AmI Design Process

Ambient intelligence: technology and design

Fulvio Corno

Politecnico di Torino, 2014/2015

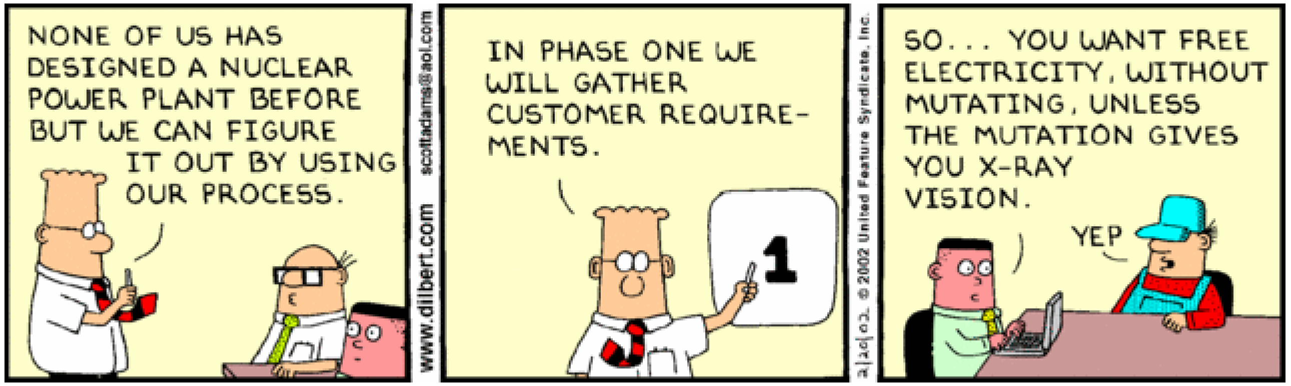
Notes on the
DESIGN PROCESS
November 2013, M.HAGAN
I've been thinking about how the DESIGN PROCESS can get to
RICH
DEEP
MEANING-FUL DESIGNS
that don't just get People to say
THAT'S COOL
But That Make the USERS feel
EMPOWERED
SATISFIED
& like The Person they WANT To BE
These Are the DESIGNS I want to Be Mak

Design process (in Engineering)

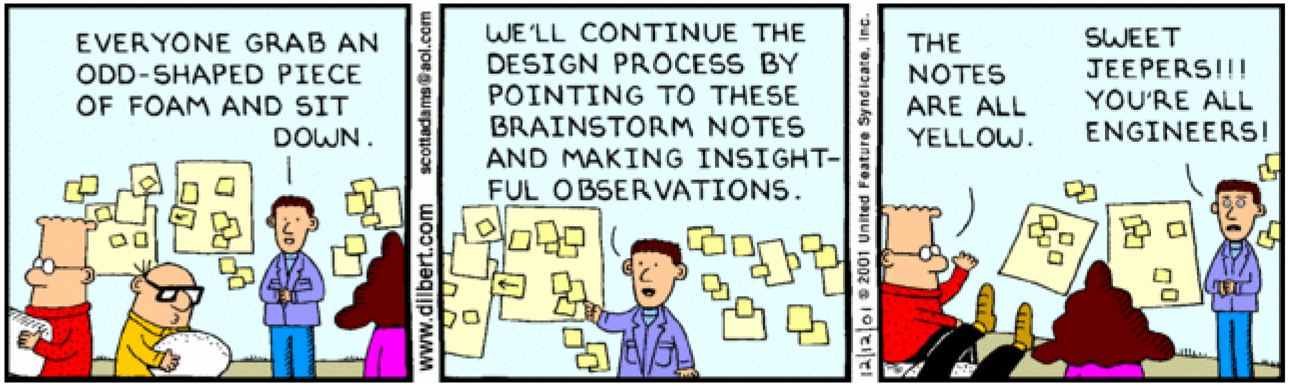
- The engineering design process is the formulation of a plan to help an engineer build a product with a specified performance goal. [Wikipedia]

- The engineering design process is the formulation of a plan to help a team of engineers build a system with specified performance and functionality goals. [improved]

Design Process


NONE OF US HAS DESIGNED A NUCLEAR POWER PLANT BEFORE BUT WE CAN FIGURE IT OUT BY USING OUR PROCESS.
IN PHASE ONE WE WILL GATHER CUSTOMER REQUIREMENTS.
1
SO... YOU WANT FREE ELECTRICITY, WITHOUT MUTATING, UNLESS THE MUTATION GIVES YOU X-RAY VISION.
YEP
www.dilbert.com scottadams@aol.com
2/20/02 © 2002 United Feature Syndicate, Inc.

http://dilbert.com/strips/comic/2002-02-20/


EVERYONE GRAB AN ODD-SHAPED PIECE OF FOAM AND SIT DOWN.
WE'LL CONTINUE THE DESIGN PROCESS BY POINTING TO THESE BRAINSTORM NOTES AND MAKING INSIGHTFUL OBSERVATIONS.
THE NOTES ARE ALL YELLOW.
SWEET JEEPERS!!! YOU'RE ALL ENGINEERS!
www.dilbert.com scottadams@aol.com
12/12/01 © 2001 United Feature Syndicate, Inc.

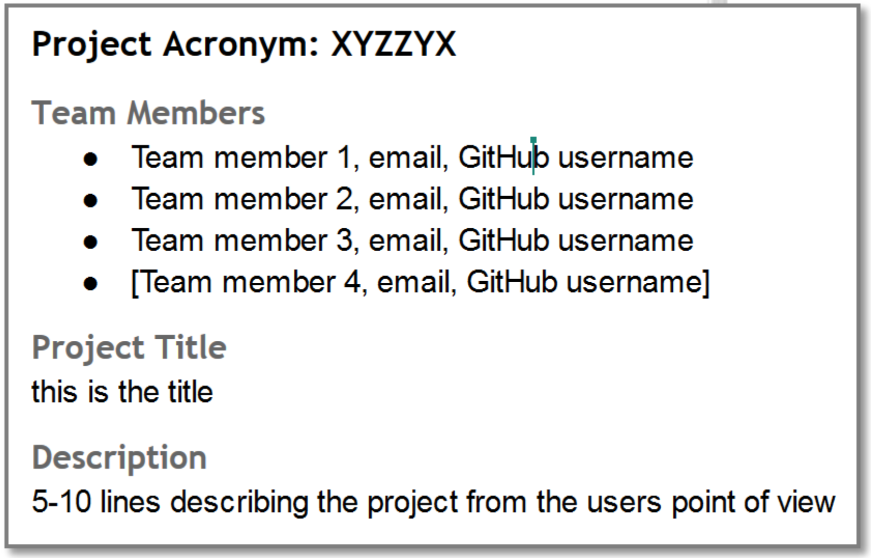
http://dilbert.com/strips/comic/2001-12-12/

Summary

- General design process
- Main steps of the process
  - Step 1: Problem Statement
  - Step 2: Requirements Elicitation
  - Step 3: Requirements Identification
  - Step 4: Architecture Definition
  - Step 5: Component Selection
  - Step 6: Design & Implementation
  - Step 7: Test and Validation
- Simplified process adopted in the AmI course

Deadline ahead

- Before 19/03
  - Group composition
  - Summary Description
- Do not wait until the last minute
  - May help forming groups
  - We'll monitor in real time
- Discussion: 19/03
- Final deadline: 20/03



Project Acronym: XYZZYX

Team Members

- Team member 1, email, GitHub username
- Team member 2, email, GitHub username
- Team member 3, email, GitHub username
- [Team member 4, email, GitHub username]

Project Title

this is the title

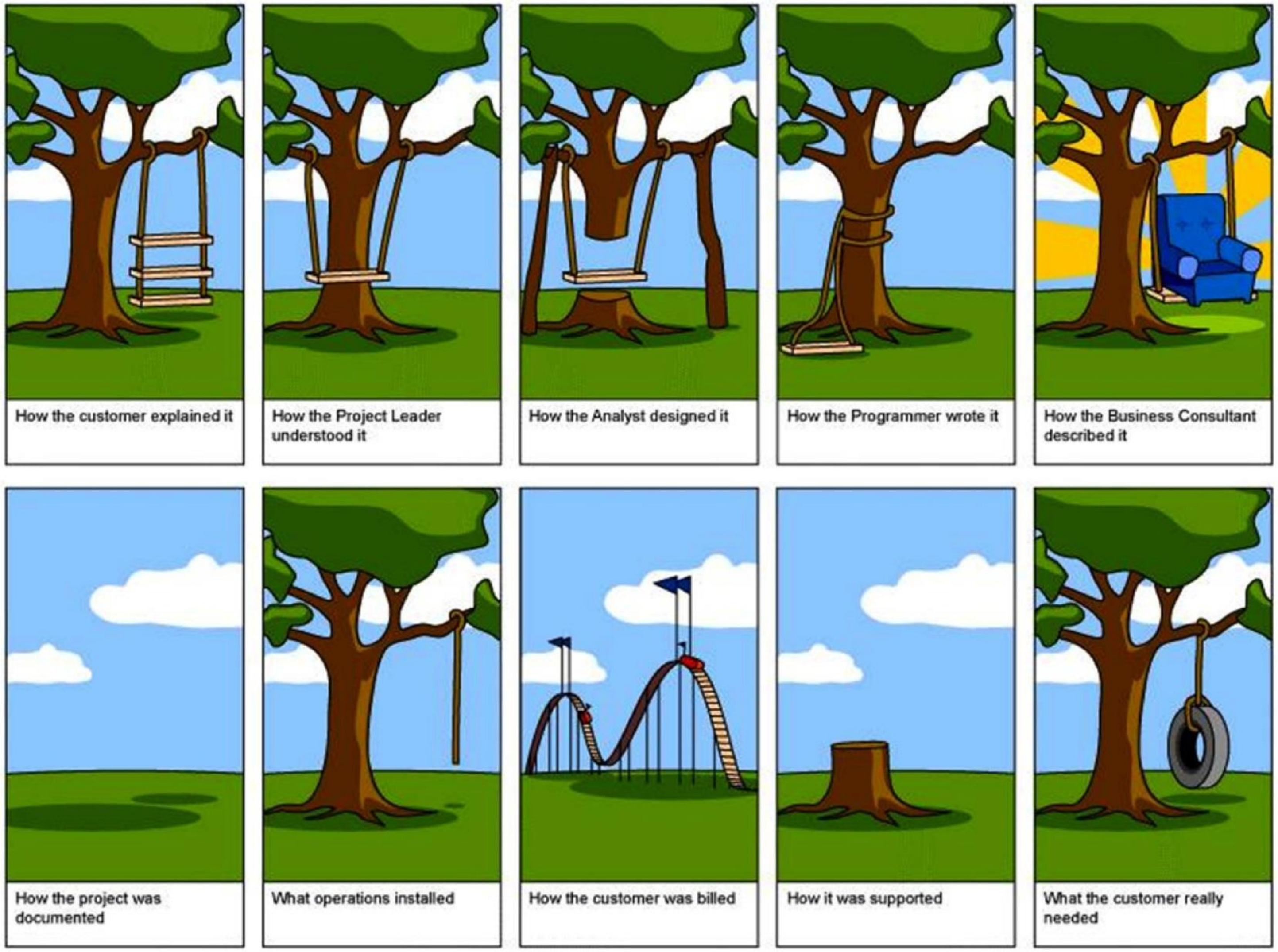
Description

5-10 lines describing the project from the users point of view

AmI Design Process

GENERAL DESIGN PROCESS

The all-too-common problem


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Goals

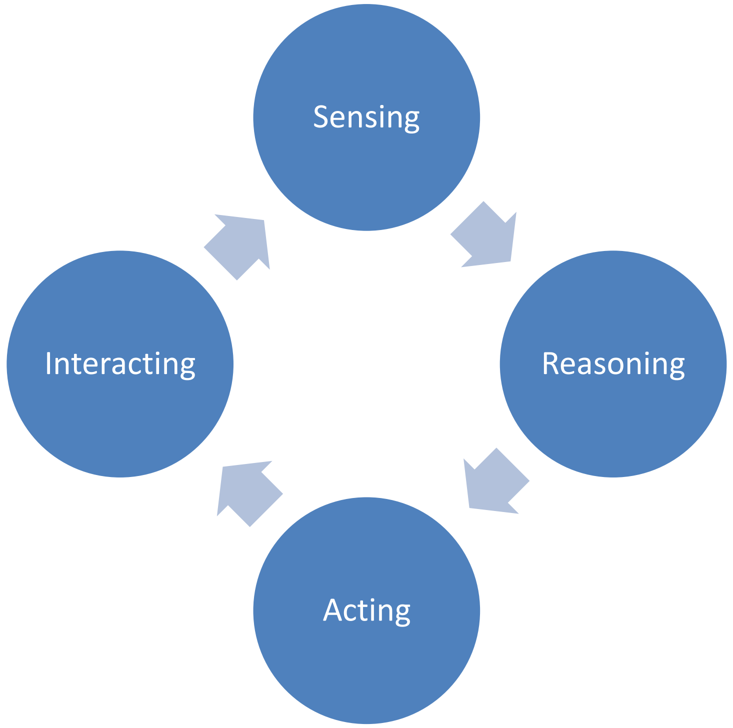
- To select **one** possible approach, among the many ones proposed, to design and realize an AmI system
- To analyze and formalize **one** possible flow of activities
- To understand the activity and the output of the main **steps**

- To define a scaled-down version compatible with the time constraints we have in the AmI course

