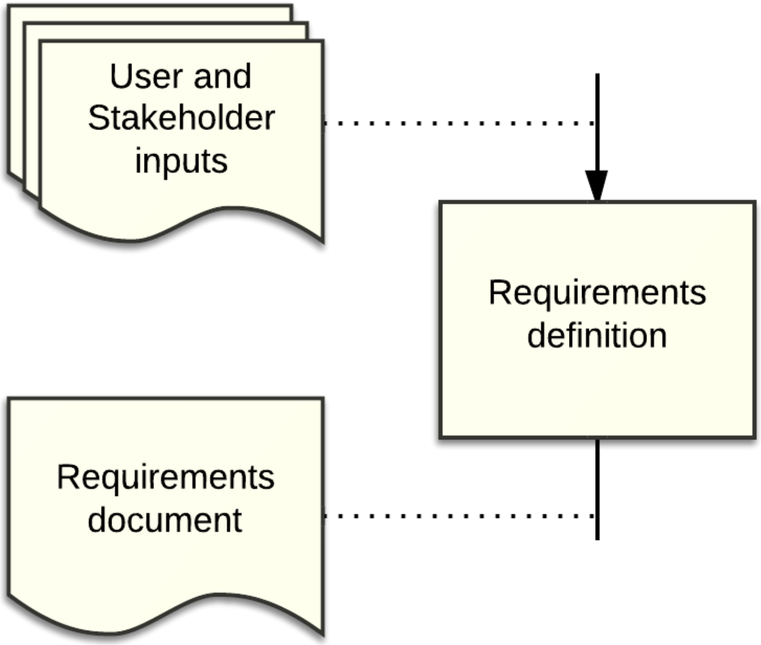
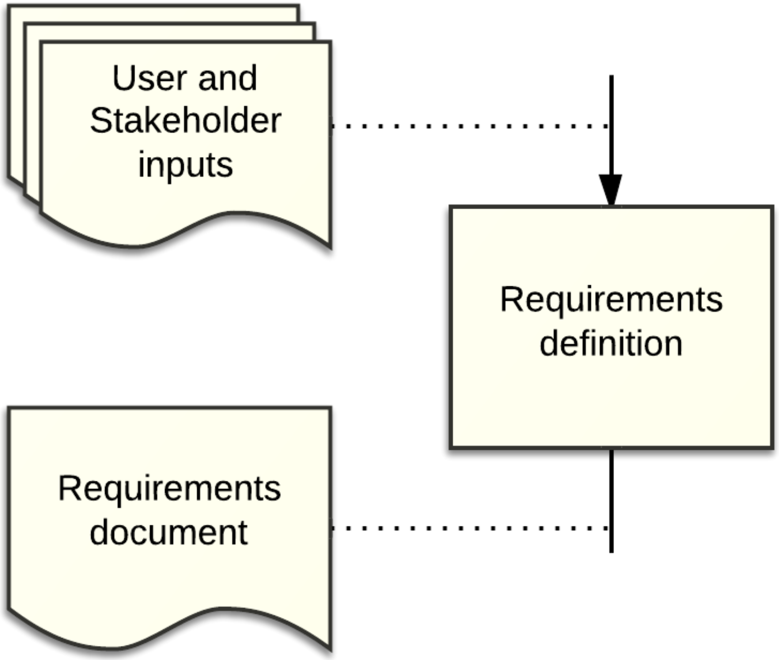


AmI Design Process

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"



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.

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
- 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
- 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

Good Requirements

- Ranked
  - Stability in the future
  - Necessity
    - Essential
    - Conditional
    - Optional
- 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
- Traceable
  - Backward
    - explicitly referencing source in earlier documents
  - Forward
    - unique name or reference number

Requirements vs. Features

Requirement

- A requirement is a capability that a product must possess or something a product must do in order to ultimately satisfy a customer need.
  - more granular
  - written with the *implementation* in mind

Feature

- A feature is a set of related requirements that allows the user to satisfy a business objective or need.
  - "higher-level" objective
  - more focused on *business/user needs*
  - something you'll print on a detailed datasheet
  - intended to be shared with your customers

http://pmblog.accompa.com/2009/07/13/features-vs-requirements-requirements-management-basics/

https://www.aha.io/roadmapping/guide/requirements-management/what-are-product-features

Product Features

- User-visible behaviors
  - data, information, acting, …
- User-callable functionality
  - commands, requests, …
- Information sensed
  - not the sensor, but the associated information
- Available customizations & preferences
- Environment modified behaviors

