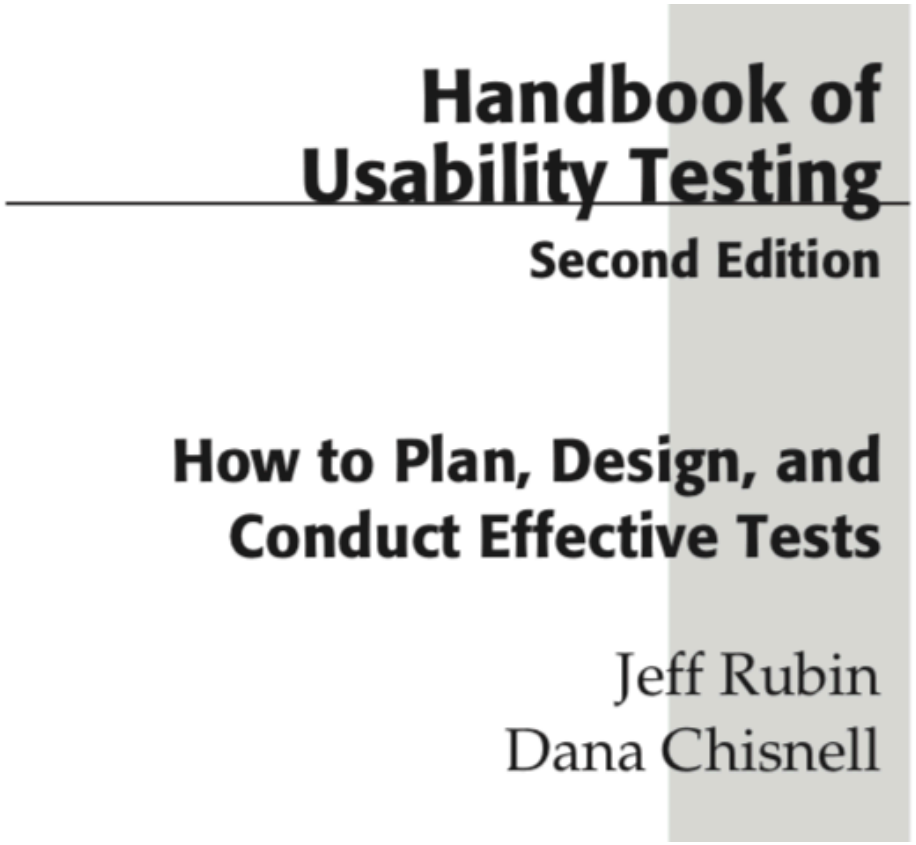
FEUP
Departamento de Engenharia Electrotécnica e de Computadores

Multimedia Systems - M.EEC057

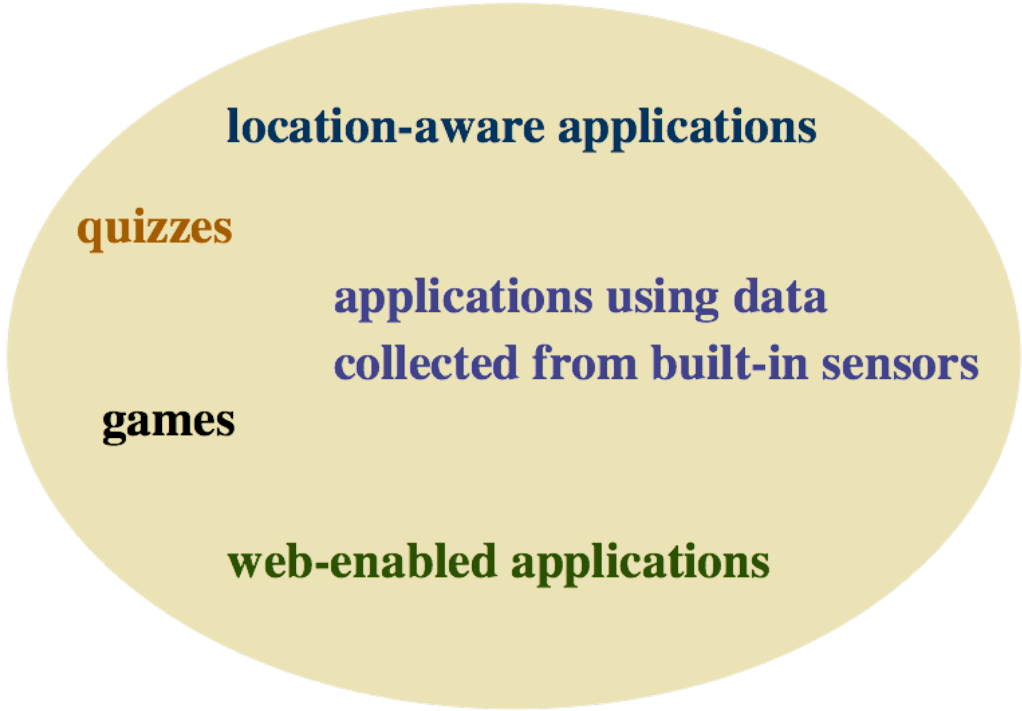
- Principles and process towards the design of multimedia applications (part 2)
- Princípios e processo para a o projecto de aplicações multimédia (parte 2)