What we want to achieve

- From initial idea…

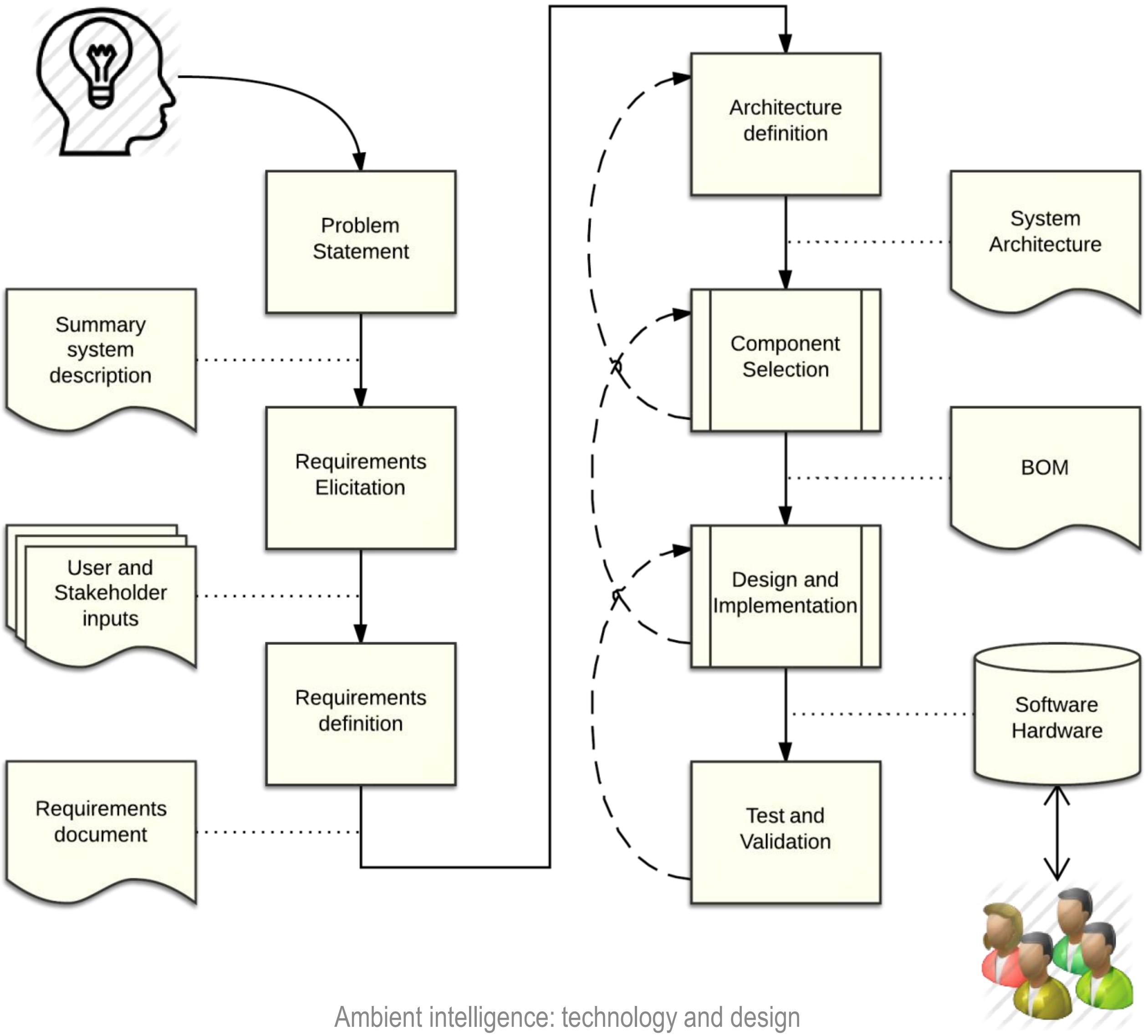
- …to working AmI system



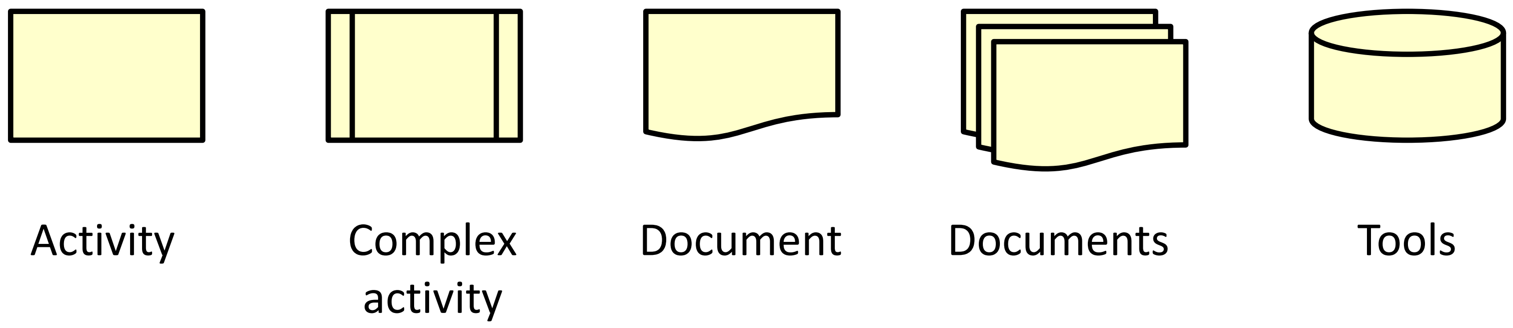
Assumptions

- The approach should be **technology-neutral**, i.e., the best fitting technologies will be selected **during** the process, and will **not** be defined **a-priori**
- When existing solutions/devices are **available and suitable** for the goal, aim at **integrating** them. When **no** suitable existing solution exists, consider developing/prototyping some **ad-hoc** device(s)