User Stories, Use Cases, User Narratives

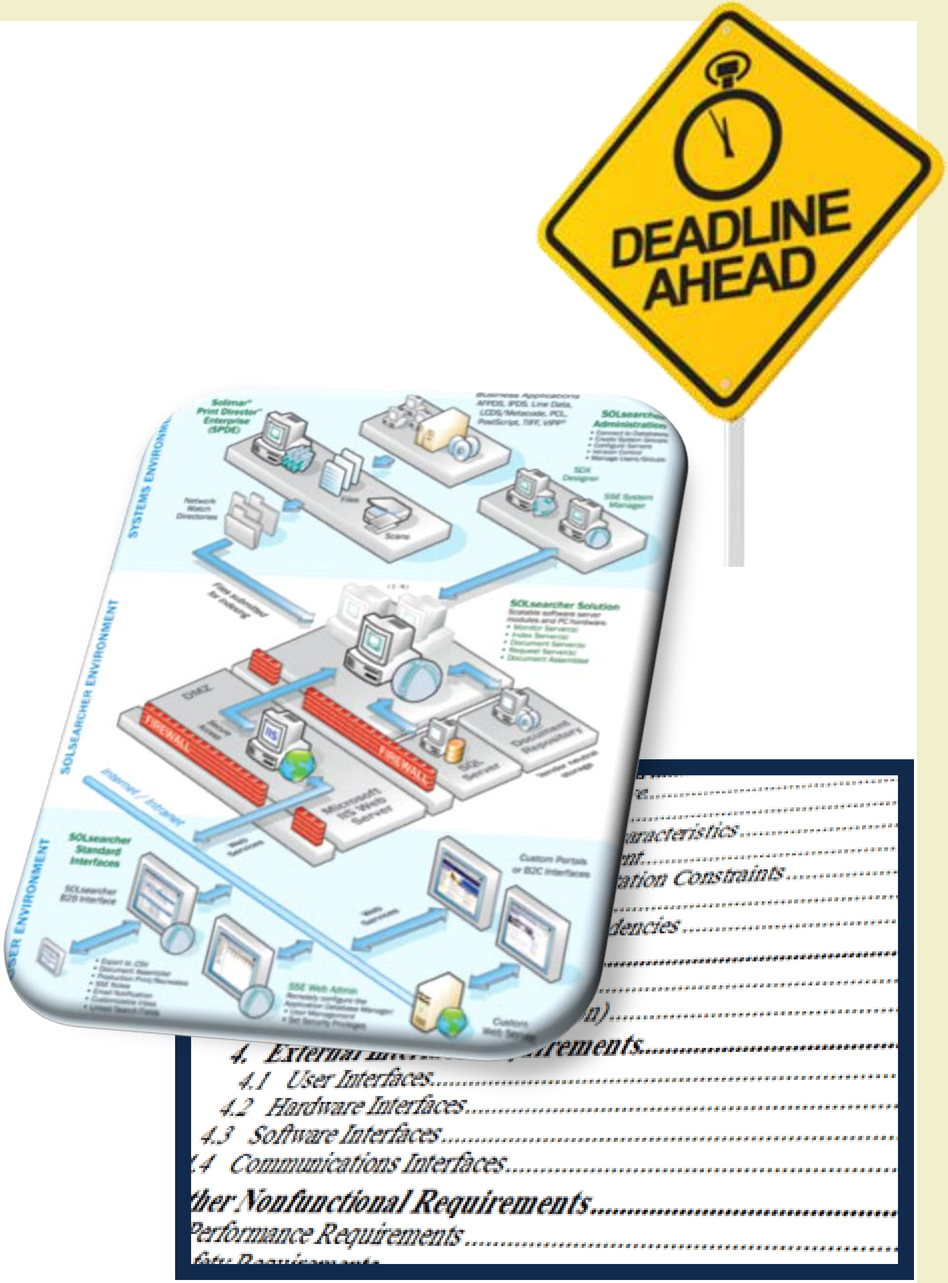
- Features may be illustrated by describing how a user is exploiting them, to reach some user goal
- A user X wants to achieve result Y so that he may get the benefit Z
  - Example: as an avid restaurant visitor I want to see unbiased reviews of a restaurant near a specific location so that I can decide where to go for dinner
  - Enabling feature: Unbiased reviews for restaurants
- User Stories are useful to put feature in context, and see how they interact.

Features (Examples)

- Define a default alarm hour
- Correct the alarm hour according to Google Calendar first appointment
- Two working modes: at home and away
- In away mode, the smartphone rings
- In home mode, music and lights are used in addition to alarm
- Alarm detects when I wake up
- May define preferred music playlist
- May associate home devices

Deliverable 2

- Before 04/05
- **Features**
- **Architecture**
- We'll provide a checklist
- Upload on the website
  - Integrate, no separate download
- You'll receive feedback on 07/05



...aracteristics........

...ation Constraints........

...dencies........

...n)........

