative activities are collected, to be executed in creative workshops. In addition, the teamwork gives participants two motivations: sense of participation in the design process and feeling of "ownership", which both are helpful in further activities.

Detailed information on facilitating user workshops can be found in the HCI literature e.g.: Sharp et al. (2019), Hartson and Pyla (2016) or Gottesdiener (2002).

Creative workshops

Creative workshops are aimed primarily at generating ideas for fresh, revolutionary solutions to be applied in a novel product. Creative workshops can be also called up as an attempt to solve a problem for which all conventional methods failed.

In IT projects creative workshops are often used for designing products for customers such as online services, websites, or mobile apps. For this reason to explore ideas possibly originating from users (customers) quite frequently separate workshops are organized for user groups, independently from workshops for professional designers from the team.

For creative workshops generally all techniques used for user workshops are applicable, especially the ones any using sticky notes. Nevertheless, for stimulating collaborative creativity the role of the moderator is critical. Interpersonal skills, ability to inspire the team and create the mood for generating ideas are the most expected personality traits. Additionally, ability to break old thinking patterns, to convert conflict situations into illuminating experience, and – last but maybe the most important – the sense of humour, are essential no matter which techniques were applied for a specific problem.

The most popular techniques for creative teamwork include:

- brainstorming: facilitated by a moderator, releases unhampered creation of ideas by participants, generating new solutions, breaking thinking patterns for innovative problem solving (figure 8.4); evaluation and selection of ideas takes can place only at the very end of the brainstorming session;
- mind mapping: drawing of concepts and ideas around the centrally located main concept; the drawing is freehand, preferably using of bright colours, funny symbols or other odd elements (Figure 8.5); mind mapping enables the projection of user expectations, regarding for instance preferred product functionality or menu structure; mind mapping should be first performed as an individual exercise, and next mind maps from all team members should be discussed and integrated as needed;

Figure 8.4. Brainstorming session – an example
(Credits: https://www.business2community.com)

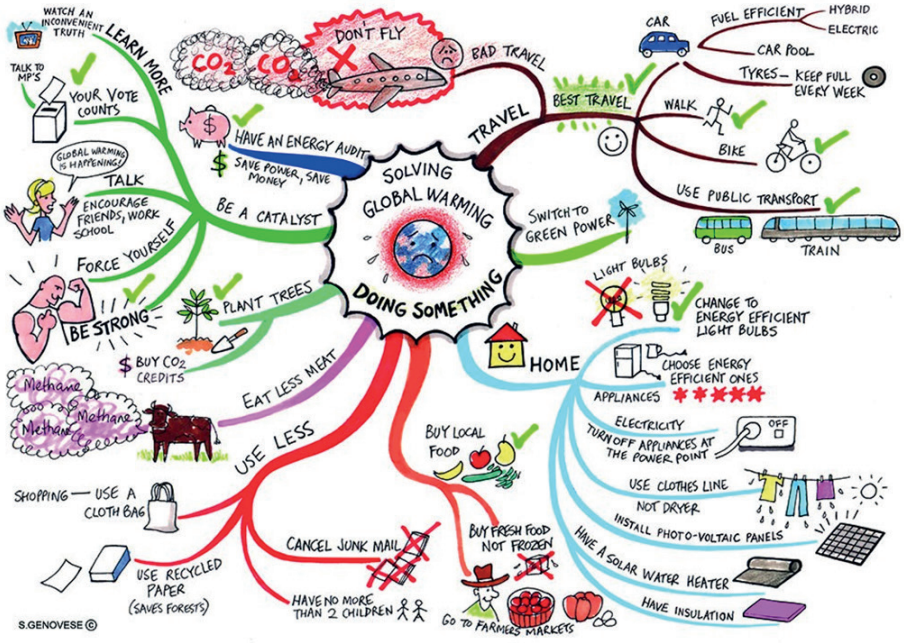

Figure 8.5. Mind mapping – an example
(Credits: https://learningfundamentals.com.au/resources/)

- six thinking hats: representing thinking styles (e.g. gut instinct, pessimistic judgement, neutral facts, enthusiasm etc.) in which the brain can be challenged when coping with a problem; during a moderated session, team members learn how to switch thinking across six distinct thinking styles roles ("thinking hats") for redirecting thoughts, the conversation, or the meeting towards creative outcomes;
- change of scenery: the team is taken away from office to a casual place, like outdoors or a time-off cafeteria where an odd scenery can help people think differently and devise new ideas; the session should be also moderated and ideas should be recorded (using sticky cards or a flipchart), and further refined when back in the office.

Nevertheless, whatever technique are used for creative teamwork, the success factor is a lucky combination of personalities, relaxed mood, casual moderation, and skilled use of visualisation techniques such as whiteboards, sticky cards and freehand sketching.

8.3. Freehand sketching

In all types of workshops, but also in individual design activity, freehand sketching of newly emerging ideas is essential for inspiration and creativity. Freehand sketching is also the best way to externalize initial concepts and to make them understood by others, therefore it an important visual communication technique in many areas of design, engineering, and management.

Freehand sketches radically differ from formal diagrams used in IT projects because they do not need to conform to any disciplined notations, being therefore informal, flexible and spontaneous. In contrary to technical, diagrammatic drawings, freehand sketching activates the right hemisphere of the brain, stimulating creative thinking, exploration, and discovery.

The study of Newman et al. (2003) found out that a generalized design process has discovery, design exploration, design refinement, and production phase.

1. Discovery is intended to determine and clarify the scope of the project and the requirements of users.
2. In design exploration phase, possible solutions to the problems identified in the discovery phase are generated and explored. Such solution can be in the form of information design, navigation design, or rough graphic design.
3. To continue to design refinement phase, it is required that a design idea be selected prior to the refinement. The design is iteratively refined and detailed.
4. Production starts when the design has reached a satisfactory level of detail – or in a worse case when the deadlines and budget dictate so. Designers prepare the design for hand-off to a client or a software development team. The handoff

may include interactive prototypes, written descriptions, guidelines, and specifications.

The study of Newman et al. (2003) also revealed that every designer in the project sketches at least once on paper, designers often annotate their sketches and printouts during the project, and that (often before the end of a project) paper sketches get discarded together with valuable ideas that may be needed some time later.

When facilitating the teamwork, freehand sketching can be used to:

- draw attention of the audience (the team) incomparably stronger than using PowerPoint slide show or any other digital media;
- inspire exploration by visual thinking and collective feedback;
- encourage the discussion and interaction among the team members.

Therefore, having whiteboards and flipcharts centrally placed in a shared workspace is essential for efficient visual communication in the team and instant recording new design ideas, concepts, or hints. In both cases of individual and collective design, sketching on paper or whiteboard, and using sticky notes, contributes much better to creativity than using any screen-based devices like tablets or displays.

More information on the use of freehand sketching and storyboarding for interaction design can be found in the HCI literature, for instance Becker (2020), Sharp et al. (2019), or Hartson and Pyla (2012).

In IT projects for ideation and developing new concepts for interaction design, a specific trio of freehand sketching techniques is frequently used, embracing screen sketches, storyboards, and user flow, briefly described hereafter.

Screen design

Ad-hoc freehand sketches of prospective screens are usually the first visualizations of an emerging user interface, expressing design ideas and general layout concepts (Figure 8.6).

Sketches are the basis for the development first screen layout templates, screen wireframes, windows, buttons etc., and eventually paper prototypes of the user interface. When about to start sketching, there is no need to worry about any personal deficiencies of drawing skills; the only importance is that the sketch is communicative to others, and it operates using simple symbols.

In IT projects freehand sketches are a commonly used tool for creating visual concepts and interaction solution concepts. They are also valuable for communicating the product concept and screen layouts to future users (consumers) but are not as useful for presenting the proposed method of system navigation.

During user workshops, when discussing and refining freehand sketches, following issues frequently arise:

- new, previously not expressed, user expectations are formulated, usually regarding non-functional requirements;

- there is a need to modify earlier task scenarios by adding details resulting from discussions and users' suggestions.

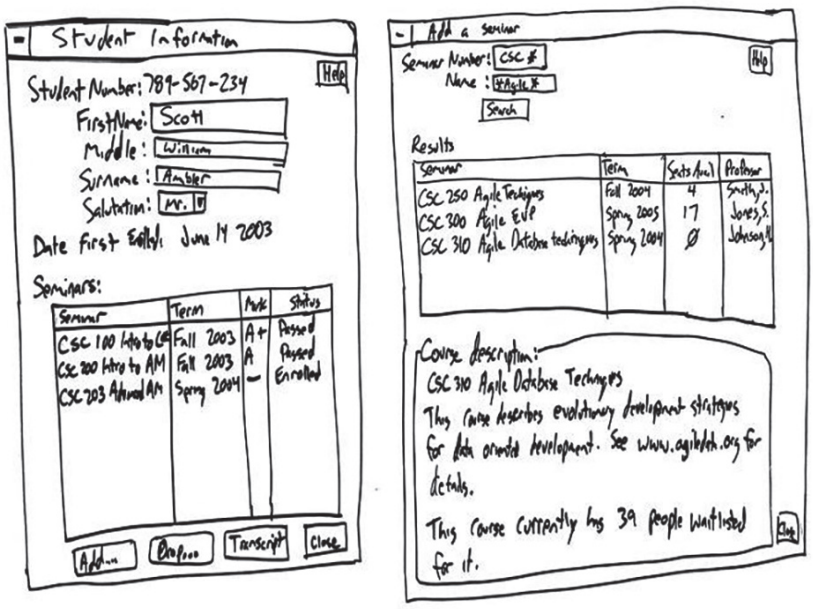

Figure 8.6. A freehand sketch of a screen template
(Credits: http://www.agilemodeling.com)

Graphic designs developed by arranging freehand sketches can then be presented in the form (even static) of screen design metaphors and further steps to be executed in the form of interactive prototypes. All screens sketches are related to specific task scenarios, so it is logical to combine them in a sequence ("storyboard") showing step-by step how the user is progressing towards the expected outcome.

Storyboards

Storyboards is the technique borrowed from cinematography as a way of drawing a sequence of scenes taking place on consecutive screens the user goes through when performing a task (Figure 8.7). Successive screens are sketched using small cards or sticky notes. Each card shows a screen that is part of a system's navigation. Usually the first drawing (screen) presents a problem situation the user wants to solve, and the last card (screen) shows the outcome of the task execution (problem solved).

