**POM 09 – Usability Management**

*Terminology:*

**Usability**: measures how well a user can utilize the system functionality and is measured by five categories

1. **Learnability**: how easy/fast can a user learn the use of the system?
2. **Efficiency**: how many steps does a user require to accomplish a particular task?
3. **Memorability**: how quickly can a user reestablish proficiency?
4. **Errors**: how many errors do users make, how severe are these errors, and how easily can they recover from the errors?
5. **Satisfaction**: how pleasant is the design of the user interface?

"**The system is easy to use**": one of the most frequently misused terms, especially in advertising (often these systems are actually unusable)

"**Unusability**": the user has extreme difficulties to learn or to use the system

*User interfaces are hard to design:*

* The developer and the user are not the same person
  + Software engineers communicate mostly with other developers
  + User interface development is about communicating with users
* The user is always right …
  + Consistent problems are the system’s fault
* … but the user is not always right
  + Users are not designers
* User interface takes a lot of software development effort
  + ~50% of design, implementation and maintenance
* Managers must be involved (usability management)

**How to address usability: prototyping:**

*Prototyping definition:*

* is externalizing and making concrete a design idea for the purpose of evaluation.
* is an early sample or model built to test a concept or process or to act as a thing to be replicated or learned from.

*Why Prototyping?*

* Instant gratification (sofortige Zufriedenstellung)
* Tangibility (Greifbarkeit): a prototype helps to understand a system early on
* Improves poor communication
* Allows early decision making
* Mistakes can be found early: **“We want instant prototypes. They allow us to make more mistakes faster”**

*Prototyping techniques:*

* Haptic prototypes
* Paper prototypes
* Wireframe prototypes (also called mockups)
  + Term from 3D computer graphics
  + Used for the schematic presentation of a screen or webpage
* Storyboard
  + Sequence of Wireframes
* Low-fidelity and high-fidelity prototypes
* Scenario based video prototypes

**User interface prototyping**

*Tips for creating paper prototypes:*

* Create as many prototypes as possible
* Invest as little time as possible for the first iterations, they don‘t have to be perfect and you get feedback anyway!
* Ask persons for feedback who are in the target group of your software
* Don‘t limit yourself by thinking „Is this even possible?“
* Involve as many persons as possible

*Failures are helpful:*

Henry Petrovski:

* Better information comes from designs that fail rather than from those that succeed
* Reason: failures draw more scrutiny; without failure, complacency sets in
* „Success in engineering is defined by its failures“

*Knowledge must be falsifiable:*

* Karl Popper (“objective knowledge”):
  + There is no absolute truth when trying to understand reality
  + One can only build theories, that are “true” until somebody finds a counter example
* The truth of a theory is never certain
  + We can only use phrases like: “by our best judgment”, “using state of the art knowledge”
* **Falsification**: the act of disproving a theory or hypothesis

