Requirements Engineering

Requirements Engineering (RE)

RE is the process of

- Understanding and defining what services are required from the desired system
- And identifying the constraints on the system's operation and development

The process leads to the production of a **requirements specification document**

RE addresses

- The development and validation processes for eliciting, representing, and analysing system requirements
- And the methods for transforming requirements into more rigorous specifications for design and implementation.

Requirements Engineering Process

The RE process considers the system as a whole, in interaction with its stakeholders.

Systems are usually seen (at least conceptually) as hierarchical, in terms of abstraction levels.

- A distinction is made between stakeholder requirements and scenarios (level of system seen by users, customers) and designer requirements and scenarios (lower level, constructed by requirement engineers).

The process of RE within software development is an *iterative process*

- A sharp borderline between defining requirements and constructing the system design is not always easy to draw.

In practice, the processes of *RE* and *system design* are often integrated by a careful co-operation between the two

System Identification

Purpose

- What is the main function of the system?
- Leads to system requirements

Scope

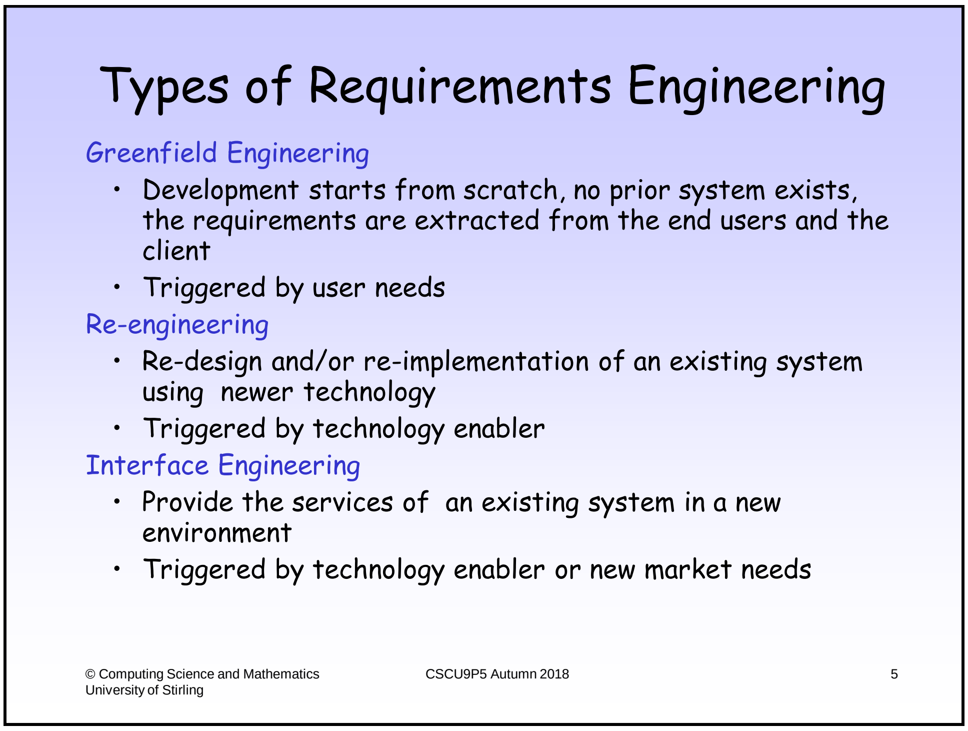
- What's included in the system?

Boundary

- What is inside, what is outside?

Context

- What is the environment of the system?
- What external entities may affect the system and its operation?