4. External Inte... ...rements........

4.1 User Interfaces........

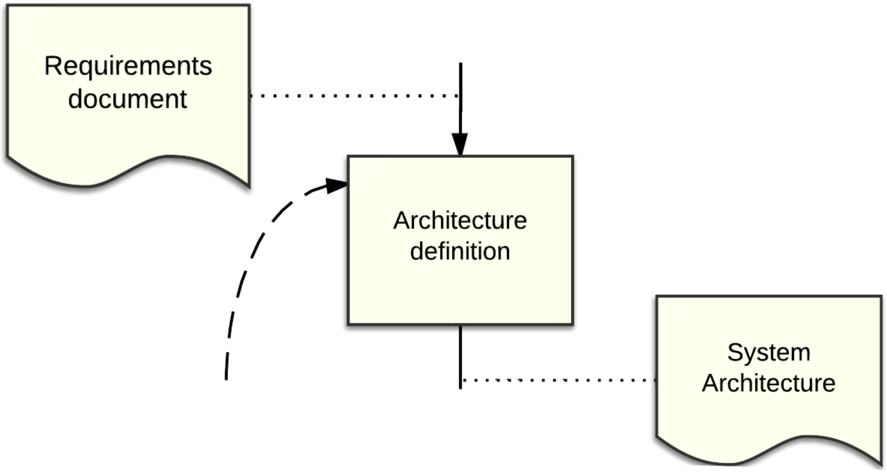
4.2 Hardware Interfaces........

4.3 Software Interfaces........

4.4 Communications Interfaces........

Other Nonfunctional Requirements........

Performance Requirements........

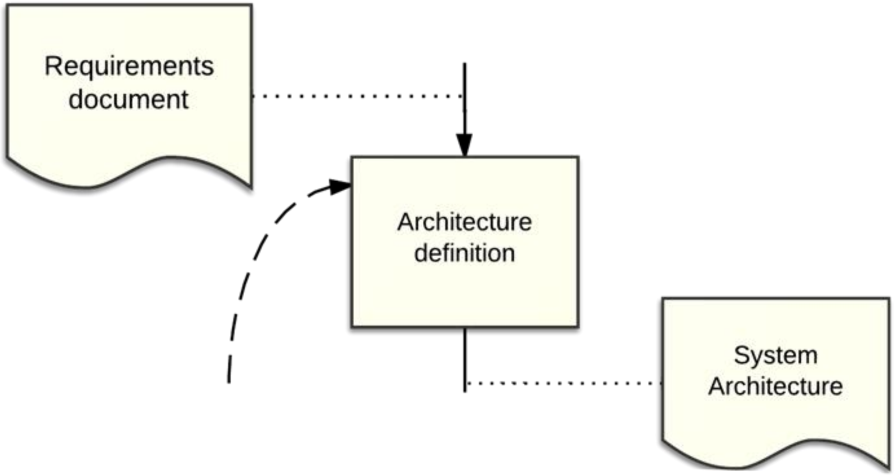


AmI Design Process

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?

