Imperial College London

Software Engineering Practice

Capturing Requirements

Lecture Overview

- From the Real-World to Software
  - Challenges
- Capturing Requirements
  - Goal-Oriented Capture
  - System Boundaries
  - Feasibility and Risk

Imperial College London

From the Real-World to Software

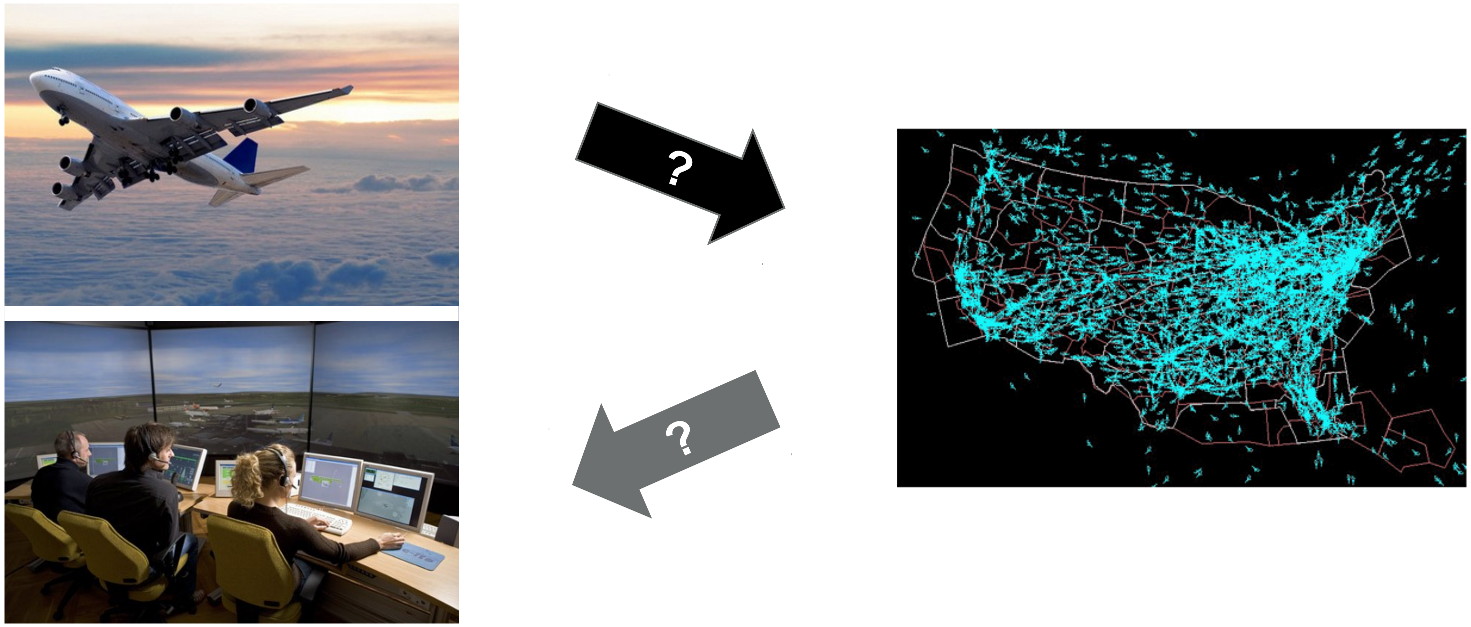
- Most software is produced to real-world demand
  - It is intended to tackle a real-world problem
  - It is intended to have a real-world effect

But!

- The real world is informal and imprecise...
- ...Software is formal and precise

From the Real-World to Software

- How do we consistently go from a description of a real-world problem and a desired real-world solution...



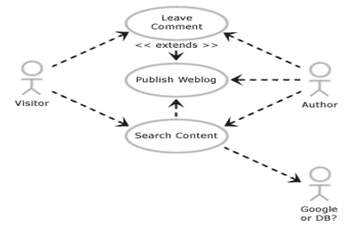
- ...to software that implements the real-world solution correctly and reliably?

