1

- Requirements Engineering
  - definition, scope
  - roles

Requirements Engineering

- Requirements Engineering deals with how to better establish the **requirements** for a software system
  - what is desired by the "customers"
  - what is feasible
  - what is of interest to the producer
- Requirements Engineering is an interdisciplinary issue
  - technical (computer science, engineering, ...)
  - social (acceptability, ethical, negotiation...)

Requirements Engineering

- The “official” definition by IEEE (1990):

Requirement. (1) A condition or **capability** needed by a **user** to solve a **problem** or achieve an **objective**. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).

Requirements Analysis. (1) The process of studying **user needs** to arrive at a definition of system, hardware, or software requirements. (2) The process of studying and refining system, hardware, or software requirements.

Requirements Engineering

- The “official” definition by IEEE (1990):

Requirement. (1) A condition or **capability** needed by a **user** to solve a **problem** or achieve an **objective**. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imp

a condition

Development problem

build a software system with given capabilities (2)
so that the problem/objective (1) is solved/reached

Requirements Analysis. (1) The process of studying **user needs** to arrive at a definition of system, hardware, or software requirements. 2) The process of studying and refining system, hardware, or software requirements.

Requirements Engineering

- The “official” definition by IEEE (1990):

Requirement. (1) A condition or **capability** needed by a **user** to solve a **problem** or achieve an **objective**. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imp

a condition

Development problem

build a software system with given capabilities (2)
so that the problem/objective (1) is solved/reached

Requirements Analysis. (1) The process of studying **user needs** to arrive at a definition of system, hardware, or software requirements. 2) The process of studying and refining system, hardware, or software requirements.

Elicitation problem

extract and articulate requirements based on interaction with users (later: customers, consultants, domain experts, etc., in general *stakeholders*)

Requirements Engineering

- An alternative definition by Pamela Zave (1997):

“Requirements engineering is the **branch of software engineering** concerned with the **real-world goals** for, **functions** of, and **constraints** on software systems. It is also concerned with the relationship of these factors to **precise specifications** of software behavior, and to their **evolution** over time and across software families.”

Requirements Engineering

- An alternative definition by Pamela Zave (1997):

“Requirements engineering is the **branch of software engineering** concerned with the **real-world goals** for, **functions** of, and **constraints** on software systems. It is also concerned with the relationship of these factors to **precise specifications** of software behavior, and to their **evolution** over time and across software families.”

Development problem

produce a precise specification of software behaviour
(implicitly: so that the goals are reached)

Requirements Engineering

- An alter (1997):

Elicitation problem

discover the goals, functions and constraints in the real world (by asking stakeholders or from other sources, e.g. documents or ethnography)

“Requirements engineering is the **branch of software engineering** concerned with the **real-world goals** for, **functions** of, and **constraints** on software systems. It is also concerned with the relationship of these factors to **precise specifications** of software behavior, and to their **evolution** over time and across software families.”

Development problem

produce a precise specification of software behaviour (implicitly: so that the goals are reached)

Basic understanding of RE

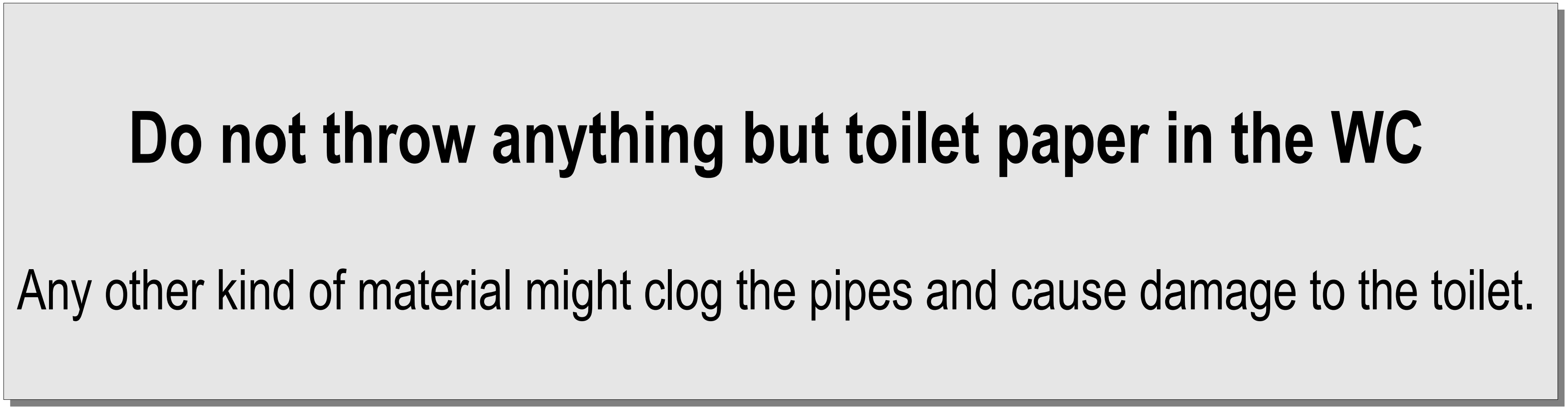
- In theory, RE is simple:
  1. Talk to users, get requirements
  2. Write a precise specification of the same
  3. Hand the spec to programmers, they will build the system accordingly
  4. Happy users will pay your fee
- In practice, it might be not!
  - in theory, theory and practice coincide
  - in practice, they do not

Major hurdles in RE

- Elicitation
  - users might not be able to articulate their needs
  - different users might have conflicting needs
  - some stakeholder might not be identified as such
  - interaction with users might produce lots of “noise” (irrelevant facts)
  - part of the needed knowledge might be tacit (hard to make explicit)

The trouble with language

- Appeared two days ago in a public toilet in our department:



Do not throw anything but toilet paper in the WC

Any other kind of material might clog the pipes and cause damage to the toilet.

