# User Experience for IoT

Human-Computer Interaction on the web
Prof. Emanuele Panizzi


DIPARTIMENTO
DI INFORMATICA
SAPIENZA
UNIVERSITÀ DI ROMA

# HCI for IoT

- HCI started on desktop
- several devices
- one service across multiple devices; cross-platform design
- many without screens (e.g. door lock)
- intermittently connected
- still technically driven field

# <UX for IoT> vs. <UX for Digital Services>

# Distributed Functionality

- Multiple devices with different capabilities
- screen / LED / sound
- no I/O: interaction via web or smartphone
- user should feel as using a coherent service
- **interusability** and not usability of single devices
  - i.e. distributed experience across multiple devices

# Locus in the service

- novel device, but…
- most of the information processing or data storage depends on the internet service
- service as critical as the device (if not more so)
- e.g. London Oyster travel card

# IoT is all about data

- embedded devices capture data from the real world
- we can use data to deliver better services
- either via traditional UIs
- or via embedded networked devices (eg. electrical activity measuring and heating system)

# Latency

- We accept failures on the internet (e.g. slow downloads)
- We expect real world objects to respond immediately and reliably
- Objects that require internet connection may have delays or fail, with unexpected consequences

# IoT asynchronous

- we assume constant connectivity on desktop, tablets, …
- many IoT devices runs on batteries
- they connect intermittently to save power
- part of a system may be out of sync wrt another part
- e.g. you set new temperature on your phone but the thermostat takes 2 min to update

# Code can run in different places

- system model: configuration of devices and code
- where can code run?:
    - one or more embedded devices
    - a cloud service
    - a gateway device
    - one or more control apps running on a traditional device

# Code can run in different places / 2

- failure in a part can make some functionality unavailable
- user needs to understand the system model
- e.g. rule to light bulbs at dusk
- designers should clarify system model while ensuring operation to keep users in control

# Programming-like activities

direct manipulation on traditional UI

vs

abstraction needed to control things that happen in the future or remotely

- abstract behaviour into a set of rules: programming
- examples: unlocking a door remotely; then?

# Complex services

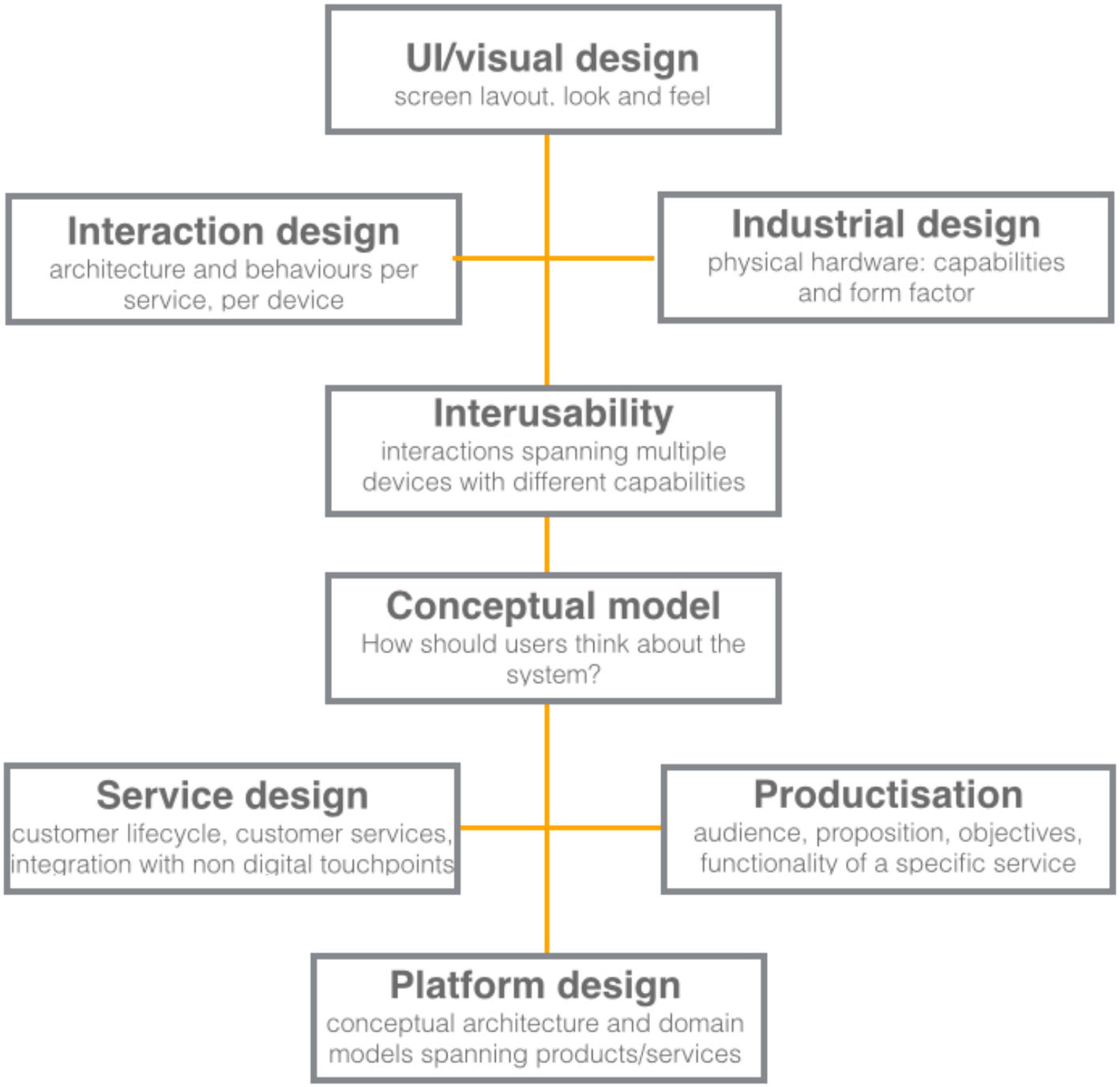
- many users
- different privileges
- multiple UIs
- many devices
- many rules
- understanding becomes challenging and time consuming

# Technical standards and interoperability

- different standards, manufacturer ecosystems
- connection on internet services rather than on devices
- consumer to look for compatibility among devices
- lowest common denominator
- user would like to trust that things work together with minimal hassle

# IoT design model

# The ‘design stack’ for IoT



# Layers in the design stack

- aspects of the user experience to be considered
- require integrated thinking (think about them together)

- **UI/Visual Design**: layout and aesthetics. Not only visual.
- **Interaction Design**: behaviours. Sequence of actions between user and device.
- **Interusability**: considerations that span multiple devices. Coherent service. Cross device user flows. Design multiple UIs in parallel.

# Layers in the design stack

- **Industrial design**: form, materials and capabilities of physical hardware. Technical constraints.

- **Service design**: addresses a holistic view of the user experience, including UX of software updates and new functionalities, customer support, in-store experience, etc.

- **Conceptual model**: enable users to figure out how to interact with service

- **Productisation**: define a compelling product proposition. Product does something of value for users.

- **Platform design**: design/use software framework that may help developers and users to discover new devices and applications, add them to the system, manage users, and manage data