Lecture 9

Communicating the Requirements

Communicating with stakeholders

Wikipedia: trawling

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 2 due Thurs, Oct. 3

Homework 2 (due 10/3)

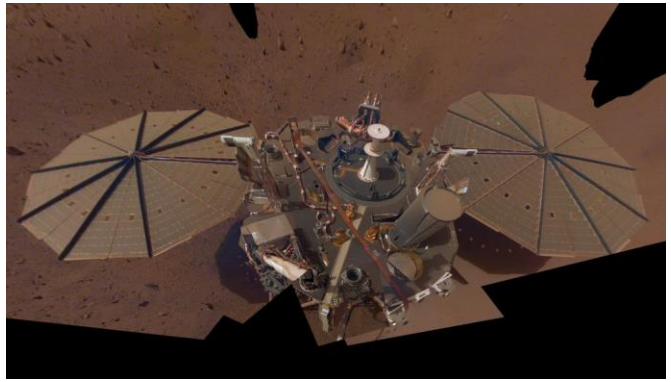
- HW#2: practice 5 skills:
  1) Derive the **scenarios** (from the product use cases)
  2) Document the **domain assumption**s (get the context interfaces right, or risk building the wrong product)
  3) Identify the **functional requirements** (what the software has to do: many FRs will come from the 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)
- 5090: research paper posted: extra HW question for grad students

Reading Assignment

Chapter 5, “Investigating the Work”

Chapter 7, “Understanding the Real Problem”

Thinking creatively



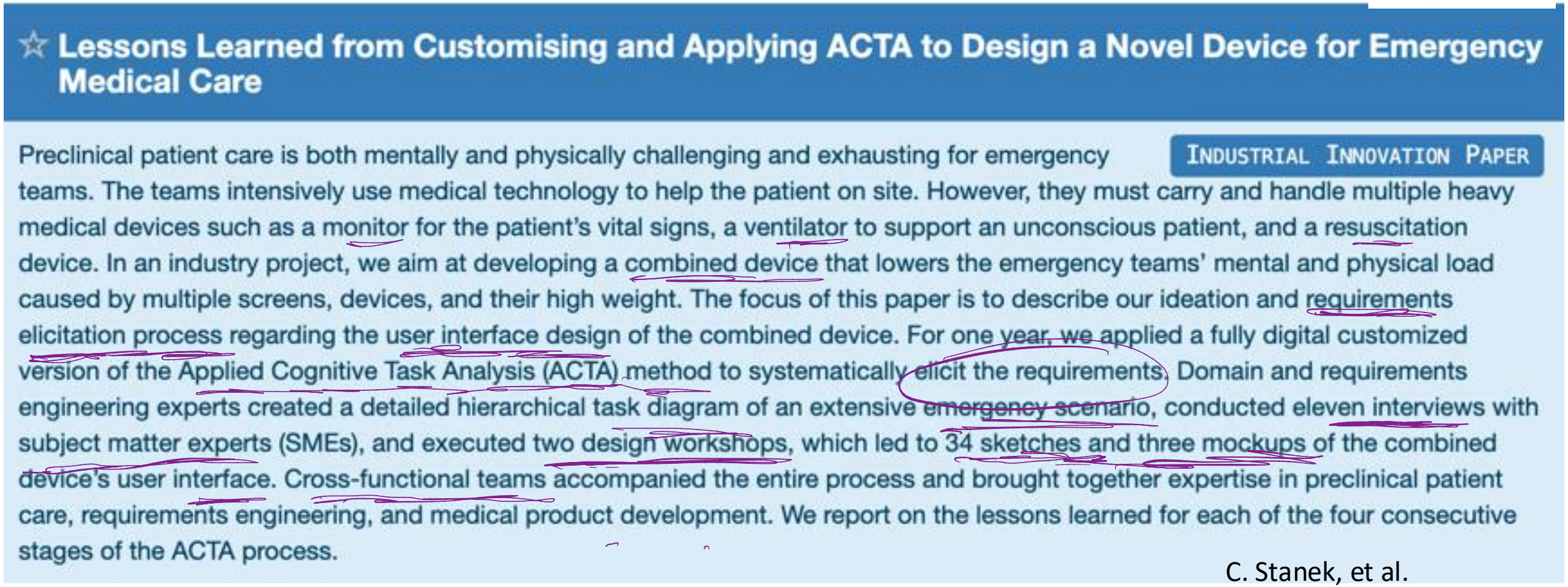
Goal: detect marsquakes

Problem: dust buildup on InSight's solar panels reduced the lander's power levels, requiring the mission to conserve energy by temporarily turning off certain instruments, including the seismometer.

Creative solution: Used InSight's robotic arm to trickle sand near one solar panel in the hopes that, as wind gusts carried it across the panel, the granules would sweep off some of the dust. The plan worked. The seismometer could stay on.