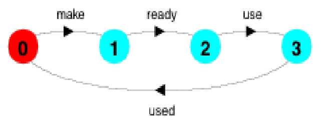
Imperial College London

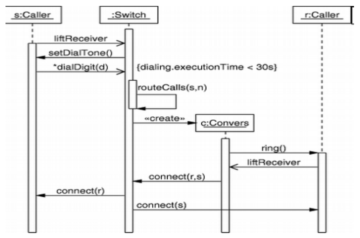
From the Real-World to Software

The Real World

The Model







The Software

A model might be informal (e.g., natural language), semi-formal (e.g., UML), or formal (e.g., state machines).

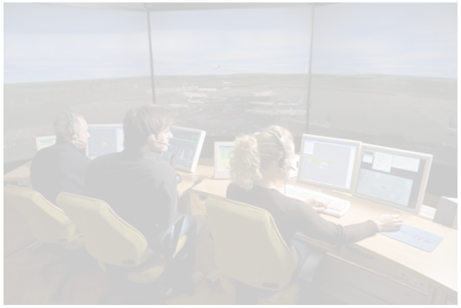
Imperial College London

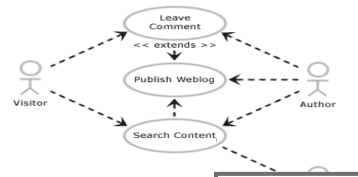
From the Real-World to Software

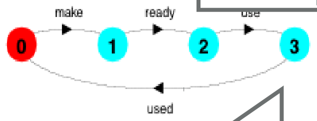
The Real World

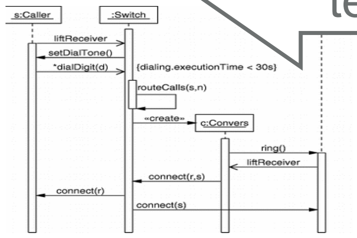
The Model

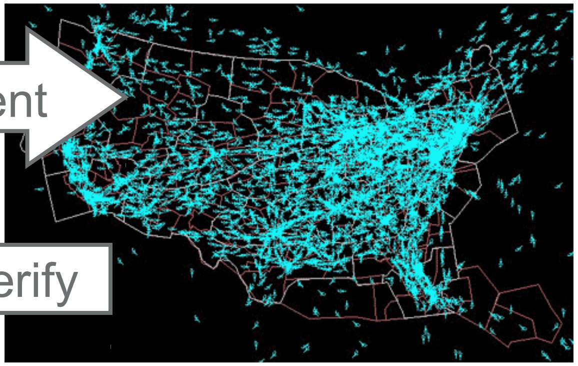
The Software











implement

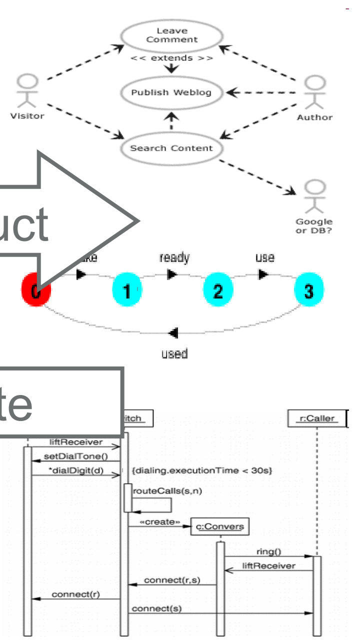
test/verify

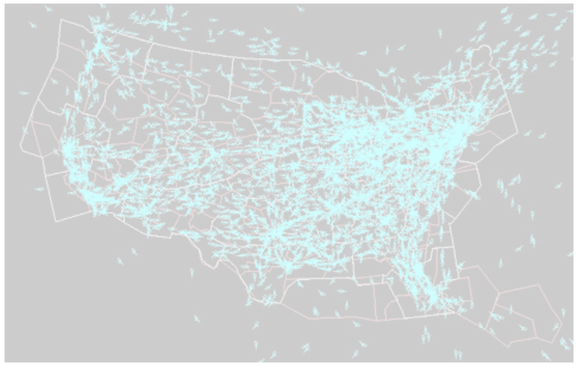
The principles on the right are relatively well understood.

