Bournemouth University

SoSyM

Process Oriented Requirements Engineering

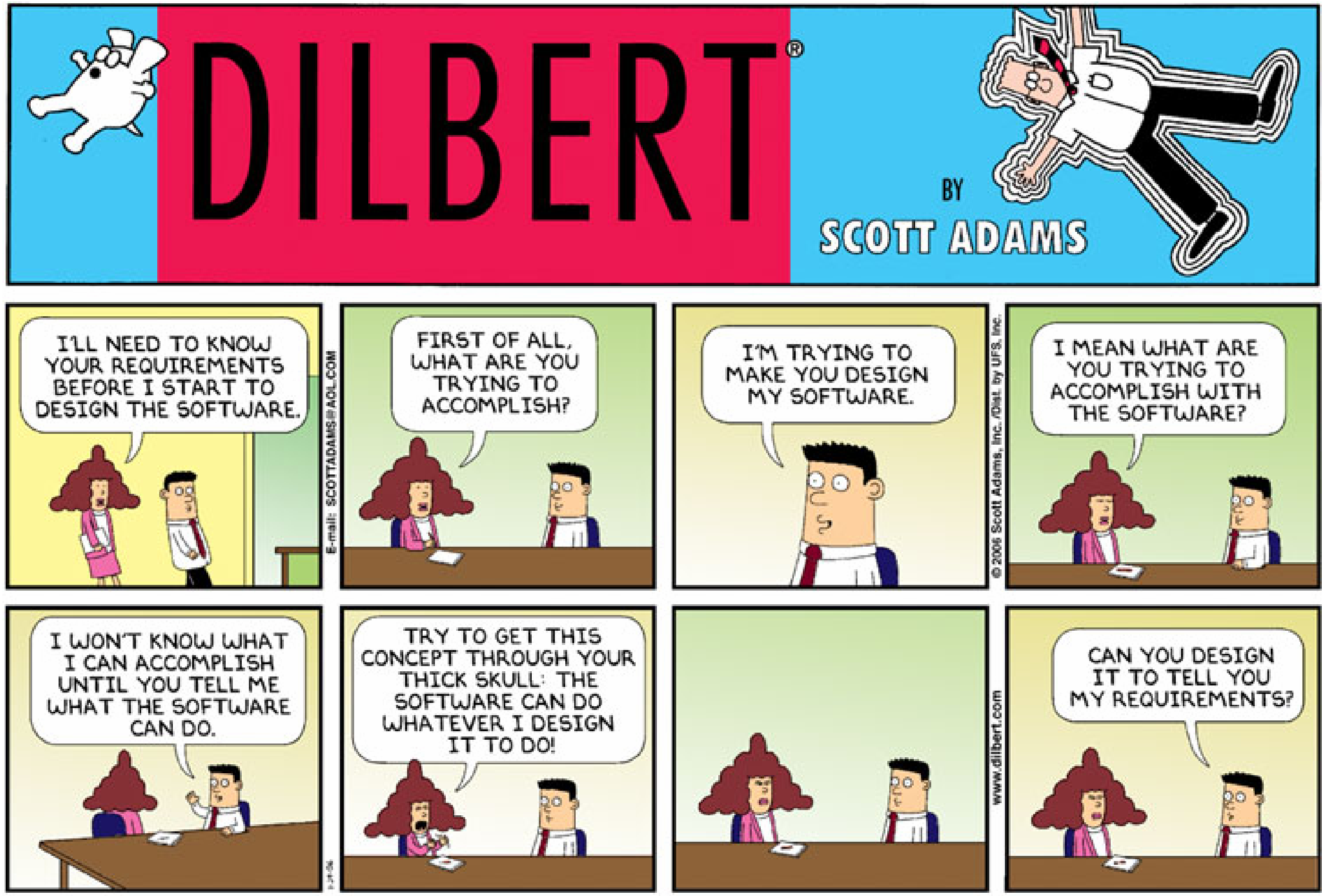
Dr Keith Phalp

The course in a nutshell

SoSyM

- Customers want systems to support their business processes. (We can argue about the b word).
- Developers build systems for clients
- “Oh dear. The system doesn’t seem to meet the client’s needs”. (It must be someone’s fault).
- This is a *requirements* problem, and ***very*** common.
- One reason is that the developers didn’t understand the problem: or what they wanted or needed.
- That’s where the process modelling comes in, and some other assorted ideas.

Who is at fault?


DILBERT®
BY SCOTT ADAMS
I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
E-mail: SCOTTADAMS@AOL.COM
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

Some themes & topics

SoSyM

- We need to unpick this problem. So we will look at:
- What did we mean by requirements.
- Does everyone agree? (Of course agreement is unlikely, even among us).
- How do current approaches fare (and past and proposed and future).
- What might we do to help:
  - What is found to be useful in practice
  - Of course it's process modelling.
  - How do we put our ideas together.
- Plus we will look at key research in the area.