Impact: allowed scientists to study the three biggest quakes they've seen on Mars, including one of the biggest. (4.2), longest-lasting (~1.5 hrs) marsquakes the mission has ever detected. [jpl.nasa.gov]

Example: RE discovery & communication advance



☆ **Lessons Learned from Customising and Applying ACTA to Design a Novel Device for Emergency Medical Care**

INDUSTRIAL INNOVATION PAPER

Preclinical patient care is both mentally and physically challenging and exhausting for emergency teams. The teams intensively use medical technology to help the patient on site. However, they must carry and handle multiple heavy medical devices such as a monitor for the patient's vital signs, a ventilator to support an unconscious patient, and a resuscitation device. In an industry project, we aim at developing a combined device that lowers the emergency teams' mental and physical load caused by multiple screens, devices, and their high weight. The focus of this paper is to describe our ideation and requirements elicitation process regarding the user interface design of the combined device. For one year, we applied a fully digital customized version of the Applied Cognitive Task Analysis (ACTA) method to systematically elicit the requirements. Domain and requirements engineering experts created a detailed hierarchical task diagram of an extensive emergency scenario, conducted eleven interviews with subject matter experts (SMEs), and executed two design workshops, which led to 34 sketches and three mockups of the combined device's user interface. Cross-functional teams accompanied the entire process and brought together expertise in preclinical patient care, requirements engineering, and medical product development. We report on the lessons learned for each of the four consecutive stages of the ACTA process.

C. Stanek, et al.

- RE's job: choose the best trawling technique(s) for the project

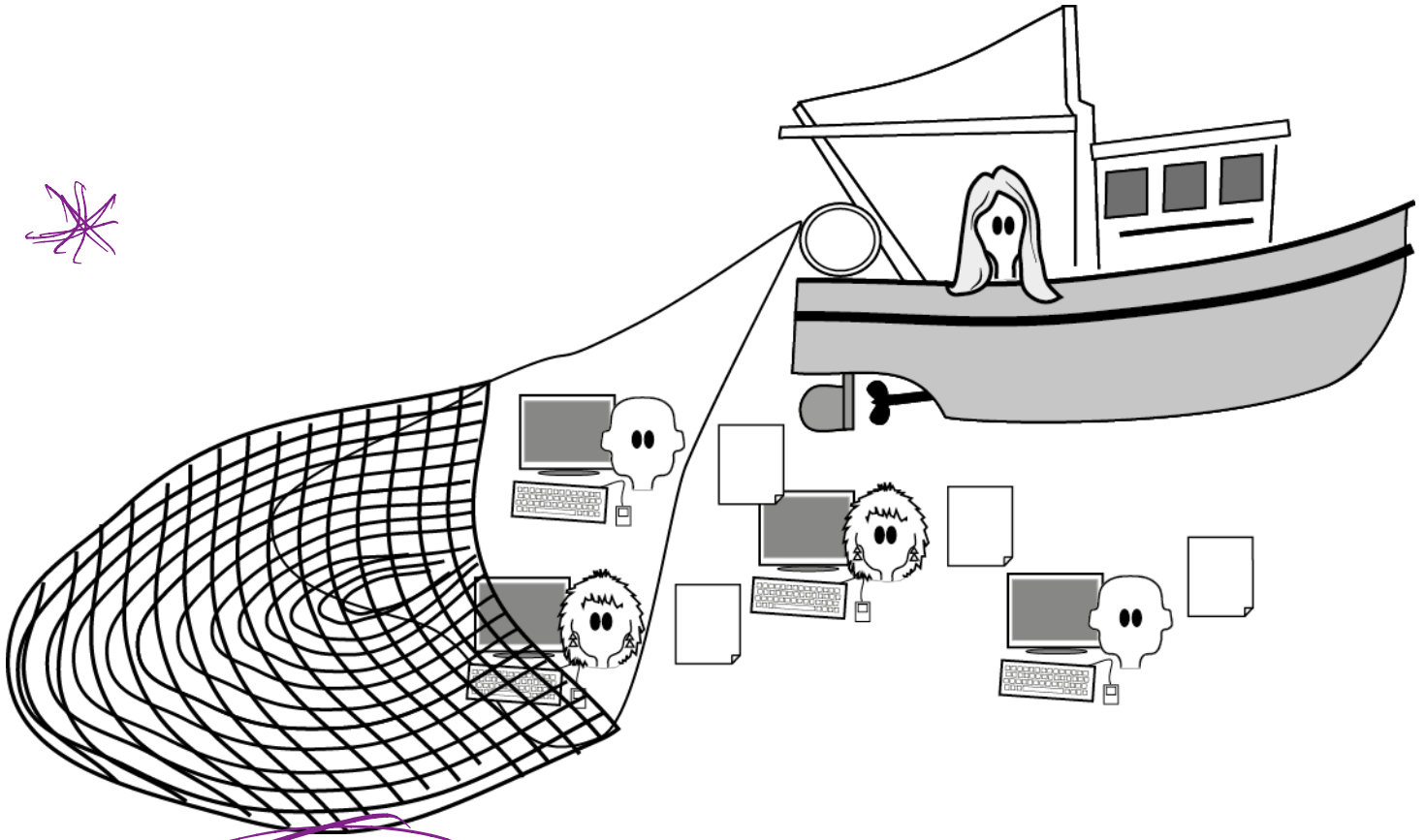


Figure 5.1
The business analyst trawls for knowledge by investigating the client's work. The analogy of running a net through the organization is appropriate: You need to sift through much of the business before you can find the best way to improve it.

