THE UNIVERSITY OF WESTERN AUSTRALIA

Requirements Elicitation

Software Requirements and Design
CITS4401
Lecture 4

The 4 Major Activities of Requirements Engineering

- **Elicitation**
- Analysis
- Specification
- Validation

Lecture Overview

- What is Requirements Elicitation
- Requirements Sources
- Elicitation Techniques

- Reference: SWEBOK 3.0 Chapter 1 Section 3

1. What is Reqs Elicitation?

- Process
- Essence
- Communication
- Scope

The Essence

"At the core of any requirements process is the ability to get people to tell you what they really need, rather than their perceived solution, or what they think you might be able to deliver."

S & J Roberson, Mastering the Requirements Process, volere.org

- https://www.volere.org/wp-content/uploads/2019/02/howNowBrownCow.pdf
- https://www.volere.org/the-brown-cow-model/. (7 min video)
- See Workshop 1

What is Requirements Elicitation?

The *process* through which the

customers, buyers, users, regulators and any others who are ***stakeholders*** in the development of a software system

discover, reveal, articulate and understand their requirements.

Requirements elicitation is concerned with the origins of software requirements and how the software engineer can collect them.

Also known as requirements capture, requirements discovery, requirements acquisition and requirement gathering

Requirements vs Specifications

"Requirements is probably the most misused word in our industry."

"Required means nonnegotiable, yet in almost every project we see changed, bartered, and negotiated requirements"

"I propose using the word "specification" instead. Specifications are changeable and are understood as the current state of our understanding."

W.Royce, IEEE Software, Sep/Oct 2005

Get a copy from http://ieeexplore.ieee.org/ via UWA

Communication

- Elicitation is fundamentally a **human activity** and is where the stakeholders are identified and relationships established between the development team and the custom

- One of the fundamental principles of a good requirements elicitation process is that of **effective communication** between the various stakeholders.

- Requirements specialists **mediate between the domain** of the software users (and other stakeholders) and the technical world of the software engineer.

Project Scope

- A critical element of requirements elicitation is informing the **project scope**.
- This involves providing a **description of the software** to be built and its purpose and
- **Prioritizing** the deliverables to ensure the customer's most important business needs are satisfied first.
- Note that project scope concerns both
  what: is to be **produced** and
  how: the **time and other resources available** to do this

Software Scope

The first activity for software project management is determining **software scope.** Scope is determined by answering the following questions:

- **Context:** How does the SW to be built fit into a larger system, product or business context? What constraints are imposed as a result of the context?

- **Information objectives:** What customer-visible data objects are produced as output from the SW? What data objects are required for input?

- **Function and performance:** What function does the DW perform to transform input data to output? Are any special performance characteristics to be addressed?

[Pressman 24.3 Managing SW Projects]

Reqs Elicitation Challenges

- Articulation Problems
  - People don’t know what they don’t know so they can’t tell you uabout it!
- Communication Barriers
  - Different “languages” in different domains
- Knowledge and Cognitive Limitations
  - Managing complexity
- Human Behaviour Issues
  - Conflicting needs
- Technical Issues
  - Change management
  - Too rigid adherence to methodology

2. Requirements Sources

- Goals
- Domain knowledge
- Stakeholders
- Business Rules
- Operational Environment
- Organisational environment

Sources: Goals

- Goals. The term “goal” (sometimes called “business concern” or “critical success factor”) refers to the overall, high-level objectives of the software.
- Goals provide the motivation for the software but are often vaguely formulated.
- Software engineers need to pay particular attention to assessing the value (relative to priority) and cost of goals.
- A feasibility study is a relatively low-cost way of doing this.

What is a goal?

- An objective that the system under consideration should achieve
- Goals are ***optative*** (express a wish) rather than ***indicative*** (stating a thing as fact), e.g.,
  - "serve more passengers"
  - "acceleration command is delivered on time"
- Notice the different levels of abstraction

Sources: Domain knowledge.