Types of Requirements Engineering

Greenfield Engineering

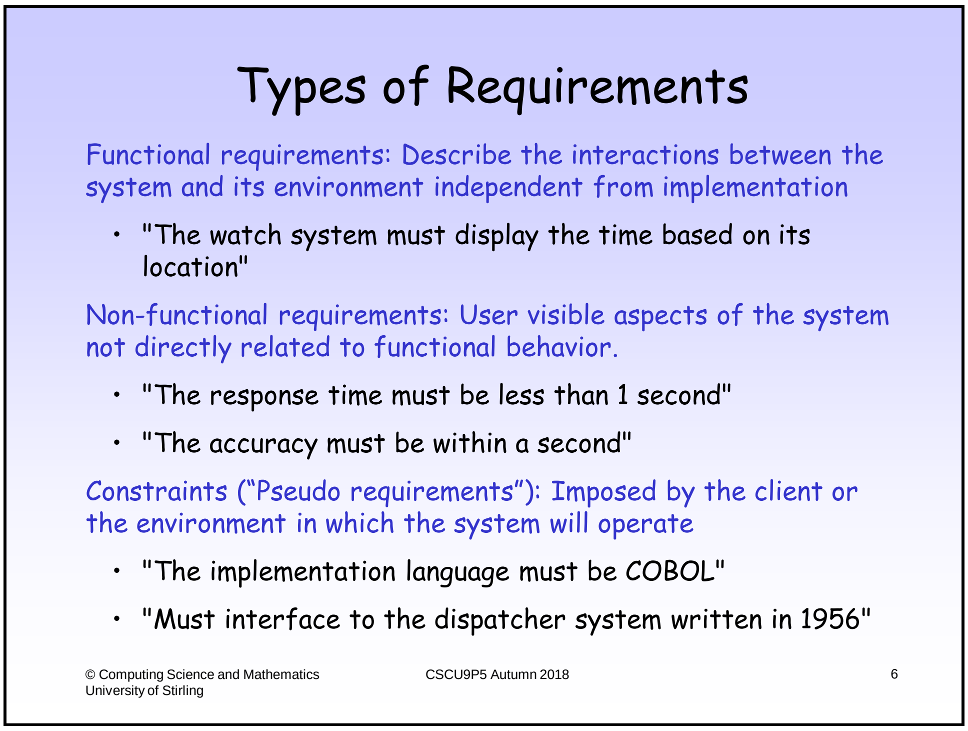
- Development starts from scratch, no prior system exists, the requirements are extracted from the end users and the client
- Triggered by user needs

Re-engineering

- Re-design and/or re-implementation of an existing system using newer technology
- Triggered by technology enabler

Interface Engineering

- Provide the services of an existing system in a new environment
- Triggered by technology enabler or new market needs



Types of Requirements

Functional requirements: Describe the interactions between the system and its environment independent from implementation

- "The watch system must display the time based on its location"

Non-functional requirements: User visible aspects of the system not directly related to functional behavior.

- "The response time must be less than 1 second"
- "The accuracy must be within a second"

Constraints (“Pseudo requirements”): Imposed by the client or the environment in which the system will operate

- "The implementation language must be COBOL"
- "Must interface to the dispatcher system written in 1956"

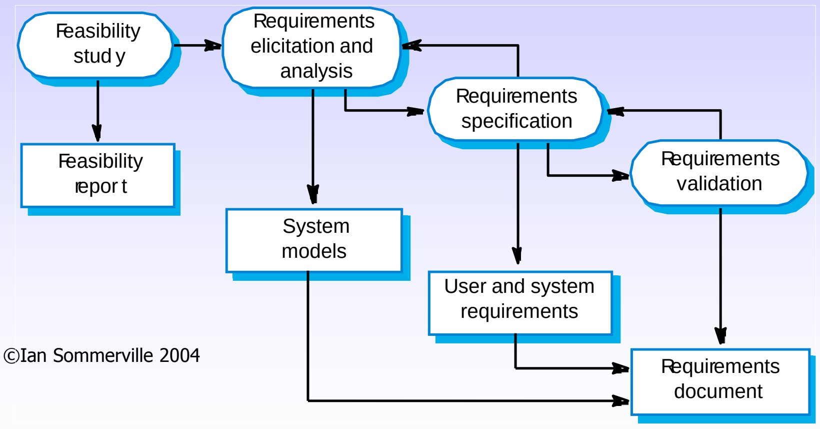
What is usually not in the Requirements?

- System structure, implementation technology
- Development methodology
- Development environment
- Implementation language
- Reusability

It is desirable that none of the above are constrained by the client...