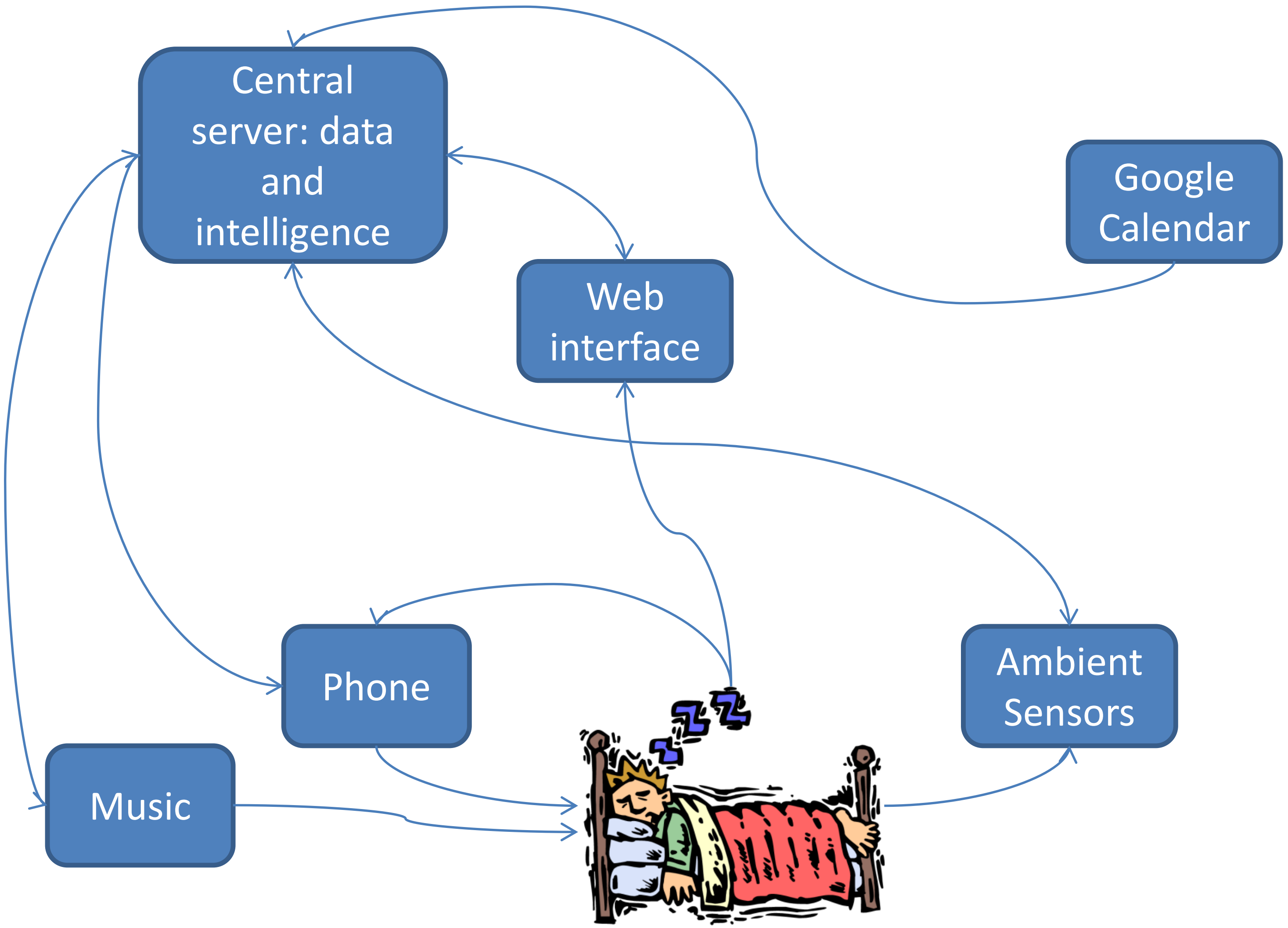
Hardware Architecture

- 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

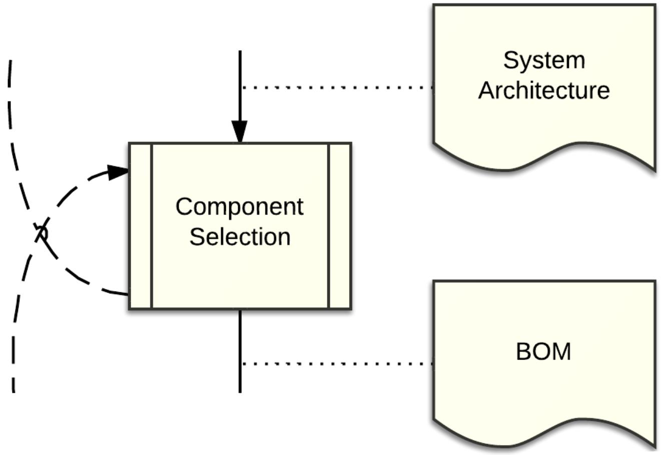
- Ambient sensors
  - Movement sensors in the room
  - Weight/movement sensors under the bed
  - Local gateway (raspberry?) for integrating sensor data
- Mobile Phone (any, Android 4+)
- Server (data storage, interaction with cloud services, web interface generation, intelligence)
  - Anywhere in the 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