- Scheme at: http://dec.bournemouth.ac.uk/ESERG/kphalp/teaching.htm

Theme 1: What requirements isn't and implications.

SoSyM

- You will have seen some of these arguments before (especially those from RAS).
- The main thing is to realise that we need to understand that *requirements is about the problem domain*, and is *not the same as specification* (which we will also define).
- Most people mix these up, but:
    - Most people start from specification
    - There is no such thing as a requirements specification.
    - Use cases are really specification.

- Let's start with some views and definitions.

What is Requirements (Analysis)

SoSyM

- Engineering is about **solving problems** (by, **constructing complex, useful artefacts)**
- Requirements (sometimes called analysis) is about **defining those problems** (deciding the **purpose of the artefact).**

- It is a **very good idea** to decide **what** you are going to build before you build it – though often Computing doesn't.
- Typically engineers and programmers want to get going (and their managers too), before figuring out the problem.

Requirements Definitions

SoSyM

- ZAVE (1997)
  - “Requirements engineering… is 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 behaviour…”
- SWEBOK (2001)
  - “A requirement is defined as a property that must be exhibited in order solve some problem of the real world.”
- Robin Goldsmith (2004)
  - “What must be delivered to provide value.”

Impact of RE

- Chaos reports: 75% of projects failed or failed to deliver key functions.
  - Why? Poor / no requirements engineering.
- 2004 JobServe report: only 16% of UK software projects are successful.
  - Why? Poor / no analysis skills.
  - *“Projects are often poorly defined, codes of practice are frequently ignored and there is a woeful inability to learn from past experience”*, says Professor McDermid

SoSyM

Requirements Matters

- ‘*the most crucial activity*’ [INCE89]
- ‘*The most important [phase] by far*’ [BRUC89]
- ‘*the most critical and problem-prone area*’ [HOOP82]

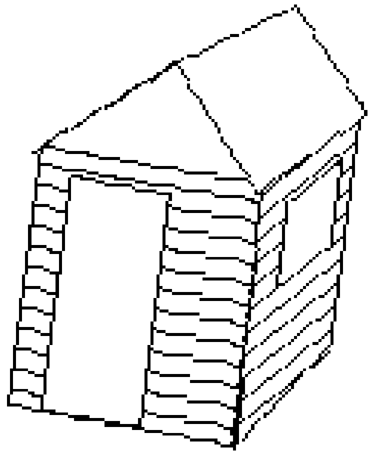
- ‘*There is little doubt that project requirements are the single biggest cause of trouble on the software project front. Study after study has found that, where there is a failure, requirements problems are usually at the heart of the matter.*’ [GLAS98]

RE is at the **start:** it forms the **foundations**

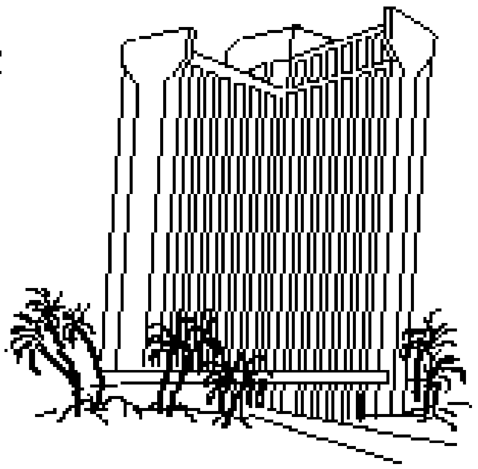
Size Really Does Matter

- Importance of getting the foundations right?

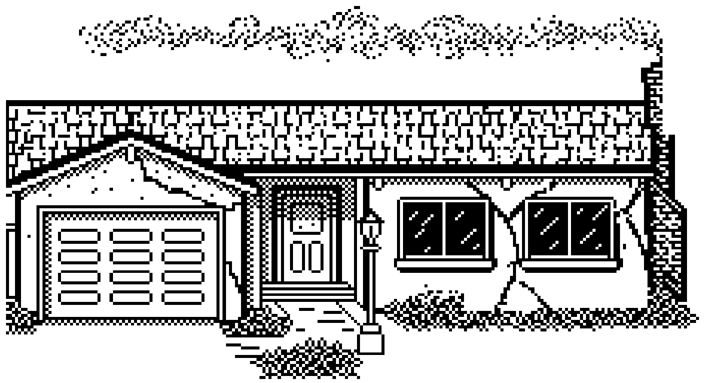
A



C



B



- All taken from:
- Bray, I ‘An introduction to Requirements Engineering’

A stitch in time…

SoSyM

- RE errors are no worse than any others
  - ***provided they are fixed there and then***
- But the cost of fixing escalates
  - Said to be approximately ten times for each major development phase.
- And more software development phases follow RE than the rest.
- The exceptions being business phases: perhaps of the strategic, and business process modelling.