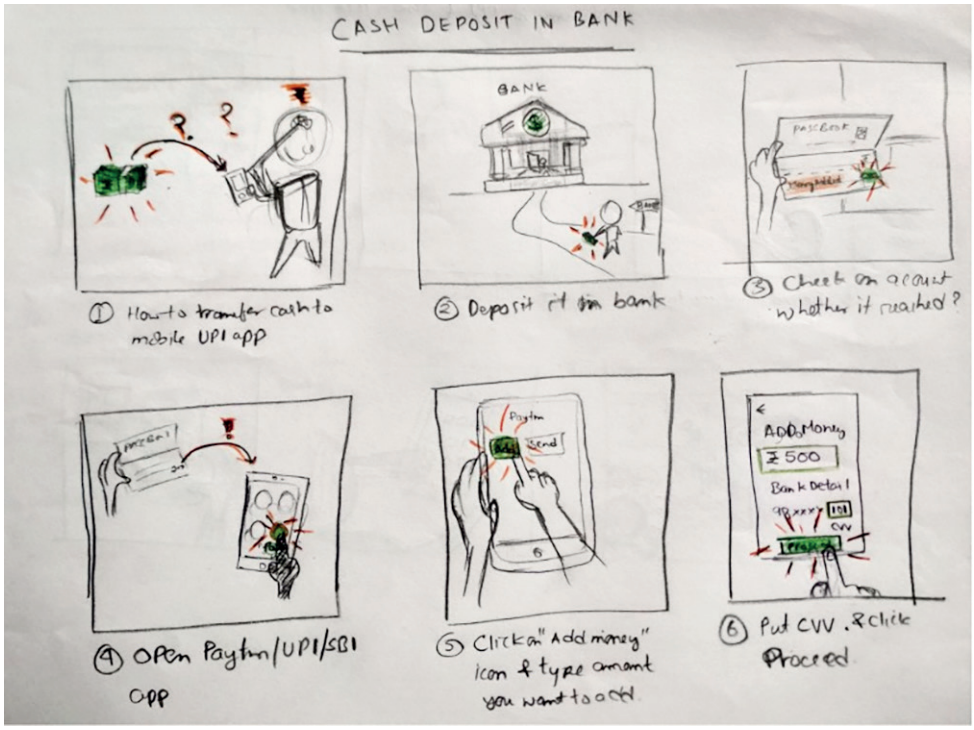

Figure 8.7. Storyboard freehand sketch – a raw version
(Credits: https://uxplanet.org)

Storyboards prepared with freehand sketching are very effective in quick visualizing the sequence of screens (actions) the user will perform when trying to accomplish the task with the product or app.

The storyboard technique is commonly used in the early stages of design for the presentation of planned interaction concepts and for developing the system navigation. It also helps in conceptual developing of prototypes intended for further testing and evaluation.

In iterative design, storyboards certainly will be subject to many discussions, improvements and changes. As a result, refined storyboard will usually not only have richer content but also better quality of graphics (Figure 8.8). Because creating a storyboard is largely based on the earlier outcomes such as task scenarios, user stories and user story mapping, it is also possible a step back is needed to correct relevant user story maps.

Figure 8.8. Storyboard – a refined version
(Credits: http://www.debaoki.com/storyboards)

User flow

User flow (known also as user journey, task flow, or navigation flow) is the visualization of user (customer) flow when operating a system (product). It is based mainly on the storyboard presenting a sequence of consecutive screens user goes through.

User flow can be prepared as a minimalist, plain flowchart with boxes and other symbols depicting specific user actions or decisions. More frequently however, user flow is built as a network of subsequent screens the user goes through when performing a task (Figure 8.9).

The first version of user flow can be constructed with a series of hand-sketched sticky cards used as miniature screens. A complete user flow should be built on a regular, consistent screen layouts and should contain all screens needed to accomplish basic task scenarios. Nevertheless, yet it may have no window messages, or screen objects for reverting user actions, or more complex actions like scrolling or zooming.

Certainly, at this stage of design freehand screen sketches and storyboards are not intended to be complete. Their objective is to give general outlook a how user interaction will look like and to visualize its overall flow. The visual details will change in further refinements, but the navigation scheme proposed in user flow is likely to remain mostly unchanged.

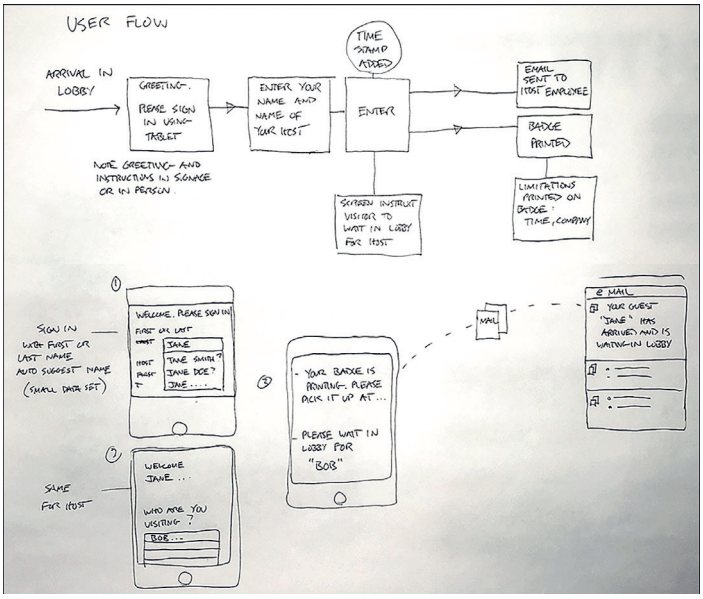

Figure 8.9. User flow sketching – an example
(Credits: http://www.visualux.design/sketching)

Trying to get through typical task scenarios, the designers can simulate user's actions and make necessary corrections. Many further annotations and improvements will be still made in design iterations and during development and testing (Figure 8.10).

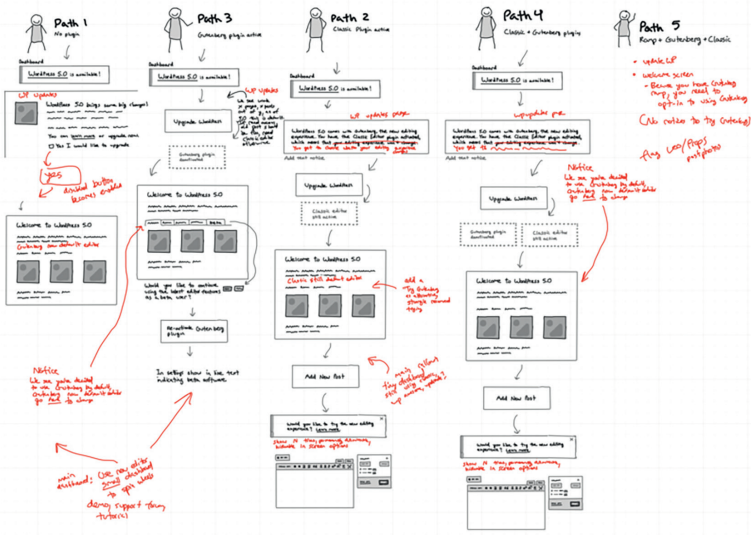

Figure 8.10. User flow reviewed and annotated – an example
(Credits: https://www.vippng.com/maxp/hJoimmm/)

Storyboards and user flow sketches are first conceptual models for designing interaction and user interface. They may be still rough and incomplete, but they are extremely useful as initial prototypes and first visualizations how the prospective system may look like and be operated by the user.

8.4. Deliverables from Design

In addition to software engineering solutions delivered from the technological stream, the Design phase feeds the project with ideas and concepts which form the user interface of a prospective product.

For interaction design area, the most important outcomes from the Design phase include:

- refined input data: task scenarios, Persona, use cases, user stories, and user story map projecting key functionality of the product;
- freehand sketches of individual screens, screen template layouts, storyboards and user flow, where key functionality of the product was combined with initial concept for user interaction and navigation.

In fact, these deliverables are initial and partial prototypes of the product, ready for review within the team by cognitive or manual simulating of user tasks. Drafts, sketches and graphical mock-ups can be also consulted with the client and with representative users (customers) as needed – the sooner the better.

Despite they will undergo further improvements and refinements, the key elements of the product are now ready for development in two areas: firstly, the implementation and coding in a programming language (the technological stream), and secondly, prototyping the user interface (the product stream) to be presented in the next chapter.

9. Development – from concepts to solutions

9.1. The outline of Development

The Development phase has following objectives:

- in the technological stream: developing software – that means converting models and diagrams into solutions such as implementation of system architectures, developing the code, programming, cleaning an optimizing software components;
- in the product stream: building prototypes – that means converting interaction concepts into solutions, such as:
  - early prototypes (sketches, paper mock-ups) for internal evaluation by team members (and preferably also by clients or users);
  - digital prototypes, based on early prototypes, ready for testing by users and acceptance by clients.

Prototyping of a prospective product (or its user interface) is based on components prepared in Design phase, such as screen templates, user stories, and user flows.

Recent popularity of prototyping in IT projects is caused by its direct benefits such as (Becker, 2020; Sharp et al., 2019; Hartson and Pyla, 2016; Dix et al., 2004):

- early validating design concepts: checking whether the model developed by the product designer is sufficiently convergent with expectations of users;
- improving communication: thanks to experiments and prototyping project shareholders such as clients, users, customers, project sponsors, can learn more about the product features than from traditional drawings or documents,
- energizing the team: by demonstrating work progress as a clickable demo satisfied users are the living proof of success, otherwise the costs of removing errors is still relatively low.

Prototyping is a vital part of contemporary IT projects, because the UX aspects of the product can be evaluated early in the development process, even with limited features of the product.

Various types of prototypes differ a lot in detail, regarding the stage of the project in which they can be applied. Nevertheless, generally they can be categorized into two major groups:

1. Low-fidelity prototypes, usually made with paper:
   – initial prototypes – screen sketches, wireframes, graphical mock-ups, storyboards and user flows;
   – paper prototypes – paper mock-ups with movable user interface parts and other user interface models animated by the designer (e.g. “Wizard of Oz”).
2. High-fidelity prototypes: developed with software tools:
   – clickable digital prototypes running on a computer screen (screen prototypes);
   – video prototypes presenting simulated use of a prospective system (product).

In another dimension, prototypes are also classified into two major classes:

- horizontal prototypes, which contain a broad selection of functions, but they are usually made as simplified, without details and often with much incomplete functionality;
- vertical prototypes, which offer a narrow selection of features, but their implementation is compete, so the user can perform all steps of the process.

In IT projects two more terms are frequently used:

- MVP (Minimum Viable Product): a “minimal” product with one principal functionality, that is ready for small-scale testing on the market to check whether users (consumers) find the product useful in real settings and attractive from a business viewpoint;
- POC (Proof of Concept): a small internal project aimed to validate a certain or theory or concept (like a new type of wireless connectivity), which serves to verify some key functional aspects of the intended design, but usually does not have all the functionality of the final product

Two main categories of prototyping, namely Low fidelity and High-fidelity, will be presented in the remaining part of this chapter.

9.2. Low-fidelity prototyping

Refined sketches

Although we have abundant software and technology supporting interaction design, often the best method is still drawing of software screens on paper. Despite

of technological advancements, paper remains still a valuable proxy for digital tools and design assumptions can be examined quickly.