Imperial College London

From the Real-World to Software

The Real World | The Model | The Software





The principles on the left are not so well understood.

Imperial College London

From the Real-World to Software

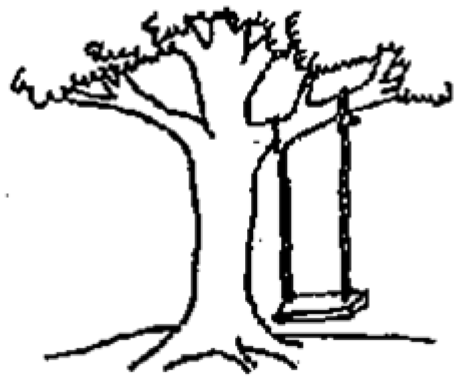
The Real World	The Model	The Software

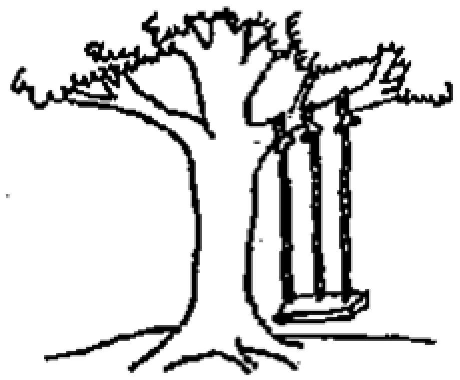
The technical subject of model theory is a study of the relationship on the right. At this point in intellectual history, we have no theory of this left-hand-side relationship.

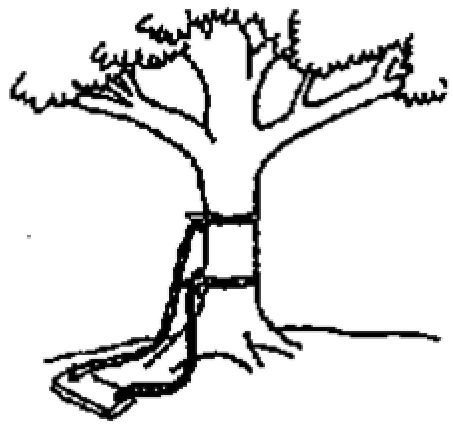
Brian Cantwell Smith, *The Limits of Correctness*, Symposium on Unintentional Nuclear War, Budapest, Hungary,1985

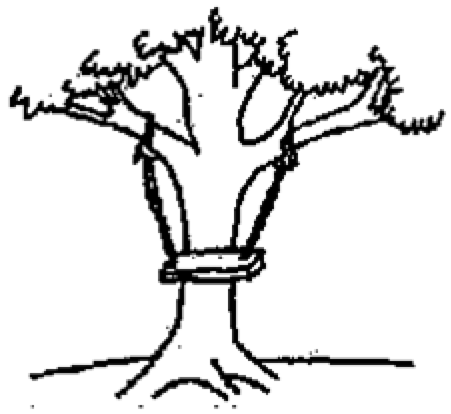
- Pretty much state of the art now, 30+ years later…

