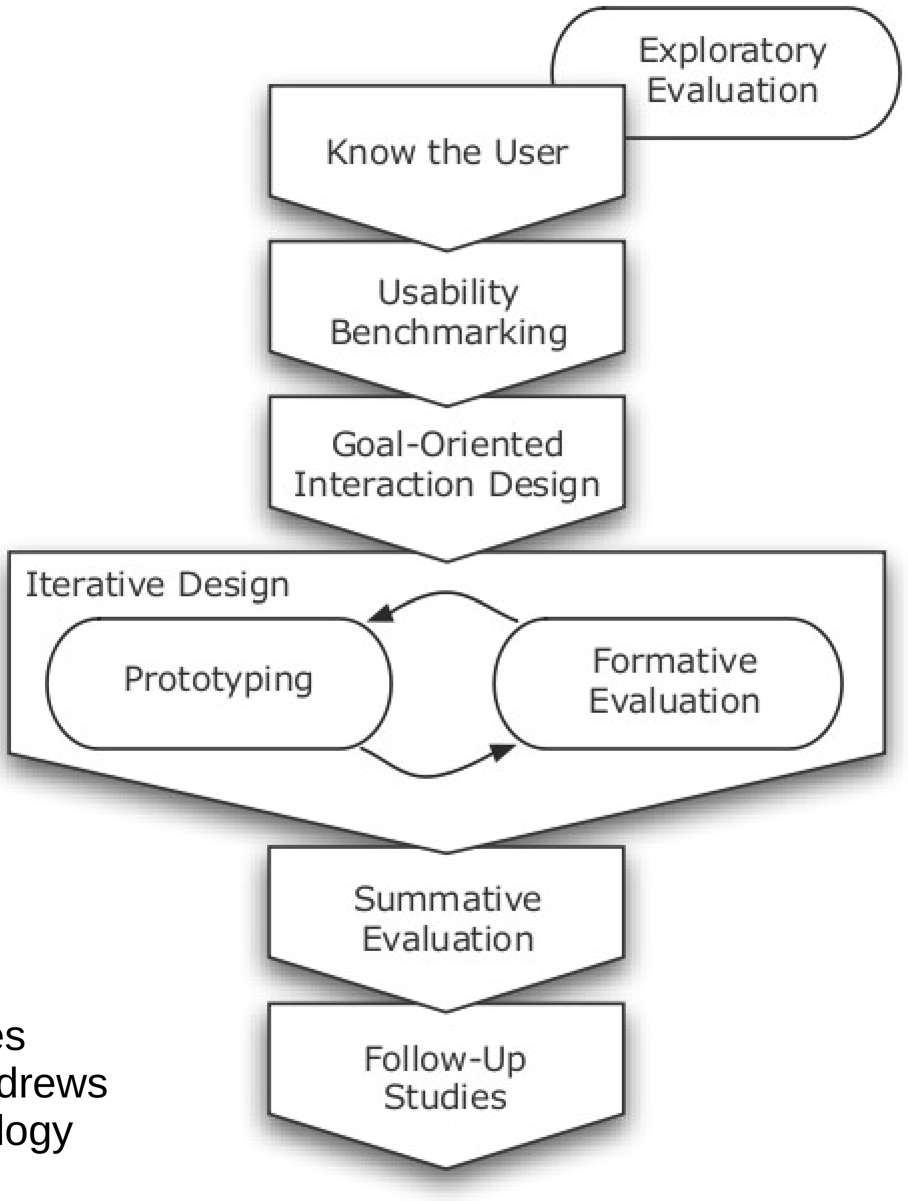
Usability life cycle
Part 1
Knowing the user
Usability Benchmarking
Goal-oriented design

Usability life cycle

- Usability in **intuitive** terms:
  - Functionality
  - Ease to learn, understand and remember
  - Interaction efficiency
  - Subjective satisfaction
- ¿How the **usability life cycle** is carried out?
  - Using methodologies to specify and validate them
  - Adapting them to users' needs
  - Reducing unsatisfaction and rework risks (fail fast)

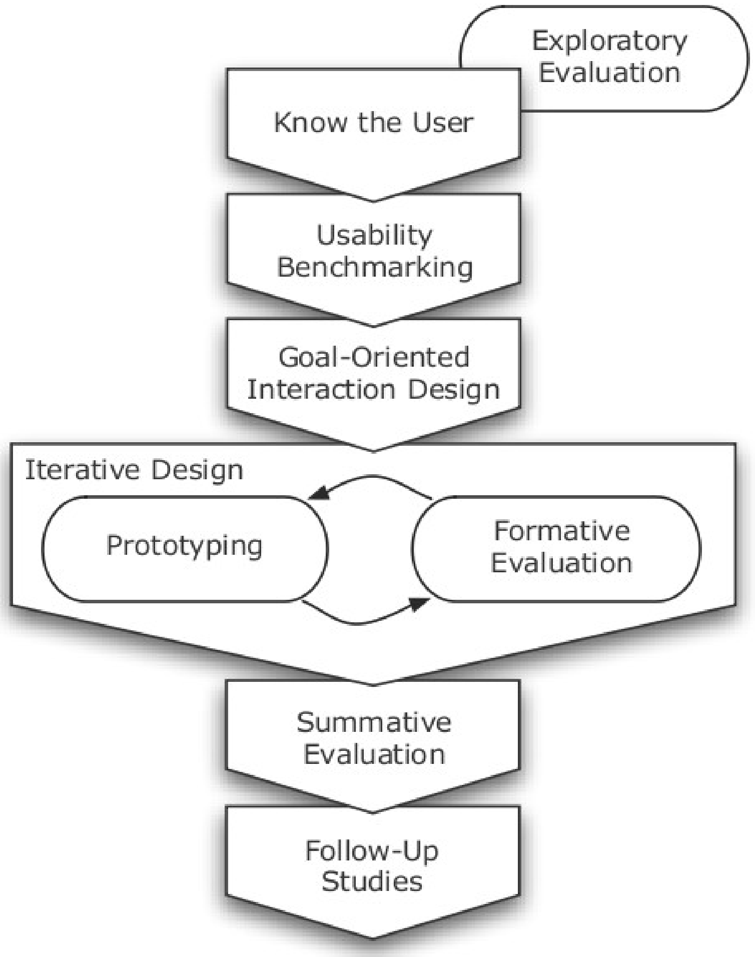
Interface creation process



Source: HCI Lecture Notes
Ao.Univ.-Prof.Dr. Keith Andrews
Graz University of Technology