The trouble with language

- Appeared two days ago in a public toilet in our department:

Do not throw anything but toilet paper in the WC

Any other kind of material might clog the pipes and cause damage to the toilet.

SHALL I TAKE MY SHIT HOME WITH ME?

Major hurdles in RE

- Specification
  - what language to write the SRS in?
    - natural language: easy to work with, imprecise
    - formal languages: precise, difficult to work with
  - how to incorporate relevant domain knowledge in a SRS?
- Verification and Validation
  - is the SRS correct? (verification)
  - will it solve the problem? (validation)
  - will the quality of the solution be sufficient?

Major hurdles in RE

- Specification
  - what language to write the SRS in?
    - natural language: easy to work with, imprecise
    - formal languages: precise, difficult to work with
  - how to incorpor
    a SRS?
- Verification and
  - is the SRS corr
  - will it solve the problem? (validation)
  - will the quality of the solution be sufficient?

Human-Computer interaction

How to properly elicit and document interaction requirements in an SRS?

A badly-engineered interaction design might make even a correct solution worthless in practice.

Major hurdles in RE

- Specification
  - what language to write the SRS in?
    - natural language: easy to work with, imprecise
    - formal languages: precise, difficult to work with
  - how to incorpor
    a SRS?

ation)

e sufficient?

Human-Computer interaction

How to properly elicit and document interaction requirements in an SRS?

ed interaction design might

rrect solution worthless in

practice.

Performance and satisfaction

In distributed applications, level of performance can make or break a product.

Transmission delays, resilience to faults, multiple appliances, cost of data transmission, execution speed, distribution channel.

Major hurdles in RE

- The hurdles mentioned above will be (in part) addressed in the following
- Other issues we will not touch:
  - how to trace requirements from source to implementation and back
  - how to collect, structure and organize requirements for whole families of products
  - how to *guarantee* certain properties, for high-assurance systems
  - how to evolve requirements in response to evolving needs and real-world
  - ... and countless others

Roles in RE

- The king: **user**
- The treasurer: **customer**
- The public: **others affected**

- The wise: **domain expert**
- The artisan: **requirements analyst**
- The worker: **developer**
- The supervisor: **quality control**
- The conductor: **project manager**

Roles in R

- **Different disciplines**
  - **HCI** is concerned mostly with users
  - **RE** is concerned mostly with customers, domain experts & requirements anaysts
  - **Soft Eng** considers developers, quality control, project managers
  - **Sys Eng** considers others affected

- The king: **u**
- The treasurer: **customer**
- The public: **others affected**

- The wise: **domain expert**
- The artisan: **requirements analyst**
- The worker: **developer**
- The supervisor: **quality control**
- The conductor: **project manager**

Roles in R

•Different disciplines

- **HCI** is concerned mostly with users
- **RE** is concerned mostly with customers, domain experts & requirements anaysts
- **Soft Eng** considers developers, quality control, project managers
- **Sys Eng** considers others affected

- The king: **u**
- The treasur **customer**
- The public: **others affected**

- The wise: **domain expert**

•Complementary approaches

- Usability requirements
- “Voice of the customer”
- Participative design
- Rapid prototyping
- Usability testing
- Social approaches

ts analyst

control

manager

Roles in RE

- “Others affected” can cover a variety of scenarios:
  - shareholders of both the customer and the developer
  - governing bodies, e.g. local councils or standard bodies
  - public at large, e.g. for environmental consequences
  - competitors in the same market
  - etc.

Roles in RE - example

- An **intensive care unit** monitoring station
  - **user**: nurses, doctors
  - **customer**: hospital
  - **others affected**: patients (& their heirs)
  - **domain expert**: physiologists & bioengineers
  - **requirements analyst**: company or independent (consultant)
  - **developer**: company or contracted (outsurced)
  - **quality control**: company
  - **project manager**: company

Human-centered design

Bring together the social and technical issues involved in inventing, marketing, deploying and operating a new technology to the maximum benefit of all involved parties, within given constraints

Human-centered design

Bring together the social and technical issues involved in inventing, marketing, deploying and operating a new technology to the maximum benefit of all involved parties, wi

Way **more** than just User Interfaces!

Human-centered design

Not only a piece of technology must be *usable*, it has also to satisfy the ***desires*** of involved humans!

dep nology

to the maximum benefit of all involved

parties, wi

Way **more** than just User Interfaces!

Human-centered design

Not only a piece of technology must be *usable*, it has also to satisfy the ***desires*** of involved humans!

de **nology**

olved

In distributed systems, expectations are still high, but technological hurdles are greater

han just User Interfaces!

RE as the study of desires

- RE is thus about the study of **desires** of **humans**
- in particular,
  - how to elicit those desires
  - how to compose conflicting desires
  - how to document them
  - how to build a software system so that the desires will come true
- ... given certain **constraints**

Sources of constraints

- Economics
  - Costs for users
  - Costs for customers
  - Costs for developers
- Technology
  - Basic components
  - Infrastructure (especially for distribution)
- Legal and social issues
  - What is socially and legally acceptable