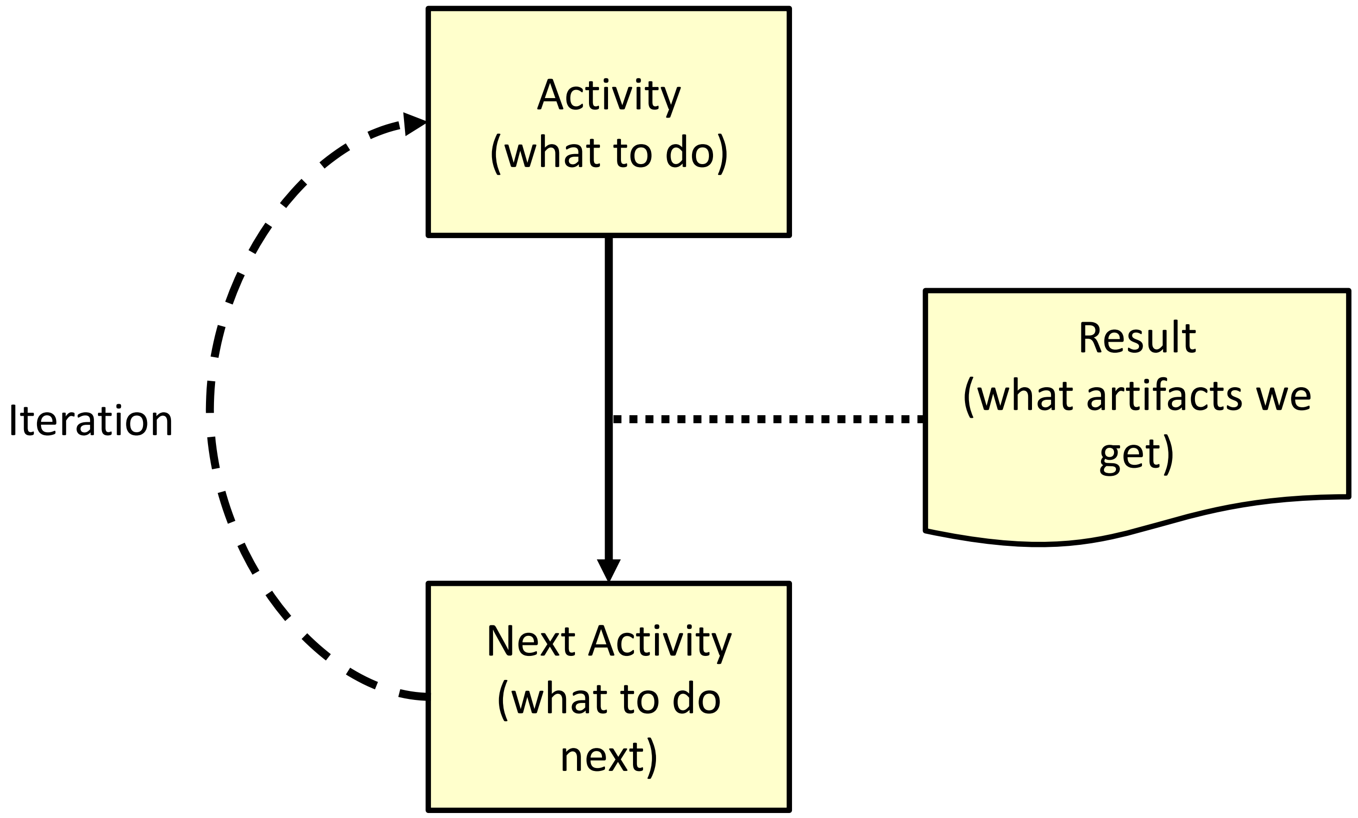
Proposed process



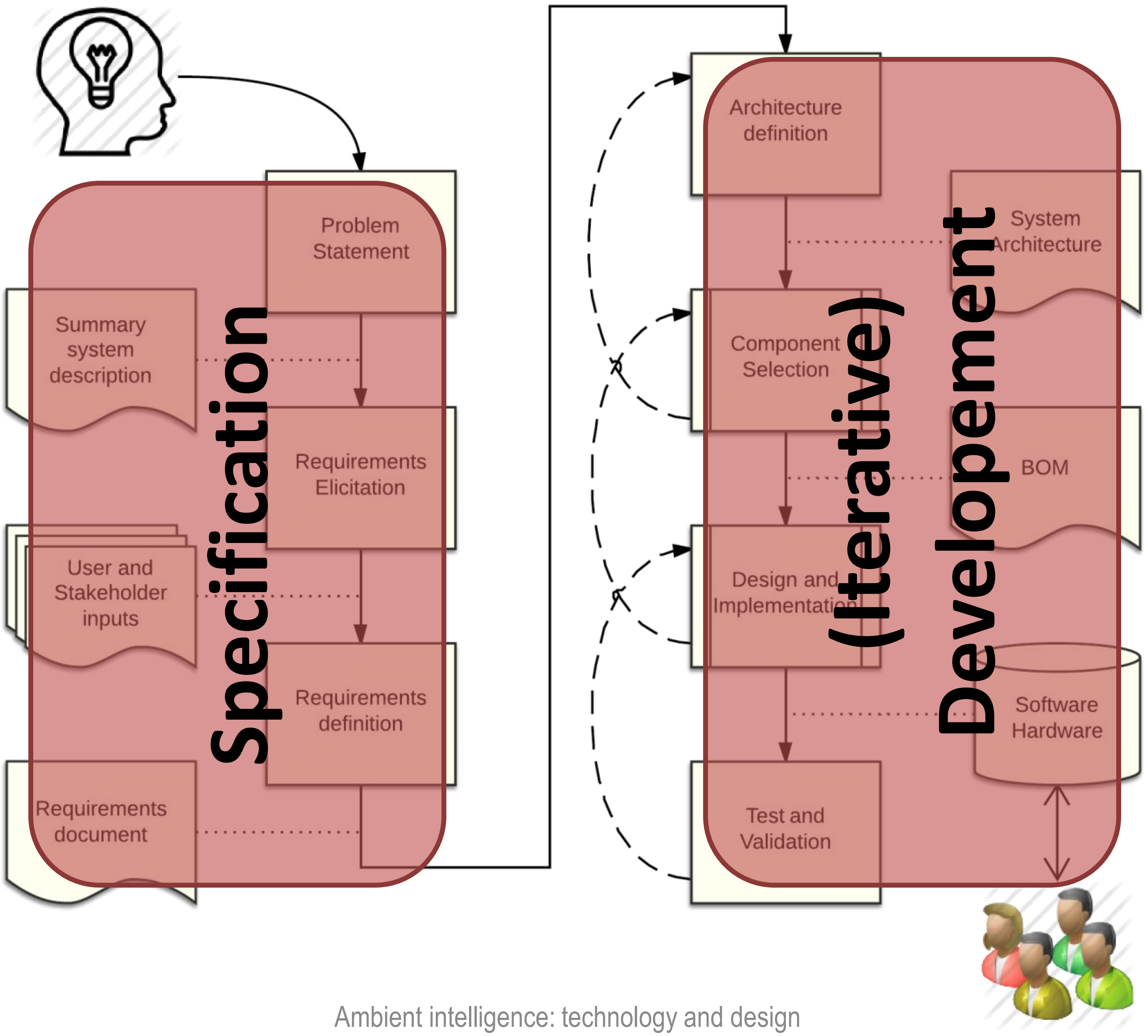
Legend



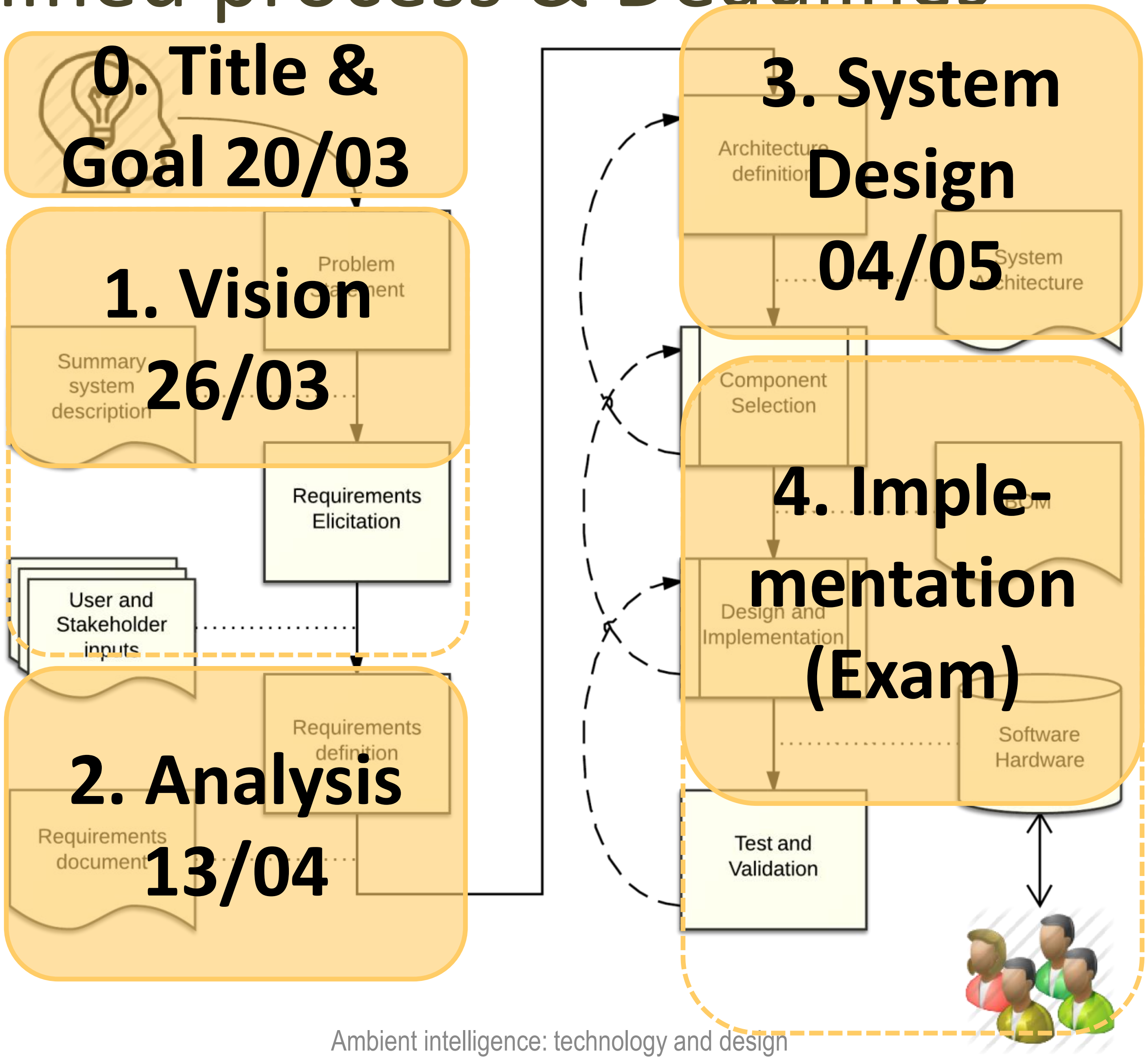
Composition of each step

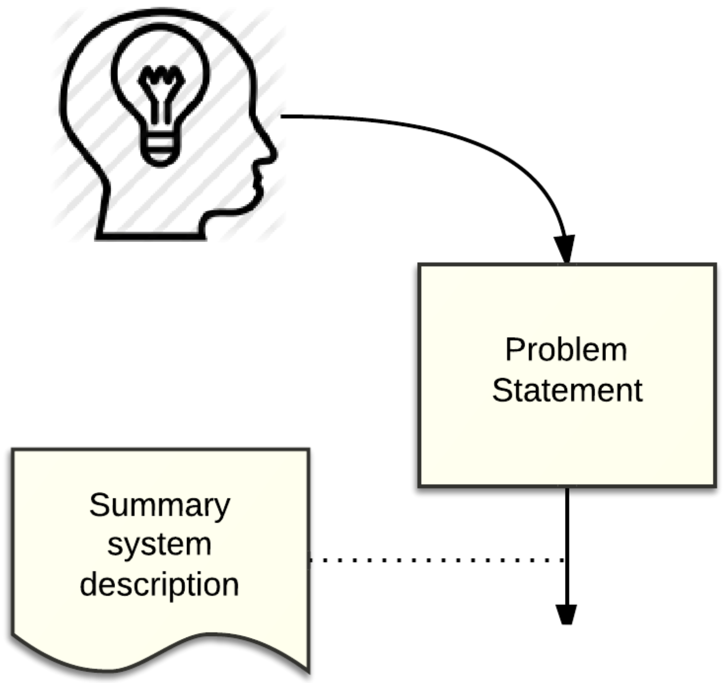


Proposed process



Simplified process & Deadlines



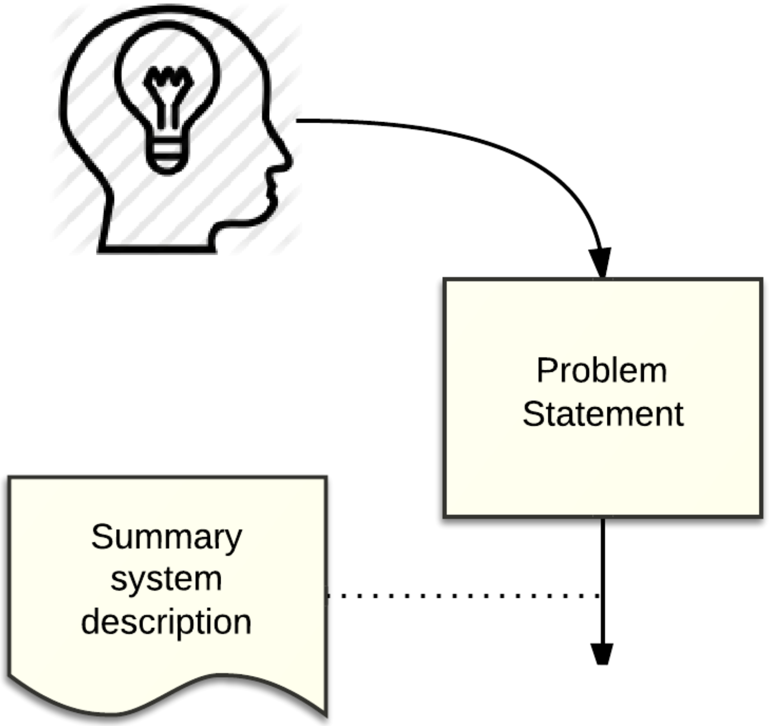


AmI Design Process

STEP 1: PROBLEM STATEMENT

Problem Statement

- Define what **problems** need to be solved/tackled
- Identify the **benefits**
  - For the users
  - For the environment
- Create a **brief summary** of **what the system does for the users**



Summary System Description

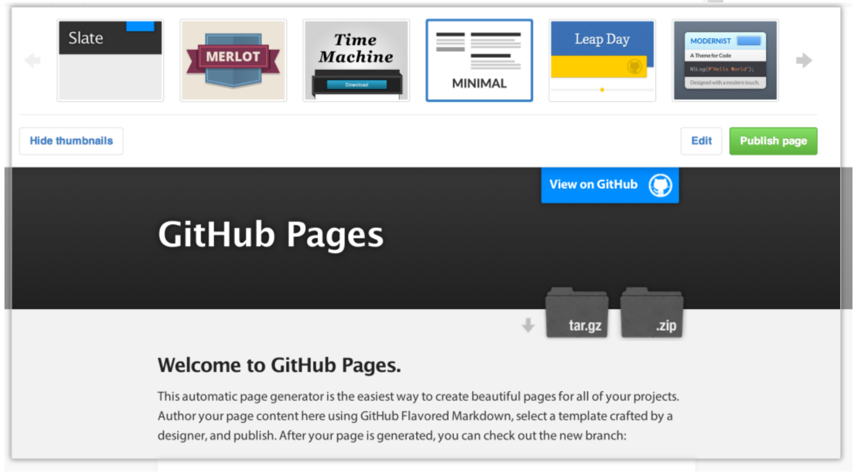
- ½ page – 1 page max of “vision”
- Absolutely **avoid** describing the **technology** or making some technical choices
- Define the target environment
- Define your **users**
- Describe **how** the environment supports the users, from the user point of view
- Try to **hint** at AmI features (Sensitive, Responsive, Adaptive, Transparent, Ubiquitous, Intelligent)
- Imagine “selling” it to a non-engineer (find someone to read it)

Tips

- No technology
  - But we must know it's feasible, somehow
- Start simple
  - Few features, few users
  - But full AmI features
- Pitch it
  - Why users should be happy to use it
  - Tell a story…
- Google it
  - Search for similar ideas / products / articles
- Involve users
  - Describe, discuss, ask, **LISTEN**
  - Users know better (except when they don't)

Deliverable 1

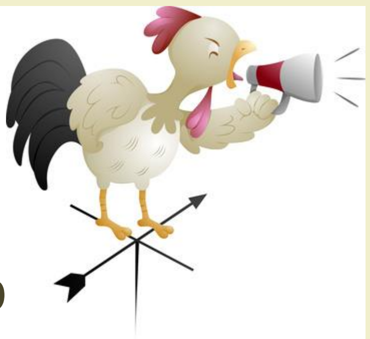
- Before 26/03
- Set-up project web site
- Develop your «Vision» document
- Upload the «Vision» on the website
- You'll receive feedback on 30/03 (in LADISPE)



Welcome to GitHub Pages.

This automatic page generator is the easiest way to create beautiful pages for all of your projects. Author your page content here using GitHub Flavored Markdown, select a template crafted by a designer, and publish. After your page is generated, you can check out the new branch:

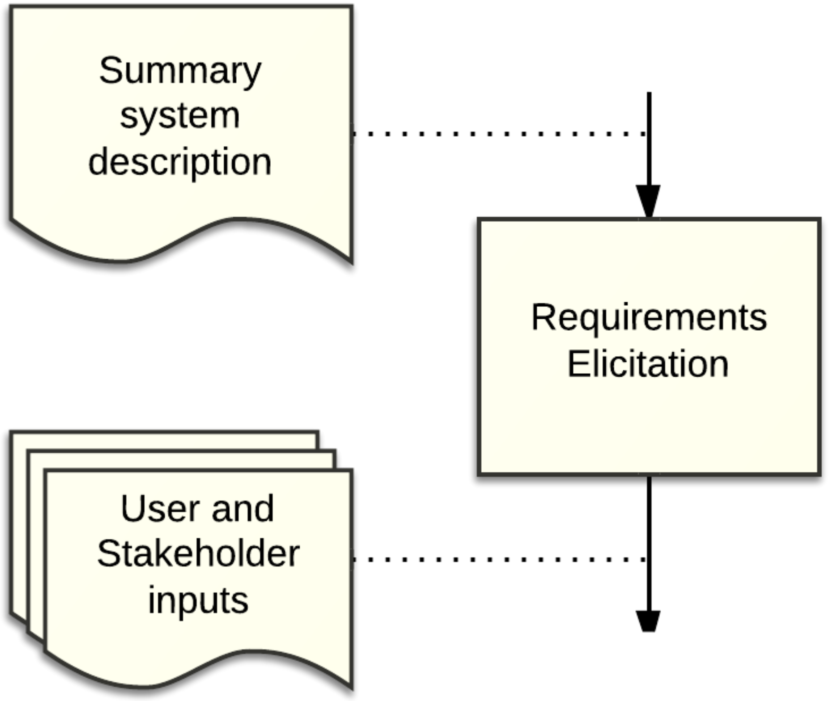
Vision: «WakeKill»



- Each user requires their own personalized wake-up experience. Users will never miss a wake-up call, every morning will be a pleasing experience and they will never be late. Your house, your devices, your calendars, will team up to personalize the optimum wake-up call, personalized to you, and personalized to your day’s schedule, location, and mood.
- The system will exploit different means to wake up users in the morning. It will combine ringing, turning on the lights, the radio, and other methods, according to the available devices and to user preferences. It will automatically adjust time according to the user’s agenda. When the user is not at home (e.g., hotel) it avoids activating at-home devices, and only users user devices. It will detect when the user actually wakes up (or is already up).

WakeKill

I absolutely love the user experience that WakeKill gives me

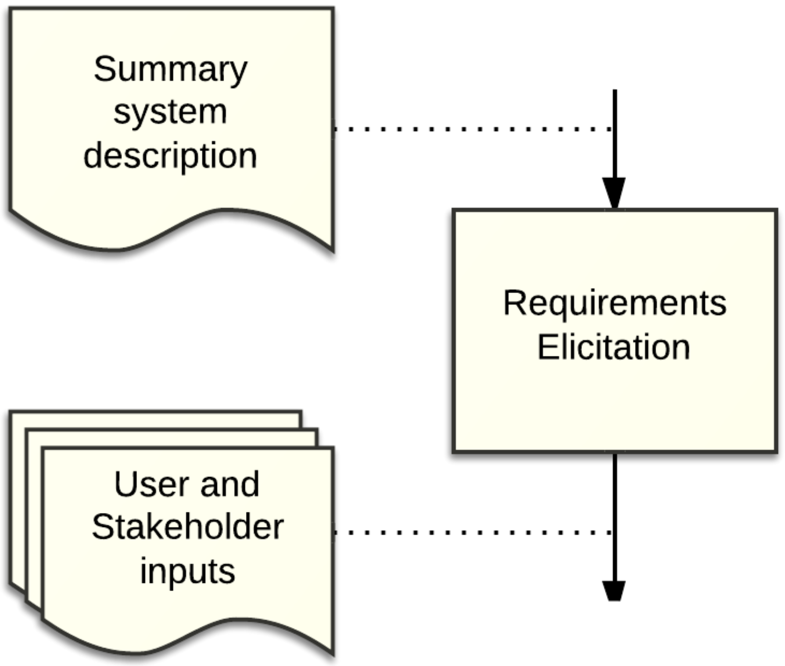


AmI Design Process

STEP 2: REQUIREMENTS ELICITATION

Elicitation

- Consider the needs and the opinions of
  - Users of the system
  - Stakeholders for the system
- Collect and evaluate carefully and objectively
- If needed, adapt your vision

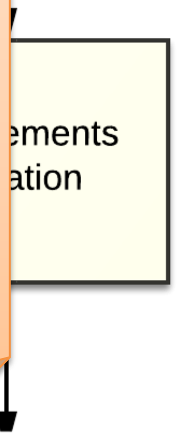


Elicitation

- Consid
  the op
  - User
  - Stak
    syste
- Collect
  careful
- If need
  vision

Due to time restrictions, this step is **not formally required** in the AmI course.
In the course, just try to get as many user inputs as possible, even in an informal and unstructured way, and consider them in building your vision.