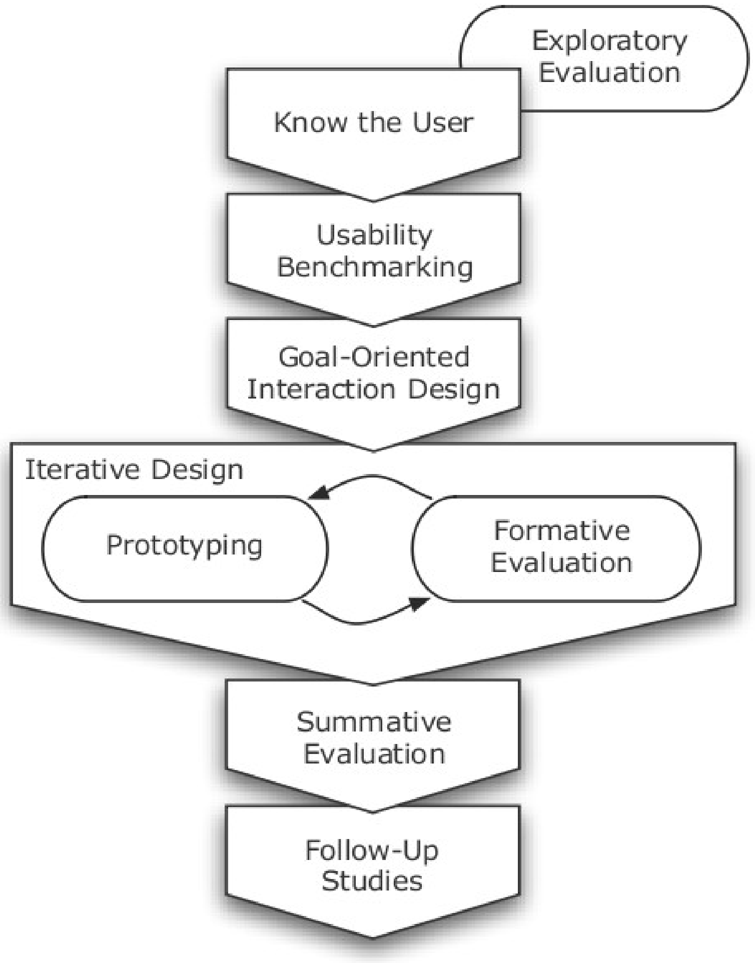
Interface creation process

- Important!
  - User interface isn't just the '**cover**' of an already programmed piece of software
  - Usability life cycle spans from the **beginning** until the **end** of the project
  - It goes hand-in-hand with the development of the other components
  - Ignoring usability leads to **incompatibility** issues between the UI and the rest of the software



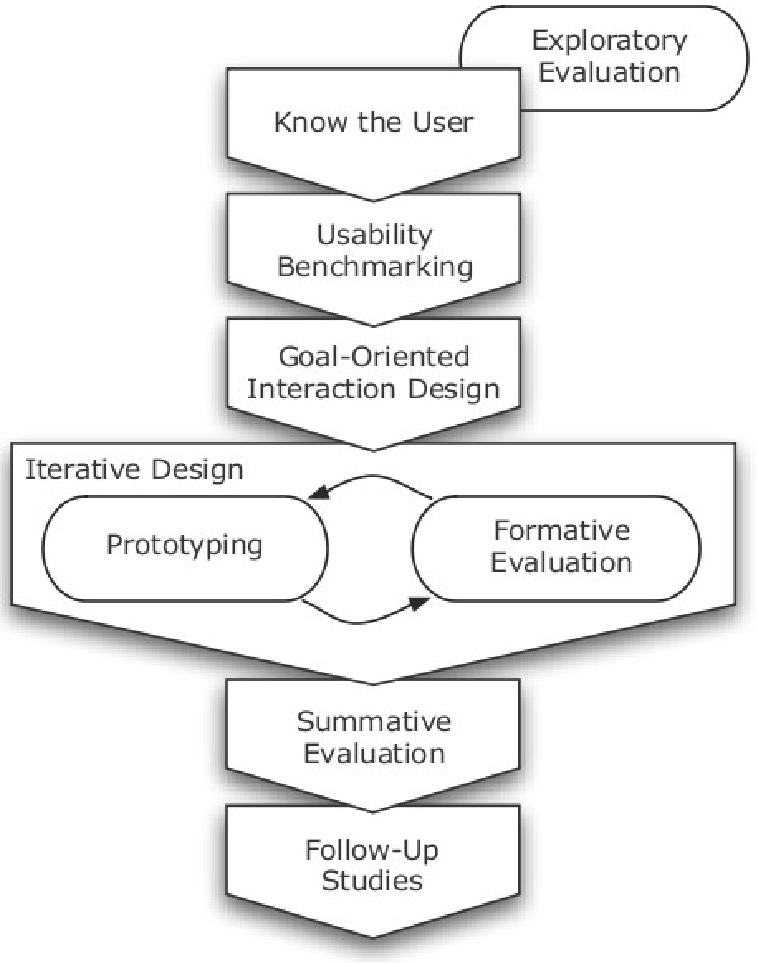
Interface creation process

- **Know the user**
  - Identify stakeholders
  - Know local culture
  - Identify needs
- Evaluations
  - **Exploratory**: Before
    - How the system will be used?
    - What tools are currently been used for this?
    - How good is the available software?
  - **Formative**: During
    - How to improve it?
    - Does it work as expected?
  - **Summative**: Towards or after the end
    - How good it turned out?
    - How was the reception of the users?