Additional reading



Some reflections on functionality and modality in mobile environments

- Picking up from last class



Using sound

- Our dominant experience when interacting with multimedia applications on a computer is through the screen, keyboard and mouse
  - exchange of information between the app and the user is usually done in a visual way
- In the real world we use other types of cognitions, namely the auditory one, which plays an extremely important role in our perception of real facts
- However, in some scenarios, the use of sound is not viable or desirable
  - e.g., we don't want to have to talk loudly into the computer/device or the computer to make noise in a workplace
- This inability to use sound translates into a significant loss in the richness of the application and the user experience
  - thus, we must bear in mind that sound is always an important component for a rich and complete cognitive experience and try to include it whenever possible

Using sound (2)

- When it is possible to effectively combine sound with visual elements, they both benefit
  - imagine images of a ball hitting a wall with or without sound
    - With sound incorporated it is possible to obtain a large amount of information about the space when the action occurs and the action itself
      - if in a room, how large the room is;
      - does it echo?
      - a notion of the speed; the consistency of the ball, etc.
  - That information helps in shaping our perception of the message that the application is trying to pass

Using sound (3)

- When creating multimedia applications it is important to provide to the user all the possible means for him/her to capture the message and to feel motivated to work with the app
  - to ask the user to interact through sound is not new or demanding per se but may require the user to be more focussed/attentive
    - *activity: argue whether this is an advantage or constraint*
  - On the other hand it may provide the app with supplemental data on the user
    - *activity: discuss what could be this information and how it could be used*

Using sound as an input to the system

- In a mobile environment, it is possible to use sound in different dimensions (not only the voice of the user)
  - Taking profit of built-in sensors
    - Using the microphone, it is possible to detect levels of surrounding noise and accordingly adapt the way information is being presented on the interface
      - Adjust the volume
      - opt by a voice-to-text conversion
    - Using other sensors, namely the accelerometer and gyroscope, the app may infer that the user is engaged in other tasks and opt by using sound to request input from the user

Using sound as an input to the system (2)

- Sound recognition brings "the sense of hearing" to smartphones, enabling new applications, such as
  - Audio event detection and notification
    - e.g., recognising a knock at the door and then alerting the user
    - in surveillance applications
  - Scene recognition and adaptation
    - combinations of sounds to infer type of location, such as a chaotic coffee shop or a calm home office
  - Sound tagging and retrieval
    - automatically identify and tag multimedia or audio based on detected sounds
      - e.g., videos taken with a mobile phone featuring persons laughing, or crying or arguing.

Using sound as an input to the system (3)

- It may solve a lot of real-world problems, but at what cost?
- For the system to detect and recognise types of sounds, the sound data needs to be acquired and processed
  - Sensing capability
    - which sensors? how often should new data be acquired?
    - battery consumption vs data accuracy
      - this is a precious resource in handheld devices, shared by many apps running on the device
  - Analysis capability
    - nowadays this it is commonly implemented resorting to AI/ML
      - requires huge amount of training data, that should be relevant and diverse to be representative
      - typically heavy processing
        - if implemented in the device, consumption will be high
        - if implemented in the cloud, privacy and security concerns will arise
          - and expensive cloud infrastructure may be required

