Aml Design Process

01QZP - Ambient intelligence

Fulvio Corno

Politecnico di Torino, 2018/2019
Design Process

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

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

• General design process
• Main steps of the process
  – Step 1: Problem Statement
  – Step 2: Requirements & Features Elicitation
  – Step 3: Requirements & Features 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 17/03
  – Group composition
  – Summary Description
  – Even >1 proposal
• Do not wait until the last minute
  – May help forming groups
  – We’ll monitor in real time
• Discussion: 18/03
• Final deadline: 24/03

GROUP NUMBER XX
Team Members
• Team member 1, email, GitHub username, role in the project
• Team member 2, email, GitHub username, role in the project
• Team member 3, email, GitHub username, role in the project
• [Team member 4, email, GitHub username, role in the project]

Project Acronym: XXXYYZZZ
Project Title
this is the title
Description
5-10 lines describing the project from the users’ point of view. Don’t mention technologies nor devices.

https://docs.google.com/document/d/1HUxItyx1alU59Bmnjuz1WtKS0rePtmCner8ualfjQu8
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
Still more accurate...

**Projet Management Crash & Burn 101**

Create your own cartoon at www.projectcartoon.com

2018/2019

Ambient intelligence
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 Aml system

![Diagram showing the process of sensing, reasoning, acting, and interacting.]

- Sensing
- Reasoning
- Acting
- Interacting

2018/2019
Ambient intelligence
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

1. Problem Statement
   - Summary system description
   - User and Stakeholder inputs
   - Requirements document

2. Requirements Elicitation

3. Requirements definition

4. Component Selection

5. Architecture definition

6. Design and Implementation

7. Test and Validation

8. System Architecture

9. BOM

10. Software Hardware

2018/2019 Ambient intelligence
Composition of each step

Activity (what to do)

Result (what artifacts we get)

Next Activity (what to do next)

Iteration
Proposed process

Specification

Idea

Problem Statement

Requirements Elicitation

Requirements definition

User and Stakeholder inputs

Requirements document

System Architecture

Architecture definition

Component Selection

Design and Implementation

BOM

Software Hardware

Test and Validation

(Iterative) Development

2018/2019

Ambient intelligence
Simplified process & Deadlines

0. Title & Goal 17/03

1. Vision 31/03

2. Features & architecture 28/04

3. Implementation (Exam)
Aml 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 Aml 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 Aml 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 31/03
• Set-up project web site
• Develop your «Vision»
• Integrate the «Vision» on the website
  – In the website content, not as a separate document
  – Conforming to the available checklist
• You’ll receive feedback on 01/04 (in LADISPE)
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.
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

• 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

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

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

- "We need to get our customers more involved in the product design cycle."
- "We only have customers who are too dumb to check product reviews online."
- "Can it wear a hat like a monkey?"
- "For the millionth time, software can't wear clothes."

WE INTERVIEWED HUNDREDS OF USERS AND TURNED ALL OF THEIR SUGGESTIONS INTO FEATURES.

AS IT TURNS OUT, EVERY USER WE TALKED TO WAS AN IDIOT, AND THEIR DUMB SUGGESTIONS RUINED OUR PRODUCT.

IN HINSDIGHT, WE PROBABLY SHOULD HAVE TALKED TO PEOPLE WHO WORK OUTSIDE THIS BUILDING.

http://dilbert.com/strip/2012-05-07
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

The Design of Everyday Things
Don Norman
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.
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
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

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 28/04
- **Features**
- **Architecture**
- We’ll provide a checklist
- Upload on the website
  - Integrate, no separate download
- You’ll receive feedback on 29/04
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

Central server: data and intelligence

Web interface

Phone

Music

Google Calendar

Ambient Sensors
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
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
Deliverable 2

• Before 28/04
• **Features**
• **Architecture**
• We’ll provide a checklist
• Upload on the website
  – Integrate, no separate download
• You’ll receive feedback on 29/04
Aml 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

System Architecture
Requirements document
BOM

Software Development
Install, Configure
Design, Build, Test, Integrate

Sensing
Acting
Reasoning
Interacting

Sensing
Acting
Interacting

Sensing
Acting
Interacting

Software
Hardware

Ambient intelligence
STEP 7: TEST AND VALIDATION

Aml Design Process
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 step 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_validat ion
• 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.

• [Link to license page](http://creativecommons.org/licenses/by-nc-sa/3.0/)