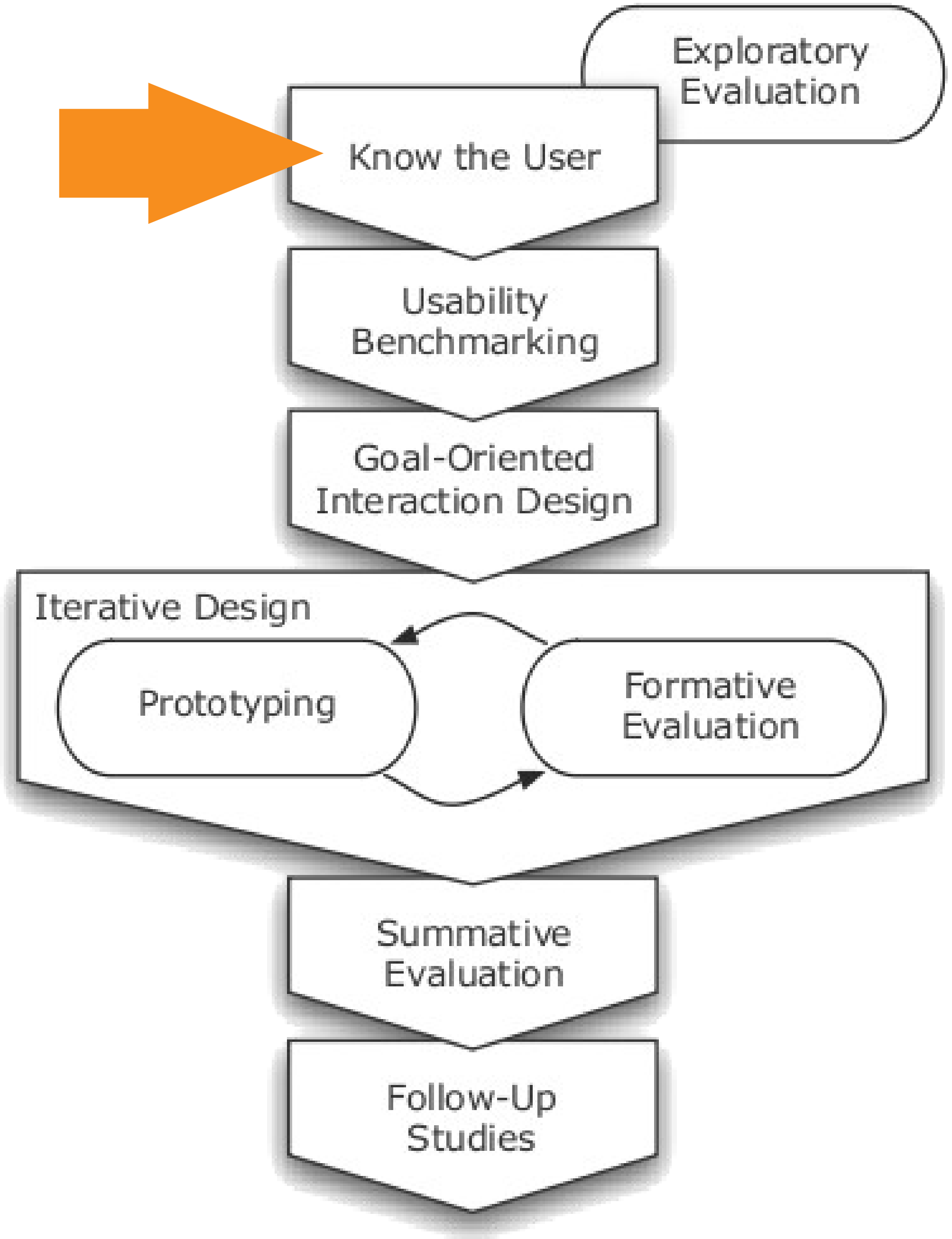


Interface creation process

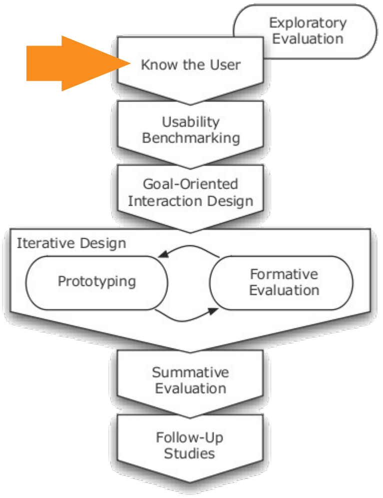
- **Usability Benchmarking**
  - Comparison with other products
  - Set metrics, identify strengths and weaknesses
- **Goal-oriented design**
  - Create generic user types (personas)
  - Set up the goals of the system
- **Prototypes**
  - Different types of prototypes
  - System mock-ups
  - Identify and solve problems early
  - Analyze user acceptance
- **Follow-up studies**
  - Observe the system in the “real world”
  - Market response, usability problems