Choice depends on:

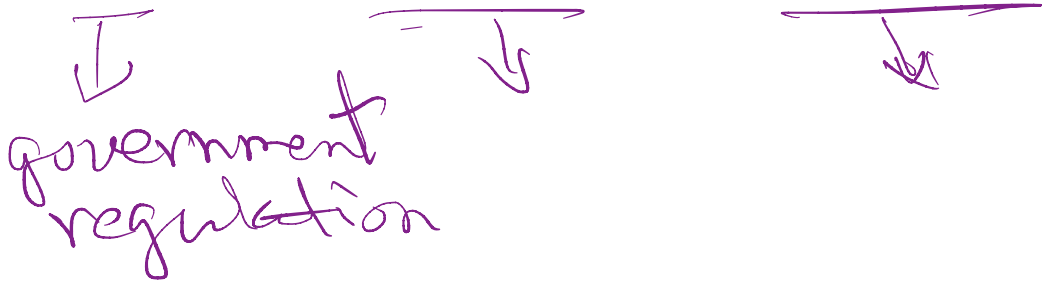
- the project,
- the stakeholders,
- if geographically distributed,
- how creative or constrained
- application domain

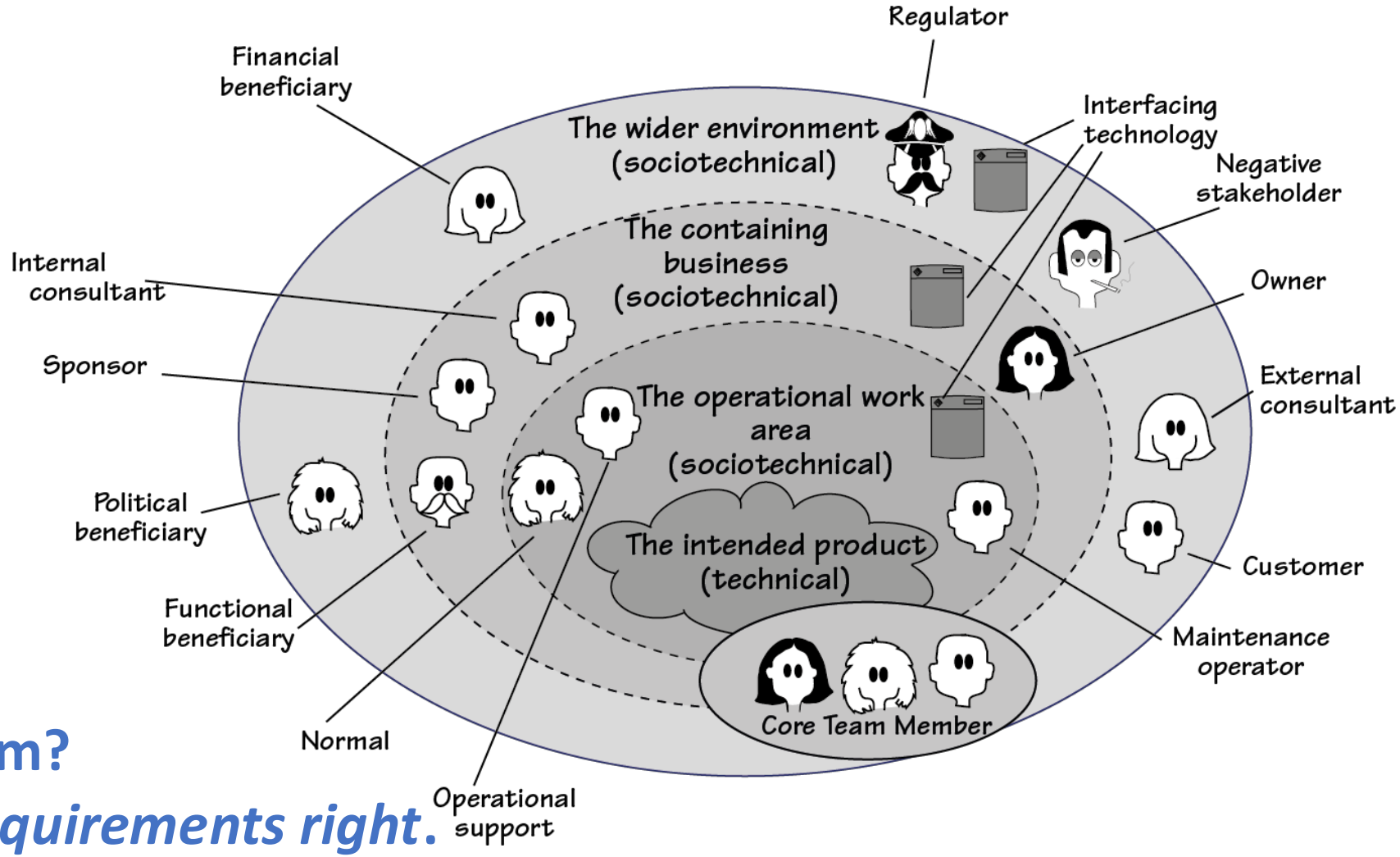
NOW

FUTURE

Who are stakeholders? (Chapters 5 & 7)

- "Any person who has an interest in the product & who therefore has requirements for it" [Glossary, back of Robertson & Robertson]
  - client
  - user
  - developer
  - some are remote: auditor, safety inspector, company lawyer





Why ask them?

