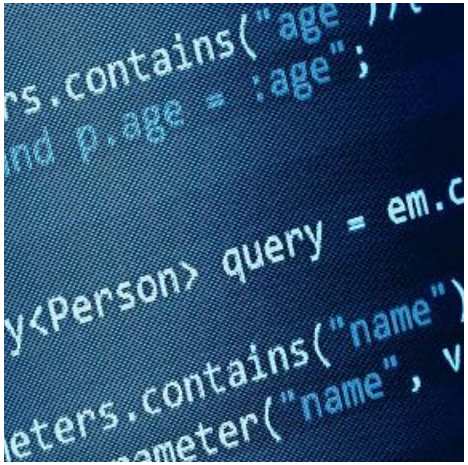
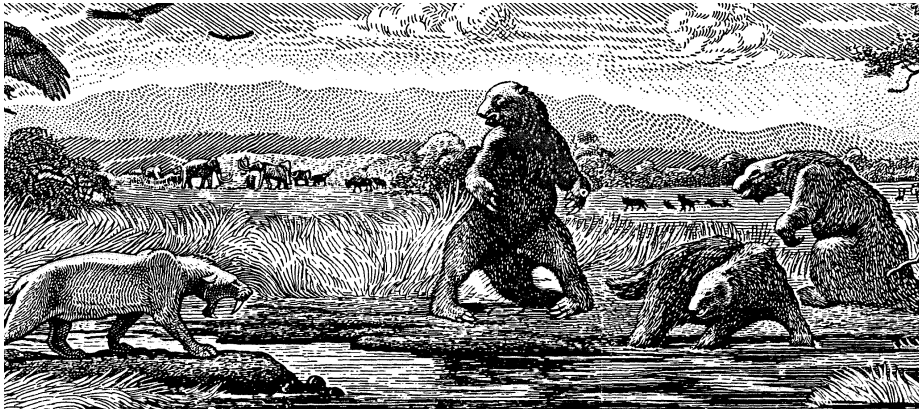
CS3500 Software Engineering



```
s.contains("age"))
nd p.age = :age";
y<Person> query = em.c
eters.contains("name")
ameter("name", v
```

2017/2018



Dept. Computer Science
Dr. Klaas-Jan Stol

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Welcome to CS3500

Important! Teams

- Please send me your team members before tomorrow.

- If you don’t have a team, please send me email, and I will make a team for you.

- Those with a schedule conflict:
    - I will email you today

Requirements Engineering (RE) Part II

After studying this material and associated papers, you should be able to:

- Define what a system is, what a requirement is, and what requirements engineering is.
- Define what stakeholders are, and describe the different stakeholders involved in a system and their concerns.
- Be able to classify requirements and explain what SMART requirements are.
- Describe and use techniques for identifying, documenting, and prioritizing requirements.

Contents

1. Definitions

2. Classifying requirements

3. SMART requirements

4. Identifying requirements

5. Requirement specification

6. Prioritizing requirements

This Lecture

1. Definitions
2. Classifying requirements
3. SMART requirements
4. Identifying requirements
5. Requirement specification
6. Prioritizing requirements

SECTION IV

Requirement Elicitation

1. Stakeholder interaction

2. Domain analysis

3. User stories and scenarios

4. Ethnography

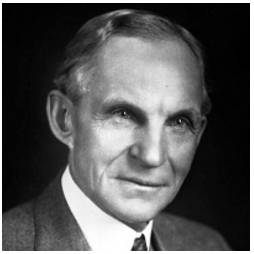
5. Prototyping

Key problems with eliciting requirements

Problem 1:

Customers don’t know what they want.

> “If I had asked people what they wanted, they would have said faster horses.
>
> Henry Ford