In IT projects paper-based sketches continue to be widely used due to their speed and ability to help designers and developers think as well as validate solutions quickly with users. Especially, when project deadlines are tense, paper sketches can be quickly assembled to address user's needs and mimic user's screen actions with a desired level of complexity and fidelity.

When trying to identify whether a certain concept or solution is meeting user's needs, there is a large set of assumptions a designer has to validate as early as possible. Therefore all prototypes of interactive products for consumers should start from freehand sketches some way. The more designers use paper to validate their ideas, the faster they learn what is working or not with their concepts.

Many designers are reluctant and sceptic to drawing because it requires pencilling. Sketching, unfortunately, is not a universal skill shared by all. Many people – especially IT professionals – say they cannot draw. This seems to be a failure of individual perception – nearly everyone can draw, it is only the matter of practice that was probably abandoned or dominated by digital tools. Therefore, any attempt to use own ability to draw is likely to improve communication with clients or users at all stages of the project (Figure 9.1).

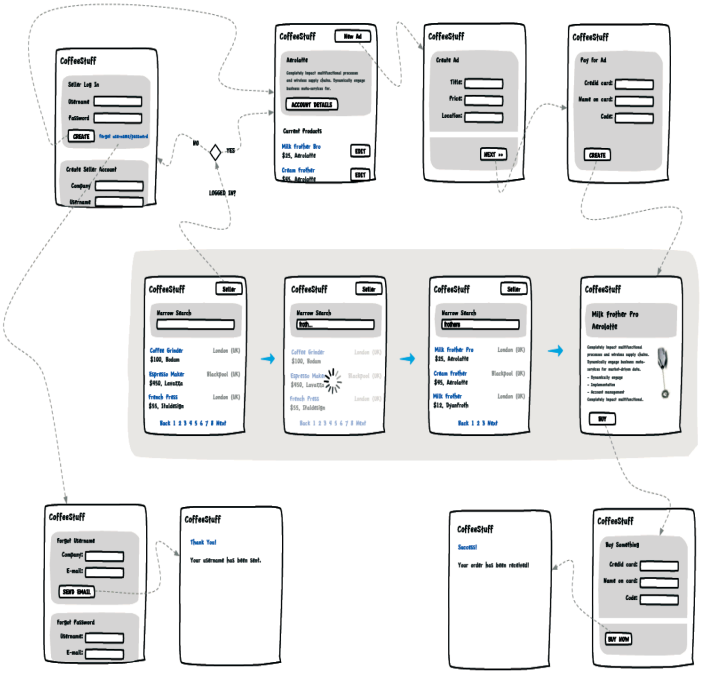

Figure 9.1. A hand sketched user flow for a mobile application (Credits: http://wireframes.linowski.ca)

The actual goal of sketching paper prototypes is not about rendering details, but rather about speed of validating your ideas. The reality of drawing is thinking, experimenting, exploring and gradually converting our visions into viable concepts. For this reason initial prototypes prepared earlier – freehand screen sketches, storyboards and user flows can be refined several times, what goes easy if they are on the papers and easily available within a reach of hand. As a result, refined screens, storyboards, and improved user flow (Figure 9.2) will lead to producing new prototyping artefacts – wireframes and paper prototypes.

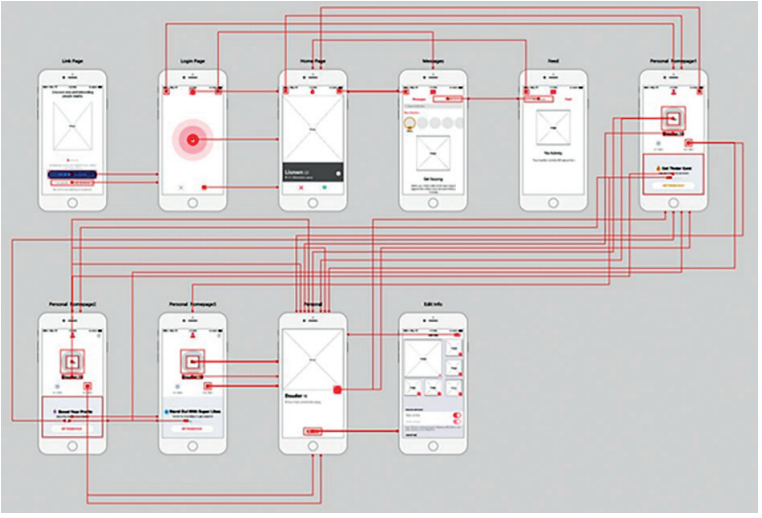

Figure 9.2. Refined user flow for a mobile application
(Credits: https://www.mockplus.com)

Wireframes

Screen wireframes are similar to freehand sketches but are usually more elaborated in detail and prepared using a computer (Figure 9.3). They are screen templates, using a regular geometric grid to show a clear allocation of the main areas for user objects and information fields, necessary to plan in detail how the screen space is to be used.

Many software tools can be used to develop wireframes from sketches, allowing quickly creating digital prototypes to be tested by users. Wireframing is especially important for web design, where a webpage should be automatically adjusted to the specific parameters of user's screen.

It should be however noted that wireframe screen prototypes and screen sketches are not very suitable for illustrating the subsequent stages of user's tasks (navigation through the process).

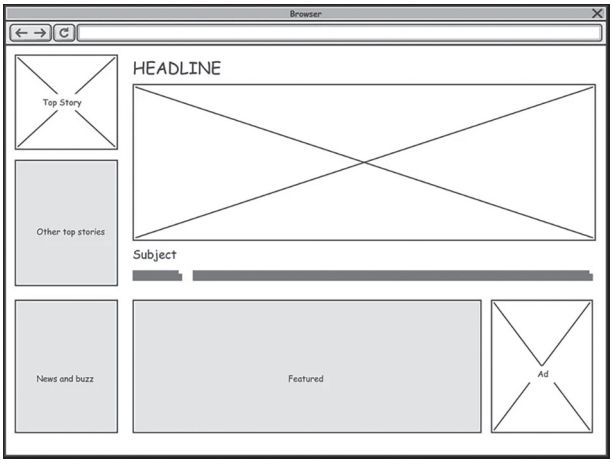

Figure 9.3. Wireframe screen prototype – an example
(Credits: https://www.archimetric.com)

For this purpose much more suitable are storyboards, preferably in refined versions, ready to be shown to the client. Improved wireframe prototypes furnished with sample contents can be also presented to the client for approval as static, graphical mock-ups (Figure 9.4), being much more informative than wireframes.

In fact, many designers are reluctant to show hand sketched screens to clients, regarding them as raw, informal and unprofessional. On the contrary, computer-generated wireframes and graphical mock-ups are treated as tidy and regular, more likely to be appreciated by the client. The advantage of screens printed on paper is that annotations resulting from discussion with client can be directly placed on the screen printout so they will not be lost. Furthermore, wireframes printed on paper can be next used for building refined storyboards and paper prototypes, reducing designer's labour especially if many screens are needed.

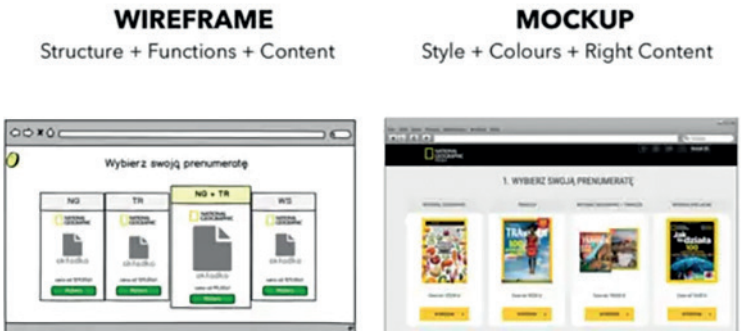

Figure 9.4. Graphical mock-up and wireframe – (static) – the difference
(Credits: https://brainhub.eu)

Paper prototypes

In contrary to graphic mock-ups which are static, paper prototypes are made with movable paper parts. Using paper prototypes it is possible to simulate performing simple tasks and thus bring interactivity to previously made sketches.

As shown in Figure 9.5., to create interactive paper prototypes easy to find, cheap raw materials are needed such as cardboard sheets, coloured paper, sticky notes, adhesive tape, scissors, glue, fluorescent markers, transparencies, etc.

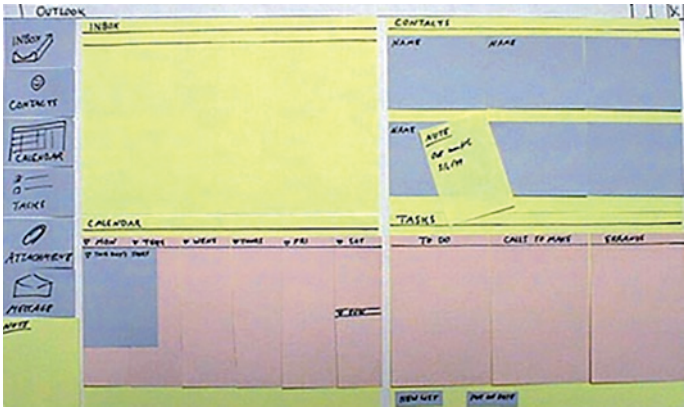

Figure 9.5. A simplistic paper prototype of a user interface
(Credits: https://community.sap.com)

Paper prototyping, to some extent, is successful to make paper sketches interactive. At this stage of interaction design, it is not about displaying how the product will look like, but rather actually showing how it should work (Becker, 2020). The key role in paper prototyping is a person acting “computer” who manipulates the paper version of the interfaces, and not the quality of graphics (Figure 9.6).

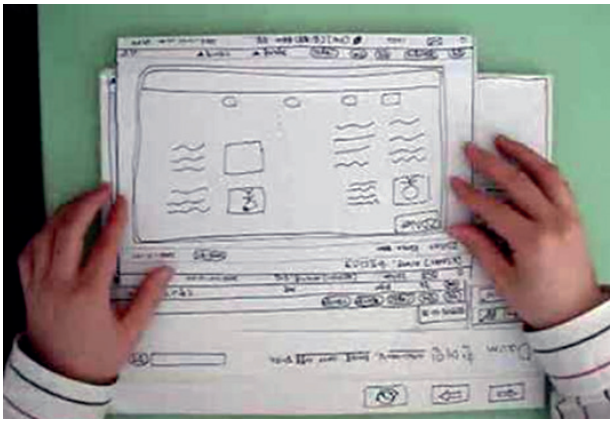

Figure 9.6. A clickable paper prototype
(Credits: https://www.connected.io/)

Through paper prototyping, designers can quickly validate whether ideas are worth pursuing into a more robust prototype. Every designer should build paper prototypes first as there is always much to learn from making mistakes on unsuccessful trials to complete a task.

A huge value of paper prototypes is found if certain ideas do not actually work. Rather, the execution is cheap, and you can discard them at once if they do not get accepted by users. Eliminating bad idea early in the design process is much better than allowing it to move into further development – or worse, making it into hands of users. Killing bad ideas at the paper stage can save a lot of work and costs for the project.