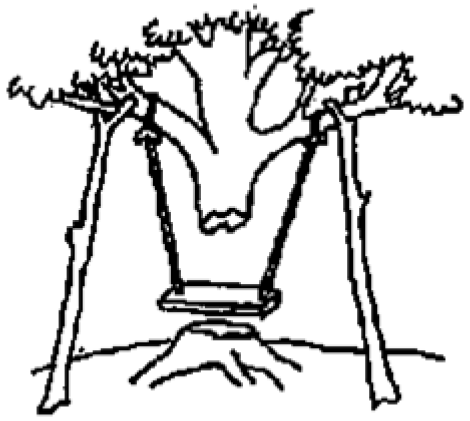
From the Real-World to Software

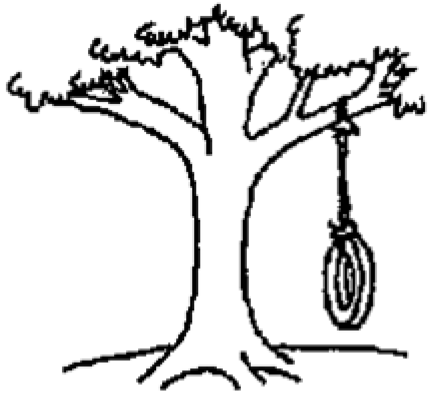

What the user asked for


As recorded in the documentation


As designed by the senior analyst


As produced by the programmers


As installed at the user's site


What the user wanted

Some Buzzwords

- **Requirements** are expressions of the needs of **stakeholders** for a **system** to achieve particular **goals**
  - Stakeholders can be anyone with an interest in the system: users, financiers, managers, technicians (Not simply those who initiate its construction, e.g. a paying client)
- Requirements typically expressed in the vocabulary of the **problem domain**, rather than the system (solution) domain
  - Problem domain is the real world

Imperial College London

Capturing Requirements

Imperial College London

Talk to Stakeholders

- Good way to capture requirements is to hold “workshops” attended by all stakeholders
  - Easier to resolve conflicts, and to agree on priorities and risks
  - A workshop for your project might include your whole group, supervisor, users of eventual system, anyone else involved (CSG? administrators?)
- Requirements are captured/updated/changed iteratively
  - Early iterations will include more of this process than later
- Document requirements as you like, but best if stakeholders and implementers both understand notation
  - Natural language, use cases, UML, etc.

Imperial College London

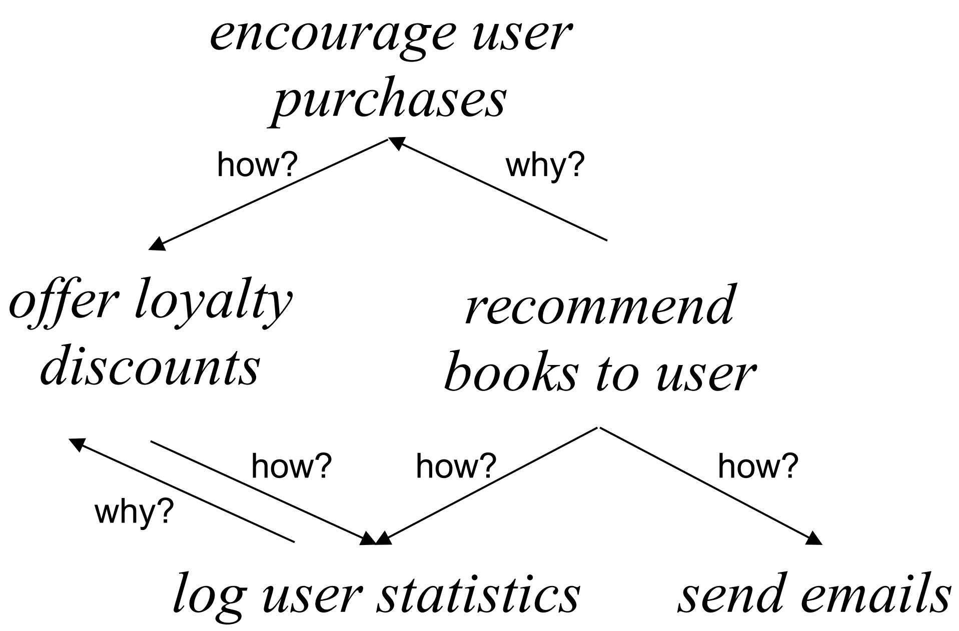
Goal-Oriented Capture

- Identify “goals” (objectives) of the system
  - Good way for developers and stakeholders to come to mutual understanding of the needs of the system
    - Why is it being built?
  - Goals will determine requirements
- For each goal, ask ***how*** or ***what*** questions to break the goal down into more specific requirements
  - Helps get at what the system really has to do
- For each requirement, ask ***why*** questions to work back to the goal behind the requirement
  - Helps get at alternative and overlooked requirements