- The software engineer needs to acquire or have available knowledge about the application domain.
- Domain knowledge provides the background against which all elicited requirements knowledge must be set in order to understand it.
- A knowledge domain *ontology* can be used to capture this knowledge
- Ontology is a “formal, explicit specification of a shared conceptualization” (Gruber, 1993).
- Relations between relevant concepts within the application domain should be identified.

- See UWA’s text to knowledge graph tool (winner of ICDM’19 competition). http://agent.csse.uwa.edu.au/text2kg/

Sources: Stakeholders

- Much software has proved unsatisfactory because it has stressed the requirements of one group of stakeholders at the expense of others.
- Hence, the delivered software is difficult to use, or subverts the cultural or political structures of the customer organization.
- The software engineer needs to identify, represent, and manage the “viewpoints” of many different types of stakeholders.

- Requirements negotiation will be discussed later in the unit

Sources: Business Rules

- These are statements that define or constrain some aspect of the structure or the behavior of the business itself.
- “A student cannot register in next semester’s courses if there remain some unpaid tuition fees” would be an example of a business rule for a university’s course-registration software.

Sources: operational environment

- Requirements will be derived from the environment in which the software will be executed.
- These may be, for example, timing constraints in real-time software or performance constraints in a business environment.
- These must be sought out actively because they can greatly affect software feasibility and cost as well as restrict design choices.

Src: organizational environment

- Software is often required to support a business process, the selection of which may be conditioned by the structure, culture, and internal politics of the organization.
- The software engineer needs to be sensitive to these since, in general, new software should not force unplanned change on the business process.

Approaches & Techniques

- Asking
- Observing and Inferring
- Discussing and Formulating
- Negotiating with respect to a standard set
- Studying and Identifying Problems
- Discovering through creative processes
- Postulating

- Interviews
- Scenarios
- Prototypes
- Facilitated meetings
- Observation
- User stories
- And others (the above list is not exhaustive)

3. Methods

1. Interviews
2. Facilitated meetings
3. Prototypes
4. Observation

- *Scenarios*
- *User stories*

1. Interviews

- Interviewing stakeholders is a “traditional” means of eliciting requirements.
- Important to understand the advantages and limitations of interviews and how they should be conducted.

Interviews

- An **interview** is a systematic attempt to collect information from a person.
- Interviewing success depends on ability to identify:
  - work flows,
  - factors that influence the operations of systems, and
  - the elements (documents, procedures, policies, etc.) that make up systems.
- Poorly performed interviews may:
  - lead to systems which do not meet the needs of the organization
  - affect the attitudes of the users and have a negative effect on the entire project effort