To *get the requirements right*.

Figure 3.7

This stakeholder map shows the organizational rings surrounding the eventual product, and the classes of stakeholders who inhabit these rings. Use this map to help determine which classes of stakeholders are relevant to your project and which roles you need to represent them.

How can we successfully communicate with stakeholders?

- Active investigation: probe for unconscious requirements and missing information
- Translate between stakeholders and developers
- Make models to record & check shared understanding
  - Context diagram, use cases, scenarios
- Apprentice
  - Observe process & patterns
- Social media/online forums/channels + natural language processing (NLP)
- Surveys/questionnaires

Video of use-case workshop for stakeholders

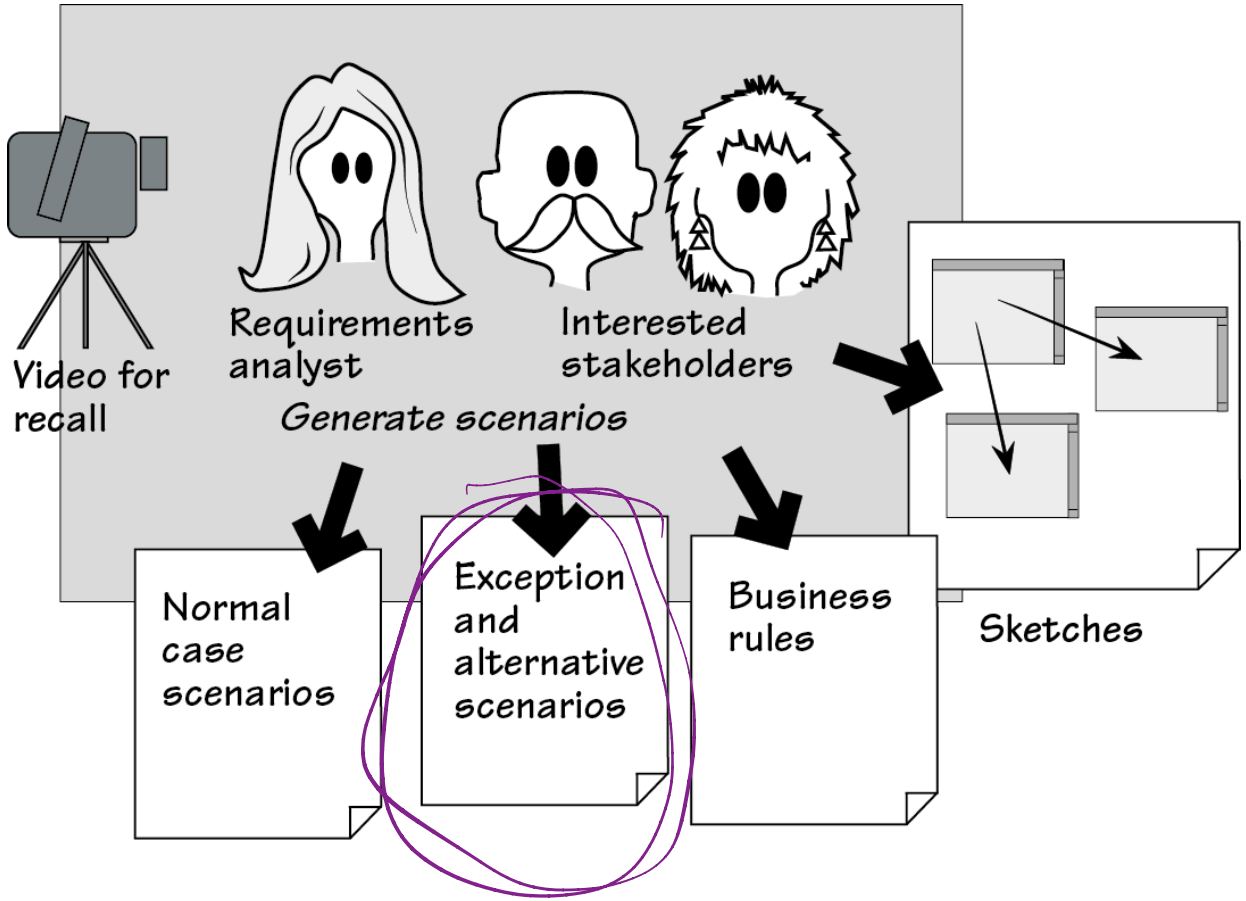


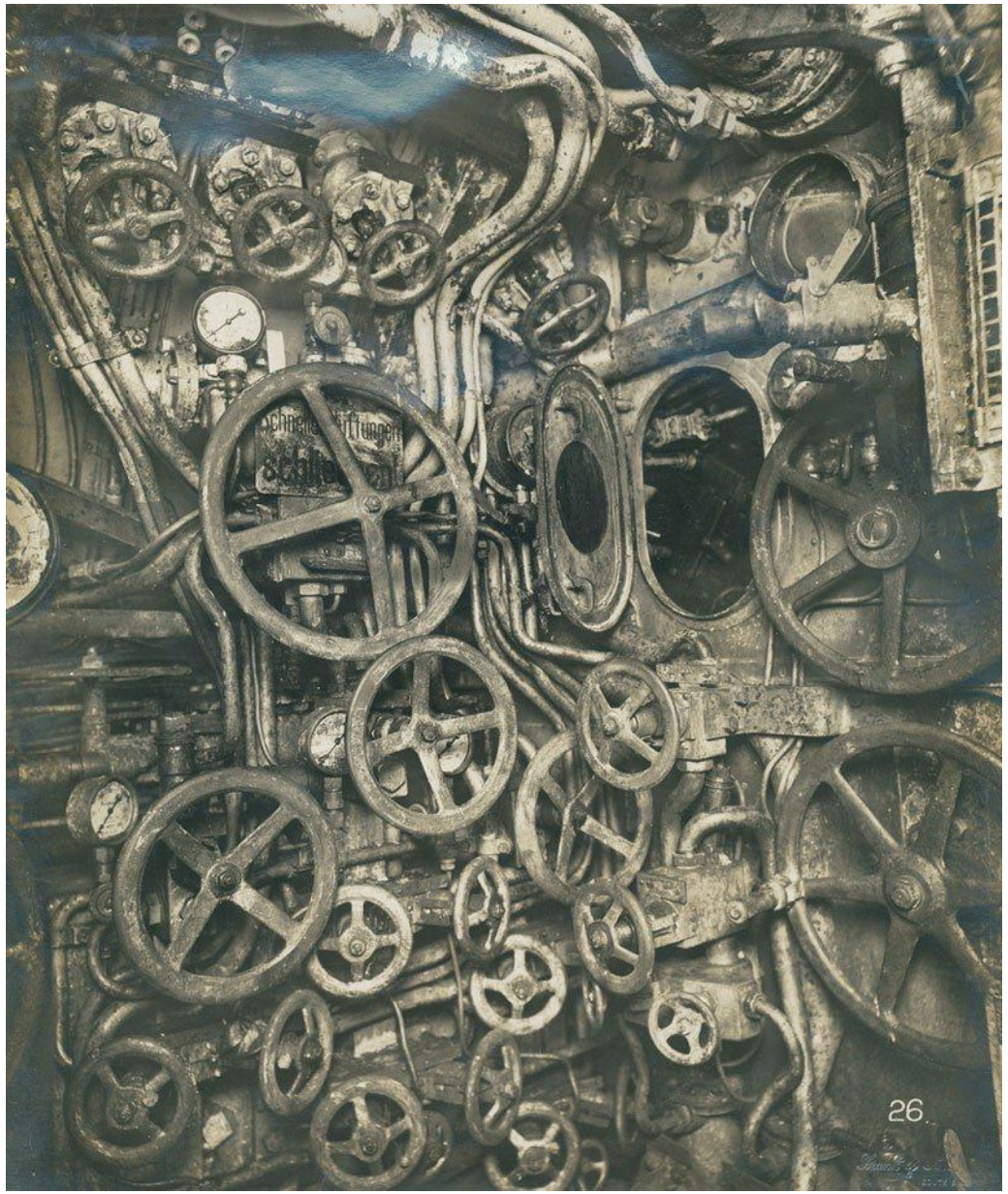
Figure 5.9

The business use case workshop records the proposed functionality using scenarios and sketched prototypes. The workshop serves as a forum for interested stakeholders to communicate effectively, express their understanding, ask questions, and give their aspirations for the work.

Figure 7.9

The team used a persona to represent their customers. By treating the persona as a real person, they were able to devise a far superior Future-What version of their work.