It is, however, **essential** for successful ICT products.



Roles

Users

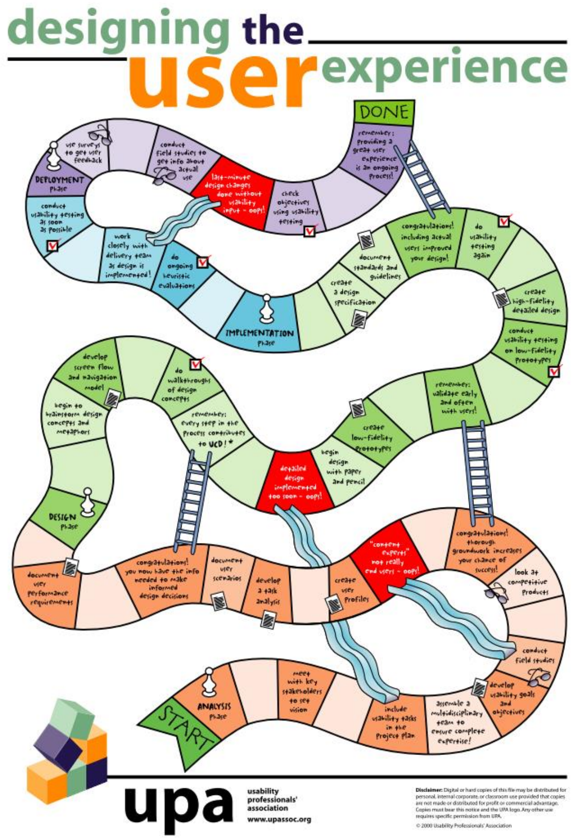
- Persons that will be the final targets of the system and will interact with the system
- Or, at least, persons with similar characteristics to the actual final targets
- Don’t need to understand how the system works
- Need to understand how they will interact

Stakeholders

- Persons (or institutions) that will have an interest in the success of the system
- May not be users
- “Interest” may be economic, better efficiency, user satisfaction, higher control or security, better understanding, …
- May be involved in funding the system

Users know better

- Serving users should be the cornerstone of AmI
- “User Centered Design” (UCD) is a methodology that includes a set of techniques for involving users throughout the design process



designing the user experience
DONE
remember: providing a great user experience is an ongoing process!
use surveys to get user feedback
conduct field studies to get info about actual use
last-minute design changes done without usability input - oops!
check objectives using usability testing
DEPLOYMENT phase
conduct usability testing as soon as possible
work closely with delivery team as design is implemented!
do ongoing heuristic evaluations
IMPLEMENTATION phase
create a design specification
document standards and guidelines
congratulations! including actual users improved your design!
do usability testing again
create high-fidelity detailed design
conduct usability testing on low-fidelity prototypes
remember: validate early and often with users!
create low-fidelity prototypes
begin design with paper and pencil
detailed design implemented too soon - oops!
remember: every step in the process contributes to UCD!*
do walkthroughs of design concepts
develop screen flow and navigation model
begin to brainstorm design concepts and metaphors
DESIGN phase
document user performance requirements
congratulations! you now have the info needed to make informed design decisions
document user scenarios
develop a task analysis
create user profiles
"content experts" not really end users - oops!
congratulations! thorough groundwork increases your chance of success!
look at competitive products
conduct field studies
develop usability goals and objectives
assemble a multidisciplinary team to ensure complete expertise!
include usability tasks in the project plan
meet with key stakeholders to set vision
ANALYSIS phase
START
upa
usability professionals' association
www.upassoc.org
Disclaimer: Digital or hard copies of this file may be distributed for personal, internal corporate, or classroom use provided that copies are not made or distributed for profit or commercial advantage. Copies must bear this notice and the UPA logo. Any other use requires specific permission from UPA.
© 2000 Usability Professionals' Association

http://www.mprove.de/script/00/upa/_media/upaposter_85x11.pdf

UCD requirements

- ISO standard Human-centered design for interactive systems (ISO 9241-210, 2010)
  - The design is based upon an explicit understanding of users, tasks and environments.
  - Users are involved throughout design and development.
  - The design is driven and refined by user-centered evaluation.
  - The process is iterative.
  - The design addresses the whole user experience.
  - The design team includes multidisciplinary skills and perspectives.

UCD tools and techniques

Conceptual tools

- Personas
  - a fictional character with all the characteristics of a “typical” user
- Scenario
  - a fictional story about the "daily life of" or a sequence of events with personas as the main character
- Use Case
  - the interaction between an individual and the rest of the world as a series of simple steps for the character to achieve his or her goal

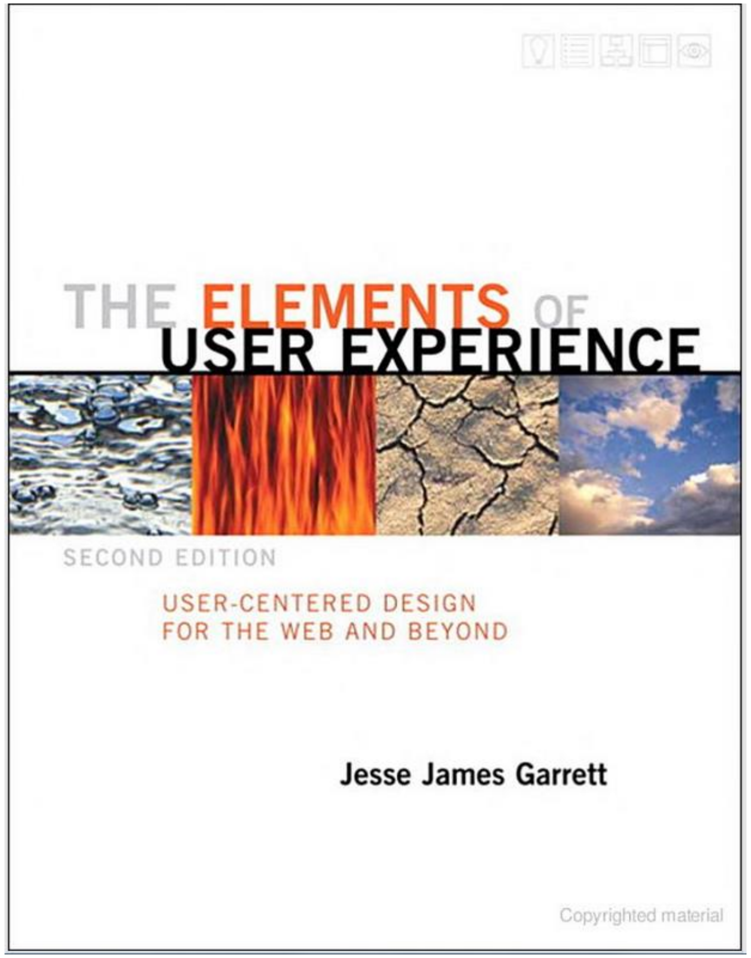
Design techniques

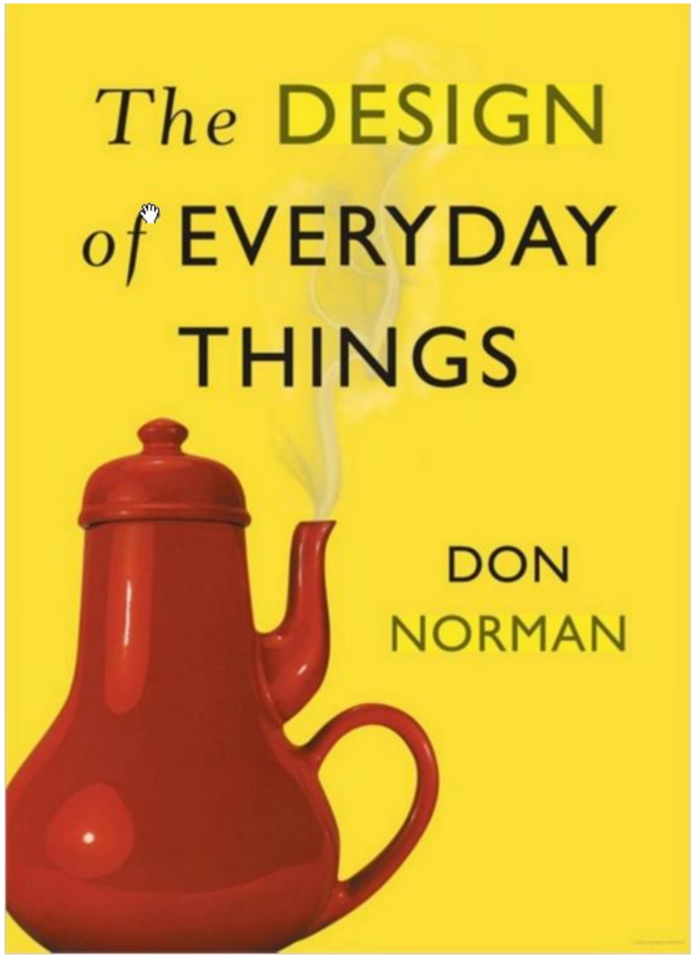
- Field research
- Focus groups
- Interviews
- Design walkthroughs
- Low-fi and Hi-fi prototypes
- Mock-up evaluation
- Usability testing

Result

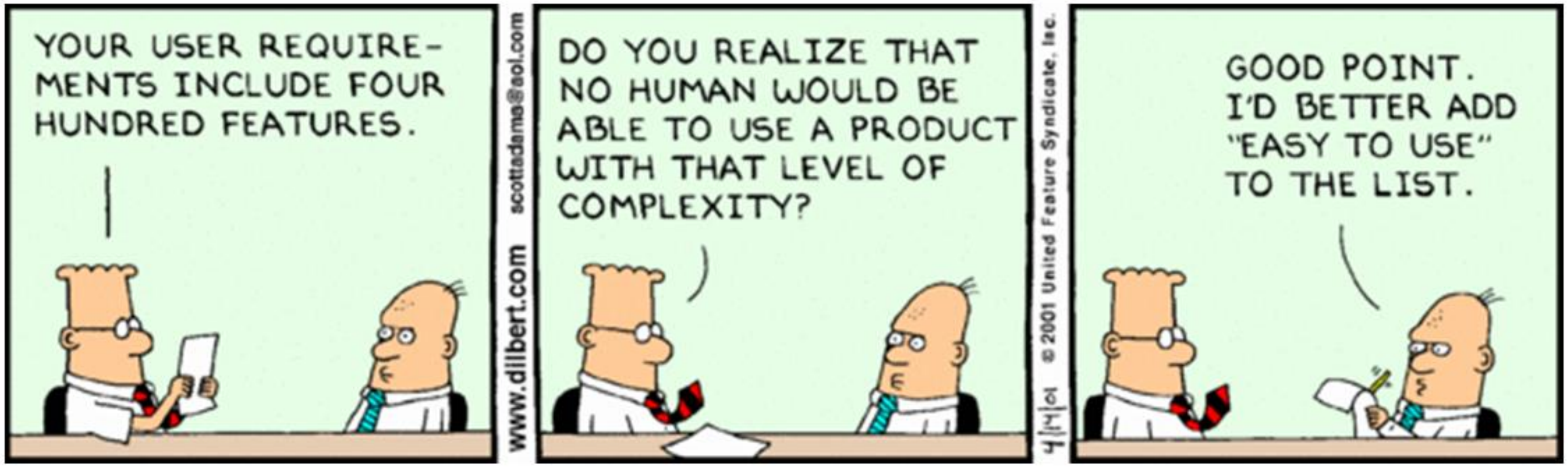
- Increased awareness of user perception in your proposed system
- Priority for different system features (some will be abandoned, some will be new)
- Gather design constraints (price, size, aesthetics,
- Mediate user inputs with product strategy
- Transform “a good idea” into “a system that users want”