Paper prototypes are often called early prototypes, as they are made early in the project, without any rendering of detail, but with an entire emphasis put on showing the projected interaction method.

Evaluation of paper prototypes is performed in the team as a collective review or a cognitive walkthrough – mentally simulating all steps of the task to be performed by the user. Less frequently paper prototypes are demonstrated for consultation with the client, instead computer-generated wireframes and static graphical mock-ups are rather preferred.

Further evaluation of digital prototypes can be made in many ways: starting from internal team reviews using cognitive walkthrough, to consultations with the client, to – most valuable – testing paper prototypes with users. Even if the digital prototype is basic and incomplete, simulating task scenarios on paper instead of screen makes possible detecting quickly major usability flaws or missing functionalities.

In theory, when the design is no longer subject to major changes – usually in later stages, high-fidelity prototype is built to represent the look-and-feel and behaviour that resemble the final product. However, it is a common temptation among designers to skip paper prototyping and start digital prototyping based on freehand sketches. In reality, experienced designers might build hi-fi prototypes early – to cope with deadlines, and then patch up the prototype with more functionality in later stages. In most cased however the risk is very high that resulting digital prototype will be loaded with major usability issues, as basic interaction patterns had not been validated on paper. Saving time this way can be illusory, and valuable ideas emerging during paper prototyping may never come to life.

More information on building low-fi prototypes can be found in the HCI literature: Sharp et al. (2019), Becker (2020), Dix (2004), Snyder (2003).

Wizard of Oz

The “Wizard of Oz” prototyping method is especially useful in design and evaluation of interactive systems, which are difficult to implement and must be tested in a very short time.

The method involves the user entering commands into a mock-up system while responses are generated by a designer “from the back” (Figure 9.7), like a magician apparently bringing a non-existing the system to life.

Figure 9.7. The idea of the “Wizard of Oz” prototyping
(adapted from Sharp et al. 2019, Dix et al., 2004)

Once popular for designing CLI user interfaces, most recently “Wizard of Oz” method is popular for testing for voice interfaces, chatbots and other solutions for which the construction of a prototype would be very labour intensive, but the tests should have been completed in a very short time.

The advantages of “Wizard of Oz” include a very short time needed to build the prototype and very high flexibility of generated responses, including reactions to unpredictable user errors. Moreover, the user does not need to know any details of the system and the scope of its functions, and can reveals his/her expectations by expressing task-related intentions.

“Wizard of Oz” is classified as a low-fidelity prototyping method because it aims to design and evaluate main interaction patters for the users-system dialogue, neglecting other technical details of user interface to be developed.

9.3. High-fidelity prototyping

From paper to digital prototypes

Paper prototypes become harder to maintain as they are added with more and more features and screens. More and more fidelity is needed to present all objects on the screen and make them clickable to test the prototype live (Becker, 2020). Digital, interactive prototypes may include some clickable buttons, links, or some dropdowns.

Clickable prototypes can be produced through multiple iterations from low fidelity (sketch or paper) to high fidelity (digital) prototype, via paper prototypes with movable objects such as windows or dropdowns.

Apparently difficult transition of design tested with a paper prototype to a digital prototype can be an issue tempting to start digital prototyping just from screen sketches and user flows, leaving paper prototyping aside. Just on the contrary, the sketch-and-snap trick can be used: with prototyping software tools, photos of paper prototypes can be easily converted directly into digital prototypes.

Digital prototypes are actually simple simulators of the product, giving an impression how it might work, despite lacking some underlying components, like a database or connection to cloud services. Digital prototypes should merely sufficiently mimic how the product might work when actually coded, along with how the interface might behave. They can include animations, transitions, and other advanced interactions to simulate the aesthetics of a proposed design and to approximate the system responses to user interactions.

As they get closer and closer to a finished product, a high-fidelity prototype can also be coded to make the product closest to the final design in terms of content and functionality. Therefore it is natural that a designer wants to build clickable prototypes early on, often to test their ideas throughout a process before users may validate them in real settings. Testing usability of digital prototypes in controlled conditions, with real users trying to accomplish specific tasks with the prototype, provides invaluable feedback for improvements in usability and UX.

For building a digital prototype from a paper prototype, however several fundamental artefacts should have been approved beforehand (Snyder, 2003):

- refined screen compositions, as visual representations that show the look of the interface, including the graphical details; and primarily approved in internal discussions of a product visual design; they may use nonsense words (“lorem ipsum”) to represent yet non-existing text and links;
- refined wireframes, which precisely defines the screen layouts, location of visual and navigation objects, showing which content goes where;
- refined storyboard, as a pictorial scenario in series of drawings showing how the interface should be used to accomplish a task;
- user flow, as a tasks flowchart showing user journey through the product when accomplishing the task.

However, all of them can be used in paper prototyping as long as they contain realistic content to support a task scenario (Snyder, 2003). In comparison with technology that tends to change rapidly, paper prototyping will never be obsolete as long as people still draw and write on paper.

Evaluation of digital prototypes that is to come later, can be made in many ways: starting from internal team reviews using cognitive walkthrough, to consultations with the client, to – most valuable – testing paper prototypes with users. Even if the digital prototype is basic and incomplete, simulating task scenarios on screen makes possible detecting usability flaws or missing functionalities that some way passed though during evaluation of paper prototypes.

Despite of many benefits, digital prototypes have also some limitations, for instance:

- users or clients may think that the prototype exactly resembles the appearance of the final product, what is usually not correct;
- the prototype has no goal to be complete – it can simulate only selected functionalities and has underneath no real software architecture, with no connection to database or cloud services;
- due to high fidelity of graphics, during evaluation with users or clients often too much attention is put to details such as icons, images, colours, losing from sights more important issues such as navigation, user guidance or tolerance for user errors.

Software tools for building digital prototypes

The role of these software tools is to create a digital, interactive imitation of a product and to let validate designer's concept in a series of simulated tasks.

Software products used for building high-fidelity digital clickable prototypes can be roughly divided into two categories: general-purpose tools and dedicated tools.

General tools

General-purpose tools include software products designed for another area of office work, but with some effort and creativity, despite some limitations, they can be adapted for building digital prototypes. General tools include office software such as:

- presentation software (like Microsoft PowerPoint, or Keynote from Apple), which allow to use slides as screens, which if suitable hyperlinked, can simulate user going though subsequent steps of the task; active graphic elements (buttons, images) are also hyperlinked, and when clicked they "move" to user another screen following specific paths of the user flow (Figure 9.8);
- graphical editors, like Adobe Photoshop, suitable primarily for high-fidelity graphical mock-ups and modelling user interface objects, however with significantly limited interactivity;
- spreadsheets, like Microsoft Excel, they have limited interactivity and need using external graphic objects, but they are excellent for prototyping systems in which calculation or charts need to be dynamically produced in real time.

General tools include also:

- HTML editors, basically intended for designing and developing websites; they enable preparing HTML-based prototypes (with CSS and JavaScript) a set of hyperlinked webpages for any kind of user interface, including small screens of mobile devices;
- coded prototypes, developed is a specific programming language, like C++ or Python; the advantage is that – in contrary to conventional digital prototypes –

the code used to create the prototype, after necessary improvements, can be reused in a finished product, saving much labour and time.

Figure 9.8. A clickable prototype in PowerPoint
(Credits: Piotr Świerczyński)

With HTML-based or with coded prototypes users can perform operations, just like fully fledged websites or applications, more realistically understanding how the product will works in a real world (Becker, 2020). Nevertheless, for developing HTML-based or coded prototypes solid programming skills in the team members are needed beforehand. They are expected to prepare a useful prototype in just few hours, and without excellent software development skills it might take too long.

Prototyping software tools

Dedicated prototyping tools fall into two categories:

- desktop tools, such as GUI Design or Axure based on a fixed licence price,
- web applications (online services), based on a monthly subscription fee.

All dedicated prototyping applications work in a similar manner: there is a library of screen objects, which can be placed on a selected screen template using with drag-and-drop. Identically as is website design, the screens and objects can be linked as needed, and external images or graphics can be added, too. After the prototype is ready, using “play” or “simulation” mode the full-screen prototype can be operated and tested.

Desktop tools include only two leading products:

- GUI Design Studio (www.carettasoftware.com), a professional application for creating user interface prototypes with rich functionality, largely for systems intended as work tools in engineering, business or administration;
- Axure RP (www.axure.com), advanced application for creating user interface prototypes, largely websites and web applications based; the prototypes are exported to HTML so they can be remotely presented to the client using only a Web browser (without Axure).

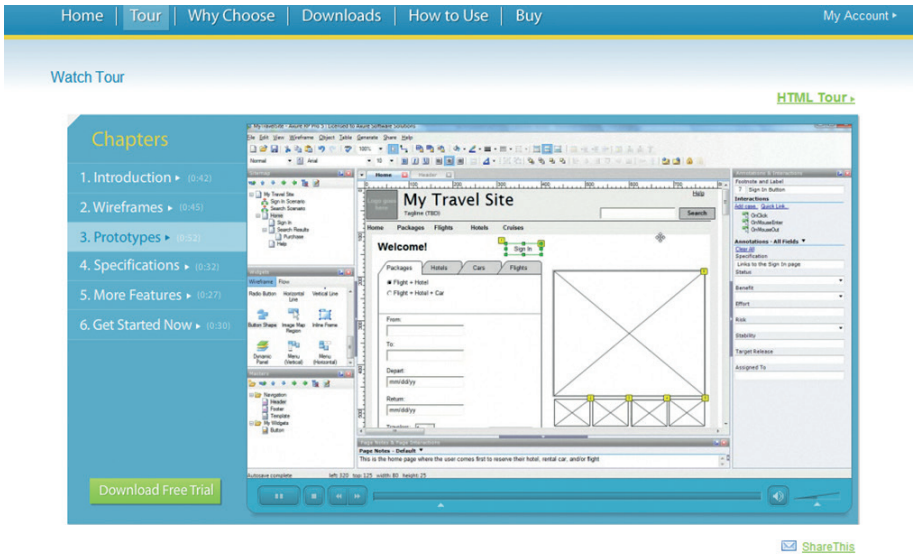

Figure 9.9. Axure – prototyping a website
(Credits: https://www.axure.com/)

Web applications for prototyping tools are usually available online via a browser or using a thin client. These prototyping tools are actually using cloud services, and they are based on a monthly subscription fee, what makes them especially suitable for agile projects.

The way how the prototypes are build is similar as in other dedicated tools for prototyping: screen objects are dragged to screen templates, which are next linked as needed and can be simulated for evaluation and testing (Figure 9.10)

On the market there is abundance of software tools for building prototypes, both fee and commercial. There are too many to list, so only a few names: Adobe XD, Figma, InVision, JustInMind or Protoshare are available options. More extensive list of prototyping applications can be found in the https://prototypr.io/prototyping/ website.

All they vary a lot regarding their functionality, available libraries for graphics and user interface widgets as well as their ability to emulate more complex opera-

tions like scrolling, panning or zooming. By default all web applications for prototyping are adjusted to designing and prototyping mobile user interface.