The requirements engineering process

A process oriented view:



©Ian Sommerville 2004

Feasibility study

This should be relatively cheap and quick

This study needs to identify:

- Whether the user needs can be satisfied using current software and hardware technologies
- Whether the production will be cost effective from a business point of view, and within budgetary constraints

The report allows the client to decide whether or not to go ahead with a more detailed, and costly, analysis

Requirements Elicitation and Analysis

Definitions:

- To elicit: *"To draw out; to extract; to bring to light facts by questioning or reasoning"*
- To analyse: *"To take to pieces; to examine critically part by part"*

Two approaches to elicitation:

- Viewpoints
- Scenarios

Viewpoints and Scenarios

Viewpoints:

- A way of classifying *stakeholders* and other sources of requirements
- For example: providers of services to the system, receivers of system services, engineers who must manage and maintain the system, marketing and sales
- Requirements can be elicited from individuals in those roles

Scenarios:

- Real-life examples of how the system will be used, eg examples of intended user interaction
- Clearly closely related to *use-cases (UML)*, but more specific, less general
- These are another source of requirements and are employed to clarify requirements

Heuristics for finding Scenarios

Ask yourself or the client the following questions:

- What are the primary tasks that the system needs to perform?
- What data will the actor create, store, change, remove or add in the system?
- What external changes does the system need to know about?
- What changes or events will the actor of the system need to be informed about?

Insist on task observation if the system already exists (interface engineering or reengineering)

- Ask to speak to the end user, not just to the software contractor
- (Expect resistance and try to overcome it)

Requirements Analysis

(more in the supplementary notes following slide 21)

The *goal of analysis* is to discover problems, incompleteness and inconsistencies in the elicited requirements

- These are then fed back to the stakeholders to resolve them through the negotiation process
- The requirements specification may be rewritten

Analysis is *interleaved with elicitation*

- Problems may be discovered when the requirements are being analysed

A problem checklist may be used to support analysis

- Each requirement may be assessed against the checklist

Analysis & Validation

Analysis works with raw requirements as elicited from the system stakeholders

- "Have we got the **right requirements**" is the key question to be answered at this stage

Validation works with a final draft of the requirements document i.e. with negotiated and agreed requirements

- "Have we got the **requirements right**" is the key question to be answered at this stage
- Important because requirements error costs are high: *"Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error"* (Sommerville)

Requirements Specification: Representing Requirements

Viewpoints and scenarios are for communicating with the stakeholders. The description of the system is used

- not only to validate the understanding of the current set of requirements and scenarios,
- but also to elicit additional information from the stakeholders.

Viewpoints and scenarios must be represented in a well-structured and easy to understand manner and need to be precise enough and detailed enough to support the development process of the system

Informal representations are easily understood but are inappropriate for the input to the design construction

Formal descriptions can be manipulated automatically, enabling verification and detection of inconsistencies – but complex

In practice a *structured, semi-formal* representation can be used

Informal & Semi-formal Representations

Different informal representations can be used to express the same requirement or scenario

- For example, in a simple graphical form,
- or a natural language, or a combination of these languages

Scenarios, for instance, can be represented using a format based on flow diagrams, or in carefully stylised natural language ("pseudo-code")

More semi-formal representations are better for input to the Design phase

- For example, in UML: Use case diagrams, sequence diagrams, activity diagrams

Formal Representations

A formalization of scenarios/requirements needs a precise, unambiguous means of expressing:

- The properties of concepts/objects
- The objects involved in (inter)actions
- The relationship between inputs and outputs

Many ways of presenting these formal representations are available

- Each has a strong mathematical flavour (e.g. algebraic, logic, set theoretic)

This kind of formalization can be used

- To disambiguate requirements and scenarios
- To perform mathematical analysis/checking of the design
- To assist in automated checking/testing of the design

Sample slide from CSCU9P6: Alloy Aircraft Boarding Example

```
sig Person { }      // a set of “persons”
sig Aircraft        // a set of “aircraft”
{
        capacity : Int,     // capacity of the aircraft
        onboard: set Person         // the set of people on the aircraft
}
{
        # onboard <= capacity       // a constraint
}
```

The signature Aircraft can be used to model different physical aircraft, or different states of the same physical aircraft.