Persona



User interface: consider persona's perspective

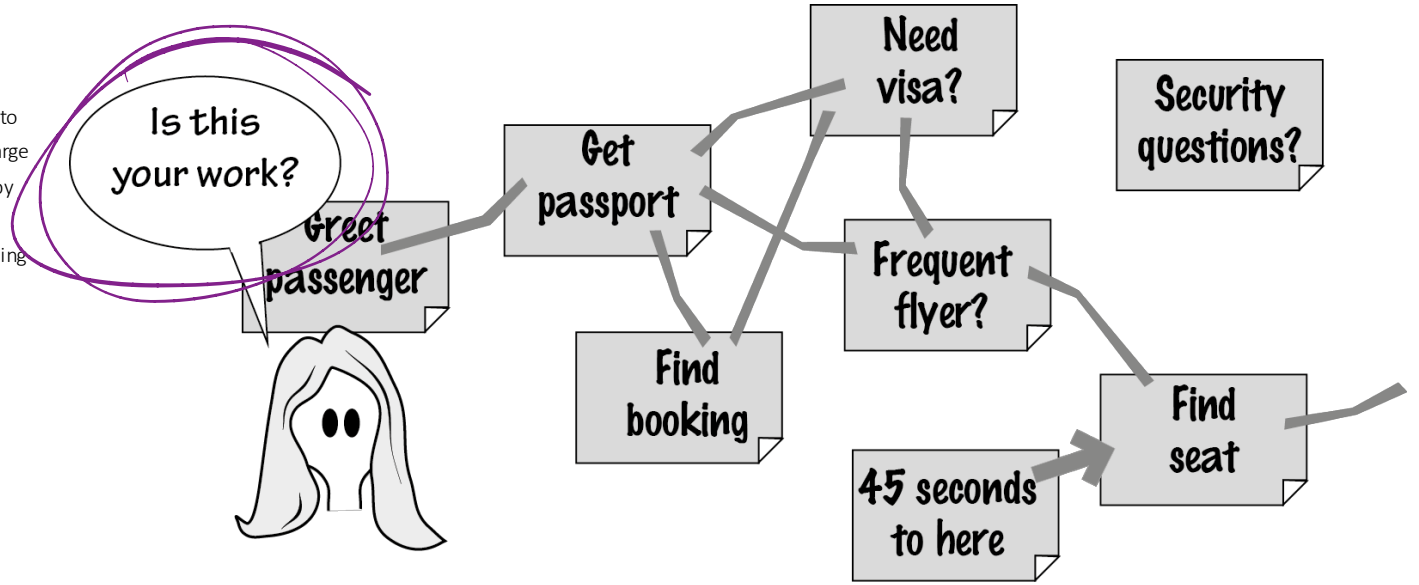
Control-room-of-the-UB-110-German-submarine-ca.-1918
https://rarehistoricalphotos.com/u-boat-control-room-1918/

Interviewing stakeholders

- Ask (using their terminology)/Listen/Feedback your understanding/Get agreement
  - What is the work the user does & what is the work they hope to do?
  - What is the underlying business reason for the new product?
  - What is the scope of responsibility of the software?
  - What are the interactions between the product & the outside world?
- Stakeholders often can communicate what the use cases are
- ``Thank the stakeholders for their time & tell them what you've learned & why it's valuable."
- Iterate until shared agreement

Figure 5.10

With quick and dirty process modeling, the business analyst uses Post-it notes to build an informal model of the work. Large notes are stuck on the wall and linked by masking tape. Naturally, interested stakeholders are partners in this modeling activity.



- Ask (using their terminology)
- Listen
- Feed back your understanding (using simple, visual models)
- Get agreement

Figure 7.5

Having achieved the correct understanding of the work as it is now, we move on to look at what it can be when we finish the project.