More detailed information on the development of digital prototypes can be found in the HCI literature such as: Becker, 2020; Hartson and Pyla, 2016; Tullis and Albert, 2008;

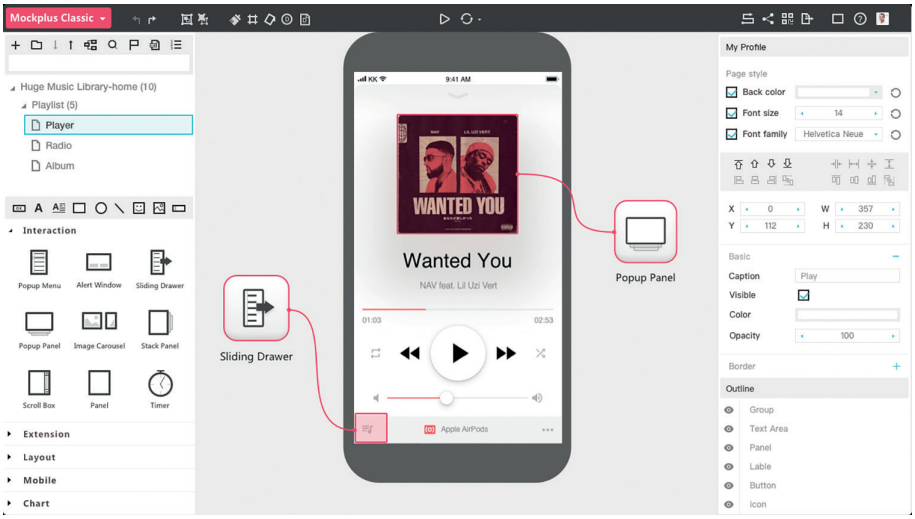

Figure 9.10. Prototyping app editing window – an example
(Credits: https://www.mockplus.com/app-prototyping-tool)

Video prototypes

Video prototype is a short video prepared in a way resembling common TV commercials, advertising benefits and novelty of a specific product. In case of video prototypes for software, online services or mobile apps, the product is presented as operated by Persona target user in a in realistic context.

The product may be yet not existing (Figure 9.11), so its operation is simulated using digital prototypes, and the main objective of video prototype is to emphasize its attractivity and business potential for specific audience, like decisionmakers, project sponsors or clients. For this reason video prototypes are largely used as a supplement for the product plan in the Strategy phase.

Video prototyping is a method delivering suggestive, impressing presentations, which however are fictious because they present yet non-existing product, only a digital prototype or a mock-up. Nevertheless, this is quite understandable, because their goals are merely promotion of product vision in the Strategy phase, as well as persuasion and influencing decision makers and project sponsors, in a way similar to classical video commercials.

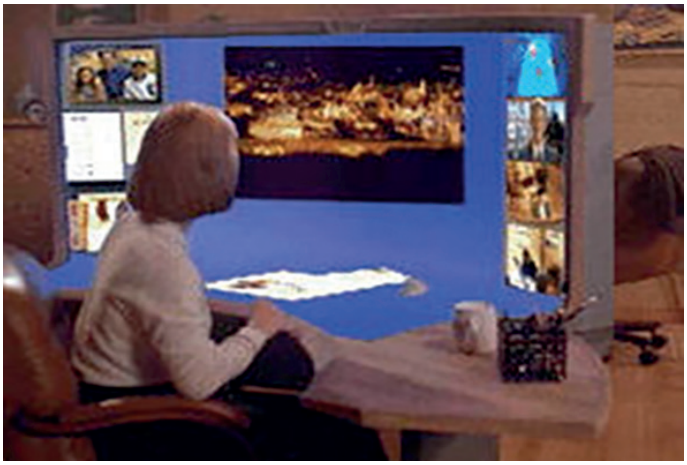

Figure 9.11. A video prototype – the Starfire system
(Credits: https://www.asktog.com)

9.4. Deliverables from Development

The Development phase feeds further stages of the project with deliverables located in two streams:

- the technological stream, with engineering deliverables such as code reviewed, running solutions, specific parts of the product fully implemented and ready for testing;
- the product stream, with interaction solutions reviewed by the team (or initially accepted by client), ready for testing by users within usability testing or UX testing to come.

Specifically, in the product stream following categories of outcomes should be handed-off and initially approved:

- initial prototypes: screen wireframes, storyboards, and user flows, approved by the client;
- low-fi prototypes: graphical mock-ups and paper prototypes, tested by walk-through in the team or reviewed by real users, ready for converting into hi-fi prototypes;
- hi-fi prototypes: digital prototypes, reviewed by click-through in the team and ready for usability testing by external users.

In the Development phase both technological and product stream become integrated by the need to combine the two types of outcomes into one product, ready to be validated with real users by evaluation and testing.

10. Validation – evaluation and testing

10.1. The outline of evaluation and testing

The last phase in an IT project is the Validation of solutions which were envisioned, designed, and developed so far. The Validation includes Evaluation and Testing performed in various forms:

- in the technological stream: a battery of tests applied to all software components delivered from the Development phase;
- in the product stream: evaluation and testing of functional components, prototypes or specific modules of a finished system.

In user centred design (UCD) evaluation should be performed on a regular basis across the whole project. Whenever possible, prospective users should be invited to evaluate, in addition to team members and clients.

Evaluation methods in IT projects may vary a lot – from internal reviews inside the team to testing solutions with large groups of users. Validation differs from verification: the latter is rather checking the conformance to engineering excellence requirements, while validation includes whether the product is what the customer really wanted (Jayaswal and Patton, 2009).

Validation covers both evaluation and testing. Evaluation is more general, more flexible and less formal than testing, remaining rather a continuous managerial activity than a disciplined process. Evaluation allows some subjectivity, while testing is more procedural and disciplined, conducted according to specific instructions and procedures, with formal reports required to be submitted.

Results of validation should result in improving the quality of the final product as well as in streamlining further activities planned in the project. Regarding interaction design, major methods useful in the Validation phase will be presented in the remaining part of this chapter.

10.2. Expert-based evaluation

Expert evaluation is aimed to determine the degree of conformity of the product (interaction solutions, user interface, functionality) with established design principles, guidelines, requirements or other earlier specified evaluation criteria.

Expert evaluation comes to play when subjective criteria or the impact of context must be considered. In interaction design area expert evaluation can be performed by an external expert of by a senior team member and is often supplemented by results of usability tests and evaluations collected from users.

The main types of expert-based evaluations (according to Wong, 2020; Sharp et al. 2019; Hartson and Pyla, 2012; Dix et al., 2004) will be discussed in the following sections.

Heuristic evaluation

Heuristic evaluation is exploratory: the expert analyses the user interface, interpreting each of the Nielsen's heuristics (Nielsen, 1993) regarding specific features of the evaluated system and the tasks assigned to the user. Heuristic evaluation is flexible: it may cover product usability, product functionality or only the user interface. Heuristic evaluation is most effective when carried out by a team of 3–5 experts working independently, while the results of their work are later compared and aggregated.

The biggest advantage of heuristic evaluation is its exploratory nature, because:

- evaluation is based on the unrestricted exploration of problem areas by an expert who interprets the contents of the Nielsen's heuristics in terms of a particular system,
- heuristic evaluation often results in constructive proposals how to improve the product, often based on expert's knowledge from other similar solutions;
- flexibility of the evaluation allows at changing its scope depending on current findings or references to other similar systems.

However, heuristic evaluation has following limitations:

- it is not formalized making difficult aggregating and comparing results from individual experts,
- experts are not always available, and their work can be expensive,
- experts may not know the software application domain (e.g. accounting), so they may overlook problems located outside their competence area.

Heuristic evaluation is useful at various stages of product development, but especially for evaluation of high-fidelity prototypes or finished systems, e.g. the competitive ones.

Internal reviews

Internal reviews are performed in the team (without the client) for design concepts, architectures or other deliverables, like the code or user interface prototypes. Internal reviews can be performed as individual reviews (by an expert of another experienced team member) or as a collective activity during a team meeting.

Peer-based review is common also in agile projects, especially XP-based ones. When software developers work in pairs, they mutually review and check the quality of code written by the other workmate in the pair.

Results of internal reviews should be added to documentation of the project and stored for future reference. Preferably, reviews performed on a regular basis should be added to the timeline of the project as fixed checkpoints.

Cognitive walkthrough

Cognitive walkthrough is a review method in which an expert is projecting imagined steps the user is most likely to go through when working with the system. These steps may be depicted on subsequent screen sketches (like in storyboarding) or just projected in expert's mind.

At first, each user task is divided into individual operations; next an expert (designer) simulates expected behaviour of the user by trying to answer following questions:

- is it obvious to the user what the next action is?
- does the user correctly interpret the descriptions of activities and connect with the functions that need to be activated?
- are the system confirmations formulated in such a way that the user is able to interpret the system response properly and which is the correct choice?

Cognitive walkthrough task scenarios are usually carried out using screen sketches. Imagined projection of subsequent screens can be supported by the think-aloud protocol about what the user is going to do, and which visual elements they would use to complete the current action.

Checklist-based inspections

In IT projects checklist-based inspections are often used for evaluating how far users' requirements have been satisfied in a specific deliverable, like a prototype. Checklist-based inspections are performed by testers employed in and IT project, or by an external inspector.

In checklist-based inspections evaluation is limited only to documented requirements, that had been identified beforehand and included in a relevant checklist. It makes evaluation less prone to subjectivity and makes easy preparing a standardised evaluation report, presenting problem areas or items which require corrections or improvements. An evaluator is expected only to identify detected deficiencies

using the checklist and present the report, and not to propose how to correct them, may be considered as a limitation of this method.

References to typical checklists for evaluation of user interfaces can be found online and in abundant HCI literature, for instance Sharp et al. (2019); Shneiderman et al., (2017); Hartson and Pyla (2012).

10.3. User-based evaluation

User-based evaluation techniques

User-based evaluation requires users to be involved in on-site or remotely, and the evaluation can include testing the product is simulated conditions. For this purpose numerous observational and empirical techniques are available, briefly presented hereafter.

Evaluation by observation

Evaluations by observation take place when a small group of users is observed, while performing a specific task in a laboratory or "in the field" (Figure 10.1). Largely qualitative data are then collected: comments, discussions, questions, emotions. Encouraging users to think aloud and express spontaneous comments makes collected data more valuable. This method can be used at any stage of system development and collected observations from user behaviour enable quick identification of difficulties encountered during system operation.

This method has also limitations such as: the presence of an observer can influence the behaviour of users, analysis of the collected data is very time consuming, and results depend on the attitude of users and their willingness to cooperate.

Evaluation by observation can be also performed within the ethnographic approach, as an observer working with the user while performing the tasks, asking questions, or performing supplementary activities.