Imperial College London

Goal-Oriented Capture

- Aim is to be comprehensive and identify everything the system is required to do



Imperial College
London

System Boundaries

- What is the environment of the system?
    - Software/hardware
        - e.g. on what hardware/OS is it expected to run?
    - System dependencies
- Who are the stakeholders?
    - Clients, users, maintainers, etc.
- Everything is a lot simpler if the boundaries of the system are established at an early stage

Feasibility

- Determine if stakeholders' requirements are feasible
  - Does your team have the needed skills (or can they be acquired in time)?
  - Does the available technology support the requirements?
  - Do you have the admin rights to implement network feature *X*?
  - Does the available graphics package support 3D visualisation?

- Don't start implementing only to discover it can't be done
- If infeasible, renegotiate requirement straight away

Imperial College London

Categorising Requirements

- High-level / low-level
  - *“Allow online booking of cinema tickets” vs. “Save current file under new name if not in read-only mode”*
- High-risk / low-risk:
  - *“Provide animation of multi-agent system in 3D with navigation and zoom” vs. “Provide command-line interface”*
- Functional / non-functional
  - *“Allow users to save their session” vs.“Allow at most 10% degradation of service under failure of one server”*
- Essential / non-essential
  - *“Allow online booking of plane tickets” vs.“Allow seat selection for booked tickets”*

Imperial College London

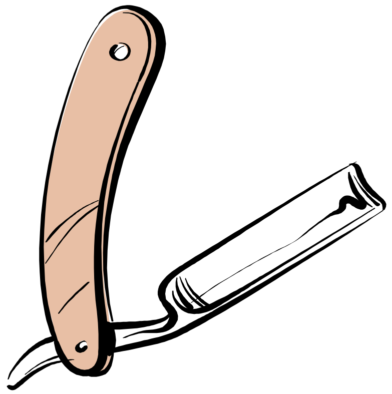
Prioritise

- Prioritise requirements that *stakeholders* deem **essential**

- Analyse high-risk requirements early
  - Determine what is or is not possible

Imperial College London

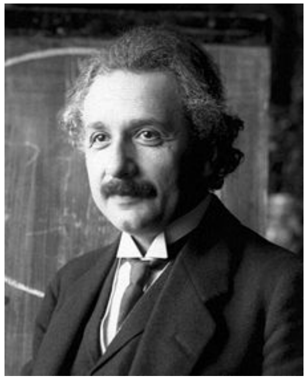
Keep It Simple: Occam's Razor



"Entia non sunt multiplicanda praeter necessitatem"
– William of Ockham (John Punch)
14th century

"Entities must not be multiplied beyond necessity"
(simplest theory is usually the best one)

"Make it as simple as possible, but not simpler"
– Albert Einstein
(paraphrasing Ockham)



Make it as simple as possible, but not simpler

- Don't over-complicate your projects
  - Distill simplest possible requirements
  - Start out with the simplest design that satisfies the requirements
    - Only once this is achieved should you think about extensions
    - And even then, only if beneficial: check with the client
- Don't make more assumptions than you need to
  - If you're unclear about a requirement, check with the client again

Imperial College London

Summary

- Capturing requirements is difficult
- Aim is to be comprehensive and identify everything the system is required to do
    - Categorise and prioritise requirements
    - Analyse boundaries, risk, and feasibility
    - Keep it simple
- Requirements determine acceptance criteria
  - Which determine how the system will be tested, and ultimately validated

Report 1

- Requirements using goal-oriented capture
- Categorise and prioritize the requirements
- Assess their feasibility and risk
- Discuss system boundaries
- Planned completion dates
- See Group Project Page for more