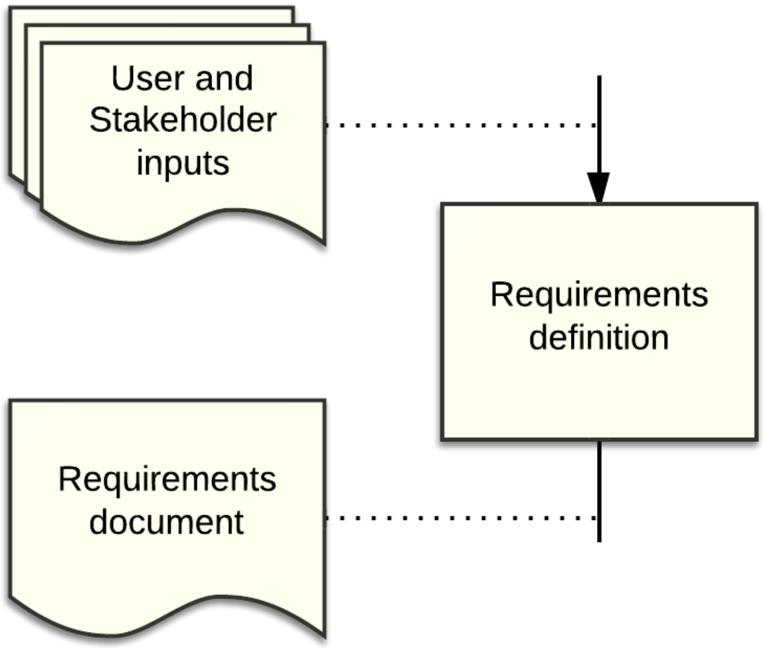
Guru References


THE ELEMENTS OF USER EXPERIENCE
SECOND EDITION
USER-CENTERED DESIGN FOR THE WEB AND BEYOND
Jesse James Garrett
Copyrighted material



Beware…


YOUR USER REQUIREMENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.
www.dilbert.com scottadams@aol.com
© 2001 United Feature Syndicate, Inc.
4/14/01

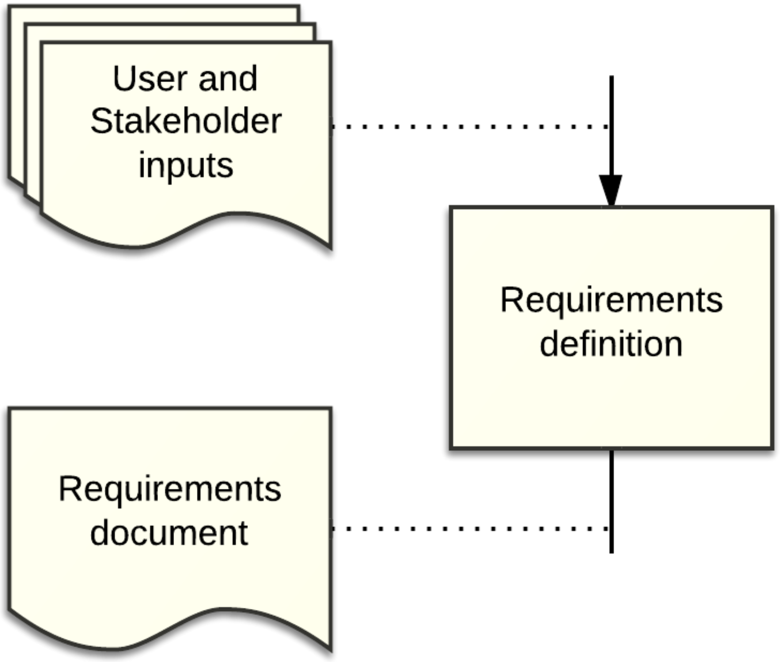


AmI Design Process

STEP 3: REQUIREMENTS IDENTIFICATION

Formalizing requirements

- The initial vision and user inputs must be “distilled” into a set of requirements
- Strategic choices: what is in, what is out
- Describes what the system does, and the external constraints
- Might be used as a “specification contract”



Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is unavoidable, as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation;
  - May be the basis for the contract itself - therefore must be defined in detail;
- Both these statements may be called requirements.

Types of requirement

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements (a.k.a. developer requirements)
  - A structured document setting out detailed descriptions of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Example

User requirement definition

The software must provide a means of representing and accessing external files edited by other tools

System requirements specification

1.1 The user should be provided with facilities to define the type of external files

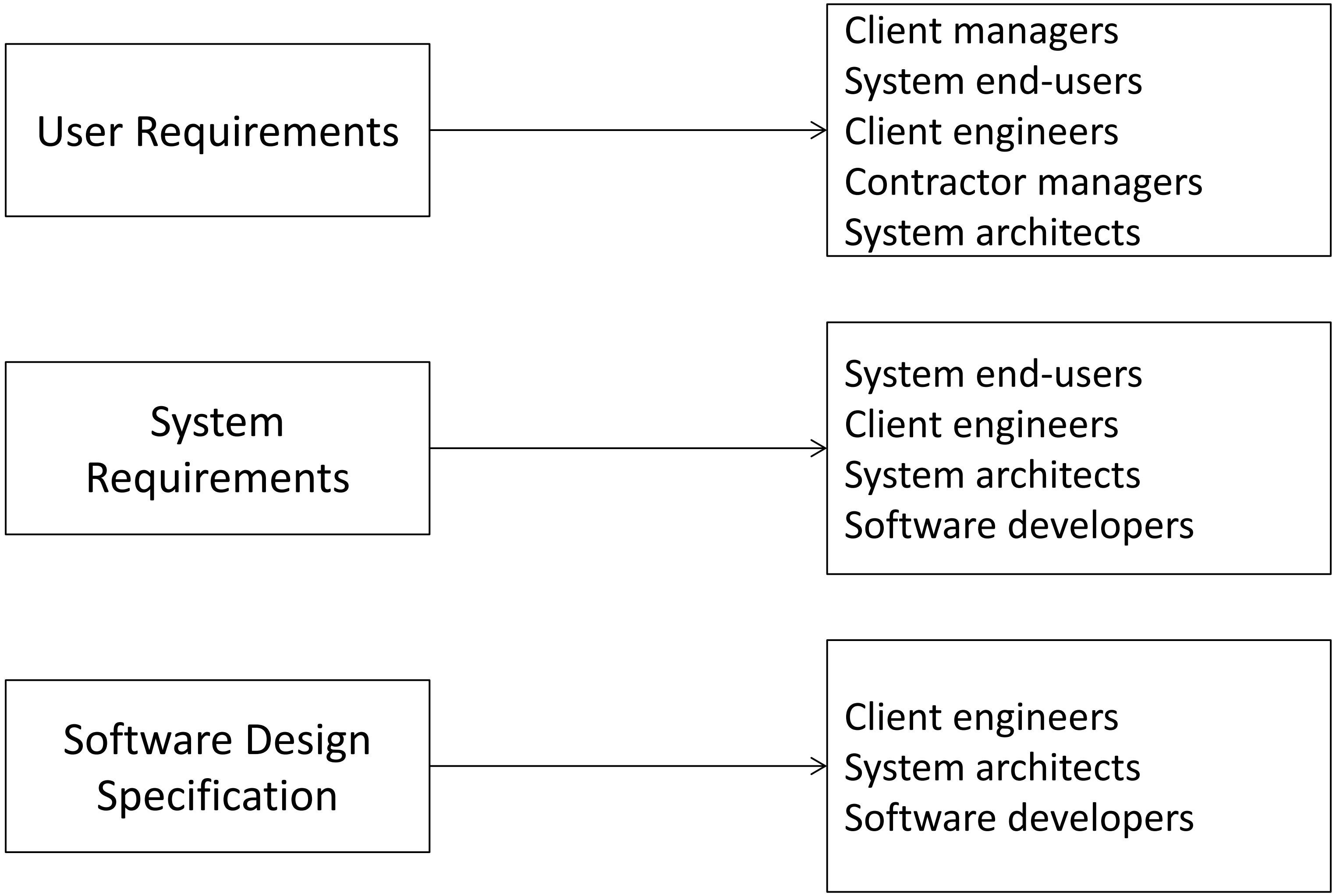
1.2 Each external file type may have an associated tool which may be applied to the file

1.3 Each external file type may be represented as a specific icon on the user's display

1.4 Facilities should be provided for the icon representing an external file type to be defined by the user

1.5 When a user selects an icon representing an external file the effect of that selection is to apply the tool associated with the external file type to the file represented by the selected icon

Requirements readers



Types of requirements

- Functional requirements (FR)
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements (NFR)
  - Aka Quality requirements
  - constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain.

Functional Requirements (FR)

- What the system does
- What functions it offers to its users
- Don’t care how they will be implemented (yet)
- A long list of “local” features (easy to identify a small portion of the system that delivers that function)

Examples

- FR3.1: The user must be able to activate and de-activate the wake-up service. This decision will be applied until the user changes it again.
- FR3.2: The user must be able to silence the wake-up service just for the next day. Service will resume automatically on the following day.
- FR4.4: The user must be able to set up an “ad hoc” wake-up call, that will run only once, will not be remembered, and will have specific settings
- FR4.4.1: The user may configure the settings of any already defined “ad hoc” call
- FR4.4.2: The user may configure the default settings for (to be created) “ad hoc” calls

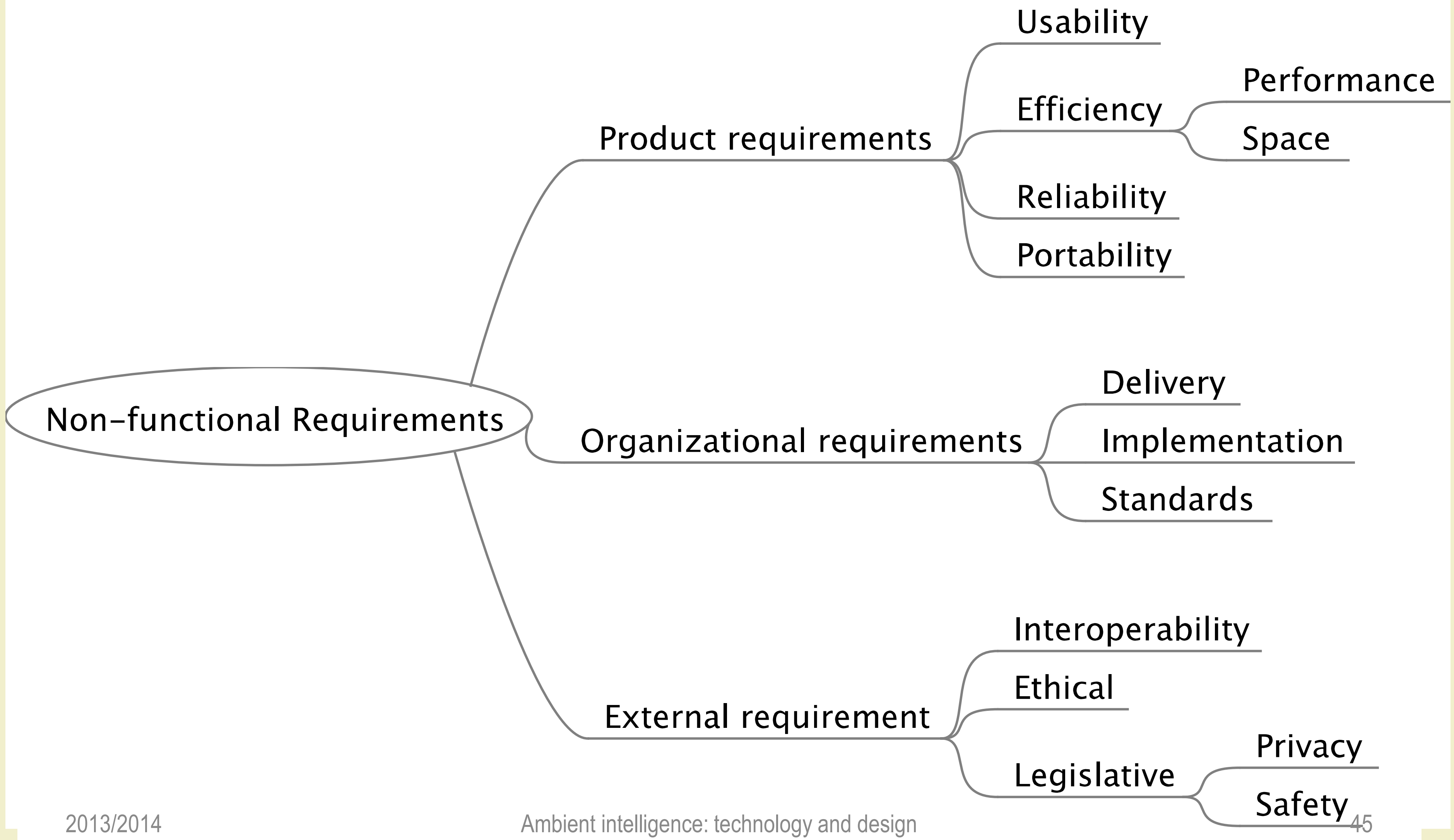
Non-functional requirements (NFR)

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, Supported devices, Usability, Language, etc.
- Process requirements may also be specified mandating a particular set of tools, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Pervasiveness in NFR

- NFR are usually “general” and cannot be localized to a single spot in system implementation
- Every function, in every module, in every screen, in every device, …. must guarantee that no NFR is broken

Non-functional requirements



Examples

- NFR1: The mobile interfaces must be compatible with iOS (8.0 and later), Android (4.2 and later)
- NFR2: The system will be localized in many languages (default: English)
- NFR18: The system should work, in reduced conditions, even if the user mobile device is switched off or disconnected
- NFR3: The web interfaces will be compatible with browsers … version ….
- NFR4: The web interfaces will be “responsive”, and will adapt to screen resolutions from 800x600 to 1920x1080

Good requirements

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked
- Verifiable
- Modifiable
- Traceable

Good Requirements

- Correct
  - Every requirement stated is one that the software shall meet
  - Customer or users can determine if the requirement correctly reflects their actual needs
    - Traceability makes this easier

Good Requirements

- Unambiguous
  - Every requirement has only one interpretation
  - Each characteristic of the final product must be described using a single unique term
  - Both to those who create it and to those who use it.