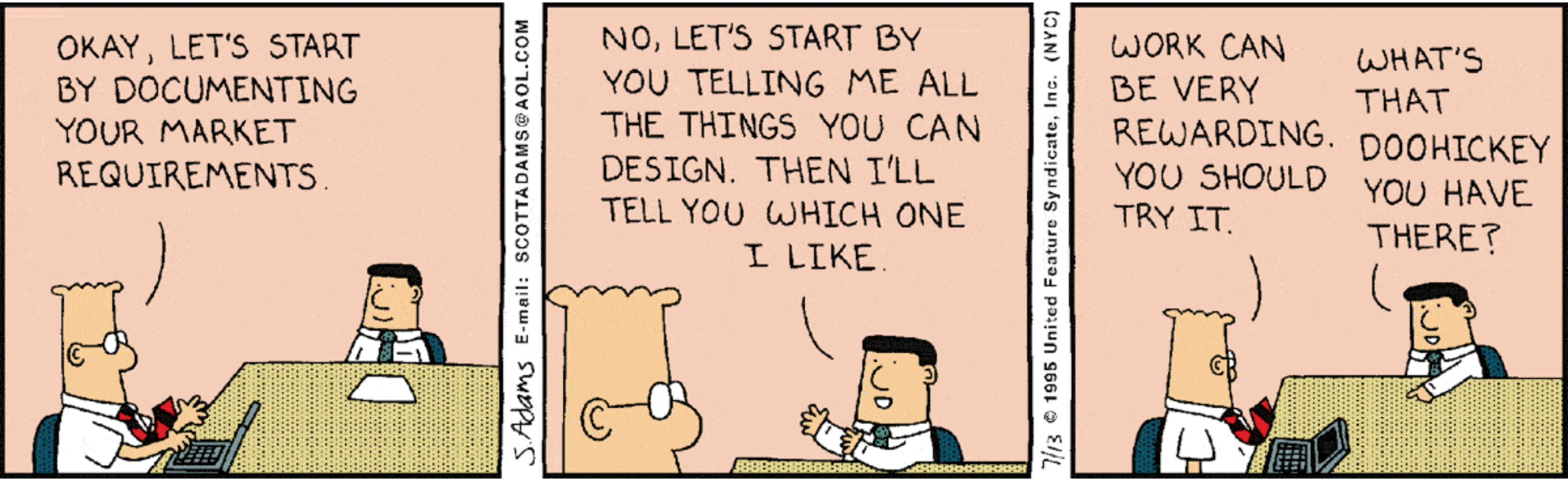
NOTE!

There is no evidence that Ford actually ever said this—but the point remains: customers often don’t know what it is they want.

Key problems with eliciting requirements

- In market-targeted software,
  the marketing dept. serves as customer proxy.

- Marketing dept. may not know either...


OKAY, LET'S START BY DOCUMENTING YOUR MARKET REQUIREMENTS.
NO, LET'S START BY YOU TELLING ME ALL THE THINGS YOU CAN DESIGN. THEN I'LL TELL YOU WHICH ONE I LIKE.
WORK CAN BE VERY REWARDING. YOU SHOULD TRY IT.
WHAT'S THAT DOOHICKEY YOU HAVE THERE?
S. Adams E-mail: SCOTTADAMS@AOL.COM
7/13 © 1995 United Feature Syndicate, Inc. (NYC)

Key problems with eliciting requirements

Problem 2:

Requirements keep changing.

ENGINEERING LIAISON
TELL ME YOUR PROJECT STATUS AND I'LL TRANSLATE FOR OUR CLIENTS.
www.dilbert.com scottadams@aol.com
THE PROJECT WILL NEVER BE COMPLETED BECAUSE OUR IDIOT CLIENTS CHANGE THE REQUIREMENTS EVERY OTHER DAY.
4/26/00 © 2000 United Feature Syndicate, Inc.
I'LL JUST SAY YOU'RE DRUNK.

Direct stakeholder interaction

1. Interviews:
   open or closed interviews with stakeholders to understand their needs.

2. Questionnaires:
   open or closed questions that are well defined.

3. Brainstorming:
   informal discussion in group setting with different stakeholders to capture as many ideas as possible.

4. Focus groups:
   group discussion that is led by a moderator following a structured approach.

Domain analysis

Created by Yu luck
from Noun Project

- Study the application domain to understand typical features.

- Sources:
  - Existing documentation / research
  - Legacy systems
  - Reusable concepts & components

Example: developing a compiler

- What does a compiler do?
- What are the main components of a compiler?
- What existing components can be reused?

User stories and Scenarios

- A user story is a brief statement that identifies user and his/her need.
- Originated in Extreme Programming (XP)

Template:

- As a <role> I want <feature> so that <reason>

Adapted from: I. Sommerville, "Software Engineering"

EXAMPLE

User Story 1

As a user, I want to upload photos, so that I can share photos with others.