Figure 10.1. Direct observation of a user working with a system (Credits: https://www.sapdesignguild.org/)

Presentation and walkthrough of a paper prototype

The presentation is facilitated by team member who moderates the course of the meeting in such a way as to obtain information most useful for designers. Due to simplified graphic design, the evaluation is limited mainly to the aesthetic aspects, invoked emotions, associations, etc. This method is often used in website design for consulting users (beyond the client) on preferred selection of colours, graphics, and indication of elements that do not fit to a specific visual style. After approval, a walkthrough over the clickable paper prototype usually will be performed in the next iteration.

The prototype walkthrough requires users to simulate performing simple tasks using a clickable paper prototype. The tasks may be based on pre-prepared scenarios or on verbal commands. User activities with a prototype are observed and can be also video recorded.

Walkthrough-based evaluation of paper prototypes is very simplified and usually limited to performing simple tasks which do not require high manual precision or analytical thinking. The objectives of paper prototypes walkthrough testing are limited merely to collecting initial user feedback: validation of the general interaction concept, first observations of users' reactions and mistakes and collecting first suggestions for improving proposed interaction methods.

Interviews

Interviews (preferably group interviews, if possible) are very useful not only in the Analysis phase, but also in the evaluation of prototypes. Interviews are best held at a user's workplace, immediately after finishing the prototype review, walkthrough, or testing (Figure 10.2).

Figure 10.2. Example of interview with a group of users
(Credits: http://cs.queensu.ca/~audrey/projects.htm)

A group interview conducted after the prototype evaluation gives a better opportunity to express their opinions and to display personal attitudes, and provides a sense of greater security in the group, than an individual interview. Moreover,

some valuable information reveals only as a result of interaction among participants, being unlikely to capture in individual interviews.

Detailed instructions on how to plan and conduct interviews with respondents are provided in numerous marketing research textbooks and in the HCI literature such as Sharp (2019); Shneiderman et al., (2017); Hartson and Pyla (2012).

Survey questionnaires

After evaluating the prototype, especially the digital one, users should fill in a questionnaire specifically designed to evaluate user satisfaction with a specific interactive product. Closed questions in the questionnaire collect numerical scores, while open questions collect comments and suggestions for improvements.

Questionnaire design should facilitate of aggregating test results in a spreadsheet, including the method of error correction due missing data points. If the questionnaire is not only a survey tool, but it is also expected to provide data for statistical analysis, it will be necessary to consult a statistician, as well as running some pilot studies, to ensure the accuracy and reliability of the questionnaire as a research tool.

Availability of electronic questionnaires is a clear temptation for the researcher to accelerate the process of gathering responses from respondents. Electronic questionnaires have obviously the following advantages: rapid acquisition of responses, rapid and automatic analysis of data, there are no copying and distribution costs involved, and data are immediately recorded in a database, easy error correction.

There are also some disadvantages: questionable representativeness of respondents, it is difficult to prevent against re-filling of the questionnaire, and to assess the survey reliability and validity.

Detailed guidelines on how to build questionnaires, choose measurement scales and conduct research, are provided in numerous marketing research textbooks and in the HCI literature such as Sharp et al.(2019); Shneiderman et al., (2017); Hartson and Pyla (2012).

Usability testing

Laboratory usability testing

Usability testing is a formal method for usability evaluation of an interactive product. It is usually performed in a laboratory, under controlled conditions (Rubin and Chisnell, 2008). Objective data from measurements during usability tests are supplemented with subjective data, collected from surveys and questionnaires. Usability testing is based on several characteristic features:

- a sample of representative users must be invited as testers;
- users will execute only predefined task scenarios using a hi-fidelity prototype or a finished product;
- users' activity will be video recorded and supervised by a researcher;

- user activity data will be stored for further qualitative and qualitative analysis;
- preferably the development team representative should be present during the test and in the interview following the testing.

The structure of a typical usability test is usually the following:

1. Introduction: the purpose of the study, the duration of the test, the session logistics.
2. Initial interview: ensuring about the representativeness of the users' group.
3. Sample tasks: trivial, no time measurement, tuning the video recording equipment.
4. Test tasks: with full video recording of time on task, errors, etc.
5. Product evaluation survey: filling in a paper questionnaire.
6. Final interview: gathering opinions and comments from test participants.
7. End of the test.

After completion of all tasks (or after approx. 60 minutes, regardless of the results), users are asked to complete a specially designed questionnaire, aimed to collect their opinions about the work experience from testing the application. User's activity during tasks execution is videorecorded using a common digital camera or with support of dedicated software (Figure 10.3).

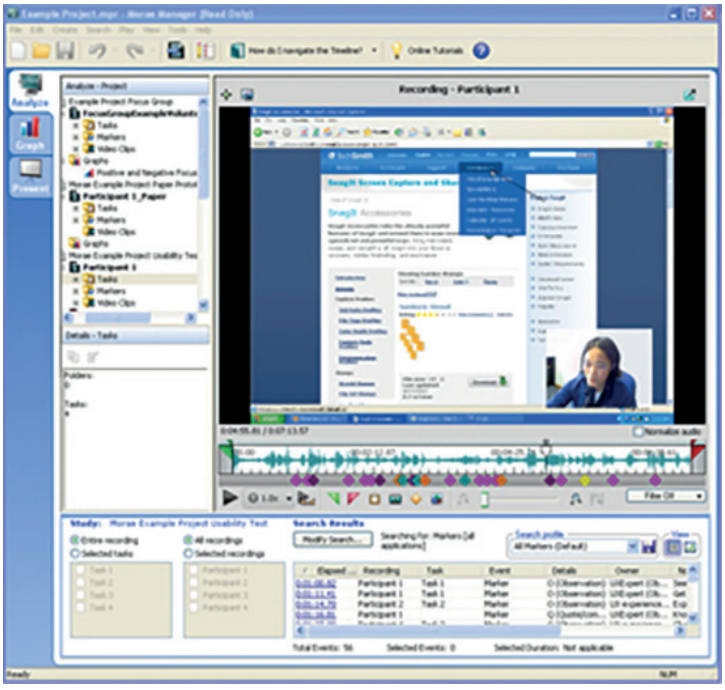

Figure 10.3. Video recording of a user working during a usability test (Credits: http://www.library.illinois.edu/sc/services/usability_testing.html

It is important to record simultaneously the operations performed by the user and the related changes on the screen. For this purpose can be used:

- the system with two cameras, one recording the user screen and the other is recording user keyboard; both video signals are next mixed in a video mixer (software based);
- special software (e.g. Noldus Observer, Techsmith Morae) can be used, which captures the image on the screen and collects all the data from the computer where user is working; the software mixes the video streams in any combination desired by the researcher (Figure 10.3).

In addition to video recording, also conventional data collection techniques can be applied, such as direct observation, time measurements, observation protocols, audio recording or system logfiles recording.

A detailed overview of technologies for usability studies is presented in the work of Rubin and Chisnell, (2008) and Albert et al. (2010).

Remote usability testing

Remote usability testing is using software tools which enable capturing on video user usability session despite the user and the research are in different locations. While the user performs scheduled tasks scenarios following on-screen instructions, remote usability testing software records mouse clicks, mouse movement, page scroll and other user activities while operating a website or an application. The users may work on their computer in their environment (e.g. at home), or using a mobile device such as tablet or mobile phone.

A detailed overview of technologies for remote usability studies is presented in the work of Bleeker de and Okoroji (2018) and Albert et al. (2010).

Web analytics

Analysis of user traffic statistics is a valuable source of information about user behaviour, especially when it comes to indicate sites where users behave in a manner inconsistent with the intention of the designers (e.g. suddenly leave the service or forgo purchase). Analysis of the traffic on a website can determine what is happening, but unfortunately it does not explain why people exhibit certain behaviours. For this reason, the analysis of user behaviour needs support of other (mostly qualitative) methods, which are expected to explain the mechanism of the observed phenomena. Per analogy to web traffic statistics, the telemetry is used for web and mobile applications, monitoring which functionalities and how frequently they are used is specific tasks.

Detailed information on how to use user traffic statistics for improving user experience (UX) is provided in the work of Beasley, (2013).

Low-cost usability testing

In agile projects usability testing should deliver results quickly, frequently, and not necessarily with superior accuracy. For this reason many low-cost usability

testing methods were created, collectively labelled a “guerrilla usability testing”, in contrary to conventional usability studies based on established research methodologies. These low-cost methods include techniques such as testing usability with 5 users, using informal tasks scenarios, integrating prototyping with testing, recording video with mobile phones, conducting very brief interviews or instant one-minute on-line surveys.

In low-cost usability studies the emphasis is not put on methodological rigour or scientific accuracy, but merely on quick informing designers which usability problems were detected and how they should be corrected.

An overview low-cost usability testing techniques can be found in various sources online and in the work of Hartson and Pyla (2016) and Albert et al. (2010).

Planning and conducting usability testing

Preparing tasks scenarios

During usability testing the participants will perform specific test scenarios, which should be carefully prepared in advance. They should be prepared in writing and made available to participants as a paper printouts. They should contain the purpose of the task and what result should be obtained, but without hints what exactly needs to be done in order to achieve the result – the users should discover the correct method by themselves.

Test tasks should be related user’s goals in a given context of use; in other words, they should relate to the most common tasks and problems that a user can try to solve in typical situations. In the initial phase of usability testing a user is often asked to perform one or two sample tasks. The goal of a sample task is to familiarize the user with the product to be tested and to check if the video recorder is working correctly.

The test tasks should be sequenced from the easiest to the most difficult one, because the users usually are not able to accomplish all tasks, especially the more difficult ones. In this situation, collected data can be fully analysed at least from simple tasks, performed at the beginning of the test.

A good task scenario should be helpful in solving a specific design problem, should provide measurable data and refer to specific elements of user interface which are likely to need improvements.

Selecting appropriate methodology

When preparing usability testing, following factors should be considered:

- the goals and scope of the usability testing, expected precision of results and the formality of report;
- the type of tested application affecting the location of testing (desktop - testing indoors, mobile – outdoors);
- ability to collect user-based data by videorecording, performing interviews or surveys;

- local logistics and organizational issues regarding the test, like ability to recruit representative users.

Finally, the key issue is choosing the right type of usability testing: the formative – for improving current design, and the summative – for evaluating a complete and finished solution.

Frequently enough, the A/B testing is popular, while two versions of the same design must be compared basing on users' opinions and task performance results.

An overview methodological issues regarding usability testing planning can be found in HCI textbooks such as Sharp et al., (2019), Hartson and Pyla (2016) and Albert et al. (2010).

Setting up an equipment for usability testing

Following equipment should be prepared in advance to planned usability testing session:

- a test device for the users – computer, tablet, mobile phone or another handheld device;
- videorecording equipment – a camera or two, a tripod, memory cards etc.;
- software tools such as:
  - a video mixer (software based) for mixing video stream from two cameras (the screen and user's face or body);
  - video recording and analytic software (Observer, Morae) - optional;
  - additional equipment such as an eye tracker of face reader - optional.