Examples of apps using sound as input

- A well-known app is the one that records an excerpt of a music that is playing on the radio and identifies the music and song player (Shazam)
  - what is the problem that it solves?
- Sound level measurements
  - Decibel X / Sound Meter / Decibel Pro / Too Noisy Pro
- Sound detection
  - Google Live Transcribe
  - recognising alarm sounds to help deaf and hard-of-hearing people
    - Wavio SeeSound
  - But also for malicious purposes
    - acoustic signals produced from typing on a keyboard can be acquired and analysed to reveal exactly what was typed (passwords …), even in noisy environments

Using sound as an output of the system

- What is more frequent to find is the usage of sound as a means to provide feedback to the user
  - Many audio elements are commonly used
    - notifications
    - alerts
    - error messages or success messages
    - keystrokes.
    - clicks
    - sound animation effects
    - narration (e.g., voice GPS and driving directions!)
    - background music

Let's look at some examples using sensed data - Example 1

- Problem
  - The cell phone rings when you're at the movies, in church or attending classes, etc.
    - what are the causes for the problem?
      - you forgot to put it in silence or to switch it off …
- Ideas to solve the problem?
  - create an agenda and the app automatically/periodically checks it to know whether or not to silence the phone; if a phone call arrives, the app automatically notifies the user via sms; and, sends a message to the caller indicating the user is busy …
  - an app that detects the user has moved to a different place and automatically asks whether it should silence the phone
  - An app that detects ambient sounds to infer the location, automatically muting , lowering or increasing the volume and turn on automatic notifications
  - …

Exemple 1 (2)

- Requirements to support / functionality to offer
  - allow the user to define the time period during which the phone should be kept in silence
  - allow the user to define exceptions (e.g., call received from some number are allowed to ring)
  - if the phone has been silentes and a call arrives, a message is sento to the caller
  - allow the user to define (personalise) multiple messages according to the place where he/she is and the calling number
  - …

Example 2

- Problem
  - Lack of physical activity among youth
  - What are the causes?
    - too much TV, computer games, and so on
    - lack of time, lack of organisation
    - lack of spaces to practice sports, lack of team mates
    - …
- ideas?
  - app to create interest groups and schedule events
  - app that monitors physical activity and alerts in case of long inactivity periods
  - app that checks weather conditions/forecasting and, cross-checking with the user's agenda, suggests walkings tracks, bicycle ridings, etc
  - app that presents a list of available places for practising sports near the user's location
  - …

Example 2 (2)

- requirements / functionality?
  - Allow the user to add members to the interest group
  - Allow the user to define periods of inactivity
  - Allow the user to define and modify sport interests
  - …

More examples

- Problem
  - fast calculation of costs when shopping in a supermarket
- ideas?
  - app that allows the user to progressively insert the products' prices, continuously performing the calculations and sending an alert when a certain amount is reached
  - …
- requirements / functionality?
  - …
- Problem
  - Memorising engagements and complying to them
- ideas?
  - app that takes a photo of the place where the commitment has been assumed or of the person with who the commitment has been made, attaching geographical data, and allowing the user to write notes on the photograph or to add a verbal description; when the time comes, the user is notified with some anticipation
  - …
- requirements /functionality?
  - …

Conceptual modelling

- The conceptual model takes the idea and identified requirements and defines the form (material/physical) that this idea will take, supporting the requirements
  - Equivalent to the model we mentally make about how something should be done
  - For example, when we think about scheduling a meeting, we think about an agenda or a diary and not a software
    - although nowadays the most common is to use software to schedule events
- Because our mental model of the "scheduling event" concept is a calendar or an agenda, the software that allows us to do this, will be graphically/visually presented as an agenda or calendar

Conceptual modelling (2)

- In practice, a conceptual model is
  - a high-level description of how a system is organised and works
    - normally in the form of a visual description, accompanied by text
  - a model that materialises the expectations of the users, meeting hopefully their/ours/most people's own mental model

Benefits of conceptual modelling

- helps explain the purpose of the application we have in mind
  - what is it intended to do and what value will it have for the user
- provides a high-level understanding of how the application will work
- the more the conceptual model relates to existing mental models, the more easily the message will pass and be understood
- by building a conceptual model of our application, we have the opportunity to assess how it fits into the users' mental models and therefore expectations
  - and, if necessary, make changes to make this happen
    - which will increase the likelihood of being intuitive and better appreciated