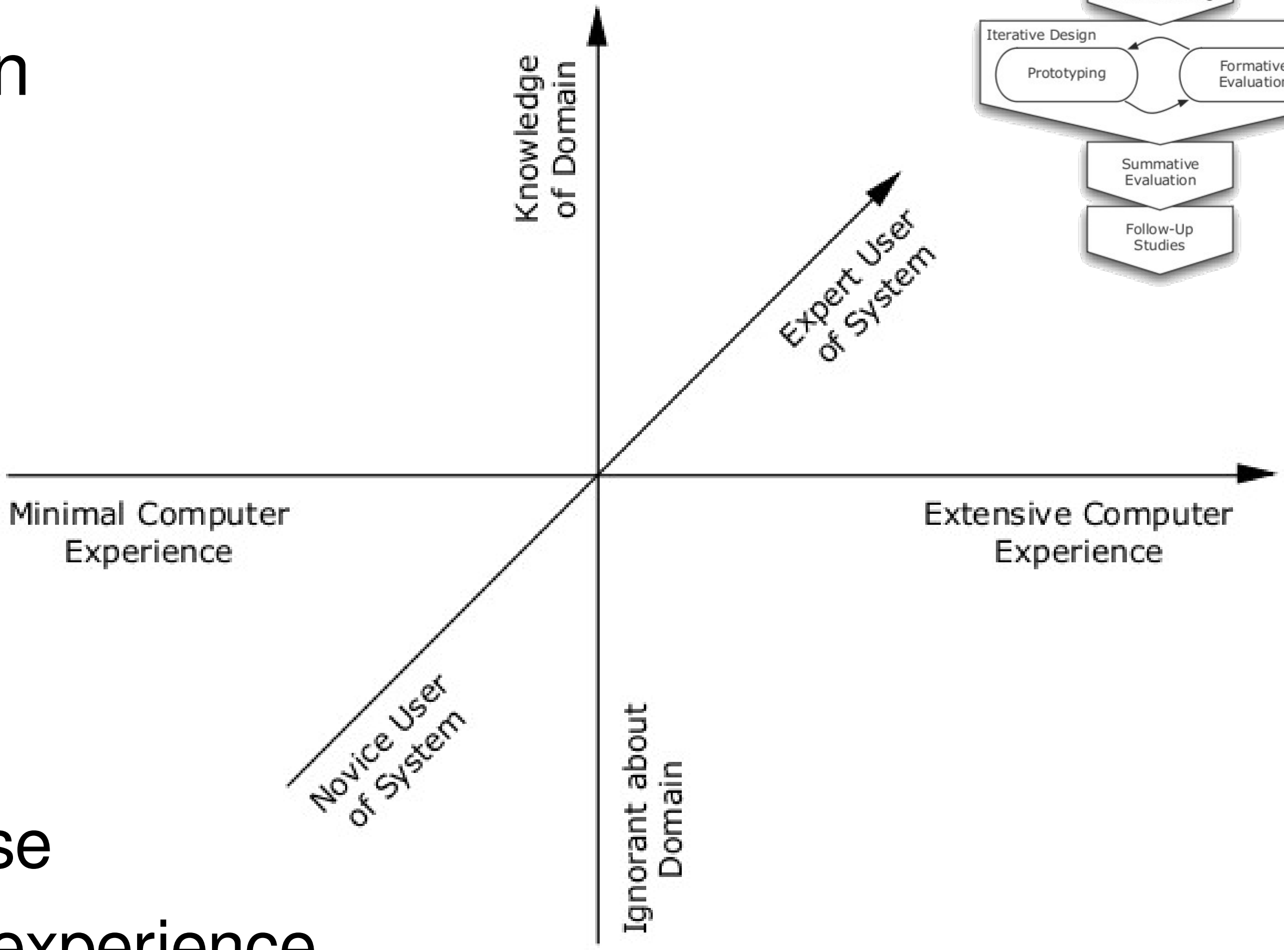
KNOW THE USER



Know the user



- User classification

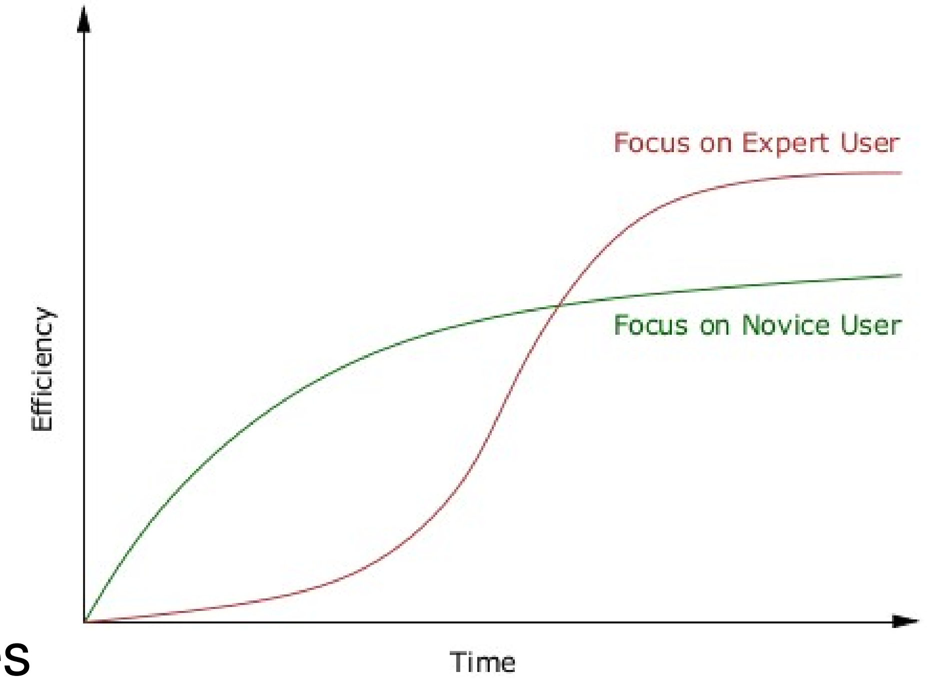


- 3 main axes:
  - System expertise
  - Computational experience
  - Domain knowledge

Learning curve

- Trade-off between functionality and complexity
- Systems may be oriented to:
  - Novice users
  - Expert users
  - Intermediate users

    (most of them)



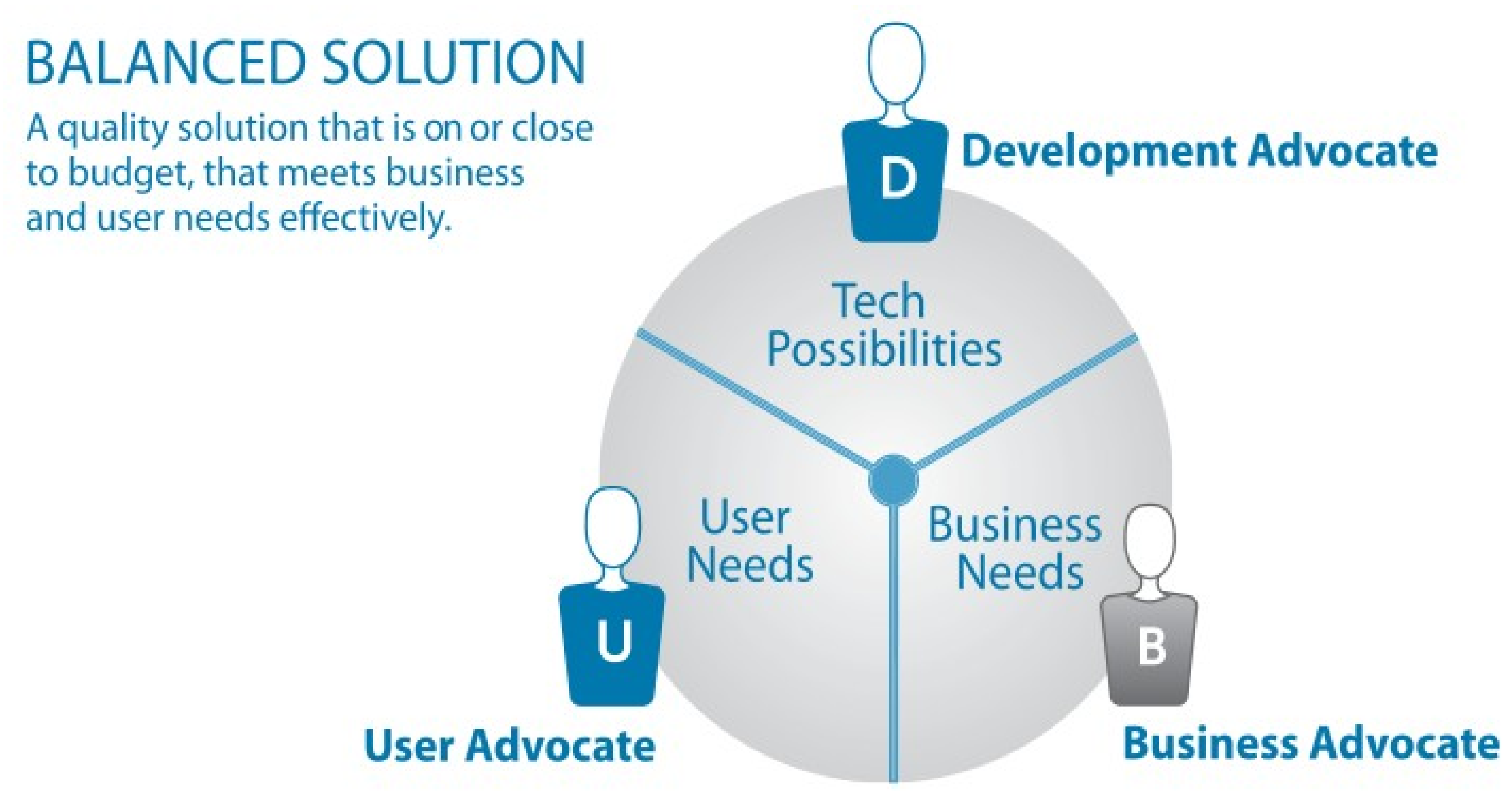
- Customization:
  - Both “simple” and “expert” modes

- In software design (an specially UI design) not only the users have influence...

Stakeholders

- Different people are involved in system and interface development:
  - **Users**: those who will actually use the product
  - **Clients**: those who will pay for the product
  - **Project managers**: those who carry out the project
  - **Domain experts** business experts
  - **Technology experts**: programmers, consultants, etc.