Bournemouth University

Requirements are Critical

SoSyM

- *‘One of the most common reasons systems fail is because the definition of system requirements is bad.’* [SCA81]
- *‘The chief villian [sic] in any software fiasco is a non-existent, vague, incomplete or poorly thought out set of requirements.’* [GLAD82]
- *‘as much as 90% of subsequent troubles can be traced back to erroneous original specifications.’* [BRUC89]

But… often carried out poorly

SoSyM

- Need domain knowledge & software development expertise (contentious)
- Communication with ‘other types’, with very different worldviews.
- Something from Nothing. It’s been suggested that one ‘invents’ requirements (rather than discovers them).
- Poor technology?
- Lack of expertise, even among software engineers.

Why care about RE

- Estimates vary, but in terms of effort (tome spent) the proportion of the development process spent in each phase is approximately

• Requirements (including Spec)	30%
• Design	20%
• Implementation	20%
• Testing	30%

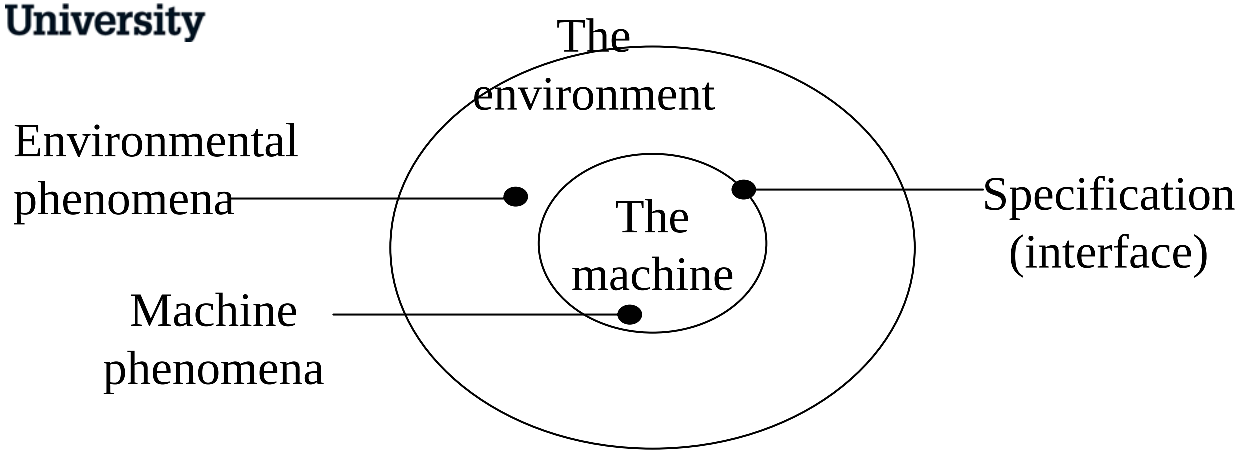
  - Requirements (and possibly testing) are reportedly the least well done phases, with (typically) the least emphasis in education.
  - Similarly, consider the quantity of books devoted even to a single programming language compared to those (relatively few) on requirements (and indeed, test).

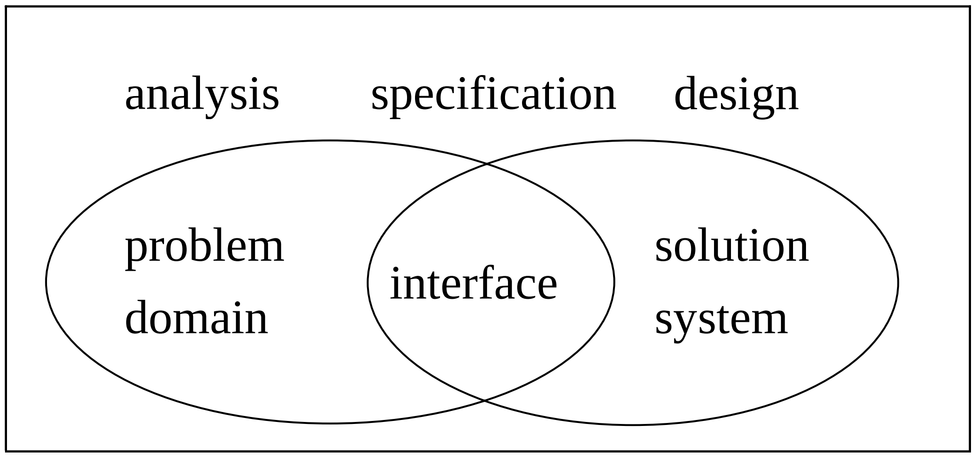
Three descriptions

- Jackson suggests the need to produce three separate, and distinct descriptions.