Benefits of conceptual modelling (2)

- It allows us to understand to what extent the way we intend to implement our idea is in accordance with the mental models and expectations of different users
- It allows us to identify aspects of our model that were not yet mentally thought out by the user and help the user to create those mental models
- For example, a user who has never used event scheduling software is likely to have never received automatic notification of a meeting
  - and so it doesn't have a mental model for this functionality

Approach towards conceptual modelling

- If the conceptual model that we develop is not in accordance with the users' mental model, it is natural that the application will cause discomfort
  - the phenomenon of cognitive friction occurs
    - it occurs when a user is confronted with an interface or device that appears to be intuitive but offers unexpected results
- This mismatch between the result of an action and the user's expectations causes frustration
  - with consequent deterioration in the quality of experience

Approach towards conceptual modelling (2)

- Thus, when designing the conceptual model for our application
  - if the objectives to attain with the system/application are already possible and related actions are common knowledge
    - we must adopt a model that matches the corresponding mental model of this objectives and action
    - intuitive procedures that match expectations
  - if the goal of the system and the actions to be performed are new, for which the user does not yet have a mental model
    - we must adopt a new model that does not get confused with something that already exists but does something else
      - and we use the conceptual model to "teach" the user
    - Minimizes the risk of creating cognitive friction

What to include in the conceptual model?

- tasks, actions, objects, attributes, terminology
- Tasks
  - purposes of the system, **what** the user will get from the system
    - high-level, more generic
  - should be presented in current language/speech
    - avoiding introducing strange concepts
- Actions
  - Operations the user needs to perform with the system to reach a certain goal or accomplish a certain task (**how** to reach a goal)
    - low-level, more granular

Tasks vs Actions

- Tasks
  - represent high-level activities or objectives that can be accomplished within a system or a process. They are typically described in a more abstract and non-specific manner and they correspond to solving a certain problem (for example, "memorising events")
  - are often used to define the overall goals or functionality offered by a system. They provide a top-down view of what the system or process is supposed to achieve
  - may not be tied to specific individuals or roles but instead describe the general activities or that can to be performed or the functionality offered
- Actions
  - represent specific, detailed steps or operations that are performed to accomplish a task or to transition from one state to another within a system or process
  - are more granular and concrete than tasks. They describe the "how" of achieving a task and can include specific instructions, algorithms, or procedures
  - are typically associated with specific individuals, roles, or components responsible for carrying them out. They describe the responsibilities and behaviors at a lower level of detail

Tasks vs actions example

- e.g., consider a library management system and the task: "Manage Library Inventory"
  - This is a high-level objective or function of the library system. It doesn't specify how the inventory management will be done but rather what the system is capable to achieve
  - Actions related to the "Manage Library Inventory" task:
    - "Scan and Catalog New Books"
    - "Check Out Books to Patrons"
    - "Update Inventory Records"
    - "Handle Returns and Overdues"
- These actions provide the detailed steps involved in achieving the task, in reaching the goal of managing the Library inventory. They specify who performs these actions and how they are executed.
- In summary
  - **tasks** represent overarching objectives or functions at a higher level of abstraction, the "WHAT" the system will offer
  - **actions** describe specific, lower-level steps or operations needed to perform to accomplish those tasks

What to include in the conceptual model? (2)

- Objects
  - physical objects that will be represented in the application and with which the user will need to interact
    - e.g., a calendar is a physical object
      - but it is still expressed in abstract way

Examples of objects and their atributes and actions

Application	Object	Attributes				
eCommerce	product	name	price	description	manufacturer	availability
	customer	name	email	shipping address	payment data	order history
Human Resources management	employee	employee ID	name	position	department	salary
	open positions	job title	description	requirements	application deadline	responsible
Library Mng System						

Application	Object	Actions				
eCommerce	product					
	customer					
Human Resources management	employee					
	open positions					
Library Mng System						

Examples of objects and their atributes

- Application Deadline
- Hiring Manager
- Patient Information System (Healthcare):
- Object: Patient
- Attributes:
- Medical Record Number
- Name
- Date of Birth

What to include in the conceptual model? (2)

