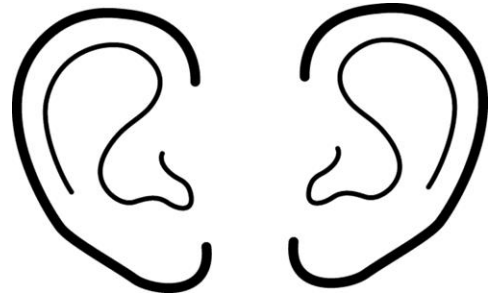
Lecture 8
Communicating the Requirements
EARS


pixy.org

(not in textbook; also work through posted A. Mavin **slides)**

Com S/SE 4090/5090

Robyn Lutz

rlutz@iastate.edu

Office hours:

Robyn: 11 Tues, 230 Atanasoff; 3 Fri (https://iastate.zoom.us/my/rlutz), & by appointment

Carter: cwunsch@iastate.edu,12 Tues (0112 Pearson); 12 Thurs (https://iastate.webex.com/meet/cwunsch), & by appointment

Arushi: arushi17@iastate.edu, 1 Wed (0112 Pearson) & by appointment

HW 1 due today, Thurs, 9/19

HW1 questions?
Q: labeled arrows on context diagram?
A: yes, as in Fig. 3.5, etc.

- Practice 4 skills:
  1) Create a context diagram for the bpMON softwre product (scope it)
  2) Develop its software product use case diagram (partition it)
  3) Identify/elicit missing domain knowledge (know what you need to find out)
  4) Decide group members' responsibilities

- 5090: RE foundational research paper posted: extra HW question for grad students
- Turn in as pdf on Gradescope
  - 4090: one per team, with the names of the team members who worked on it at the top
  - 5090: one per person, with the names of the team members who worked on problems 1-4 at the top AND with your ***individual*** solution to problem 5 appended at the end

Homework 2 (posted later today; due Thurs, 10/3)

- HW#2: practice 5 skills:
  1) Derive the **scenarios** (from the product use cases)
  2) Document the **domain assumption**s (get the context interfaces right, or risk building the wrong product)
  3) Identify the **functional requirements** (what the software has to do: many FRs will come from the steps in scenarios)
  4) Specify the functional requirements in **EARS** (better than text for clear & unambiguous communication)
  5) Document the **nonfunctional requirements** (in English text)
- 509: research paper posted: extra HW question for grad students

Why are requirements so hard to write?

[slides adapted from A. Mavin, *IEEE Software*, 2012; Mavin & Wilkinson, 2010]

1. You have to figure out what's needed, & that's hard.
2. **Today: You have to figure out a clear way to express it**, & that's hard

- Fact: most requirements are written in text (English, etc.)
- Frequent problems: too wordy, ambiguous, vague, hard to understand, partial, premature implementation decisions, untestable
- Projects need clear & concise textual specification of functional requirements
- Technique that works: **EARS: Easy Approach to Requirements Syntax**

Specifying requirements using EARS

- EARS classifies all requirements into **5 basic templates**
- Results from industry show that requirements improve with EARS use
  - Problems/cost/developer misunderstandings are reduced
  - Helps projects build the right product
- We'll use EARS in Homework 2

1. Ubiquitous

- A ubiquitous requirement is something that the system must **always do** (unconditional & continuously active)
- Ex: ”The <system name> shall comply with regulation XXX.”
- Template: The <system name> shall <response>

2. Event-driven

Event-driven keyword: **WHEN**

- System response is initiated by a triggering event the system detects at the system boundary
- Ex: WHEN commanded by the aircraft, the Engine Control System shall dry crank the engine.”
- Template: WHEN <trigger>

3. State-driven

State-driven keyword: **WHILE** [SEP]

- Active while a particular state or states remain true,
- continuous as long as the state holds
- Ex: “WHILE the aircraft is in flight and the engine is running, the Engine Control System shall maintain engine fuel flow above *x* lbs./sec.” [SEP]
- Template: WHILE <in specific state>, the <system name> shall <response>

4. Option

Option keyword: **WHERE**

- A system response is needed only in applications that include a particular feature
- used as a simple way to handle product or system variation, and
- Ex: “WHERE electronic components are used in the Engine Control System, they shall comply with DO-254.”
- Template: WHERE <feature is included>, the <system name> shall <response>

5. Unwanted Behavior

- Unwanted Behavior keyword: **IF/THEN**
- Required system response to unwanted events (such as failures, disturbances, and any unexpected behavior of interacting systems or users)
  - variation of the event-driven requirement
- Ex: “IF the engine fails to start during a third attempt, THEN the Engine Control System shall terminate the autostart sequence.”
- IF <unwanted trigger>, THEN the <system name> shall <response>

Combining these 5 templates

- Can state more complicated requirements by combining templates
- Ex: "**WHILE** the aircraft is on the ground, **WHEN** reverse thrust is commanded, the Engine Control System shall enable deployment of the thrust reverser"
- Ex: "**WHILE** the aircraft is in flight, **IF** reverse thruster is commanded, **THEN** the Engine Control System shall inhibit thrust reverser deployment."

Benefits of EARS & more examples

- Using EARS templates gives you simple, clear statements of the requirements
- EARS is a structured way to write better textual requirements

➢ *Check your understanding*: work through the examples in A. Mavin's EARS tutorial slides, posted on Canvas