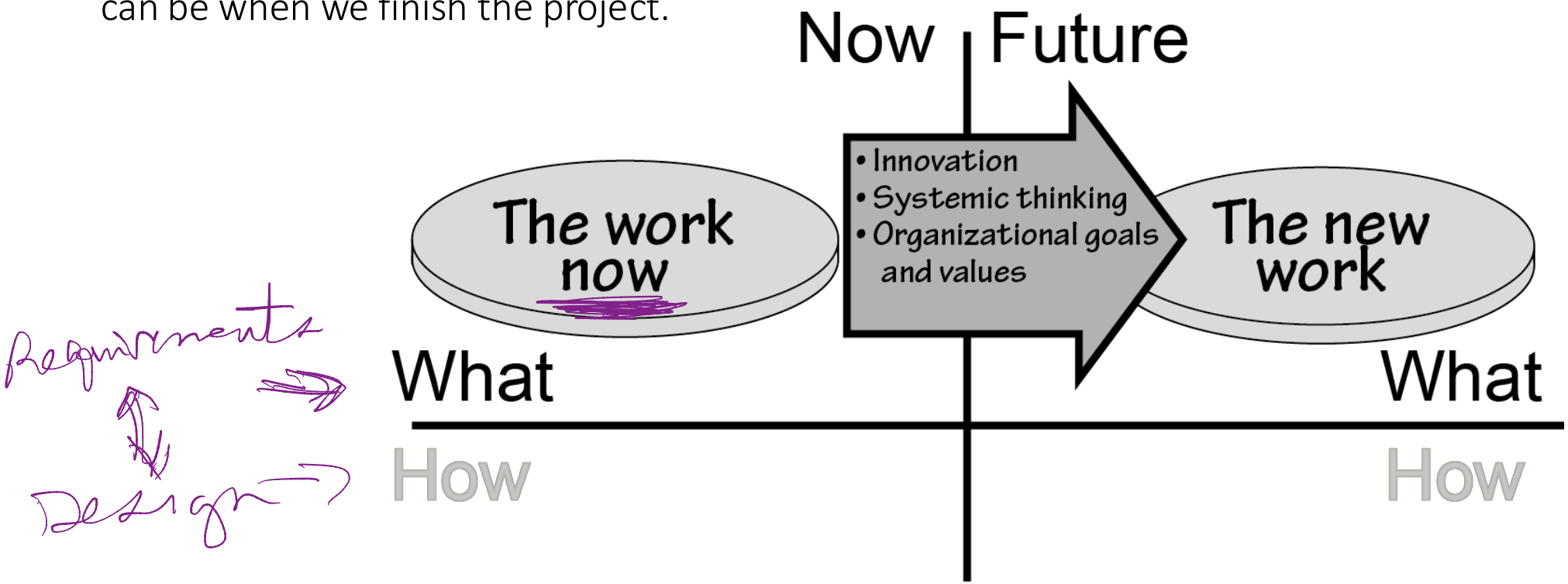
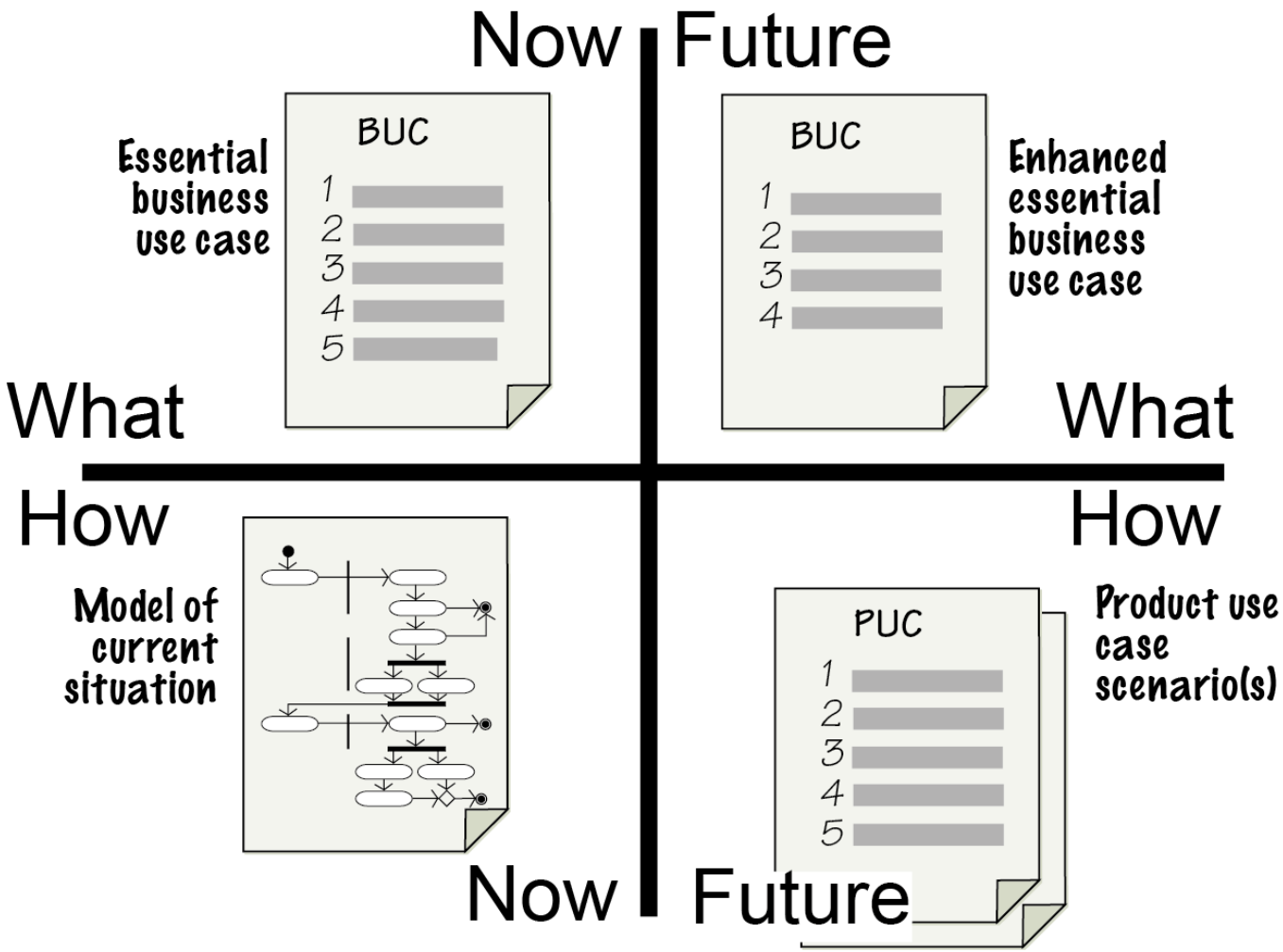


Figure 5.4

The Brown Cow Model. This shows four views of the work, each of which provides the business analyst and the stakeholders with information that is useful at different stages of the requirements discovery process.

"Brown Cow" model



Risks

- Stakeholder may propose solution:
  - may not match actual problem
  - may be premature buy-in to a specific technical design
- Stakeholder's description may not be accurate
- Stakeholders' requirements may well conflict with each other (we'll handle this later)
- Dead Fish: stakeholders know from the beginning that project is doomed & yet no one says it
  - make fact-based GO/NO GO decision

➢Check your understanding: know trawling techniques in **Table 5.1**

☆ Most Influential Paper: Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development

Elizabeth Bjarnason
Krzysztof Wnuk
Björn Regnell

Communication is essential for software development as its efficiency throughout the entire project life-cycle is a key factor in developing and releasing successful software products to the market. This paper reports on findings from an explanatory case study aiming at a deeper understanding of the causes and effects of communication gaps in a large-scale industrial set up. Based on an assumption of what causes gaps in communication of requirements and what effects such gaps have, a semi-structured interview study was performed with nine practitioners at a large market-driven software company. We found four main factors that affect the requirements communication, namely scale, temporal aspects, common views and decision structures. The results also show that communication gaps lead to failure to meet the customers' expectations, quality issues and wasted effort. An increased awareness of these factors is a help in identifying what to address to achieve a more efficient requirements management, and ultimately more efficient and successful software development. By closing the communication gaps the requirements may continue all the way through the project life-cycle and be more likely to result in software that meets the customers' expectations.