Validation techniques

(more in the supplementary notes following slide 21)

Requirements reviews

- Systematic manual checking of the requirements
- Clients/stakeholders should be involved if possible

Prototyping

- Using an *executable* model of the system to check requirements
- Clients/stakeholders can try out dummy scenarios

Test-case generation

- Developing tests for requirements to check testability
- Can expose untestable requirements

Requirements Based Testing

Each requirement should be testable

- It should be possible to define tests to check whether or not that requirement has been met

Inventing requirements tests is an effective validation technique

- Missing or ambiguous information in the requirements description may make it difficult to formulate tests

Each functional requirement should have an associated test

Hard to test Requirements

System requirements

- Requirements which apply to the system as a whole – for example performance or useability. In general, these are the most difficult requirements to validate irrespective of the method used as they may be influenced by any of the functional requirements.

Exclusive requirements

- These are requirements which *exclude* specific behaviour. For example, a requirement may state that system failures must never corrupt the system database. It is not possible to test such a requirement exhaustively.

Some non-functional requirements

- Some non-functional requirements, such as reliability requirements, can only be tested with a large test set. Designing this test set does not help with requirements validation

Further notes on Requirements Engineering

Also see Ian Sommerville's book, Chapter 4 "Requirements Engineering

Analysis Checklist

Premature design

- Does the requirement include premature design or implementation information?

Combined requirements

- Does the description of a requirement describe a single requirement or could it be broken down into several different requirements?

Unnecessary requirements

- Is the requirement 'gold plating'? That is, is the requirement a cosmetic addition to the system which is not really necessary.

Use of non-standard hardware

- Does the requirement mean that non-standard hardware or software must be used?

Analysis Checklist (cont)

Conformance with business goals

- Is the requirement consistent with the business goals defined in the introduction to the requirements document?

Requirements ambiguity

- Is the requirement ambiguous, i.e. could it be read in different ways by different people?

Requirements realism

- Is the requirement realistic given the technology which will be available at the time the system is implemented?

Requirements testability

- Is each requirement testable, that is, is it stated in such a way that test engineers can derive a test which can show if the system meets that requirement?

Requirements Validation

Requirements validation criteria:

- Correctness:
  - » The requirements represent the client's view.
- Completeness:
  - » All possible scenarios through the system are described, including exceptional behavior by the user or the system
- Consistency:
  - » There are no functional or nonfunctional requirements that contradict each other
- Clarity:
  - » There are no ambiguities in the requirements.

Requirements Validation Criteria (cont)

Realism:

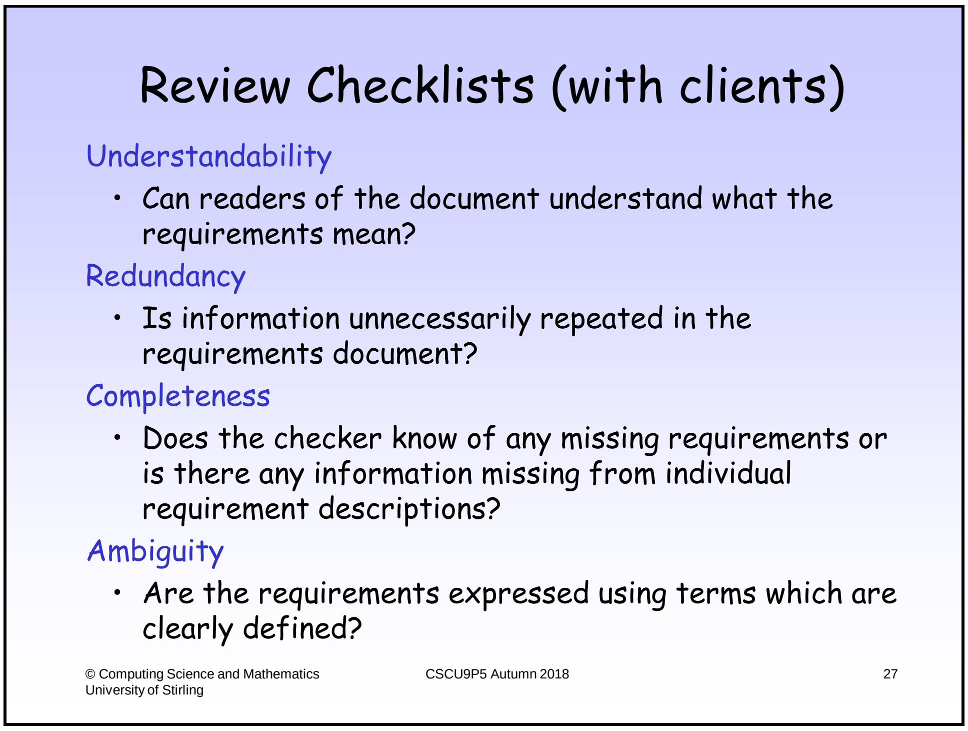
- The requirements can be implemented and delivered