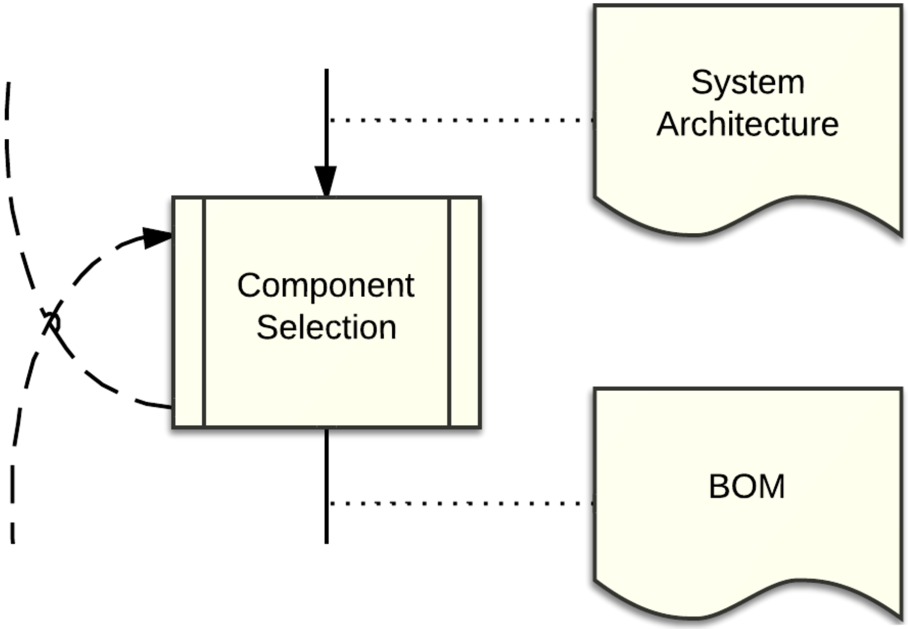


AmI Design Process

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

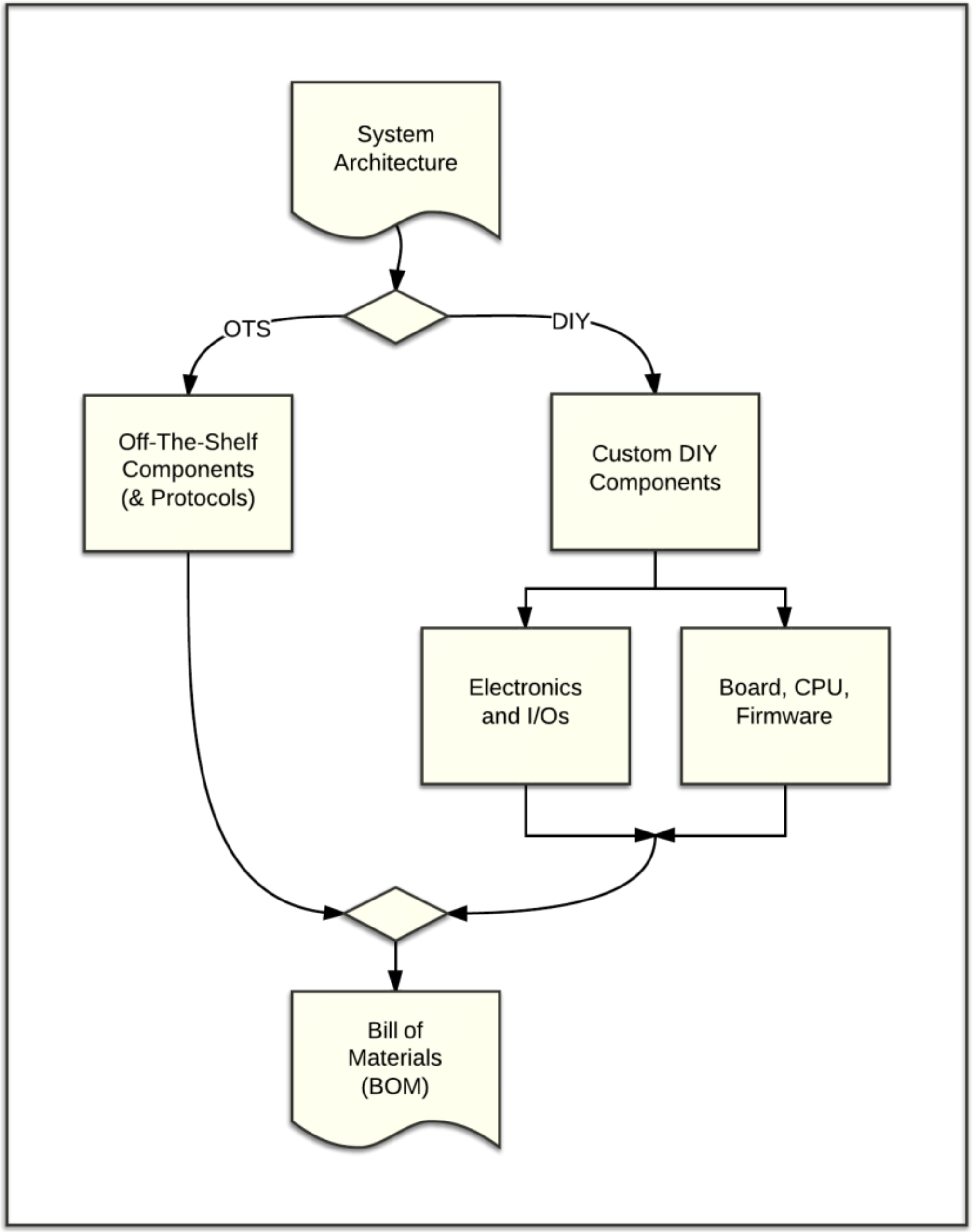
Off-the-shelf

- 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



System Architecture

OTS

DIY

Off-The-Shelf Components (& Protocols)