Causes of communication gaps

[Bjarnason et al. 21]

1. Scale: complex product & large organization ⇒ Tool support
   - large (5,000 employees) company developing embedded systems for international market
   - 40-80 developers per project team
   - complex and large requirements database (20,000 at various levels of abstraction)
2. Missing common views:
   - Missing tacit requirements due to lack of understanding of customer's domain
   - Not communicating due to weak understanding of impact on other units' work
3. Temporal aspects: hand-over points among units — Breakdown of communication.
   - Insufficient knowledge of requirements passed along
   - Development detaches from requirements, thus some are misunderstood
4. Decision structures: no clear common goal
   - Software Development Unit controls which requirements get implemented
   - Modifies without consulting clients' needs or understanding of domain/context

Effects of communication gaps

[Bjarnason et al. 21]

(Excerpts)

Communication gaps contribute to a number of consequences for the project and for the resulting software. The communication gaps during requirements definition contribute to an unstable, unclear and ambiguous SRS.

Software Requirements Specification

All this results in not always meeting the customers' expectations either due to lack of desired functionality or quality issues. In addition, effort is wasted.

Ex: "Advancing the Ability of Robots to Help" (CACM 9/22) Interview with roboticist Prof. Howard

Credit: Ayanna Howard

"You've made the point that designing educational tools for kids with special needs is actually a good way of designing educational tools for all kids. Can you elaborate on that?

A lot of times, even at the college level, instructors teach things based on the way that they learn. So, if I like to write and I learned by reading a lot, then I'm going to give my students a lot of reading assignments. But students have very different ways of consuming and processing information. People know that about children with special needs, but I don't think they realize that children have these nuances across the board. So when you design for what I call the extremes, you also incorporate the different learning styles of children who don't necessarily fit into the box that the teacher is teaching from."

perspective

"How does that philosophy work when you're designing outside of an educational context?

I encourage people to think about who their opposite is in terms of attributes, and design for that person. If I'm a technologist living in a high-SES neighborhood, then I need to think about designing solutions for someone who is in, say, rural America. Then what happens is, even though that's not my lived experience, it makes me sit back and start rethinking my design choices. It doesn't get you to the other extreme, but it does break the habit of designing based on what you know, and it makes you explore other things you might otherwise not have."

Ex: "Advancing the Ability of Robots to Help" (CACM 9/22) Interview with roboticist Prof. Howard

"I encourage people to think about who their opposite is in terms of attributes, and design for that person. . . . it makes me sit back and start rethinking my design choices. . . . it does break the habit of designing based on what you know, and it makes you explore other things you might otherwise not have."

- What requirements-discovery approaches can help?

Table 5.1

Ex: “Advancing the Ability of Robots to Help” (CACM 9/22)
Interview with roboticist Prof. Howard

Let's talk about your research into overtrust. Can you summarize the problem and share some of your recent findings about using explainable AI methods to counteract it?
Prior research from my group and others has shown that when you're using robots and AI agents, and they are dependable, you start believing that they are always dependable, and you won't even second-guess yourself after a while. One thing we're looking at is explaining to individuals when the system itself is uncertain in a way that just makes people better reflect on their trust in the decisions being thrown at them by these agents.

In other words, more of a contextual prompt, rather than a broad disclaimer.
Right. We've started to examine this approach for use primarily in the self-driving domain. We have all this data about dangerous intersections—two highways that merge, for example, where the accident rate is higher. You can take that information and decode it. Let's say you're in your car, on autonomous mode. It is 3 P.M., and you're about to enter an intersection that is hazardous. Now, 3 P.M. is when the kids are out. So maybe the car says, "Hey, school kids are on the road and a child died last week."

What we're finding is that people make better decisions when risk is quantified in terms that are more personal. They either pay closer attention to the road, for example, or they'll do a full override of the autonomous system to get through that intersection.

Ex: “Advancing the Ability of Robots to Help” (CACM 9/22)
Interview with roboticist Prof. Howard

Let's talk about your research into overtrust. Can you summarize the problem and share some of your recent findings about using explainable AI methods to counteract it?
Prior research from my group and others has shown that when you're using robots and AI agents, and they are dependable, you start believing that they are always dependable, and you won't even second-guess yourself after a while. One thing we're looking at is explaining to individuals when the system itself is uncertain in a way that just makes people better reflect on their trust in the decisions being thrown at them by these agents.

In other words, more of a contextual prompt, rather than a broad disclaimer.
What we're finding is that people make better decisions *when risk is quantified in terms that are more personal*. They either pay closer attention to the road, for example, or they'll do a full override of the autonomous system to get through that intersection.

- Choose an AI-enabled system
- How to discover effective *requirements* to counteract user overtrust?