Trade-off between interests



Source: "A Project Guide to UX Design", Unger, Chandler, 2009, Peachpit Press

Example: Hospital software

- **Purpose**: Log the patients reception at ICU (Intensive Care Unit)
  - **Users**: nurses
  - **Clients**: Hospital managers
  - **Project management**: software enterprise
  - **Domain experts**: Doctor in charge of ICU, experts from the Ministry of Health, etc.
  - **Technology experts**: Java experts, network experts, usability experts, etc.
- How to consider their advice in UI design?
- What questions ask to each one?

All stakeholders may help

- Ask **clients**:
  - General needs and goals
  - Problems of current system
  - Project budget and time constraints
  - Buy decision process (who decides)
  - Who will use the system
  - Environment constraints (corporate rules, liability limits, etc.)
- Ask **domain experts**:
  - Frameworks (policies, regulations, etc.)
  - Best practices
- Ask **technology experts**:
  - Technological possibilities
  - Associated risks

Focus on user

- Once the framework defined, the focus must be placed on the user
  - Common mistake: ask the boss (who won't use the system) the required features
  - Warning! Few times the users know exactly what they want
  - It's better to **observe** them than ask them what/how they do their work
  - Usually there's not a single user, but several with different interests

- Ethnographic Interview
  - ***Ethnography***: science that studies social phenomena on human groups
  - It aims to understand the culture where the software will be used
  - A representative set of future users must be identified

- It is important to consider all stakeholders (and not only users) because:
  - i. Some user requirements may be lost otherwise
  - ii. Users have not all the required information
  - iii. Someone has to pay for the system

  - A. i and ii
  - B. i and iii
  - C. ii and iii
  - D. all the above

```
/poll "stakeholders" "A" "B" "C" "D"
```

Ethnographic Interview

- At the workplace
- Just with final users (without supervisors or clients)
- Try to gather the following information:
  - Main goals of the work
  - Tasks carried out
  - Constraints and exceptions
  - Identify problems that need to be solved
    - What needs more time or money, what is critical, what has to be known, etc.
  - Vocabulary
  - Business context
  - Photographic or audio record (if needed)

  - https://www.youtube.com/watch?v=FBIdwhrnIZw

VIDEO!

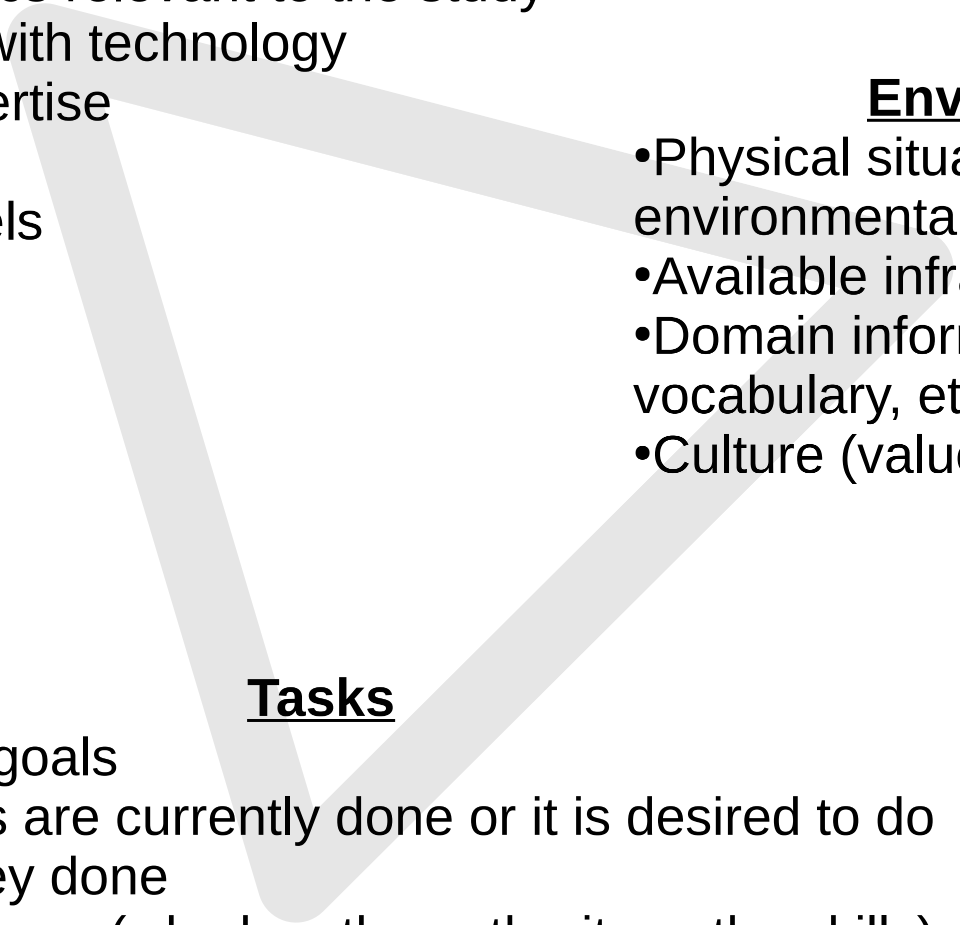
In the shoes of a user

- Recommendations:
    - Make yourself an apprentice, do what users do, in the same way
    - Rather focus in what they do that in what they ask or declare
    - Notice how they deal with their artifact errors or difficulties (e.g., post-it)
    - Make yourself *invisible* (reduce Hawthorne effect)

- Ask yourself:
    - What do **they** know?
    - What are **their** values and goals?
    - What are **their** activities?, how do **they** fit in a wider contexts?
    - What are the differences and similarities between **them**?

(Based in Scott Klemmer's HCI course, Stanford University, 2012)

Ethnographic Interview



Users

- Characteristics relevant to the study
- Experience with technology
- Domain expertise
- Motivation
- Mental models

Environment

- Physical situation (placement, environmental conditions)
- Available infrastructure
- Domain information (processes, vocabulary, etc.
- Culture (values, traditions, etc.)