- Actions and attributes
  - associated to the objects
    - action: what will the user be able to do with the object?
      - e.g., select a date for an event in/with the calendar
    - attributes: what should the object have so that the user may perform the action?
      - e.g., to select a date for an event, the calendar should have starting and end date, contacts, etc.
- Terminology
  - establish a single terminology for naming tasks, objects, actions and attributes
    - e.g., if on one screen, scheduling is referred to as "booking meeting", in another screen it should not be named as "booking appointment"

Results of the conceptual modelling

- The conceptual model can be used to create use cases
  - and model the user interaction with the application
- Provides clues about the approach to take to UI design
- In general allows establishing more precise ideas about a number of aspects:
  - What tasks can users perform?
  - What is the value of the application to the user?
  - What are the objects you will work with?
  - What relationships exist between objects?
  - What will users be able to do with the objects?
  - What characteristics should the objects have?
  - What terminology to use?

Results of the conceptual modelling (2)

- The conceptual model is not the UI
  - it is the set of concepts associated with the application and the relationships that exist between them
    - e.g., in a Calendar application, the associated concepts can be creating and managing calendars, events, reminders, addresses, contacts, etc.
    - with the relationship between them: "an event is marked on the calendar indicating address and contacts and a reminder must be sent to the contacts for each event marked."
    - which should be as familiar as possible to the user
- The design of all these concepts is done later by designing and implementing the UI based on the conceptual model

Results of the conceptual modelling (3)

- The conceptual model obtained is then represented by the application interface
  - the UI will be the realisation of that conceptual model, it will transmit the conceptual model to the application user
  - objects and tasks will appear in the UI and will be presented to the user as defined in the conceptual model
- If the conceptual model does not follow the mental model it can create friction/confusion/difficulties for the user

Conceptual modelling - summary

- Conceptual modelling is a crucial step in software development
  - it enables to create a high-level representation of the essential concepts and relationships within a system
  - used during the initial phases of software development
    - to understand and formalize the requirements of the system
    - to create a high-level architectural design of the software system, identifying the major components, their interactions, and the overall structure
    - helps developers clarify what the software should do and how it should behave
      - ensuring that user's expectation can be met, that no friction will occur

Use cases

- After having designed the conceptual model, use cases can be developed
  - detailing the possible interaction scenarios of the user with the application
  - correspond to the execution of a series of tasks by the user with a certain sequence and the corresponding reactions of the application
- e.g., for the application "Agenda", use cases may be defined to
  - create an event
  - add a contact to the list
  - delete an event
  - visualise the calendar
  - etc.

Use cases (2)

- Use cases provide the description of the user-application interaction sequence
  - they don't include any details about the GUI, about graphical elements that will appear on the screen
  - e.g, for the "Create event" use case the description would be
    - "the user opens the calendar and selects the date; the application asks for a name for the event; the user provides the event name; the application saves the event."
  - nothing would be said about how to open the calendar such as double-clicking on the "calendar" icon; nor on how to select the date to create the event such as, right-click on the desired date"; nor on how to provide the event name

Sketching the UI

- Building on the conceptual model and uses cases, the UI can be drafted using the following already acquired knowledge:
  - tasks the user can execute/perform
  - objects with which he/she can interact
  - relationships between objects
  - features of objects
  - actions that may be operated upon the objects
  - terminology to use
  - how the user should perform the tasks
    - the sequence of actions that should be executed for each task (from the use cases)

Sketching the UI (2)

- sketching the UI corresponding to converting the abstract concepts, tasks, actions, etc., into concrete elements of the UI
  - buttons,
  - icons
  - menus
  - text boxes
  - layouts
  - etc.
- and while in that process, use cases may be adjusted to better correspond to the structure of the UI (two way process)

Sketching the UI (3)

- Finally a prototype(s) can be built before the final application

- Next topic:
  - Prototyping

References

- https://uxmag.com/articles/the-secret-to-designing-an-intuitive-user-experience
- https://www.sciencedirect.com/science/article/pii/S0003682X17309945
- https://neurosciencenews.com/smartphone-typing-attack-14881/
- http://personales.upv.es/thinkmind/dl/conferences/achi/achi_2020/achi_2020_3_10_28007.pdf
- https://webapphuddle.com/conceptual-model/
- https://www.interaction-design.org/literature/article/we-think-therefore-it-is-conceptual-modelling-for-mobile-applications
- https://uxplanet.org/how-to-create-personas-step-by-step-guide-303d7b0d81b4
- https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them