Good Requirements

- Complete
  - Include all significant requirements
    - Address external requirements imposed by system specification
  - Define response to all realizable inputs
    - Both correct or incorrect
  - Define all terms and unit of measure

Good Requirements

- Internally Consistent
  - No subset of requirements is in conflict
    - Characteristics of real-world objects (e.g. GUI)
    - Logical or temporal
    - Different terms for the same object

Completeness and consistency

- In principle, requirements should be both complete and consistent.
  - Complete
    - They should include descriptions of all facilities required.
  - Consistent
    - There should be no conflicts or contradictions in the descriptions of the system facilities.
  - In practice, it is impossible to produce a document that is both complete and consistent

Good Requirements

- Ranked
  - Stability in the future
  - Necessity
    - Essential
    - Conditional
    - Optional

Good Requirements

- Verifiable
  - there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement.
    - Ambiguous requirements are not verifiable

Good Requirements

- Modifiable
  - structure and style such that any changes can be made easily, completely, and consistently while retaining the structure and style
    - Well structured
    - Non redundant
    - Separate requirements

Good Requirements

- Traceable
  - Backward
    - explicitly referencing source in earlier documents
  - Forward
    - unique name or reference number

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases
4. The technology to be used
5. Other various requirements
6. Human backup, legal, political, organizational issues

Requirements Document

1. Purpose and scope

- What is the overall scope and goal?
- Stakeholders (who cares?)
- What is in scope, what is out of scope

sues

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases
4. The technology to be used
5. Other various requirements
6. Human backup, legal, political, organizational issues

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases

- The primary actors and their general goals
- The business use cases (operations concepts)
- The system use cases
- Collects most **functional requirements**

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases
4. The technology to be used

- What technology requirements are there for this system?
- What systems will this system interface with, with what requirements?

- Development process
- Business rules
- Performance
- Operations, security, documentation
- Use and usability
- Maintenance and portability
- Unresolved or deferred

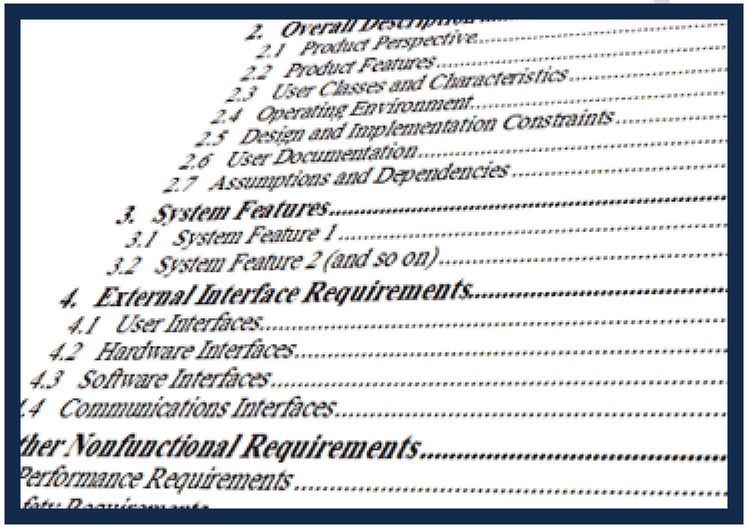
5. Other various requirements
6. Human backup, legal, political, organizational issues

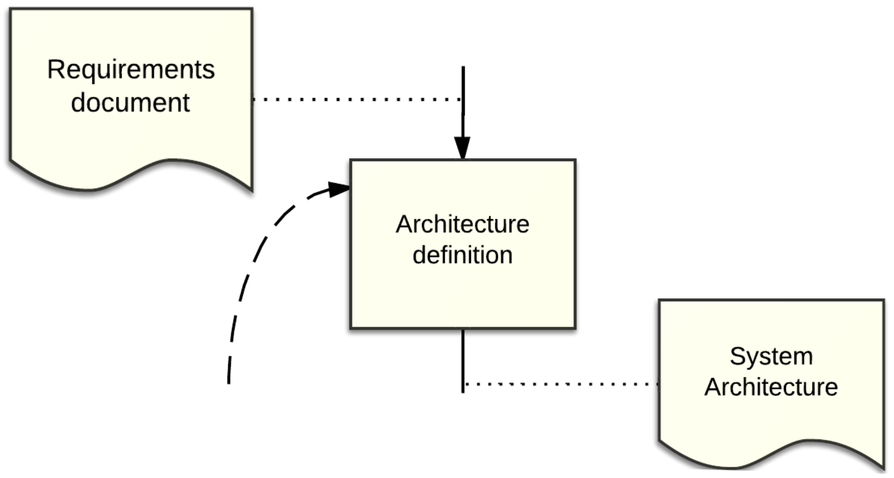
- What is the human backup to system operation?
- What legal, what political requirements are there?
- What are the human consequences of completing this system?
- What are the training requirements?
- What assumptions, dependencies are there on the human environment?

6. Human backup, legal, political, organizational issues

Deliverable 2

- Before 13/04
- **Functional Requirements**
- **Non Functional Requirements**
- We'll provide a template and an example
- Upload on the website
- You'll receive feedback on 27/04



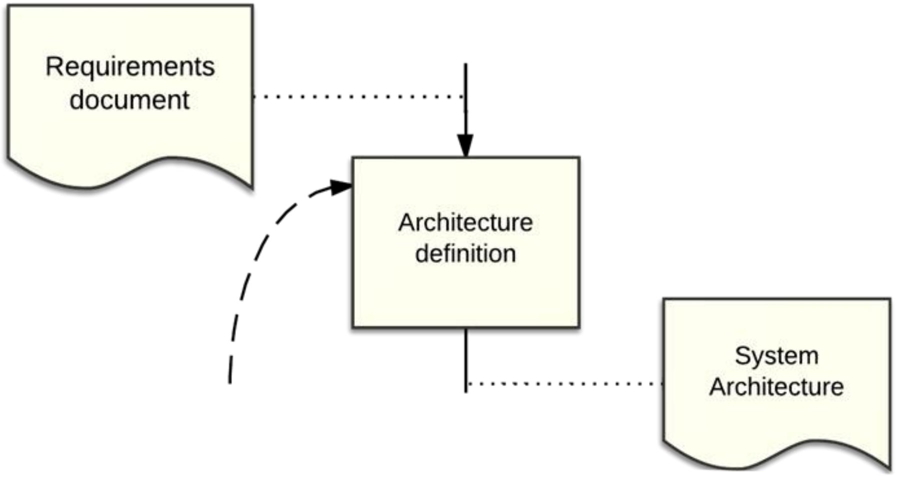


AmI Design Process

STEP 4: ARCHITECTURE DEFINITION

Defining the Architecture

- System Architecture
- Hardware Architecture
- Software Architecture
- Network Architecture



System Architecture

- What are the main system components, what is their nature, and what kind of information they exchange with the environment, the user, and other components?

- Computational nodes (One? Many?)
- Sensors/actuators (which physical interactions? Where installed? How interconnected?)
- User interfaces (Where? What functions?)
- Which functions are deployed on which nodes?

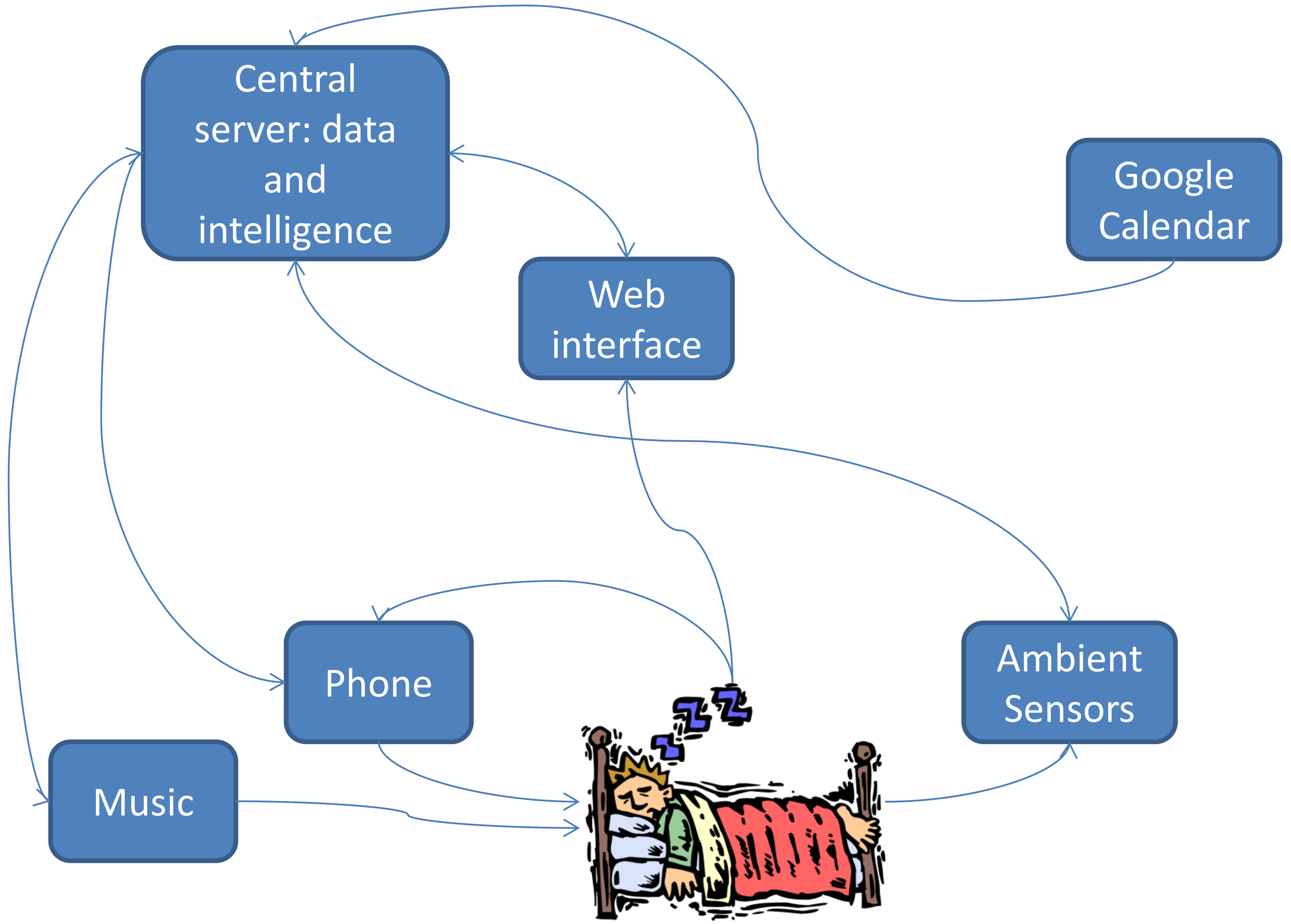
Hardware Architecture

- Computational nodes
- Devices (sensors/actuators)
  - types, function, location
  - not yet brand & model
- User interface devices
  - type, function, location

Software Architecture

- Major software architectural modules
  - what functions (mapped to a subset of functional requirements)
  - where are running (deployment)
  - how they interact (APIs)
- May be existing components, or new SW to be developed
- Adopted libraries and frameworks

Example System Architecture



Example Hardware Architecture

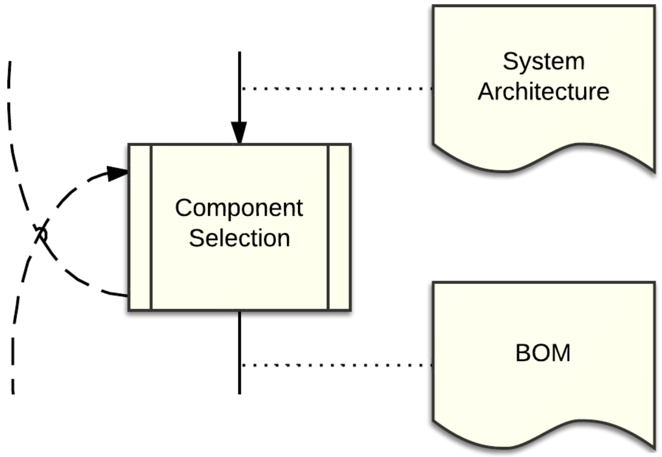
- Ambient sensors
  - Movement sensors in the room
  - Weight/movement sensors under the bed
  - Local gateway (raspberry?) for integrating sensor data
- Mobile Phone (any, Android 4+)
- Server (data storage, interaction with cloud services, web interface generation, intelligence)
  - Anywhere in the web, always-on system.
  - Raspberry-PI? PC? Virtual cloud server?
- Music server (raspberry PI + audio amplifier)

Example Software Architecture

- Data sensor collection software (on local gateway)
  - Sends data to central server
  - Some local processing for detecting situations ???
- Music server software (on local gw)
  - Accept commands from central server
- App (on mobile phone)
  - Settings
  - Ringing
  - Relaying user info (GPS, accelerometer) to central server
- Web application (on central server)
  - User settings
  - Analytics and statistics
- Data storage (on central server)
  - Store sensor data and calendar data
- Intelligent core (on central server)
  - Receive inputs, analyze data, decide what action to perform, send commands to devices

Example Network Architecture

- Local Gateway on home LAN, connected to Internet via ADSL NAT
  - Port forwarding, open tunnel or VPN for being reached BY the central server
- Wireless sensors (e.g., Z-Wave), connected to local gateway (acting as a mesh controller)
- Phone connected to local wi-fi or to 3G network (all functions supported in both cases?). Connects to central server, only
- Central server: world-accessible public IP address