“In principle you really need three descriptions: the common description; the description that’s true only of the machine and the description that’s true only of the application domain. If you make all of those descriptions, and separate them carefully, you’ll be all right.”

Descriptions and phases





SoSyM

Tasks in Requirements

- **Analysis** (Requirements) - concerns study of the problem domain and the problem(s) in it
  - Much misunderstood – as we shall see
- **Specification** - defining the interaction between the solution system and the problem domain
- **Design** - (**not** part of requirements engineering) *largely* concerns inventing the internal workings of the solution system.

Separation of concerns

- Vital to separate distinct tasks, especially, requirements and specification.
- We start by studying the problem domain and producing a **description of the problem domain** and a statement of **the effects that the new system should produce** in the problem domain (i.e., the **requirements**)
- Only then **specify a behaviour** of the **new, solution system,** such that it **will produce the required effects** in the problem domain (the **specification**)

Bournemouth University

Requirements and Spec

SoSyM

RECAP, or the brief version

- Through a sound understanding of the *problem domain* can we understand and develop a set of REQUIREMENTS for the proposed system.
- Then a SPECIFICATION can be written that should enable system design to occur.

- *Of course this is a rather purist view.*

Requirements

- Requirements: The desired effects that the machine (system) is to have on the problem domain or environment.
  - Purpose: Fulfils the goals.
- Of course there can be different ‘types’ of requirements.

SoSyM

Functional Requirements

- **Functional requirements**
  - The “ordinary” requirements
  - Can be met by appropriate system functionality.
  - **Requirement**: The doors are to be cycled whenever a lift stops at a floor
  - **Corresponding function**:***When the controller detects that a lift has stopped at a floor, it sends the signal to cycle the doors***
- ***But choosing level of abstraction is difficult.***

Performance Requirements

SoSyM

- Parameters of functionality – determine how quickly, how reliably etc. functions must operate
- Often (wrongly) called ‘non-functional’ or ‘non-behavioural’ requirements.
- Usually separated from other (functional) requirements because they are relatively *volatile*.
- Common sub-categories: Speed, Capacity, Reliability, Usability or **SCRU.**

Performance (Speed)

- **Defined in terms of :-**
  - **throughput** (off-line (batch) systems)
  - OR **response times** (interactive or real-time systems)
- E.g.,
  - must process 10,000 transactions per hour
  - must respond to flame-out by closing gas valve within 0.2 seconds

SoSyM

Performance (Capacity)

- The **quantity of data** that can be stored in the system, the **number of simultaneous users, etc, for example:**
    - It will be possible to store at least 10000 transactions
- **Not** to be confused with **constraints** upon the **size of the system, such as:**
    - The new system shall occupy no more than 10 Gbytes of RAM

SoSyM

Performance (Reliability)

- Common to use mean time between failure (really between fault discovery).
  - Note: Software doesn't wear out, faults don't occur, they get found.
- Usually better to specify in terms of **availability**: the proportion of the time (within specified periods) that the system is performing correctly (or, at least, useably).

Performance (Usability)

DILBERT by Scott Adams
YOUR USER REQUIREMENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.
www.dilbert.com
scottadams@aol.com
© 2001 United Feature Syndicate, Inc.

Best to think in terms of **testability**

SoSyM

Constraints

- The **true** non-functional requirements: (constrain) how the system is built but **not** what it does
- *Under normal usage, would it be apparent to the eventual users of the system whether the requirement has been met?* If **not**, then it is a design constraint.
- Ideally (as developers) we want **no** design constraints…
  - but it doesn't always work out that way

Common Constraints

- target machine(s) upon which the-system must operate
- memory size within which it must operate
- operating system(s) under which it must operate
- programming language(s) that must be used
- other software packages that must be incorporated
- development standards that must be applied
- design methods that must be employed
- algorithms that must be incorporated
- Processes or procedures that must be followed

SoSyM

Summary (so far)

- Problem domain (PD) contains the problem.
- Requirements are the desired effects required in the PD.
- Specify a system to produce the desired effects.
- Activities in RE
  - Analysis - Describes PD and the requirements.
  - Specification - the interface between the PD and the System. OR inventing functions to meet the requirements.

Implications for Requirements

SoSyM

- Need to understand the (whole) problem and the client goals, needs and processes.
- Need to identify the system boundary (and model interactions outside and across).
- Elicitation to directly produce requirements
- Use of process models (existing or new) to inform requirements.
- All before use cases and specification.