Custom DIY Components

Electronics and I/Os

Board, CPU, Firmware

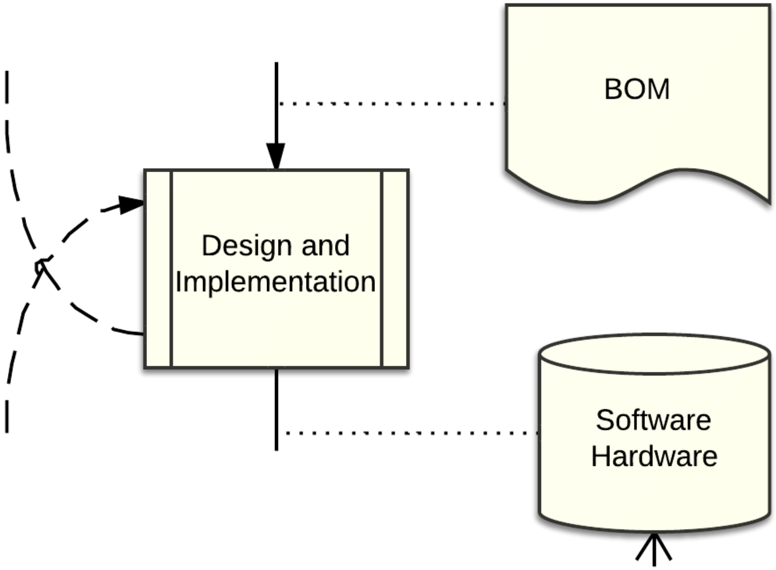
Bill of Materials (BOM)

Deliverable 2

- Before 04/05
- **Features**
- **Architecture**
- We'll provide a checklist
- Upload on the website
  - – Integrate, no separate download
- You'll receive feedback on 07/05

SYSTEMS ENVIRONMENT
SOLSEARCHER ENVIRONMENT
USER ENVIRONMENT
SOLsearcher Solution
DMZ
FIREWALL
FIREWALL
Internet / Intranet
Microsoft IIS Web Server
SQL Server
Document Repository
SOLsearcher Standard Interfaces
Custom Portals or B2C Interfaces
SSE Web Admin

e
aracteristics
nt
ation Constraints
dencies
n)
4. External Inte... rements
4.1 User Interfaces
4.2 Hardware Interfaces
4.3 Software Interfaces
4.4 Communications Interfaces
her Nonfunctional Requirements
Performance Requirements