For usability testing A suitable ambient environment should be also prepared, like an office room or another location resembles actual user's workplace or home. Otherwise outdoor testing should be performed during weather conditions appropriate for usability testing goals (for instance the sunlight when testing mobile application outdoors).

User recruitment

Recruitment of participants for usability testing can be done in various ways. In the case of prototype testing, participants are often recruited from among users who earlier took part in requirements workshops, analytical meetings or other forms of cooperation with designers. Sometimes participants are recruited using the "snowball" method, but sometimes are recruitment is outsourced to an agency that specializes in market research.

The problem of the optimal number of users for usability testing is not uniformly treated in literature. A practical approach to the problem suggests that the optimum sample size is from 5 to 10 participants, but it depends on the type of system to be tested, the size of available laboratory room, availability of participants and the budget available for testing, as well as on methods planned for further data analysis.

More comprehensive guidelines about recruitment of participants can be found in HCI literature: Albert et al. (2010), as well as Rubin and Chisnell (2008).

Ethical aspects

Planning usability testing should include essential ethical aspects regarding users' comfort, security and privacy protection. The researcher should ensure the participation of members on a voluntary basis and obtain the written consent from each participants to take part in the test and agree for the video recording of their work.

Before the test starts, it should be emphasized to the participants that the system is evaluated, not the skills of the users. It is also important to ensure the users that the anonymity of personal data was secured.

Usability tests can be stressful for users, thus test time is limited to a maximum of 90 minutes. Participants should be informed about the possibility of quitting the test at any time. In all situations researchers should present polite treatment of participants and avoid exerting any pressure on them.

Conducting usability testing sessions

Gathering observation data

During the test - in addition to automatic video recording – user behaviour should be observed, including manual note taking, including characteristic user comments observed during the task execution. The problem with usability should be noted is situations such as for instance:

- a user specified an action objective, but despite efforts there is no progress within a certain time (help is necessary from the observer),
- a user specified an action objective and must try several approaches to find a solution, because the system does not suggest how to begin,
- user specified an action objective, has unsuccessfully tried several approaches, and then resigned from the task,
- participant expressed surprise at the specific behaviour of the system, a negative opinion, a problem statement, or a redesign suggestion.

After the test tasks are completed, subjective data should be collected from users using survey questionnaires and interviews.

Usability measurements

Objective data (measurements and observations) collected during the test can be used for calculating results such as:

- time of completion of each task, time values distribution among the participants,
- percentage of tasks completed by the user at a given time,
- percentage of tasks completed correctly by the user,
- percentage of users who have completed a specific task within a given time,
- the ratio of positive to negative comments,
- number of users who showed frustration or dissatisfaction.

In addition to the objective test results, subjective data concerning user satisfaction should be gathered, e.g.:

- evaluation and opinions expressed in the questionnaires and during interviews,
- percentage of users expressing opinions of a particular type,
- aggregated indicators which describe most frequently expressed opinions.

More techniques on data collection during usability tests have been discussed in detail by Albert et al. (2010), and Rubin and Chisnell (2008).

Analysis and reporting user-bases data

Detailed description of techniques useful for analysing data from user-based studies can be found in HCI textbooks such as Sharp et al., (2019), Hartson and Pyla (2016) and Albert et al. (2010). Hereafter only selected issues will be outlined.

Retrospective analysis is a simple technique for an in-depth analysis of video recording made during an usability testing session. Shortly after the sessions (preferably next morning) one of users-testers is invited to play the recording and watch own activity during executing task scenarios. Usually the user is keen to comment own actions, responses of the systems and other issuer making task execution more difficult. These comments are valuable in explaining erroneous behaviours (system response other than expected, or user's intention not adequately mapped to existing functionality) and most importantly, in drawing redesign suggestions.

Analysing quantitative data is simple with a spreadsheet and charts. Quantitative results inform *what*, *when*, *how many*, but leave room for interpretation and explanation to be made by the researcher. On the contrary, qualitative data convey texts, verbal statements, videos or observation notes. They are difficult to analyse and need coding of specific categories, what is subjective and prone to errors. Nevertheless, qualitative analysis is able to inform *why* something happens. Usually the researcher has to perform manual analysis and interpretation, what takes time. Most recently, software tools for qualitative analysis have been available (like Nvivo and other products), using techniques such as text mining for sentiment analysis, and visualizations of factors shaping users' attitude to the product.

Reporting results from user-based studies

There are three types of reports from user-based studies:

- a written report (document) addressed to the project manager or client;
- a slide show to be presented to team members or project managers;
- a research paper, to be published in an academic journal.

In agile IT projects currently the slide show presented at the meeting is the most popular format for reporting results of usability evaluations. This is caused by the pressure of time and the need for direct conclusions, driving further research activities.

Furthermore, decisions can be taken during the meeting when the results were shown and discussed, so the decision process is quick and agile.

Finally, the PowerPoint slides have a landscape format, quick to read and handy for electronic distribution to team members or to other recipients. While paper documentation is no longer popular, slide-based reports are stored in the archive of the project together with other project files.

10.4. Deliverables from evaluation and testing

The Validation phase combines many methods, but for usability-related issues with only three sources of data: the team, the client, and users (customers).

Evaluation methods can be classified as expert-based and user-based, and the latter include many more methods, including empirical testing and surveys.

In IT projects evaluation results form user-based testing need to be compared (and sometimes combined or confronted) with results of expert-based evaluations, or with the outcomes from internal reviews from the team. Some inconsistencies are unavoidable, so again it is the task for designers and project managers to adequately balance different viewpoints primarily for the benefit of clients and users (consumers).

In the Validation phase follows further integration of the technological and the product streams. As a result of validation, certainly some rework will have to be done, and further iterations will be certainly needed. Afterwards, the product should be ready to be handed off to the client for deployment or integration with other systems. After that, the agile team should gather again at the Retrospective meeting, intended to reflect on just completed process.

The Retrospective

In agile projects the retrospective reviews are conducted frequently, typically after each sprint, regarding two aspects: firstly, the quality of a recent outcome (increment), and secondly, the quality the development process – particularly communication and organization issues important for planning the next sprint. Finally, a "big" retrospective is performed for the whole product, regarding product backlog and how far its goals and requirements have been accomplished.

Also, for this book, it is worth looking back – what was done, and forward – highlighting the challenges emerging ahead.

When looking back, this book had an ambition to present how the theories and methodology of HCI (Human-Computer Interaction) evolved in rapidly changing world of digital transformation and prevailing agile IT projects. The pressure from business clients has much contributed to the emergence of agile design and agile management, making them now the leading concepts relevant to IT projects and software development. Similarly, matured methods of user centred design (UCD) had to evolve to deliver results faster, be more cost-effective and more relevant to the needs of agile design teams.

As a result, similarly like in business and IT practice, in this book more emphasis is put to qualitative methods in interaction design, and support they can offer for IT project managers, than to strictly quantitative methods and measurements. Across all stages of a generalized IT project presented in this book, frequent evaluations and intensive communication with users are promoted as a rapid and most efficient user-centred quality management framework regarding product usability and user experience (UX).

Certainly, methods of agile design will be continuously refined, based on even more intensive teamwork and communication with clients and end-users. Furthermore, since the role of remote teamwork keeps growing, future development of agile design methods depends a lot on advancements in remote collaboration between the team and other remotely located project stakeholders. Beyond the Strategy, Analysis and Design phases, further developments can be expected especial-

ly in methods for conducting remote evaluations, testing, validation and providing user feedback for the team.

When looking ahead, we can see serious challenges caused by latest technological advancements, such as:

- natural interaction methods, voice user interfaces, and gesture-controlled multimodal interfaces;
- adaptive components, based on Artificial Intelligence (AI), which are autonomous in learning new behaviours upon data collected from the environment;
- Internet of Things (IoT) paradigm, enabling devices to communicate online, exchange data and autonomously adapt their behaviour;
- industrial robots, autonomous vehicles and cyberphysical systems in industry, and software robots automating office and administrative procedures;
- AI-based surveillance of traffic, transport and public space;
- social media which enrich contemporary social life, but at the same time are very prone to abuse, trolling, violence or spreading fake news.

Among many, newly emerging challenges for IT projects management include:

- adapting the agile approach for very big projects:
  - synchronization of numerous agile teams working in big projects;
  - synchronization of agile teams (usually design and development) in an organization working in traditional design or in administrative, procedural environment;
- evaluating the impact of smart systems on project management:
  - designing human interaction with autonomous (smart) systems that learn and change;
  - adding mandatory risk assessment for evaluating possible impact of smart systems on human safety, security and attitude to autonomous robots;
- evaluating the impact of smart systems to social life:
  - possibility of undermining social trust and stability by malicious campaigns in social media;
  - possibility of AI-based, autonomous systems to be hijacked or reprogrammed for malicious use.

There are many other possible challenges, but they all seem to be related to the fact that digital services become invisible and smart, and thus able to penetrate users' privacy and behaviours in many spheres of life.

It means that future developments regarding interaction design need to address two vital issues: human interacting with autonomous systems operating beyond users' control, and social impact of digital solutions. In relevance, recent report of Stephanidis and Salvendy (2019) specified seven grand challenges that will shape developments in interaction design:

- Human-Technology Symbiosis: humans much more interacting with smart devices, services, materials, and environments, than with “computers” as we know them today;
- Human-Environment Interactions: human living a world which is concealed and ubiquitous continuum between the physical and the digital;
- Ethics, Privacy and Security: designing and implanting solutions which will be beneficial to people living in technologically augmented and intelligent environments;
- Well-being, Health and Quality of Life: using IT for fostering a healthy life, psychological well-being and quality of life;
- Accessibility and Universal Access: using IT for improving quality of life of disabled and older persons, especially in the view of aging populations;
- Learning and Creativity: using IT for supporting people in learning, creating, and sharing knowledge for development in both individual and social dimensions;
- Social Organization and Democracy: shifting IT developments from coping with earthy problems such as economy, poverty, climate change towards smart societies related issues such as social participation, social justice, and e-democracy.

The above list of challenges will be never exhaustive. All the time new challenges will be emerging, but – hopefully – also new opportunities will be available to provide effective solutions.

As a deep retrospective, for IT business community a permanent reflection is therefore needed on issues such as:

- the impact of new technological developments on social life;
- developing a regulatory framework how smart systems are designed, implemented and operated;
- developing ethical responsibility of designers of smart systems, as well as multidisciplinary skills of development team members;
- rethinking the role of training and education of IT professionals with regards to social perspective.

Last but not least, in the design activity of each IT professional and manager, an ongoing professional retrospective is needed to see relevant digital products and IT projects as aimed at improving the quality of life in all human activities, from work, education, entertainment, to business and social life.