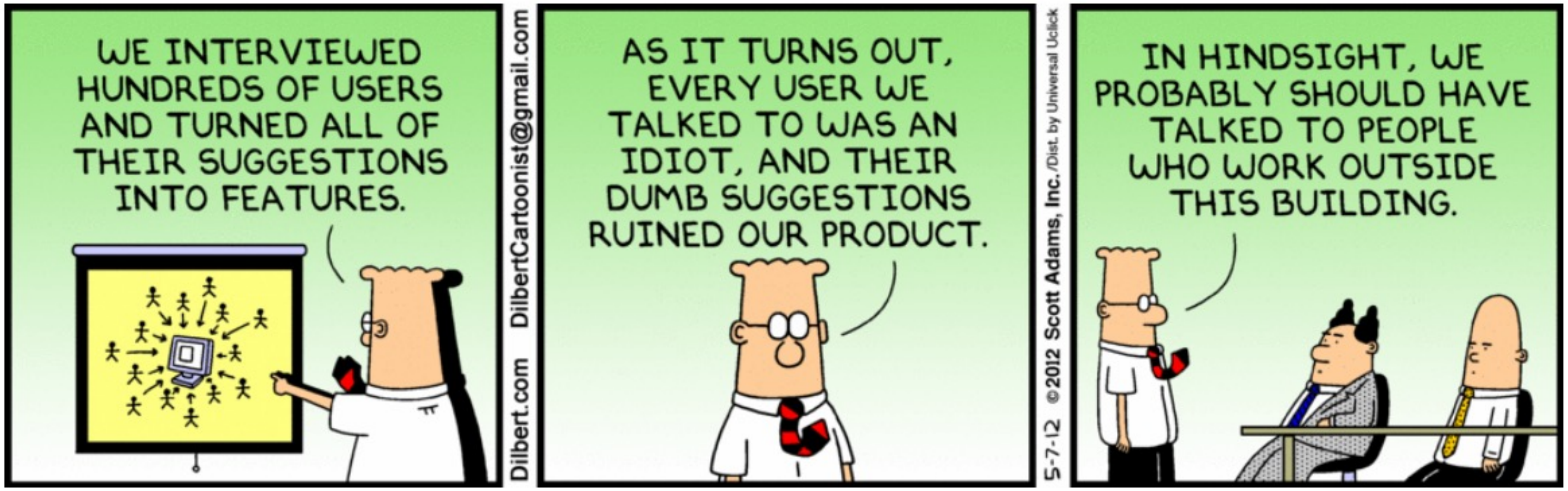
5 Steps of the Interview Process

1. Preparing for the interview
2. Planning and scheduling the interview
3. Opening and closing the interview
4. Conducting the interview
5. Following up for clarification

Problem: Resistance may be found

I'LL DESIGN THE SYSTEM AS SOON AS YOU GIVE ME THE USER REQUIREMENTS.
BETTER YET, YOU COULD BUILD THE SYSTEM. THEN I'LL TELL YOUR BOSS THAT IT DOESN'T MEET MY NEEDS.
I DON'T MEAN TO FRIGHTEN YOU, BUT YOU'LL HAVE TO DO SOME ACTUAL WORK.
THAT'S CRAZY TALK.
www.dilbert.com scottadams@aol.com
© 2003 United Feature Syndicate, Inc.

Problem: Conflicting requirements


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 HINDSIGHT, WE PROBABLY SHOULD HAVE TALKED TO PEOPLE WHO WORK OUTSIDE THIS BUILDING.
Dilbert.com DilbertCartoonist@gmail.com
5-7-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick

2. Facilitated meetings

- Purpose: try to achieve a summative effect, whereby a group of people can bring more insight into their software requirements than by working individually.
- Advantage: Brainstorm and refine ideas that may be difficult to bring to the surface using interviews.
- Advantage: conflicting requirements surface early on in a way that lets the stakeholders recognize where these occur.
- Advantages: may result in a richer and more consistent set of requirements than might otherwise be achievable.
- Disadvantage: meetings need to be handled carefully (hence the need for a facilitator) to avoid poor group dynamics
- Disadvantage: Meetings are time consuming (hence the need for a facilitator)

Brainstorming

- A simple group technique for generating ideas
- Allows people to suggest and explore ideas in an atmosphere free of criticism or judgement
- Works best with 4 to 10 people – 1 person to get the session started, but not constrain it

Brainstorming

- *Generation Phase:* participants encouraged to offer as many ideas as possible without discussion of the merits of ideas
- *Consolidation Phase:* ideas are discussed, revised and organised

Brainstorming (cont.)

LET'S BRAINSTORM NEW PRODUCT IDEAS.
DilbertCartoonist@gmail.com
REMEMBER, THE MOST IMPORTANT RULE OF BRAINSTORMING IS *NO CRITICIZING.*
I'LL GO FIRST.
© 2012 Scott Adams, Inc./Dist. by Universal Uclick
RESEARCH SHOWS THAT BRAINSTORMING IS LESS EFFECTIVE THAN PEOPLE WORKING BY THEMSELVES AND LATER COMPARING IDEAS.
MY IDEA IS TO USE STEM CELL TECHNOLOGY TO DESIGN BOSSES WHO AREN'T IGNORAMUSES.
REMEMBER, YOU'RE NOT SUPPOSED TO CRITICIZE IDEAS.
BUT IF YOU DECIDE TO DO IT ANYWAY, IT SORT OF PROVES MY POINT.
www.dilbert.com
7-8-12
I UNDERSTAND WHY BRAINSTORMING HAS A BAD REPUTATION, BUT IT DOESN'T STOP ME FROM ENJOYING IT.