Role:	user
Feature:	upload photos
Reason:	sharing photos

EXAMPLE

User Story 2

As an administrator, I want to approve photos before they are posted, so that I can make sure they are appropriate.

Role:	administrator
Feature:	approve photos
Reason:	ensuring appropriateness

User stories and Scenarios

A scenario is a real-life example (a vignette) that illustrates a concrete system interaction.

Is more extensive than a user story.

Includes description of:

- Initial assumptions & expectations
- Normal flow of events
- Exceptions & errors
- Other parallel/background activities
- System state after scenario is finished

Adapted from: I. Sommerville, "Software Engineering"

EXAMPLE

Scenario for a patient record system

INITIAL ASSUMPTION

Patient sees receptionist, to make appointment with doctor.

NORMAL

Receptionist enters current morbidities.

EXCEPTION

If patient doesn’t have a record yet, a new patient record with name, dob, address […] is created.

OTHER ACTIVITIES

Previous records may be consulted by receptionist but not edited. […]

SYSTEM STATE

Patient receives an appointment confirmation. […]

Adapted from: I. Sommerville, “Software Engineering”

Ethnography

- Ethnography is a research method where the researcher is immersed in a culture, taking the point of view of the study subject.

- Develop understanding through:
    - Interviews
    - Observation
    - Participation
    - Longitudinal immersion

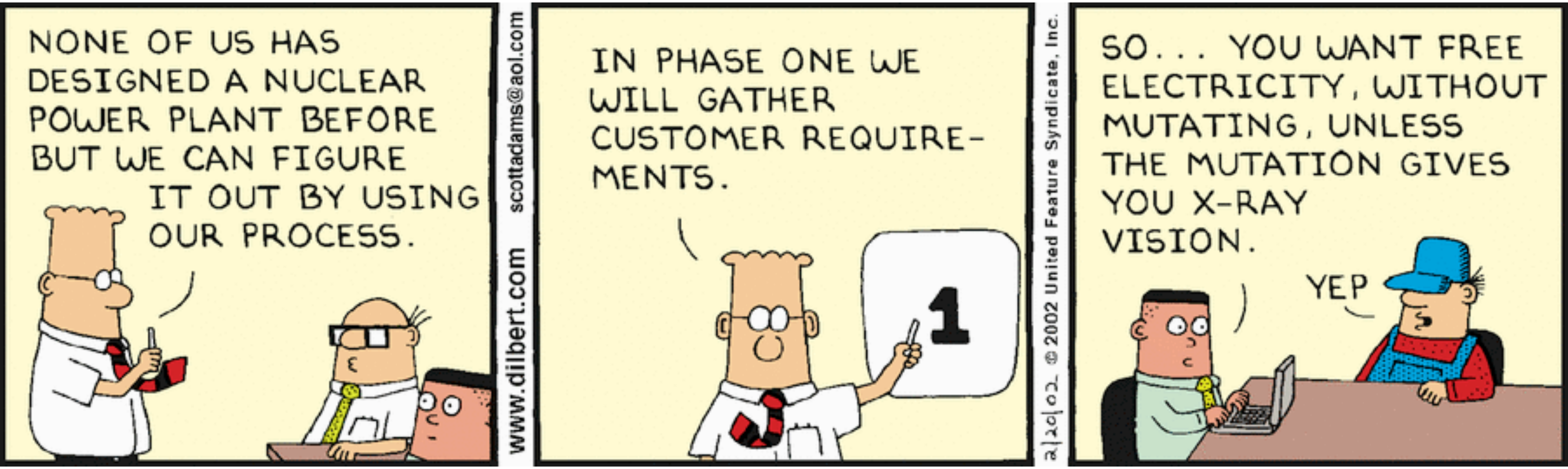
Spot the ethnographer

Prototyping

- Develop prototypes of the system to solicit early feedback.
- Prototype based on:
  - Preliminary requirements
  - Existing examples / similar systems
- Useful for:
  - User-interfaces
  - Greenfield development
  - Overcome IKIWISI syndrome [see reading]