References

1. Adobe (2013). Adobe 2013 Mobile Consumer Survey results. URL: https://empoweryou.ca/wp-content/uploads/2013/12/Adobe-2013-Mobile-Consumer-Survey-Result.pdf (Accessed: 11 April, 2021)
2. Adobe (2015). Mobile Consumer Report. URL: https://wwwimages.adobe.com/content/dam/acom/en/solutions/pdfs/adobe_mobile_consumer_study.pdf (Accessed: 7 June, 2021)
3. Albert B., Tullis T., Tedesco D. (2010). Beyond the Usability Lab. Conducting Large-Scale Online User Experience Studies. Morgan-Kaufmann.
4. Beaird, J., and George, J. (2014). The Principles of Beautiful Web Design. SidePoint Pty. Ltd., Fitzroy, Australia.
5. Beasley, M. (2013). Practical Web Analytics for User Experience. Morgan Kaufman.
6. Becker, C.R. (2020). Learn Human-Computer Interaction. Packt Publishing, Birmingham.
7. Bittner, K., Kong, P., Naiburg, E., and West, D. (2017). Nexus Framework for Scaling Scrum, The: Continuously Delivering an Integrated Product with Multiple Scrum Teams. Addison Wesley.
8. Bleeker de, I., and Okoroji, R. (2018). Remote Usability Testing. Packt Publishing, Birmingham.
9. Csontos, B., Heckl, I. (2020). Improving accessibility of CMS-based websites using automated methods. Universal Access Information Society. URL: https://doi.org/10.1007/s10209-020-00784-x.
10. Chmielarz, W. (2016). Information Technology Project Management. Warsaw University.
11. Cobb, C.G. (2011). Making Sense of Agile Project Management: Balancing Control and Agility. Wiley.
12. Cohn, M. (2013). Succeeding with Agile. Addison-Wesley.
13. Dix, A., Finlay, J. Abowd, G., Beale, R. (2004). Human-Computer Interaction. Prentice Hall.

14. Dumas, J.S. and Redish, J.C. (1999). A Practical Guide to Usability Testing. Intellect.
15. Galitz, W. (2013). The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. Wiley.
16. Grobelny, J., Michalski R. (2020). Investigating human visual behavior by hidden Markov models in the design of marketing information. In: Cassent D.J (ed): Proceedings of the AHFE 2019 International Conference on Human Factors and Simulation, July 24-28, 2019, Washington D.C., USA. Springer, 2020. 234–245.
17. Gottesdiener, E. (2002). Requirements by Collaboration: Workshops for Defining Needs. Pearson.
18. Hartson, R. and Pyla, P. (2012). The UX Book: Process and Guidelines for Ensuring a Quality User Experience. Waltham, MA: Morgan Kaufman.
19. Hassenzahl, M. (2008). User experience (UX): Towards an experiential perspective on product quality. ACM International Conference Proceeding Series. 339, pp. 11–15. 10.1145/1512714.1512717.
20. Hassenzahl, M., and Tractinsky, N. (2006). User Experience – a Research Agenda. Behaviour and Information Technology, Vol. 25, No. 2, March–April 2006, pp. 91–97.
21. Krug, S. (2005). Don't Make Me Think: A Common Sense Approach to Web Usability. New Riders Publishing.
22. Hix, D. and Hartson, P. (1993). Developing User Interfaces. Wiley.
23. Humphreys, A., and Grayson, K. (2008). The Intersecting Roles of Customer and Producer: A Critical Perspective on Co-Production, Co-Creation and Prosumption. Sociology Compass 2: pp. 963–80.
24. Jacobsen, J., and Meyer, L. (2019). Praxisbuch Usability und UX. (in German). Rheinwerk Computing, Bonn.
25. Jayaswal, B.K., and Patton, P.C. (2009). Design for Trustworthy Software. Prentice-Hall.
26. Kearney, M., Gash, D., and Boxhall, A. (2020). Introduction to ARIA. URL: https://developers.google.com/web/fundamentals/accessibility/semantics-aria (Accessed: 9 June, 2021)
27. Malewicz, M., and Malewicz, D. (2019). Designing User Interfaces. 4Hype, Warsaw.
28. Mendoza, A. (2013). Mobile User Experience: Patterns to Make Sense of it All. Morgan Kaufman.
29. Meroni, A. and Sangiorgi, D. (ed.). (2011). Design for Services. Farnham: Gower.
30. Morvile, P. (2004). User Experience Design. Semantic Studios. URL: https://semanticstudios.com/user_experience_design/ (Accessed: 17 March, 2021)
31. Neil, T. (2015). Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps. O'Reilly.

32. Newman, M.W., Lin, J., Ho, J.I., & Landay, J.A. (2003). DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. Human-Computer Interaction, 2003. 18(3): pp. 259–324.
33. Nielsen, J. (1993). Usability Engineering. San Diego, CA: Academic Press.
34. Nielsen, J. (1994). How to Conduct a Heuristic Evaluation. URL: https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/ (Accessed: 7 January, 2021)
35. Nielsen, J. (2000). Designing Web Usability. New Riders.
36. Nielsen, J., Loranger, H. (2006). Prioritizing Web Usability. New Riders.
37. Nielsen, J., Molich, R. (1990). Heuristic Evaluation of User Interfaces. Proc. ACM CHI'90 Conf. (Seattle, WA, 1–5 April), 249–256.
38. Nielsen, J. and Budiu, R. (2013). Mobile Usability. Pearson.
39. Norman, D. (1999). The Design of Everyday Things. Basic Books.
40. Olsen, D. (2003). Developing User Interfaces. Morgan Kaufmann.
41. Osterwalder, A. (2010). Business Model Generation. John Wiley and Sons.
42. Patton, J. (2014). User Story Mapping. O'Reilly Media.
43. Pearrow, M. (2000). Web Site Usability Handbook. Charles River Media.
44. Phyo, A. (2003). Return on Design. Smarter Web Design That Works Pearson.
45. Pinhanez, C. (2009). A Service Science Perspective on Human-Computer Interface Issues of Online Service Applications. International Journal of Information Systems in Service Sector, 1(2), pp. 17–35.
46. Pressman, R.S. (2000). Software Engineering. A Practitioner's Approach. Prentice-Hall.
47. Rothenburg, S. (2007). Sustainability Through Servicizing. MIT Sloan Management Review, Winter 2007. URL: https://sloanreview.mit.edu/article/sustainability-through-servicizing/ (Accessed: 3 March, 2021)
48. Rubin, J., Chisnell, D. (2008). Handbook of Usability Testing. How to Plan, Design and Conduct Effective Tests. Wiley.
49. Sharp H., Rogers Y., Preece J. (2019). Interaction Design. Beyond Human-Computer Interaction. Wiley.
50. Shneiderman, B., Plaisant, C., Cohen, M., and Jacobs, S. (2017). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson.
51. Shore, J., and Warden, S. (2008). The Art of Agile Development. O'Reilly Media.
52. Sikorski, M. (2008). HCI and the Economics of User Experience. In: Law, E., Hvannberg E., Cockton, G. (eds): Maturing Usability, Springer-Verlag, London, 2008, pp. 318–343.
53. Sikorski, M. (2012). User-System Interaction Design in IT Projects. Gdansk University of Technology.
54. Sikorski, M. (2013). Evolution of End-User Participation in IT Projects. In: Pańkowska, M. (ed.): Frameworks of IT Prosumption for Business Systems Development. IGI Global Hershley, New York, pp. 48–63.
55. Snyder, C. (2003). Paper Prototyping. Morgan-Kaufmann.

56. Sommerville, I. (2016). Software Engineering. Pearson Education.
57. Stellman, A., and Greene, J. (2013). Learning Agile: Understanding Scrum, XP, Lean, and Kanban. O'Reilly.
58. Stephanidis, C., Salvendy, G. and 30 others (2019). Seven HCI Grand Challenges. International Journal of Human-Computer Interaction, 35:14. 1229–1269. DOI: 10.1080/10447318.2019.1619259.
59. Stickdorn, M. and Schneider, J. (2010). This is Service Design Thinking. Amsterdam: BIS Publishers.
60. Thallmaier, S.R. (2015). Customer Co-Design. Springer-Gabler. Wiesbaden.
61. Tognazzini, B. (1995). Tog on Software Design. Addison-Wesley Professional.
62. Toffel, M.W. (2008). Contracting for Servicizing. Harvard Business School. URL: http://dx.doi.org/10.2139/ssrn.1090237.
63. Tullis, T., Albert, B. (2008). Measuring the User Experience. Morgan Kaufman.
64. Vargo, S.L., and Lusch, R.F. (2008). Service-dominant logic: continuing the evolution. Journal of the Academy of Marketing Science, 36(1), 1–10.
65. Wong, E. (2020). Heuristic Evaluation: How to Conduct a Heuristic Evaluation. URL: https://www.interaction-design.org/literature/article/heuristic-evaluation-how-to-conduct-a-heuristic-evaluation (Accessed: 3 June, 2021)
66. W3C (2020). Mobile Accessibility. World Wide Web Consortium. URL: https://www.w3.org/WAI/standards-guidelines/mobile/ (Accessed: 9 May, 2021)

Standards

1. ISO/IEC 9126. Software Quality Characteristics. International Standard.
2. ISO 9241-11. Ergonomics of Human-System Interaction. International Standard.
3. ISO 13407:1999. Human-Centred Design Processes for Interactive Systems.

Internet Resources (all accessed: 19 May, 2021)

https://www.scrum.org
https://wdrfree.com
https://teamquest.pl
http://ui-designer.net/usability/usersgoals.htm
https://commons.wikimedia.org/
https://freepik.com
https://uxdesign.cc
https://www.pngwing.com
https://docs.microsoft.com/en-us/windows/uwp/
https://interaction-design.org
https://www.microsoft.com
https://robodk.com
https://i-scoop.eu
https://bbc.co.uk

https://www.oreilly.com
https://www.ssa.gov/accessibility/andi/help/install.html
https://www.packtpub.com
https://hardrock.com
https://chatbotsmagazine.com
https://www.pinscreen.com/vitualassistant
https://www.exoplatform.com
https://www.runtastic.com
https://www.androidpatterns.com
https://www.justinmind.com/ and https://wiki.smu.edu.sg/is480/
https://historypin.org
https://aaptiv.com
https://www.freepik.com
https://www.scrum.org
https://www.liveworkstudio.com
https://psdrepo.com
https://freepik.com
https://4ba.pl
https://xnsio.com/
http://www.msrblog.com/
https://uxdesign.cc
http://www.interaction-design.org
https://www.interaction-design.org
https://www.business2community.com
https://learningfundamentals.com.au/resources/
http://www.agilemodeling.com
https://uxplanet.org
http://www.debaoki.com/storyboards
http://www.visualux.design/sketching
https://www.vippng.com/maxp/hJoimmm/
http://wireframes.linowski.ca
https://www.mockplus.com
https://www.archimetric.com
https://brainhub.eu
https://community.sap.com
https://www.connected.io/
https://www.axure.com/
https://www.mockplus.com/app-prototyping-tool
https://www.asktog.com
https://www.sapdesignguild.org/
http://cs.queensu.ca/~audrey/projects.htm
http://www.library.illinois.edu/sc/services/usability_testing.html