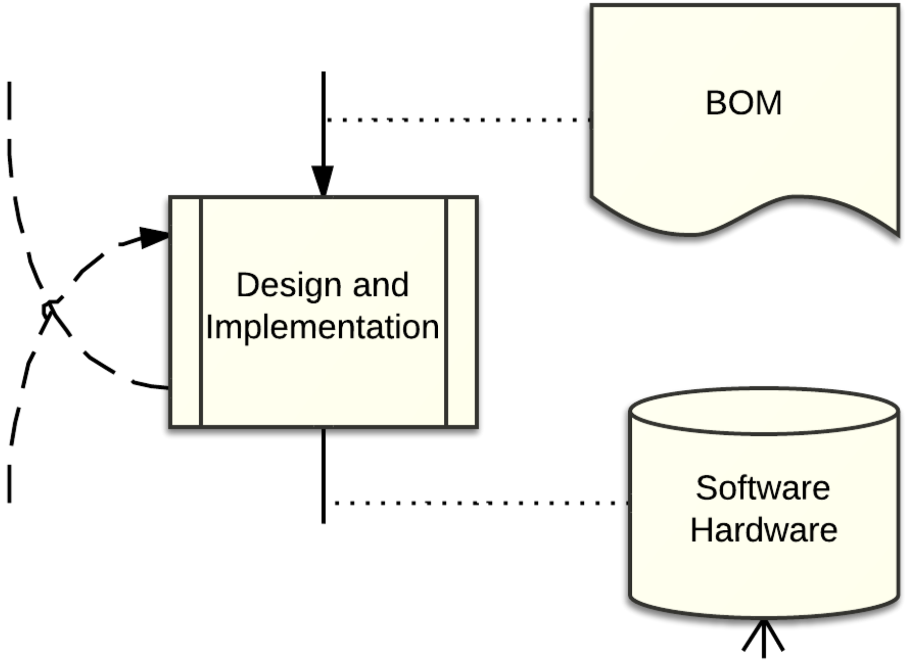


AmI Design Process

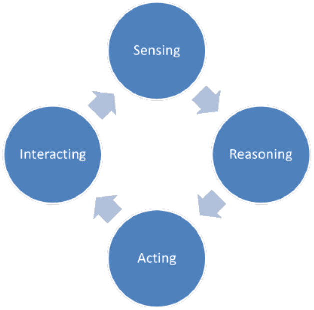
STEP 6: DESIGN & IMPLEMENTATION

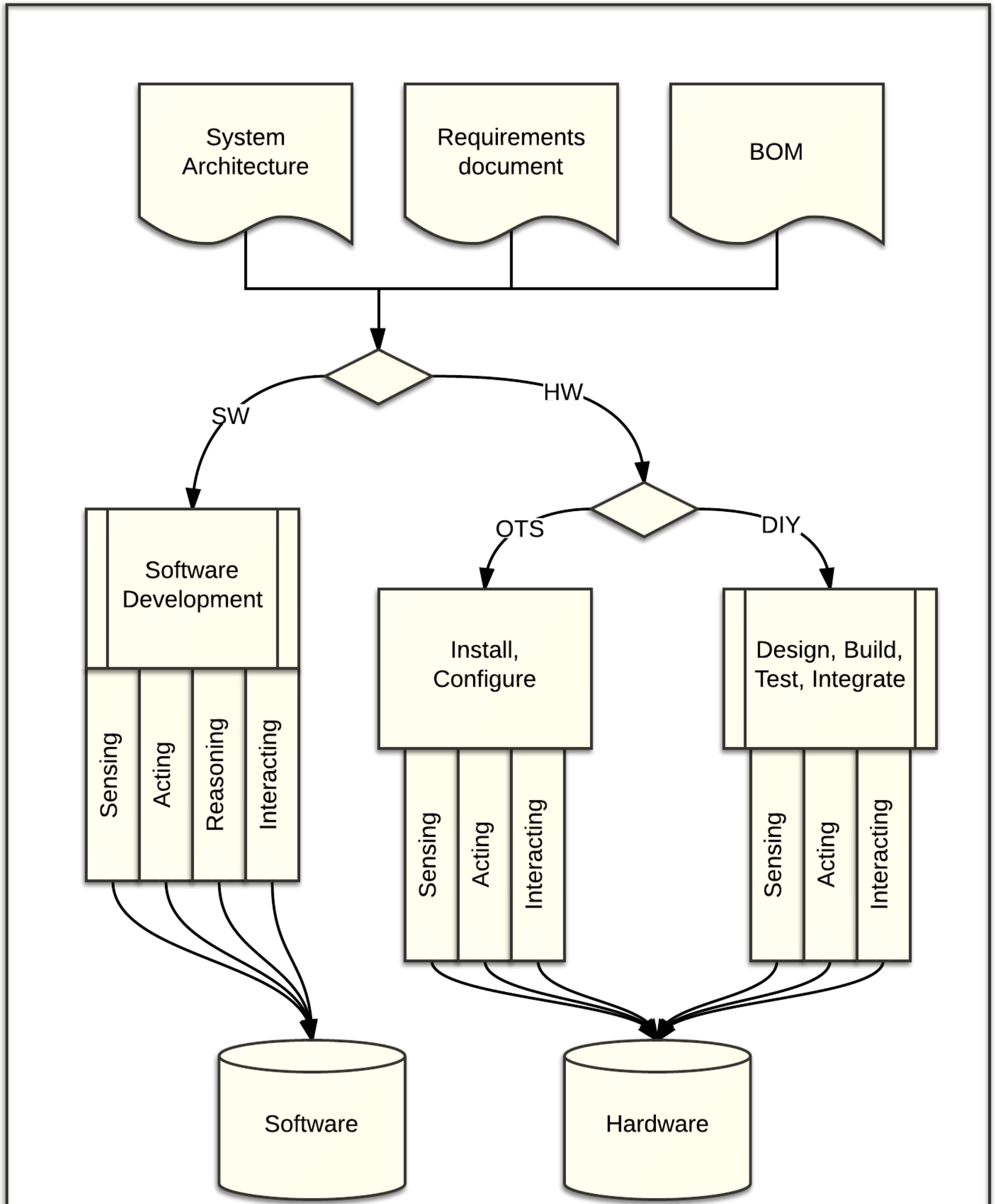
Implementation

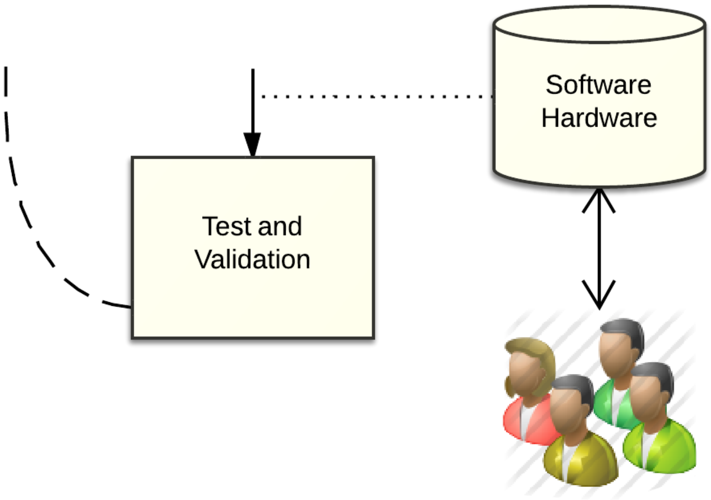
- Realize the HW and SW components defined in the previous steps
  - Implement DIY Hardware
  - Install and/or configure OTS Hardware
  - Develop Software
  - Integrate the SW architecture
- Parallel activities for different disciplines