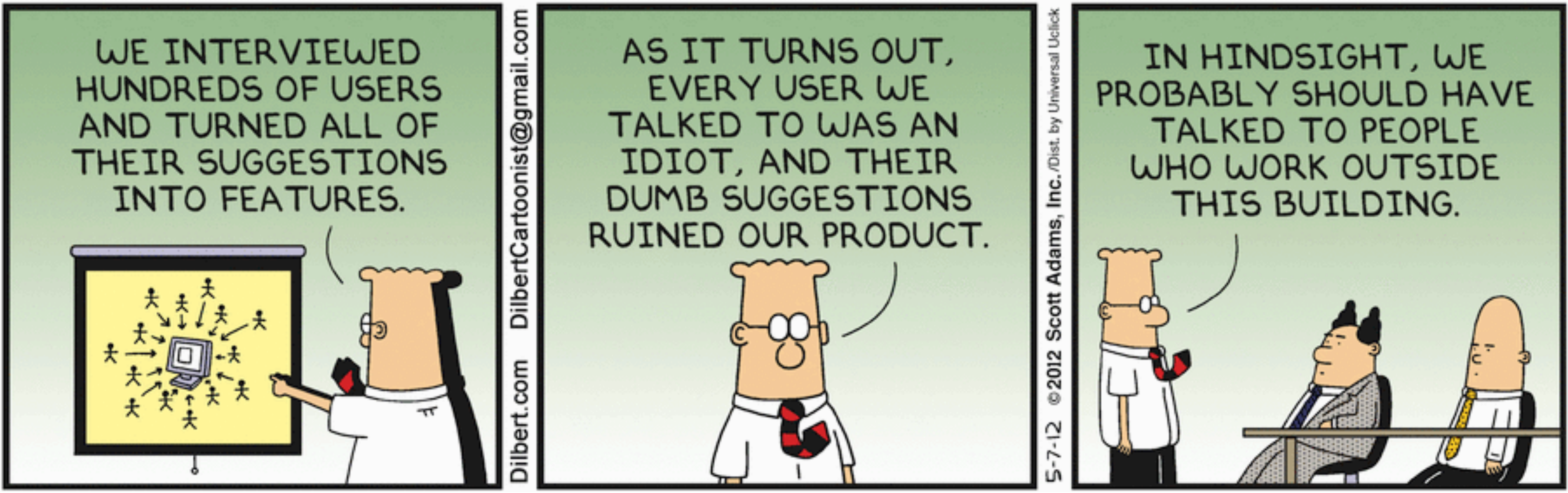
More on prototyping later!

Which approach to use? It depends (1)


NONE OF US HAS DESIGNED A NUCLEAR POWER PLANT BEFORE BUT WE CAN FIGURE IT OUT BY USING OUR PROCESS.
scottadams@aol.com
www.dilbert.com
IN PHASE ONE WE WILL GATHER CUSTOMER REQUIREMENTS.
1
© 2002 United Feature Syndicate, Inc.
2/20/02
SO... YOU WANT FREE ELECTRICITY, WITHOUT MUTATING, UNLESS THE MUTATION GIVES YOU X-RAY VISION.
YEP

Source: dilbert.com

Which approach to use? It depends (2)


WE INTERVIEWED HUNDREDS OF USERS AND TURNED ALL OF THEIR SUGGESTIONS INTO FEATURES.
Dilbert.com DilbertCartoonist@gmail.com
AS IT TURNS OUT, EVERY USER WE TALKED TO WAS AN IDIOT, AND THEIR DUMB SUGGESTIONS RUINED OUR PRODUCT.
5-7-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick
IN HINDSIGHT, WE PROBABLY SHOULD HAVE TALKED TO PEOPLE WHO WORK OUTSIDE THIS BUILDING.

No “right” answer—the right way is one that works.

Source: dilbert.com

SECTION V

Requirement Specification

1.

The SRS document

2.

Notations

A structure of an SRS

Section	Description
Preface	Define readership, version history
Introduction	Context and need for system incl. business / strategic objectives
Glossary	Define technical terms used in document
User requirements	Services / functionality provided to user
System architecture	High level overview of expected system architecture
System requirements specification	Describe requirements
System models	Models showing relationships between components and environment.
System evolution	Fundamental assumptions underpinning the system and anticipated (future) changes (e.g. hardware, users)
Appendices	Additional information re. hardware and database specifications.
Index	Several indexes (e.g. by topic, figures, tables, etc.)

