**[INTRODUCTION]**

**Types of Systems**

* Soft – difficult to define precisely
  + Depends on viewpoint of person describing it
  + Difficult to agree on boundaries and behaviour
  + Eg, all human activity systems
* Hard – well defined
  + Easy to agree on boundaries
  + Eg, mechanical operation of car

**Problem** **statement** – clear and concise description of the issue(s) that need(s) to be addressed by a problem solving team. Issues, Goals and Objectives, Method, and Scope.

**Requirements** **engineering** – set of activities concerned with identifying and communicating the purpose of a software-intensive system and the context in which it will be used

Requirement analyst must identify the problem/opportunity

* Which problems need to be solved? (problem boundaries)
* Where is the problem? (context/problem domain)
* Whose problem is it? (stakeholders)
* Why does it need solving? (goals)
* How might a software system help? (scenarios)
* When does it need solving? (development constraints)
* What might prevent us solving it? (feasibility/risk)

Typical problems in requirement specifications:

* Noise
* Silence
* Over specification
* Contradiction
* Ambiguity
* Forward reference

Things you want:

* Valid/correct
* Unambiguous
* Complete
* Understandable
* Consistent
* Ranked
* Verifiable
* Modifiable
* Traceable

**Requirements** **Specification** – Purpose

* Communication
* Contractual
* Baseline for evaluating the software
* Baseline for change control

Re**q**uirements **Specification** – Audience

* Customers and users
* Systems analysts
* Developers/programmers
* Testers
* Project managers

Specification should include

* Functionality
* External influences
* Required performance
* Quality attributes
* Design constraints

**Critical** **System** – a computer, electronic or electromechanical system that the failure of which may have serious consequences (human death/injury, failure of goal directed activity, financial loss/failure of business)

* Safety-critical systems
* Mission-critical systems
* Business-critical systems

**Validation** – ensure that the specification of the software system satisfy the requirements of the stakeholders

**Verification** – ensure that the finished product satisfies the specification

**[REQUIREMENTS ANALYSIS]**

Requirements 🡪 Design 🡪 Implementation 🡪 Verification 🡪 Maintenance

**Requirements** – description of something a product must do/quality it must have

* Functional – things the product must do or provide to be useful for users
* Non-functional – qualities the product must have
* Design – how the system will be designed and implemented

**Constraints** – global issues that shape the requirements

**Assumptions** – describe properties of the environment of a software product that the software can’t directly control, but necessary for overall functioning.

**Verifiable** – A requirement should be verifiable.

* Should be a clear way of checking that an implementation can satisfy requirements
* Eg, by defining test cases for that requirement

**PROCESS**

**1. Scope** – provides a high level view of the planned product: Need, Goals, Case, High-level operational concepts, Stakeholders. Purpose:

* Helps keep bigger picture in view
* Ensure all stakeholders have a common view
* Ideally should remain fixed throughout development process

**2. Use Case** – abstractions from scenarios for usage of the system typically describing series of interactions between users and the system to be developed.

* Help people quickly gain an understanding of what is intended
* Convenient way of generating requirements
* Helps to avoid omitting requirements

**3. Interfaces –** points where the system interacts with environment (users, other software components, physical devices)

* Helps to define boundaries of responsibility for the development teams
* Important for defining ways of testing implemented systems

**[Software Modelling and UML]**