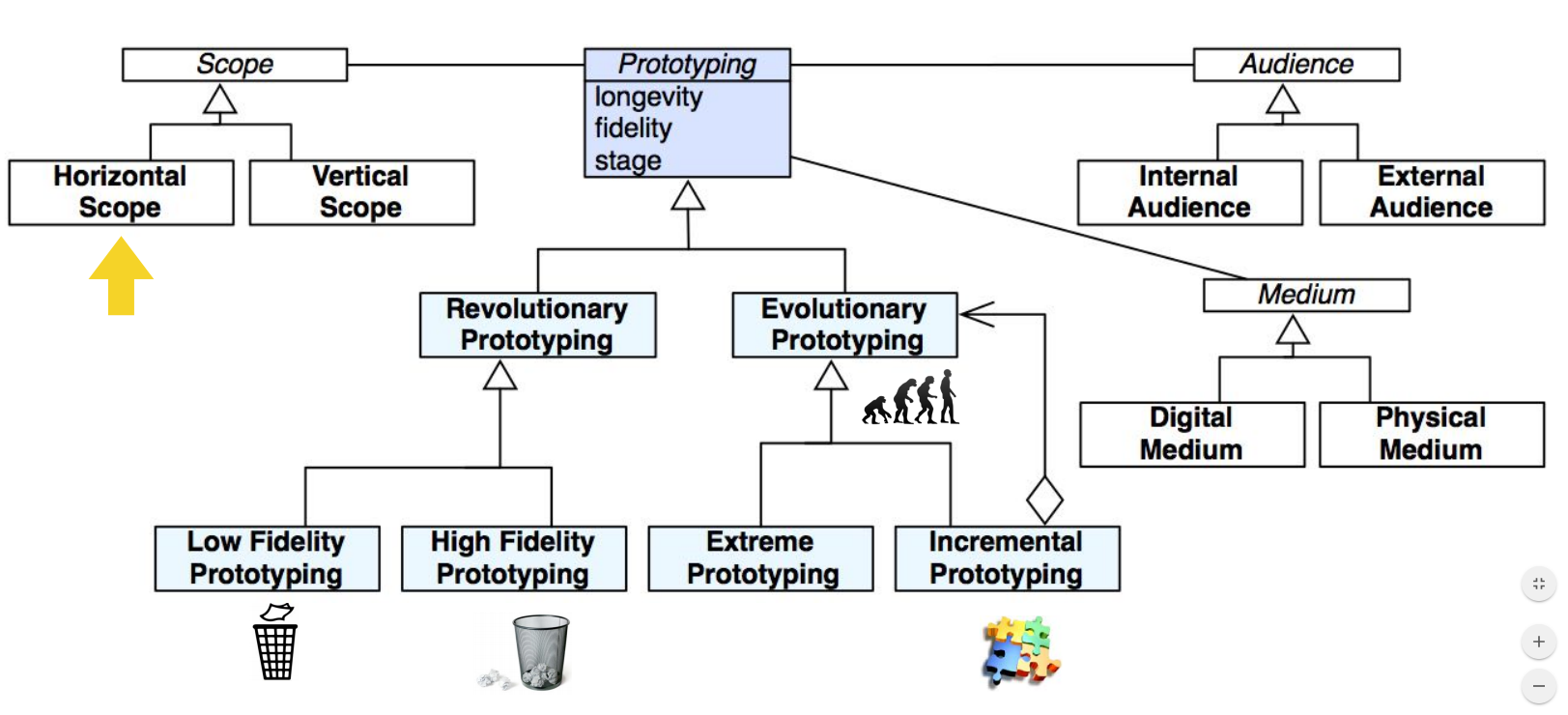
*Consequence for software systems:*

* In software engineering, any system, including a user interface, is a model and thus a theory:
  + We build models to find counter examples
  + Techniques: requirements validation, user interface testing, review of the design, source code testing, system testing, etc.
* **Testing**: the act of disproving a model
* **Usability testing**: the act of testing a user interface, i.e. disproving a user interface model
* We can do these tests with prototypes