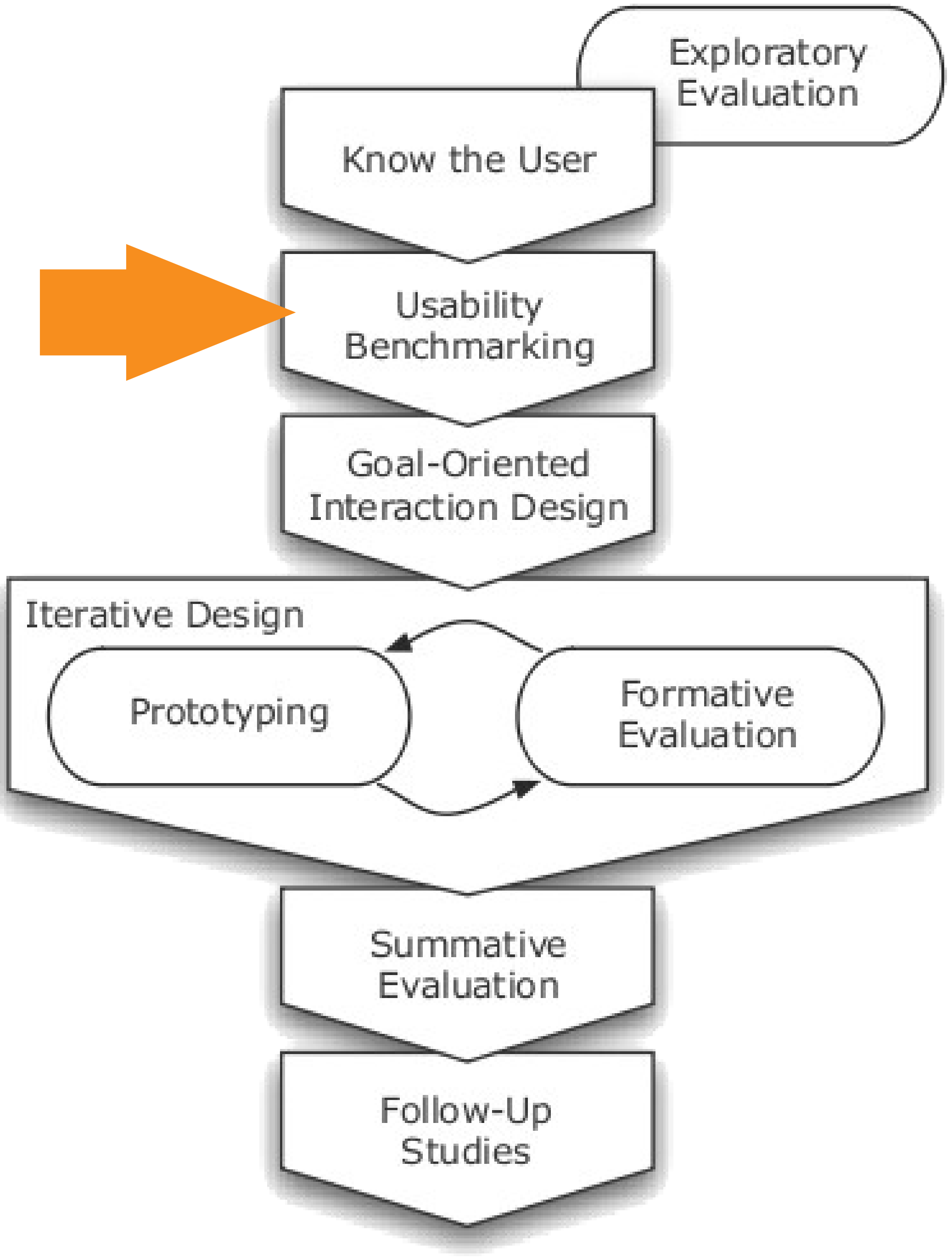
Tasks

- Underlying goals
- Which tasks are currently done or it is desired to do
- How are they done
- Who does them (who has the authority or the skills)
- Constraints (time, available information, etc.)
- How do they impact the environment

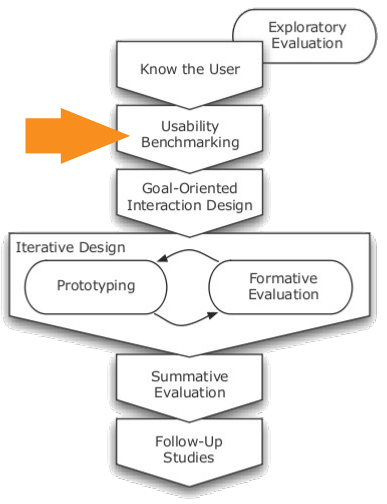
Ethnographic Interview

- Study **body language**
- **Let the user talk**, show interest
- **Open questions** to raise unexpected subjects
  - *"How do you register the patient?"*
  - *"Could you tell me more about this"*
- **Closed questions** to clarify processes
  - *"Do you authorize the hospitalization?"*
- **Summarize** what you understood to check if it's right
  - *"So -if I understood well- when a patient arrives..."*
- *Do not push to get quick answers.* ***2-3 seconds of silence*** *may awake a interesting or unexpected memory in the user*