3. Prototypes

- Valuable tool for clarifying ambiguous requirements.
- Act in a similar way to scenarios by providing users with a context within which they can better understand what information they need to provide.
- Wide range of prototyping techniques—from paper mockups of screen designs to beta-test versions of software products
- Protypes can also be used requirements validation (see later)
- Low fidelity prototypes are often preferred to avoid the stakeholder “anchoring” on minor, incidental characteristics that could limit design flexibility
- Disadvantage: Choose implementation too early
- Risk: Rough prototype becomes the product

Scenarios and Use Cases

Scenarios and use cases are commonly used in planned methodologies, especially in the object-oriented UML setting

Scenarios provide a valuable means for providing context to the elicitation of user requirements.

They allow the software engineer to provide a framework for questions about user tasks by permitting “what if” and “how is this done” questions to be asked.

The most common type of scenario is the use case description.

See week 1 and later

User Stories

User stories are commonly used in adaptive or agile methodologies

Short, high-level descriptions of required functionality expressed in customer terms.

A typical user story has the form: "As a <role>, I want <goal/desire> so that <benefit>."

Just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

The aim is to avoid some of the waste that often happens in projects where detailed requirements are gathered early but become invalid before the work begins.

See week 1

4. Observation / Ethnography

- Analyst immerses herself in the working environment where the system will be used
- She observes the day-to-day work and notes the actual tasks in which participants are involved
- This helps discover implicit system requirements that reflect the ***actual*** rather than ***formal*** processes in which people are involved
- Advantage: discovers many user tasks and business processes that are too subtle and complex for their actors to describe easily.
- Disadvantage: Expensive (analyst works in client environment)
- Disadvantage: Observer should be detached: end-user based, non-judgemental so not appropriate for discovering organisational or domain requirements

Pulling this all Together
Some Guidelines

A Generic Requirements Process

ELICITATION

- ***Identify*** relevant sources of requirements
- ***Ask*** appropriate questions to gain an understanding of their needs

AND THEN

- ***Analyse*** the gathered information, looking for implications, inconsistencies or unresolved issues
- ***Confirm*** your understanding of the requirements with the users (validate)
- ***Synthesize*** appropriate statements of the requirements (specify)

Systems Thinking

1. What objectives are we trying to achieve?
2. What decisions do we control which affect those objectives?
3. What items dictate constraints on our range of choices?
4. What criteria should we use to evaluate candidate solutions?
5. What decision provides with the most satisfactory outcome with respect to those criteria?

Sommerville & Sawyer Elicitation Guidelines

1. Assess system **feasibility**
2. Be sensitive to **organisational** and **political** considerations
3. Identify and consult system **stakeholders**
4. **Record** Requirements Sources
5. **Define** the system's operating environment
6. Use **business concerns** to drive requirements elicitation

Elicitation Guidelines (cont)

7. Look for **domain constraints**
8. Record requirements **rationale**
9. Collect requirements from **multiple viewpoints**
10. **Prototype** poorly understood requirements
11. Use **scenarios** to elicit requirements
12. Define **operational** processes
13. **Reuse** requirements

Lecture Summary

- What is Requirements Elicitation?
  - Process; Essence; Communication; Scope
- Requirements Sources
- Elicitation Techniques
  - Interviews; Facilitated meetings; Prototypes;
  - Scenarios; User Stories see next week
- When eliciting requirements, ensure you have the correct people in the room…often you won't and the result is poor/invalid requirements

Recommended reading

- SWEBOK 3.0 Chapter 1 Section 3
- R. S. Pressman, *Software Engineering: A Practitioner's Approach,* 9th ed., 2020
  - Chapter 5 Understanding Requirements
- B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering – Using UML, Patterns, and Java,* 3rd ed., Prentice Hall, 2010
  - Section 4.3.1 Software Specification
  - Section 7.2 Requirements Elicitation and Analysis