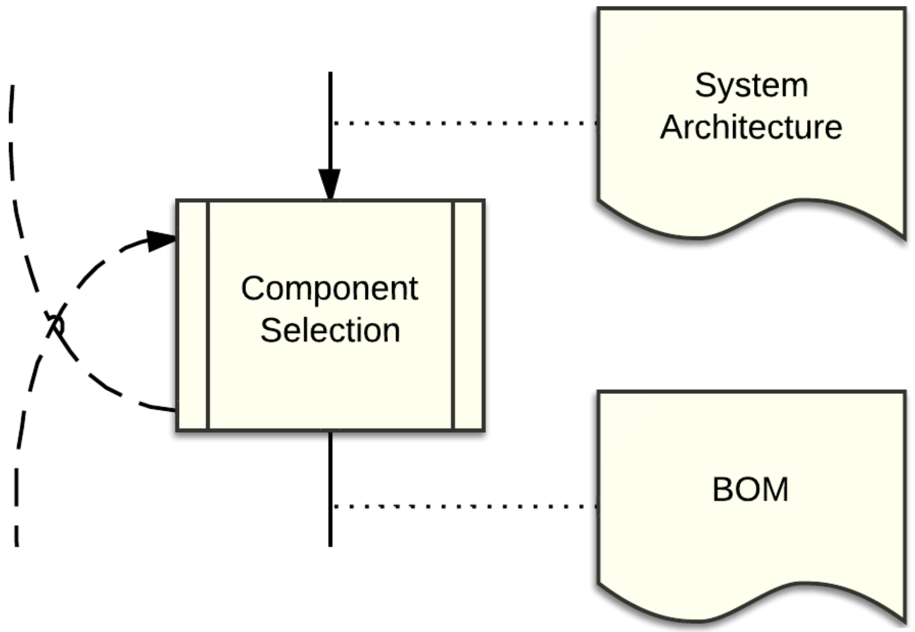


AmI Design Process

STEP 5: COMPONENT SELECTION

Selecting components

- Identifying actual products to populate the chosen architecture description
- Evaluating cost-integration-functionality-design tradeoffs
- Identifying needs for DIY HW and for SW development
- Usually iterates over the definition of the architecture



Selecting HW components

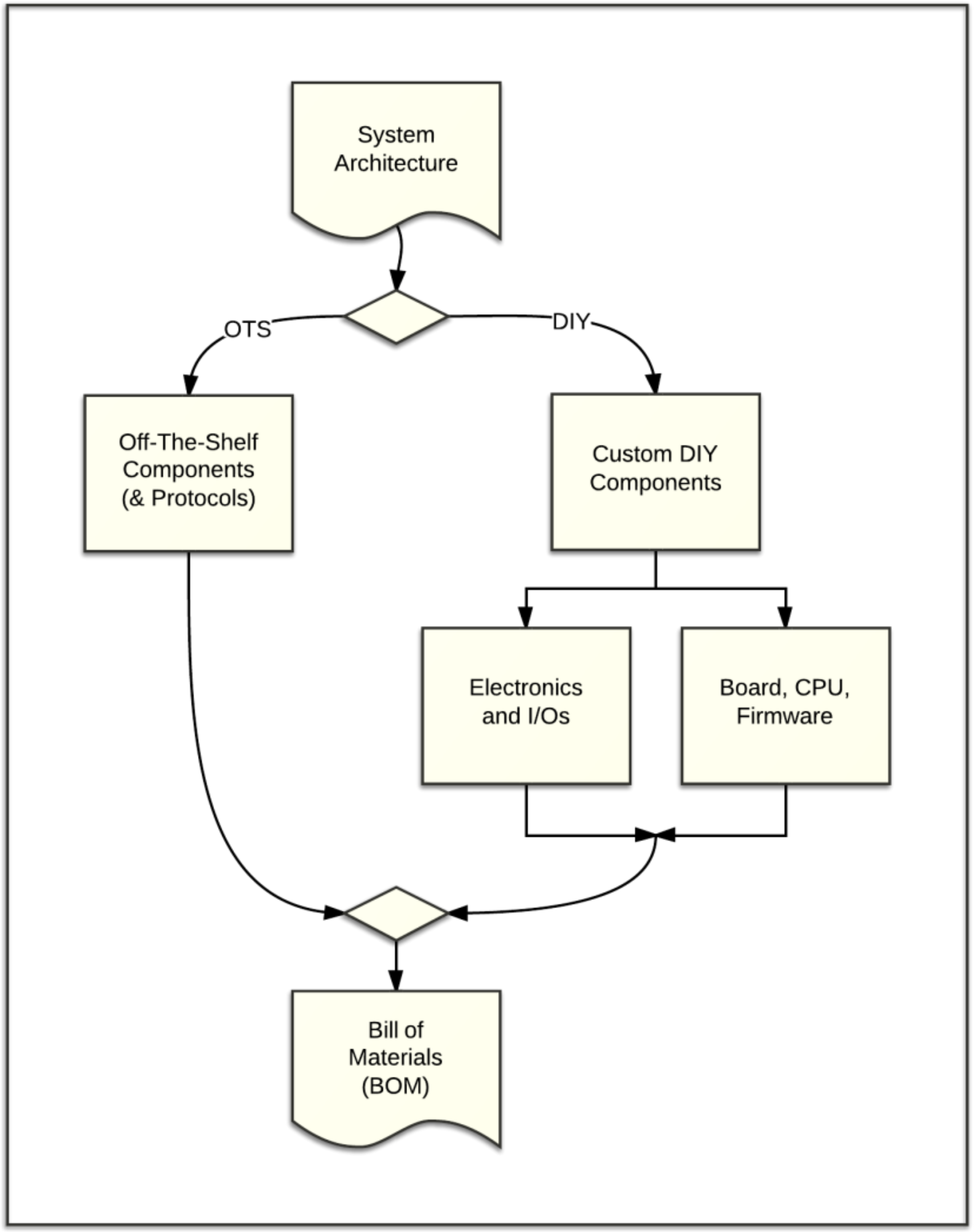
Off-the-shelf

- Which existing OTS components may fit the requirements and the design constraints (also considering budget)
- Aim at selecting, as much as possible, components that share the same communication protocol
- Includes Computational nodes

Custom

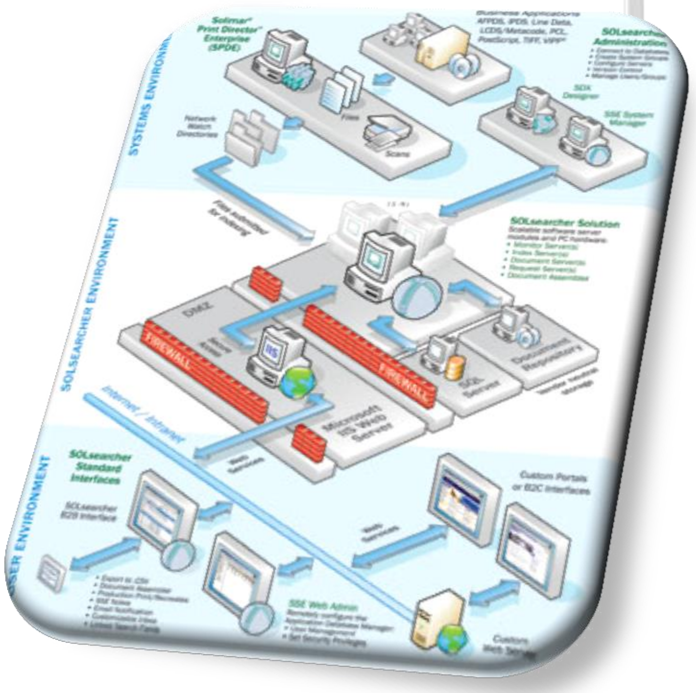
- Which components must be built with DIY techniques
- What kind of hardware (electronics, I/O, …) is needed
- What kind of computational node is required to support the hardware

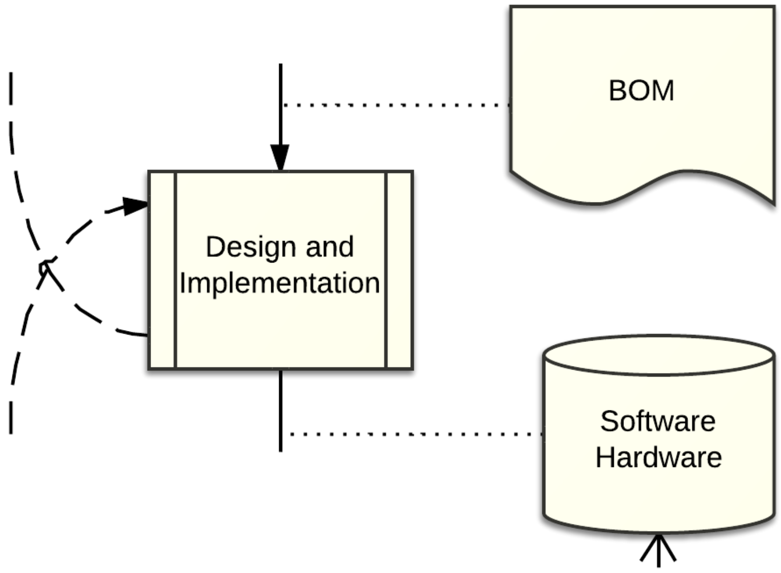
Component Selection



Deliverable 3

- Before 04/05
- **System Architecture**
- **SW Components**
- **HW Components (OTS & ad hoc)**
- We'll provide an example
- Upload on the website
- You'll receive feedback on 11/05