USABILITY BENCHMARKING

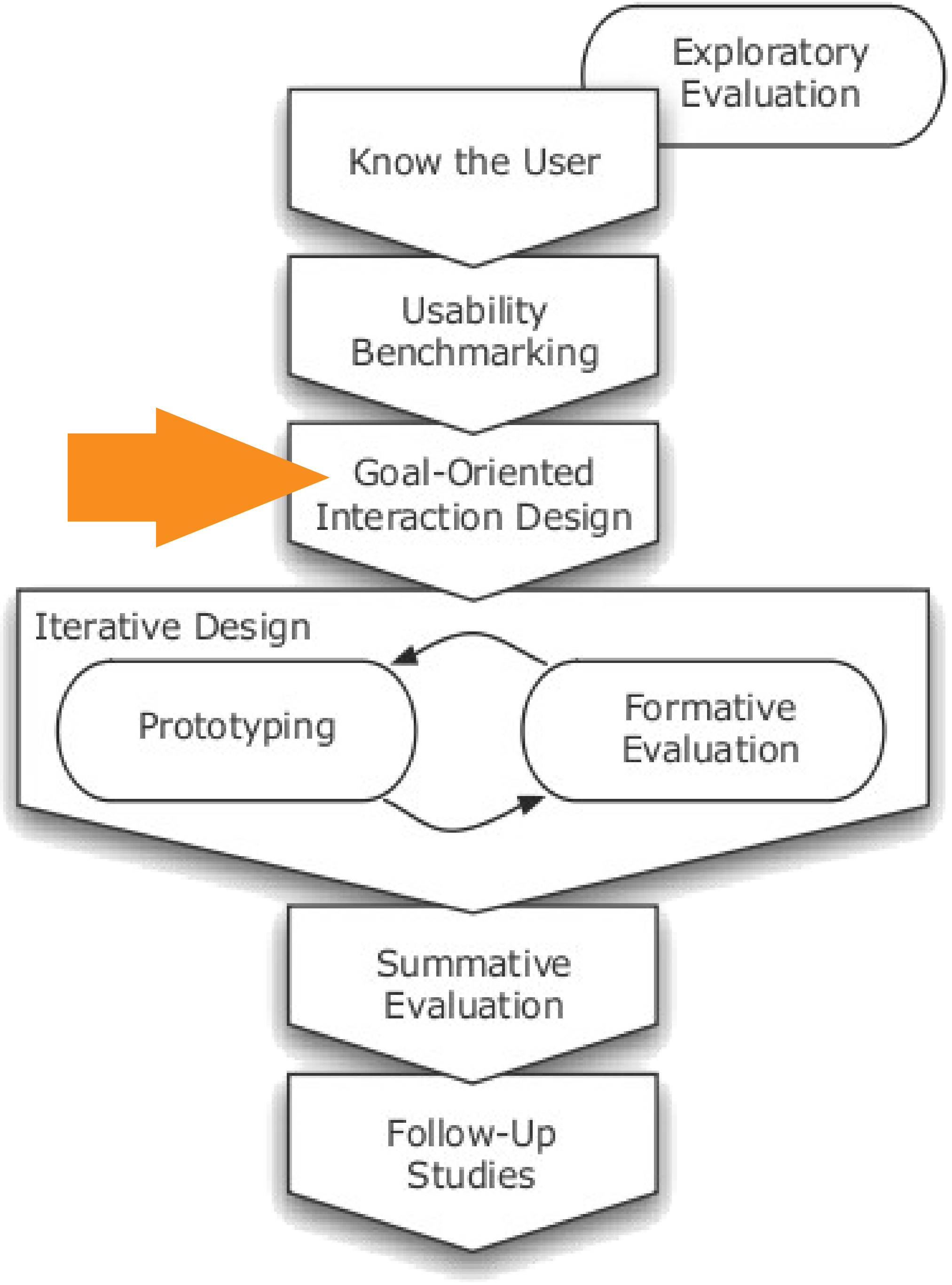


Usability benchmarking

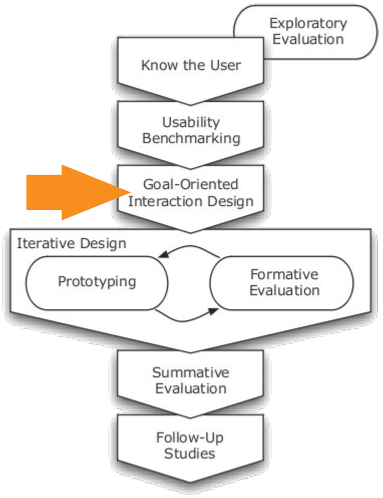


- Study the **competition**
- Detect issues to **improve**
- Set interaction goals, for example:
  - Mean time of interaction
  - Success rate in task execution
- **Financial analysis**
  - Project ROI
  - What improvement in the interface is worth to implement?
  - Focus effort on important improvements

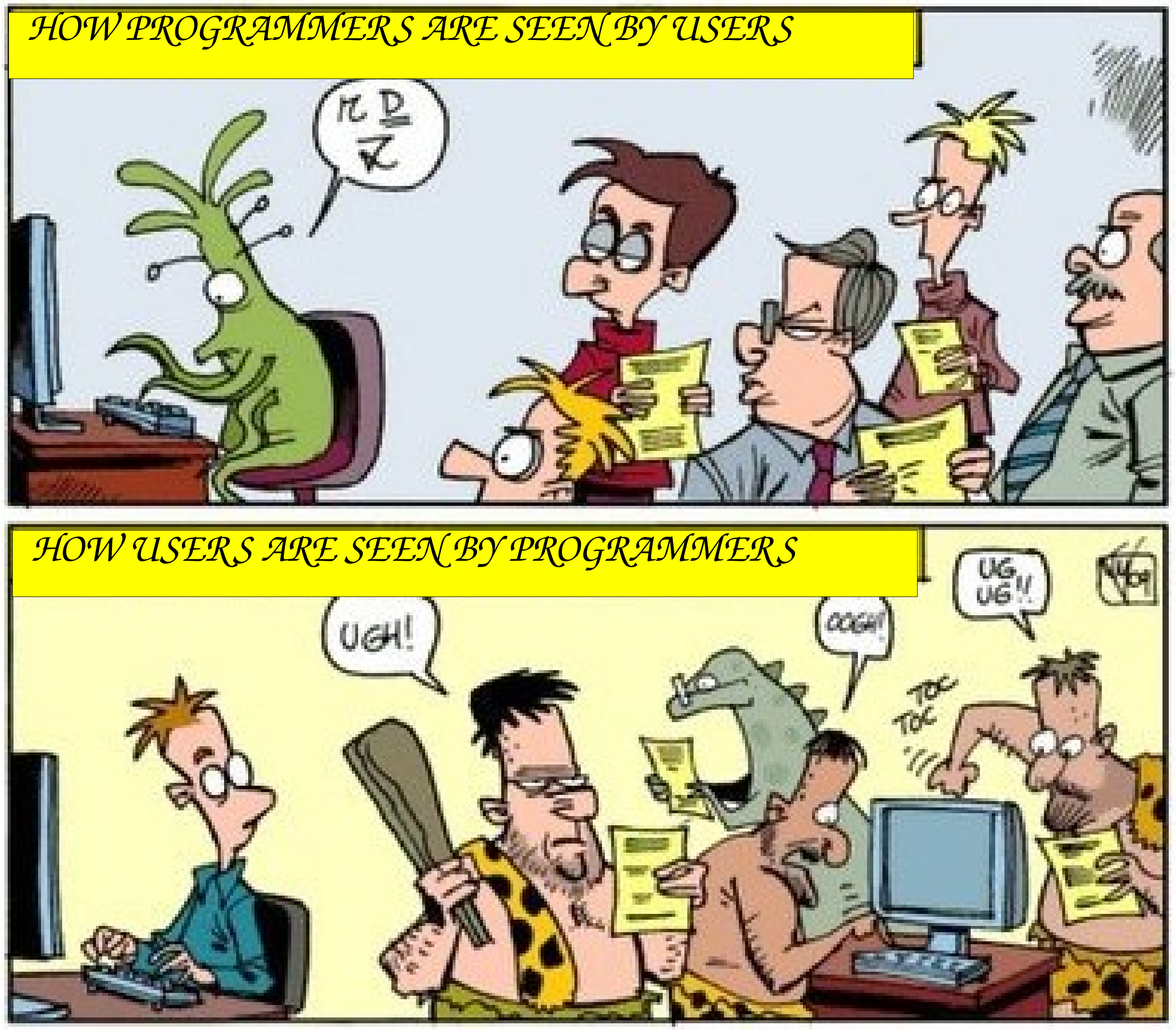
GOAL-ORIENTED DESIGN



Goal-oriented design



- Programmers vs. users


HOW PROGRAMMERS ARE SEEN BY USERS
HOW USERS ARE SEEN BY PROGRAMMERS
UGH!
OOGA!
UG UG!!
TOC TOC

Goal-oriented design

- Interface design is an iterative process
- It is crucial to focus in users' criteria
- ***Programmer != User***

Programmer	**User**
Wants **control**, accepts complexity	Wants **simplicity**, accept lack of control
Seeks to **understand**, accepts failure	Seek **success**, accepts don't understand
Concerned about **completeness** (all the cases), accept preparation	Concerned about **the most common task**, accepts incompleteness