Adapted from I. Sommerville, “Software Engineering”

Requirement notations

Requirements can be expressed in various ways:

- Natural language
- Structured language
- Graphical notations

Others, not discussed in CS3500:

- Formal languages, e.g. Z
  - Used in specific domains e.g. embedded

Whatever notation,
requirements should always be SMART!

Notation: Natural language

Expressive and universal, but also potentially vague and ambiguous.

Suggested guidelines:

- Standardized format
  - Requirement as a single sentence
  - Include rationale to explain why needed
  - Origin (who proposed?)
  - Number each requirement
- Use language and terms consistently
- Highlight important parts
- Avoid jargon and include definitions

Adapted from: I. Sommerville, "Software Engineering"

EXAMPLE

Natural language requirements

R9:

No longer than one hour shall be required to produce an optimised plan for a period of 4 orbits (11 days).

This time is a trade-off between precision and timeliness, and facilitates the mission leader to make timely decisions during mission planning.

Adapted from:
Keepence et al. "SMARTre Requirements: Writing Reusable Requirements". 1995. IEEE

EXAMPLE

Structured language requirements

Insulin Pump/Control Software/SRS/3.3.2

Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. [… further details …]
Requirement	2 previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the max. allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side-effects	None

Adapted from: I. Sommerville, “Software Engineering”

EXAMPLE

Structured requirements: tables

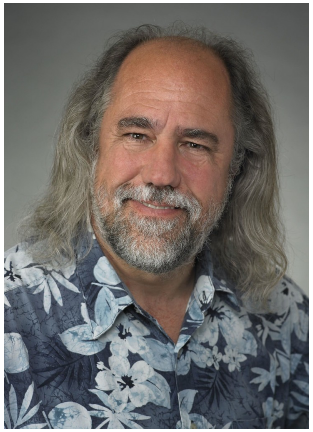
Condition	Action
Sugar level falling (r2 r1)	CompDose = 0
Sugar level stable (r2 r1)	CompDose = 0
Sugar level increasing and rate of increase CompDose 0 decreasing ((r2 r1) (r1 r0))	CompDose = 0

Sugar level increasing and rate of increase stable or increasing ((r2 r1) (r1 r0))	CompDose = round ((r2 - r1)/4)
	If rounded result == 0 then
	CompDose = MinimumDose

Adapted from: I. Sommerville, "Software Engineering"

Graphical notation: Unified Modeling Language (UML)

Why Unified?



Grady Booch
(Booch's method)

James Rumbaugh
(Object Modeling Technique)

Ivar Jacobson
(Object-Oriented Software Engineering method)

- Joined forces to create an Object Management Group (OMG) standard
- 1997: Version 1.0
- 2005: Version 2.0
- 2015: Version 2.5 (current)

UML

- UML originated in methods for object-oriented analysis and design.
  - Therefore strongly focused on OO

- UML suitable for:
  - Visualizing
  - Specifying
  - Constructing
  - Documenting (all artifacts, incl. requirements)

- UML is a standard, but most people don’t use it formally—only selectively

UML: Use cases

A use-case identifies an actor and an interaction with a system.