*A typical prototyping process:*

1. Low-fidelity prototype
2. High-fidelity prototype
3. Application

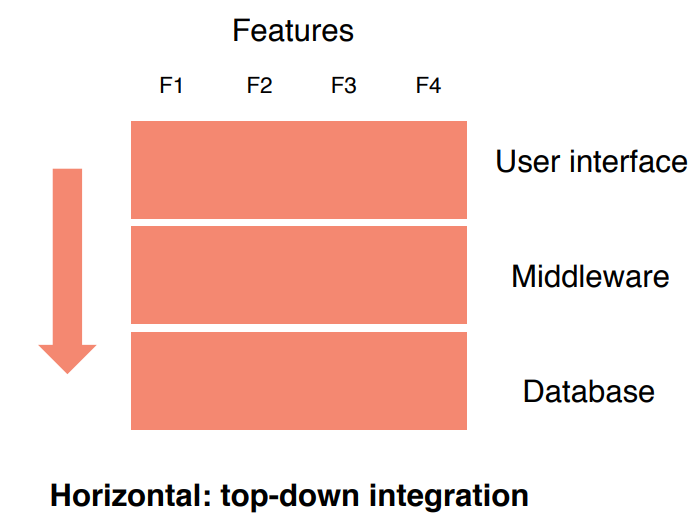
*Types of Prototyping:*



*Horizontal vs. vertical prototypes:*

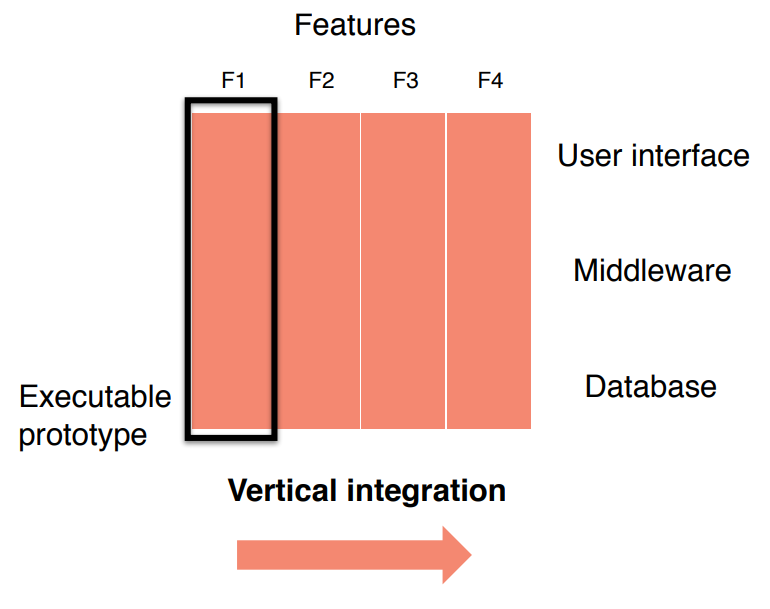
**Horizontal prototypes**

* Show wide range of features
* Horizontal integration
* Bottom up, top down
* Used in linear processes:
* No full implementation up to the end

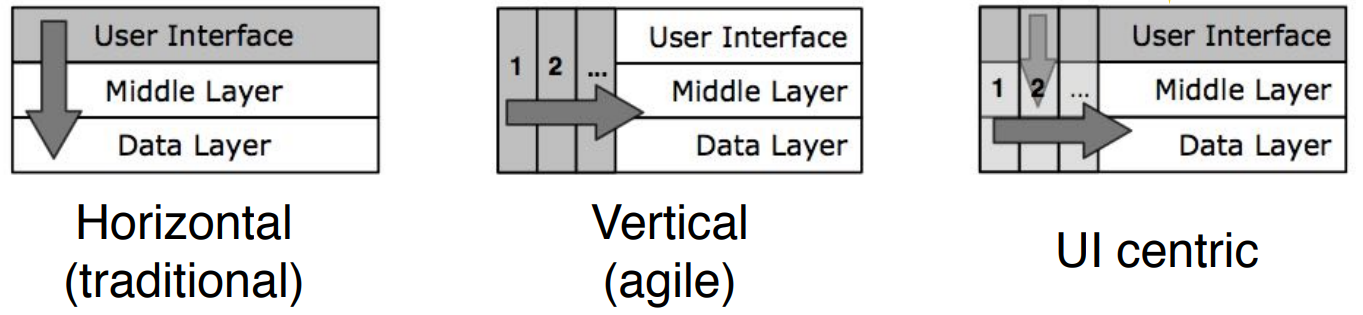


**Vertical prototypes**

* Show small range of features (e.g. scenario, user story)
* Full implementation of these features
* Vertical integration
* Used in agile processes



*Integration approaches used during prototyping:*



*Methods to reach good usability:*

* **Usability testing**: Watching a user interact with the user interface of the system
  + Usability testing uses scenario-based design
  + Involves the creation of a test scenario
  + The user performs a list of tasks while the observer watches and takes notes, and compares the observed with the specified/expected behaviour
* **Heuristic evaluation**: A usability engineering method to find usability problems in a user interface design
* The difference between usability testing and heuristic evaluation is similar to the difference between walkthrough and inspection (review techniques)