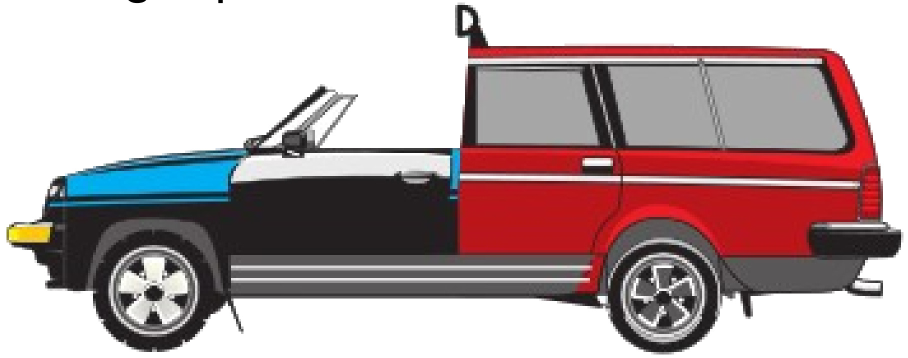
- Instead of design the UI right away, the goals and the tasks needed to accomplish them must be defined

The elastic user

- Many times the user profile is not properly defined
- This allow programmers to “adjust” user needs to fit her interest or possibilities
- In order to avoid this, users profiles are defined (called “**personas**”)
- Each persona has her own objectives
- Then pretend that they are real users.

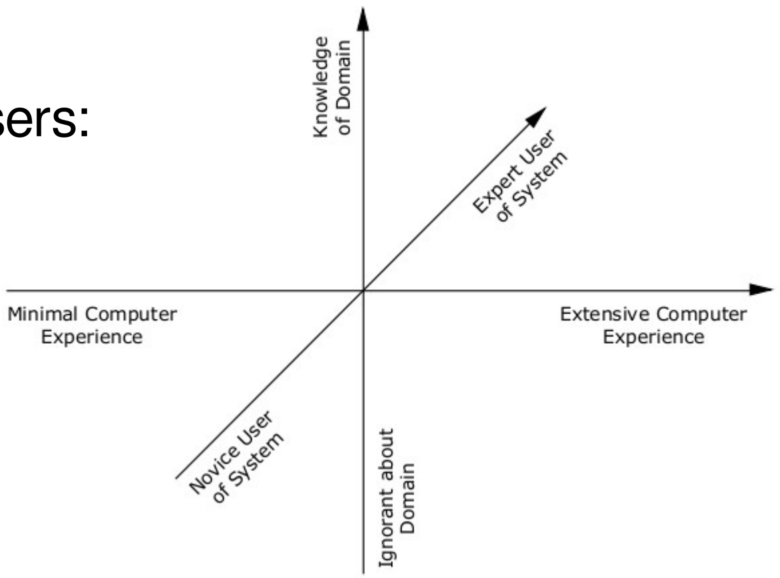
Designing personas

- Do not create an “average” persona



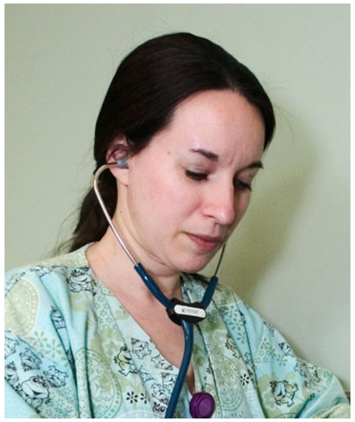
- A persona isn't real, but is specific and detailed (provide age, profession, personal interests, a picture, personality traits, etc.)

- Use extreme cases from the existing users:
  - “The least skilled”
  - “The eldest”
  - “The only woman”
  - “The foreigner”, etc.



Designing personas

- Claudia Fernández
  - 37 years old, ICU nurse, shift supervisor.
  - Married, 2 children. At home she uses the computer only to check her email. She dedicates her spare time to her family.



- Patricia di Francesco
  - 22 years old, Medicine 4th year student, intern.
  - Single, she came from Argentina, she often uses computers and other gadgets. She has just a few years of medical practice.

VIDEO!

https://www.youtube.com/watch?v=W1kw5xK1C30

Personas and use scenarios

- Each persona has her own goals
  - “To log an urgency”
  - “To obtain the list of available beds”
  - Similar to a UML *use case*, but including characteristics of the specific persona
- To model these goals (contextualized in a situation), ***use scenarios*** are defined
  - There are a precise description of a persona using an interface
- Example:
  - Use scenario: “*Claudia (shift supervisor) has worked for 10 hours when a pileup produces the arrival of 8 patients, 3 of them in critical state. While she enters the data of the patients, she must convoke the medical team and communicate with the pharmacy. Also, she must delegate to a colleague the attention of the relatives that arrive to the hospital or call*”

- Regarding use cases and use scenarios:
  - i. Use cases are related to actors as use scenarios to personas
  - ii. Use scenarios capture emotional details
  - iii. Use scenarios are better than use cases

  Are true:
    - A. i and ii
    - B. i and iii
    - C. ii and iii
    - D. all the above

```
/poll "scenarios" "A" "B" "C" "D"
```

Generating solutions

- Once personas and scenarios were defined, a UI that satisfy them must be designed
- How to produce an appropriate solution?
  - **Parallel efforts** (lateral thinking)
    - Avoid to “stick” with an initial –possibly sub-optimal– solution
    - Explore “absurd” alternatives, identify out-of-the-box ideas
  - **Brainstorming**
    - Carry out outside the traditional work context (office, classroom, etc.)
    - Include different professional profiles if possible
    - Materials: paper, whiteboard, etc.
  - Brainstorming rules:
    - EVERYTHING is possible, do not constraint by technology or budget
    - It is forbidden to criticize ideas, only to offer alternatives
    - Be spontaneous (be crazy) y generous (do not hide your thoughts)

Generating solutions

- Techniques to not get stuck:
  - **Magic**: “What the system would do if it were magical?”
  - **Human**: “What the system would do if it were a human?”
  - **Look for help**: If you run out of ideas, ask someone outside the team
  - **Analogies**: Propose an analogy of the system in a different context. How do things work in that context?
  - **Incomplete Information**: get a vague idea from other solutions and reinvent the rest
- Once the brainstorming is over, categorize the collected ideas, study their viability and select the most feasible ones
- GUI techniques
  - We will review a number of techniques to design GUIs in the following classes

Summary

- **Life cycle**: frames the process of interface creation in practice
- Compared to (mainstream) Software Engineering:
  - More focused in the UI
  - More engaged with the user and her environment

- First, it is necessary to **know the user** and her context
  - Classify users according to their expertise
  - Different learning curves
  - Stakeholders: each one has something to say
  - Ethnographic interview: characterize users, environment and tasks

Summary

- **Benchmarking**: study of the competition
  - Evaluate if it's worth to compete and at which level
  - Economic feasibility

- **Goal-oriented design**
  - The user usually does not know exactly what he wants, while the programmer suffers from “professional deformation”
  - **Personas** (fictitious –but credible– users): they represent diversity
  - **Use scenarios**: Use case + state of the persona
  - Solution generation: first, do a brainstorming to open the mind