Traceability:

- Each system function can be traced to a corresponding set of functional requirements

Verifiability:

- Repeatable tests can be designed to demonstrate that the system fulfills the requirements



Review Checklists (with clients)

Understandability

- Can readers of the document understand what the requirements mean?

Redundancy

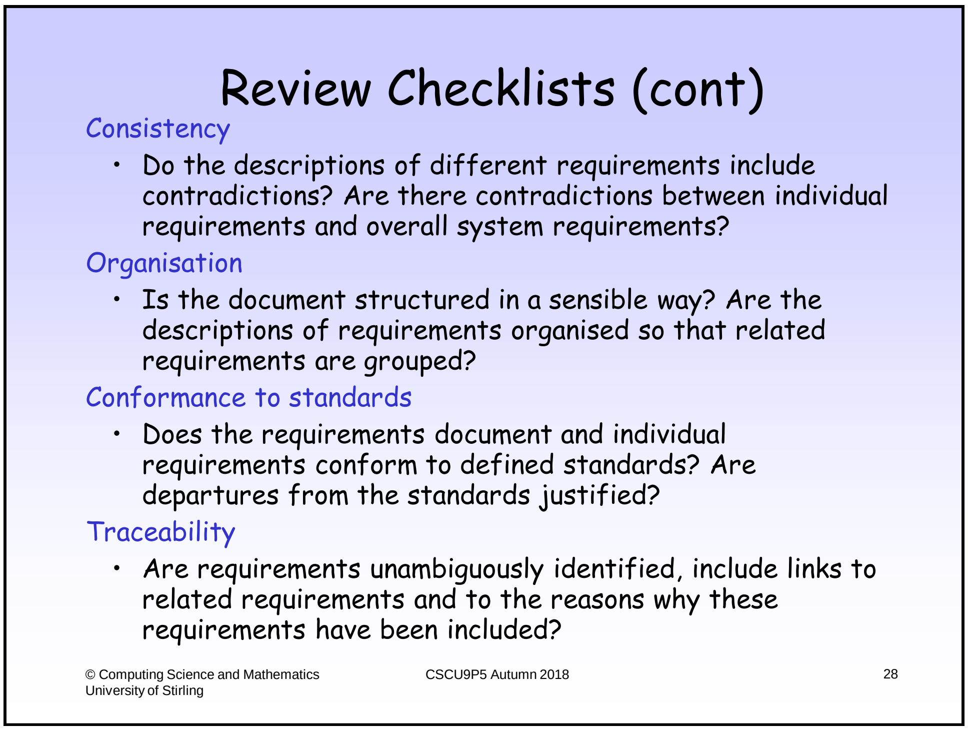
- Is information unnecessarily repeated in the requirements document?

Completeness

- Does the checker know of any missing requirements or is there any information missing from individual requirement descriptions?

Ambiguity

- Are the requirements expressed using terms which are clearly defined?



Review Checklists (cont)

Consistency

- Do the descriptions of different requirements include contradictions? Are there contradictions between individual requirements and overall system requirements?

Organisation

- Is the document structured in a sensible way? Are the descriptions of requirements organised so that related requirements are grouped?

Conformance to standards

- Does the requirements document and individual requirements conform to defined standards? Are departures from the standards justified?

Traceability

- Are requirements unambiguously identified, include links to related requirements and to the reasons why these requirements have been included?

End of lecture