*Nielsen’s 10 heuristics:*

* **Meet expectations**

1. Match the real world
2. Consistency & standards
3. Help & documentation

* **The user is the boss**

1. User control & freedom
2. Visibility of system status
3. Flexibility & efficiency

* **Handle errors**

1. Error prevention
2. Recognition, not recall
3. Error reporting, diagnosis, and recovery

* **Keep it simple**

1. Aesthetic & minimalist design

*Heuristic evaluation:*

* An application of Nielsen’s 10 heuristics
* An inspection method
* Performed by usability experts
* Basic steps

1. An evaluator inspects the user interface thoroughly
2. Compares the user interface against the 10 heuristics
3. Provides a list of usability problems

*Good heuristic evaluation:*

* Justify every problem with a heuristic
  + You cannot just say “I don’t like the colors”
  + Better: “Too many choices on the home page violates Nr. 10 Aesthetic & Minim.”
* List every problem
  + If an interface element has multiple problems, list them all
* Go through the interface at least twice
* Don’t limit yourself to the 10 heuristics
* Nielsen’s heuristics are a good start to compare against

*Evaluating prototypes:*

* Heuristic evaluation can already be used for the evaluation of prototypes
  + It should start early in the development process
  + Do not wait for the final product
* Heuristic evaluation works on
  + Sketches
  + Paper prototypes
  + Early software prototypes
* “Missing element” problems are harder to find on sketches
  + Because you’re not actually using the interface, you aren’t blocked by feature’s absence
  + Look harder for them

*Review: Revolutionary vs. evolutionary prototyping:*

* **Revolutionary prototyping**: get experience with a throwaway prototype
  + Advantage: can be developed in a short amount of time.
  + Disadvantage: features in the prototype are more expensive to implement
* **Evolutionary prototyping**: use the prototype as basis for the implementation of the final system
  + Advantage: shorter time to market
  + Disadvantage: can be used only if the target system can be constructed in a prototype

*Risks of prototyping:*

* Developers may become attached to the prototype
  + Excessive development time of the prototype
* Customers might not appreciate how much work must be done to turn a prototype into a fully functional system
* Users may confuse prototype and finished system
  + The prototype may actually perform the functions it implements better than a fully functional system (e.g. due to reduced functionality, easier access)
* Negative feelings toward the software if the prototype has problems
* Cost of implementing the prototype

*Managing expectations:*

* “If you show a nonprogrammer a screen which has a user interface that is 90% worse, they will think that the program is 90% worse.”
* “If you show a nonprogrammer a screen which has a user interface which is 100% beautiful, they will think the program is almost done.”

*Storyboarding: modeling the user interface as finite automaton:*

* Makes it possible to navigate through the user interface design with the user
* Often the model can be use to generate code
* Allows to combine evolutionary prototyping and low-fidelity prototyping
* Example: storyboarding in Xcode

*Low vs. high fidelity prototypes:*

|  |  |  |
| --- | --- | --- |
|  | **Advantages** | **Disadvantages** |
| **Low Fidelity** | - Easy to produce  - More feedback  - No design decisions | - Mostly not reused  - Important details are ignored |
| **High Fidelity** | - More realistic  - More detailed problems can be identified  - More impressive | - May cause much effort  - Less feedback  - High expectations (especially with interactive Prototypes) |

*Summary:*

* Usability is important for the success of your software project
* Prototyping allows you to quickly evaluate the user interface design
  + The earlier you know that something does not work as intended, the faster you can improve
* Heuristics help to evaluate / test the usability of a system
  + Example: Nielsen’s 10 heuristics
* Benefits of prototyping
  + Prototypes allow to get early feedback —> save time while developing
  + Prototypes are easier to understand than text or diagrams —> easier communication
  + Prototyping improves the usability