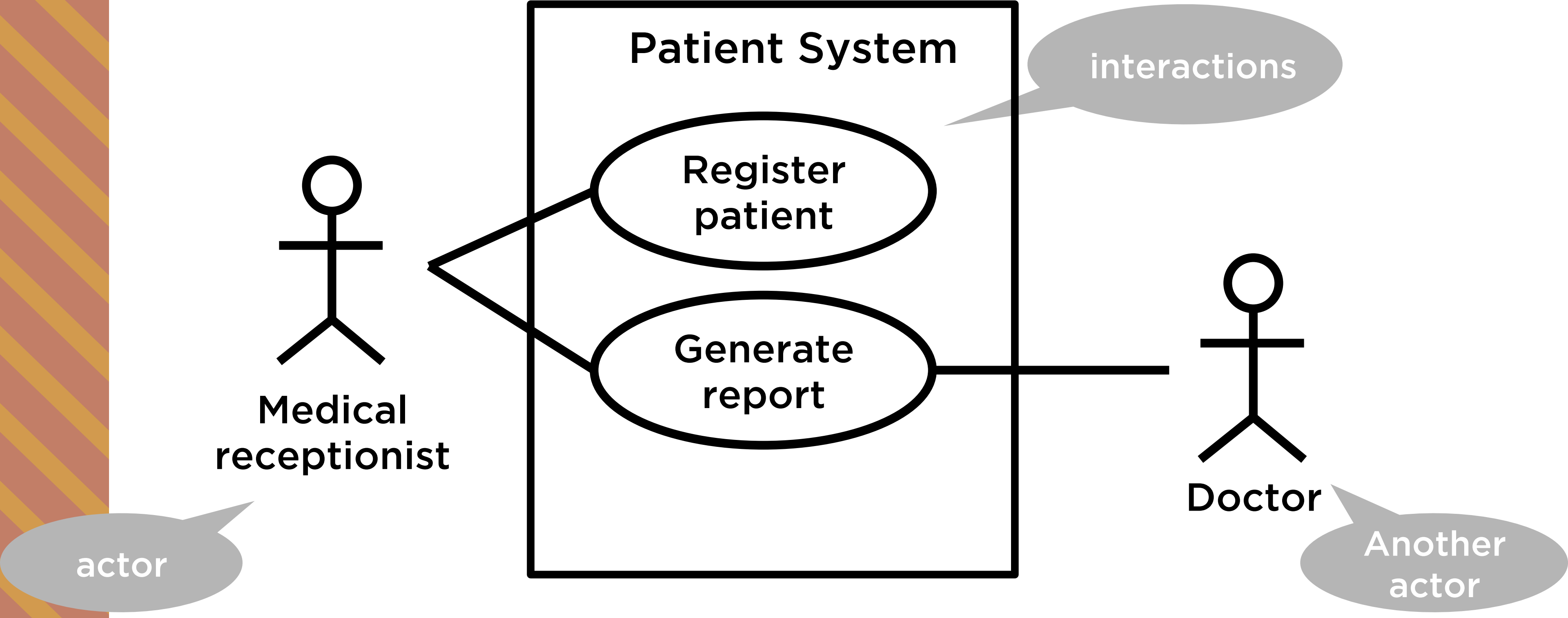
Capture intended behavior without specifying how to implement it

More on use-cases later!

Adapted from:
I. Sommerville, "Software Engineering"

EXAMPLE

Use-case



More on use-cases later!

Adapted from:
I. Sommerville, "Software Engineering"

SECTION VI

Requirements Prioritization

1. Simple ranking

2. MoSCoW

3. Planning Poker

4. $100 method

5. Bubble sort

Simple ranking

- N requirements are ranked 1 to N, in order of decreasing priority.

- Example:
  - R1: most important requirement
  - R2: like R1, but less important
  - R3: like R2, but less important
  - R4: like R3, but less important
  - …
  - N: like N-1, but less important

Requirements prioritization: MoSCoW

- **Must have**
  required to ensure project success
- **Should have**
  “would be nice to have”
- **Could have**
  like “should” but less important
- **Won’t have (this time)**
  “wish list” – maybe next time!

Planning Poker

- A practice of the Extreme Programming method (XP)
  - Discussed in 2nd half of the semester.

- Planning Poker is a simple way to reach consensus on effort estimation

- When 2 features have same value, then prioritize the one with least effort (cost)

$100 method

- Give all stakeholders $100 (imaginary money / tokens)
- $100 to be distributed over requirements:
  - more important ones get higher value
- For each requirement count sums
  - Results in order of prioritized reqs.
- Use other amounts as appropriate.

Bubble Sort

- Just like sorting an array of numbers, but instead sort requirements.

  [r1, r2, r3, … , r256]

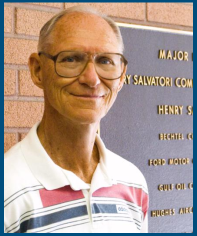
- Take 2 requirements rX, rY

  If priority (rX) < priority (rY) then swap

Reading Assignment

"Requirements that Handle IKIWISI, COTS, and Rapid Change"

By: Barry Boehm



Originally published in:
IEEE Computer, July 2000, pages 99-102.

Estimated reading time:
3¼ pages, 45 min.

Summary

- SRS Template—implies considerable detail on implementation

- Different notations available to document requirements: natural language, graphical language

- Various techniques available for Identifying requirements—choose the ones that make sense for your situation.

- Many techniques available to prioritize requirements—$100, ranking, sorting, planning poker, MoSCoW

Thank you for your attention

Questions & suggestions can be sent to:
k.stol@cs.ucc.ie