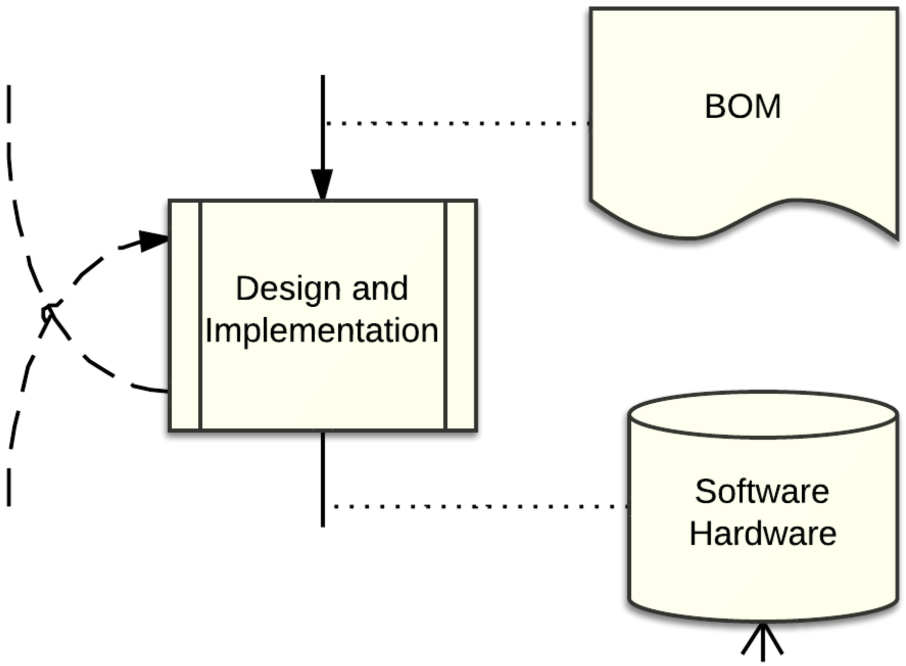


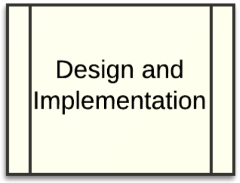
AmI Design Process

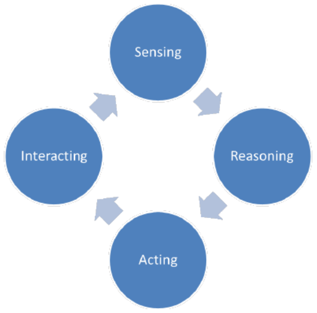
STEP 6: DESIGN & IMPLEMENTATION

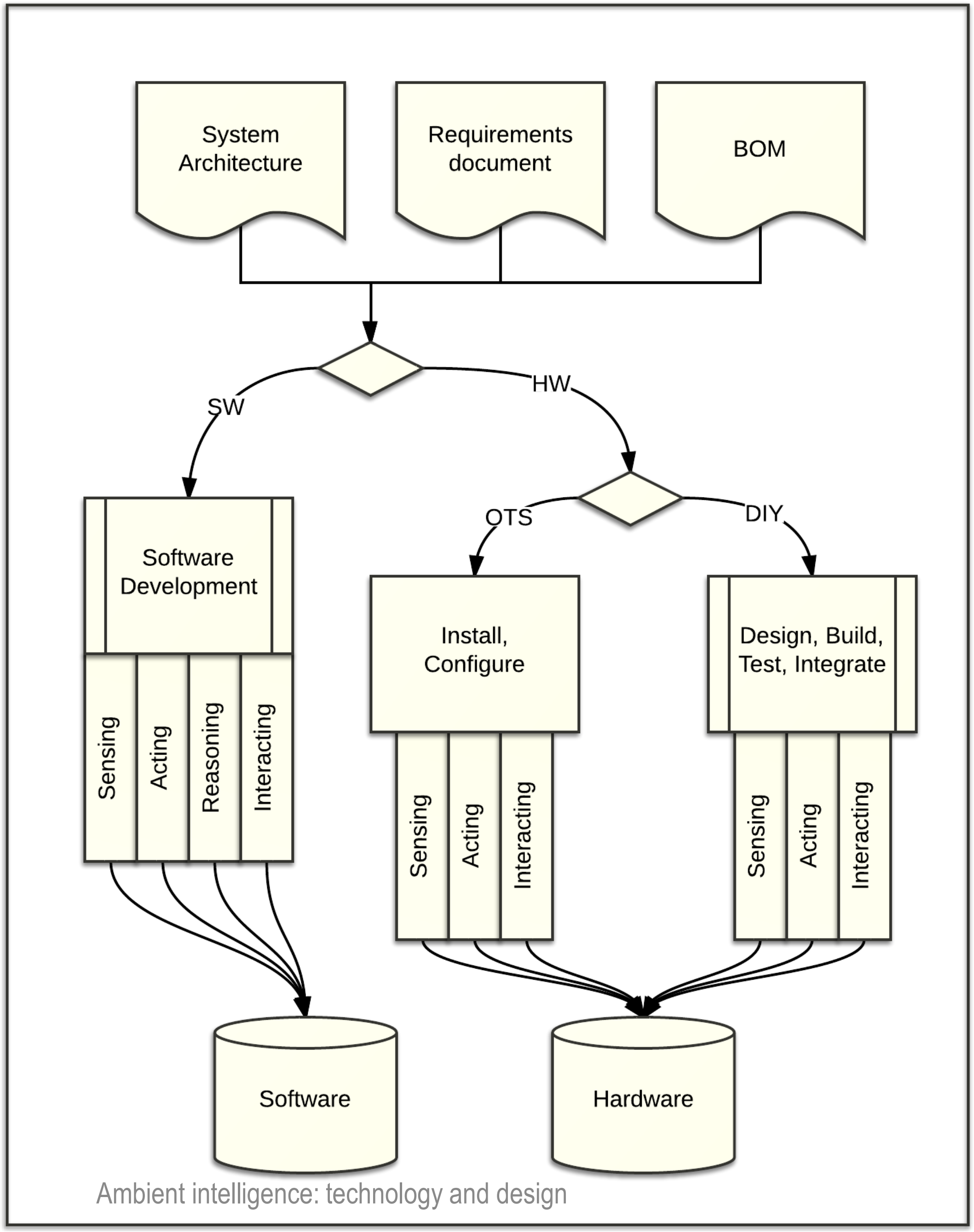
Implementation

- Realize the HW and SW components defined in the previous steps
  - Implement DIY Hardware
  - Install and/or configure OTS Hardware
  - Develop Software
  - Integrate the SW architecture
- Parallel activities for different disciplines




Design and Implementation

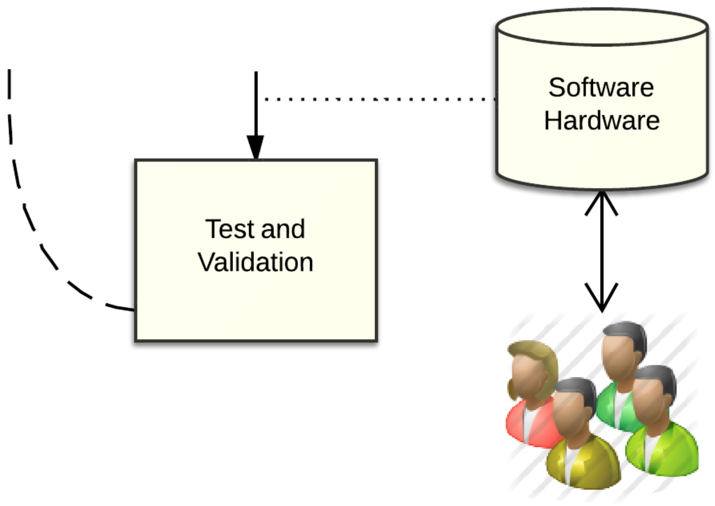

Sensing
Reasoning
Acting
Interacting


System Architecture
Requirements document
BOM
SW
HW
OTS
DIY
Software Development
Sensing
Acting
Reasoning
Interacting
Install, Configure
Sensing
Acting
Interacting
Design, Build, Test, Integrate
Sensing
Acting
Interacting
Software
Hardware

Final project review

- During class of 14/05

- Show state of advancement
- Ask final questions
- Get feedback and directions

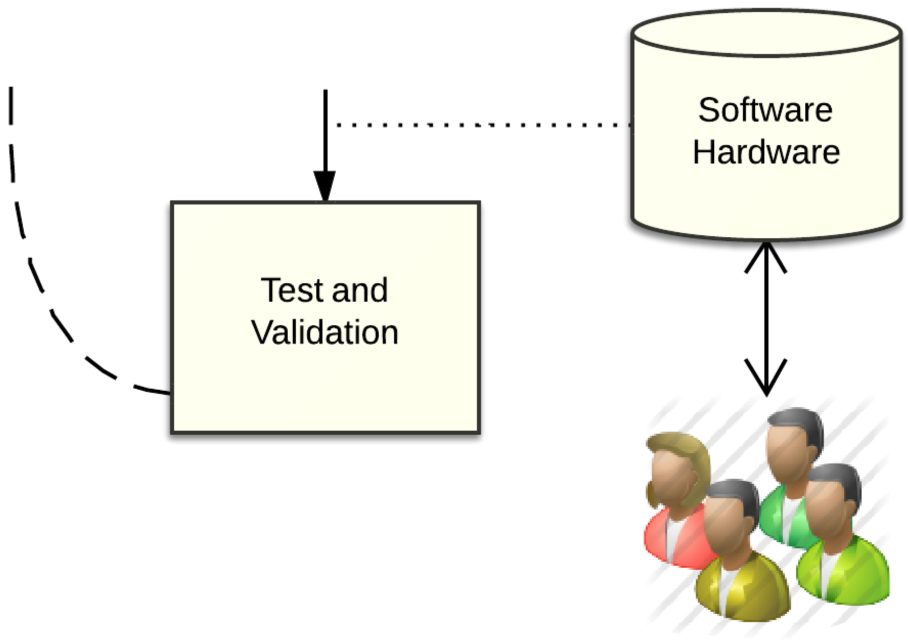


AmI Design Process

STEP 7: TEST AND VALIDATION

Testing the system

- Deploy the prototype of the system (carefully)
- Verify whether requirements are satisfied.
- Verify whether users and stakeholders are satisfied.
- Test should be executed by means of small iterative improvements



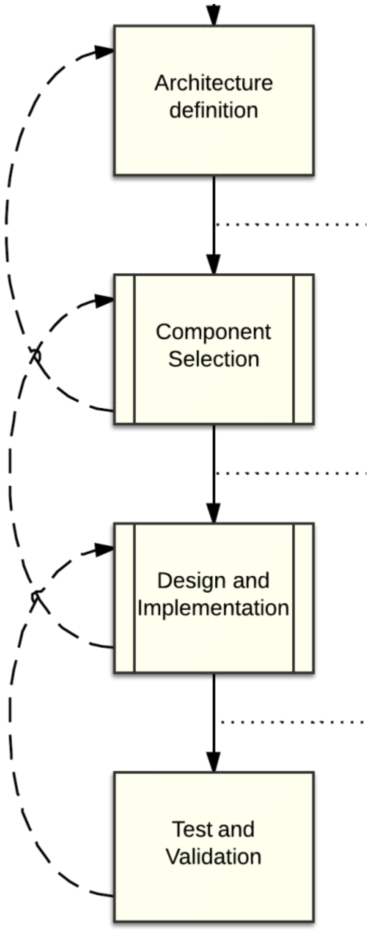
What are we testing?
(aka Verification and Validation)

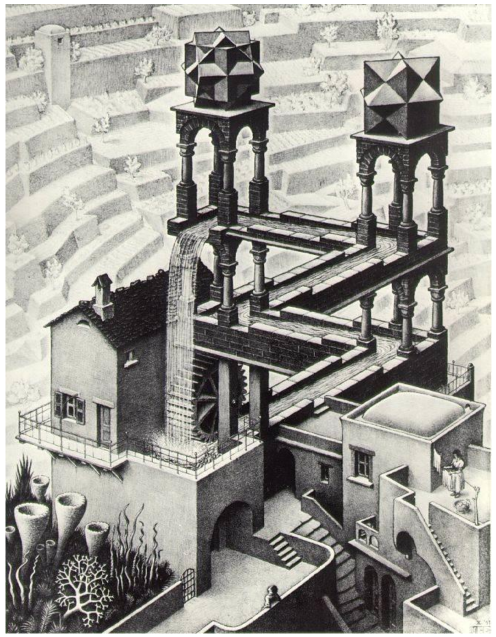
- **Verification** is intended to check that a product, service, or system meets a set of design specifications.
- Test with respect to the Requirements document
- «Am I building the system right?»

- **Validation** is intended to ensure a product, service, or system result in a product, service, or system that meets the operational needs of the user
- Test with respect to Users and Stakeholders inputs
- «Am I building the right system?»

Loops and iterations

- Every design steps should be re-considered, if the need arises
- "Agile" methodologies encourage iterative discovery of system design
- Suggestion: loop over small improvements.
- Aim at a minimal working system, then add features





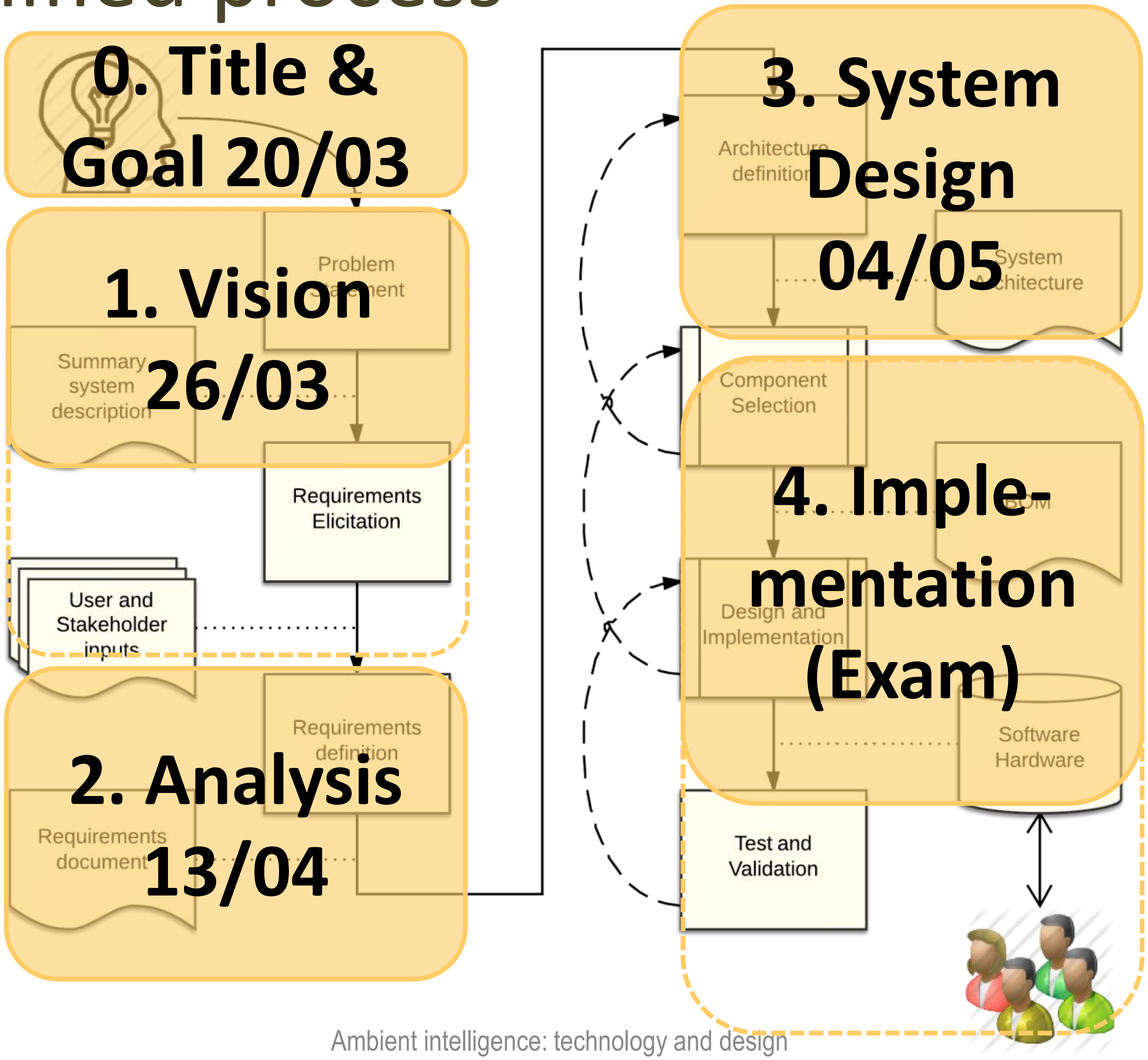
AmI Design Process

SIMPLIFIED PROCESS ADOPTED IN THE AMI COURSE

Principles and constraints

- Propose a simplified process for the Work Groups to be developed in the course
- No time for ‘proper’ user testing
- Need to concentrate resources on development and testing
- Component selection constrained by available devices

Simplified process



Simplified process (in this course)

- 0. Title and Goal
- 1. Vision and Website
  - Deliverable 1: System vision and summary description
- 2. Analysis phase
  - Deliverable 2: Functional Requirements, Non Functional Requirements
- 3. System Design
  - Deliverable 3: System Architecture and Components
- 4. Design & Implementation
  - Deliverable: source code (constant updates to GitHub repository)
- 5. Exam
  - Deliverable: video, presentation and demo

Practical issues

- All deliverable should be submitted through the GitHub platform
- We will provide “templates” for the deliverables
- Deliverables will be checked, and we will provide feedback.
  - If you have questions or doubts, you are responsible for asking
- Intermediate deliverables will be evaluated during the exam.

Resources

- http://en.wikipedia.org/wiki/Verification_and_validation
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications

License