Design and Implementation





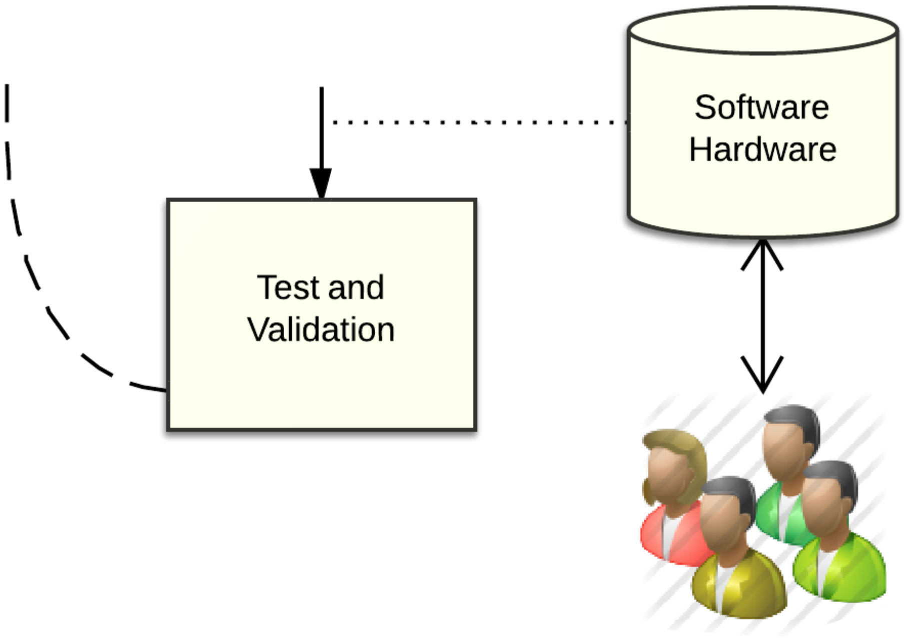


AmI Design Process

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



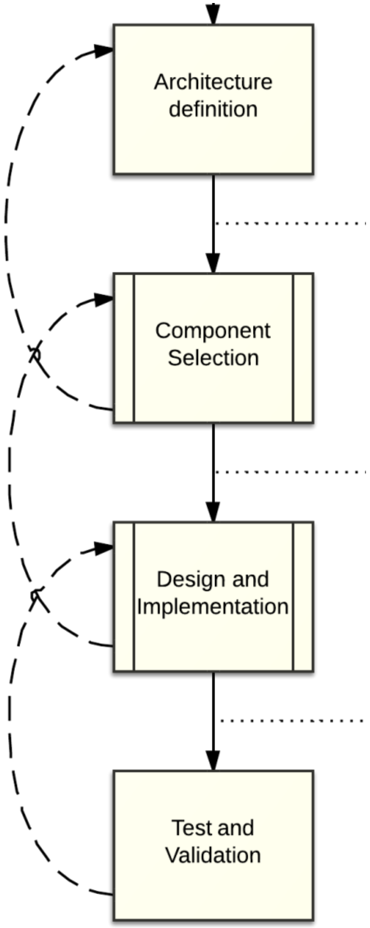
What are we testing?
(aka Verification and Validation)

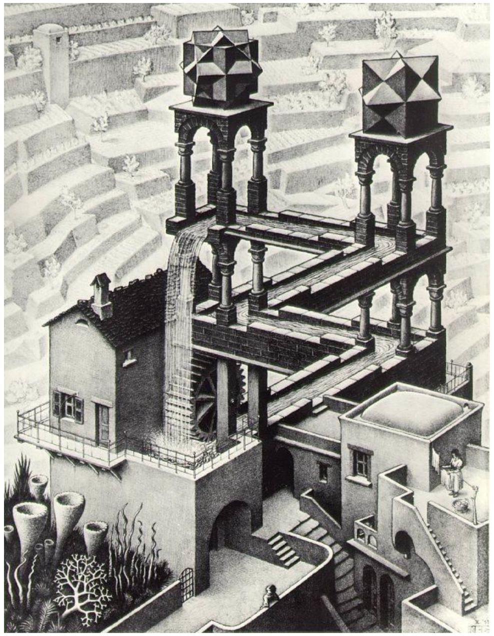
- **Verification** is intended to check that a product, service, or system meets a set of design specifications.
- Test with respect to the Requirements document
- «Am I building the system right?»

- **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





Practical issues

- All deliverable should be submitted through GitHub
  - GitHub project(s) for source code
  - Public project website for deliverable contents
- We provide “templates” for the required contents of the deliverables
- Deliverables will be checked, and we will provide feedback.
  - If you have questions or doubts, you are responsible for asking
- 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

- These slides are distributed under a Creative Commons license "**Attribution – NonCommercial – ShareAlike** (CC BY-NC-SA) 3.0"
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- http://creativecommons.org/licenses/by-nc-sa/3.0/