Daniel **Fitz**

43961229

University of Queensland

CSSE3002 – The Software Process

Lecture 1

Table of Contents

Software Engineering

Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintence of software.

- This is, the application of engineering to software
- IEEE Standard 610.12-1990 Concerned with theories, methods and tools that enable professional software development. "The topic that we call software engineering is both exciting and frustrating. Exciting because it draws on many technical disciplines and provides a harness that binds each discipline to the next. Frustrating, because it demands knowledge in a multitude of topic areas and seems to be infinitely expandable." (Roger Pressman, 1992) Discipline behind the process (you actually know what you're doing and why) Being able to
- choose appropriate tools and techniques
- work effectively in a team
- manage your own work and the process

Software Engineering Process

A structured set of activities followed to develop software system.

- Tools
- Methods
- Practices

A lot of companies have fucked up

Well Engineered Software

- Usable
- Dependable
- Maintable
- Efficient
- How do costs come into this?
  - Trade-offs may be involved
    - ∗ appropriate
    - ∗ cost-effective

Process Models

- Abstract representation of a process
- Plan Driven
  - Structured / Traditional
- Incremental
- Agile
- Lean
- Formal

Plan Driven Processes

- Waterfall
- V Model
- Spiral

Waterfall

- Introduced iteration between phases

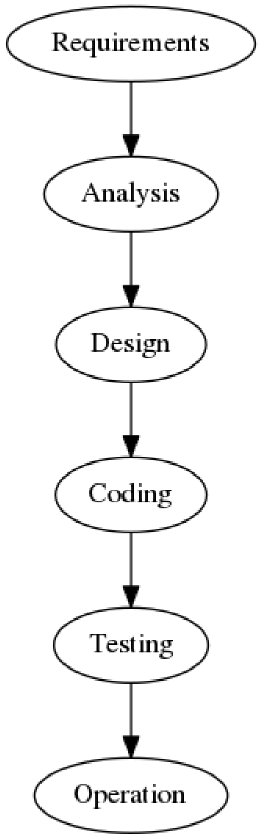

Figure 1: Diagram explaining Waterfall

- Prototyping
  - Requirements
  - Design
- See:
  `http://www.allaboutagile.com/dr-royce-and-waterfall/`

V Model

Spiral

- Focus on process control
- See
  `http://csse.usc.edu/TECHRPTS/1988/usccse88-500/uscsse88-500.pdf`

Incremental Processes

- Unified Process
- OPEN
  - Object-oriented Process Environment and Notation

Unified Process

Unified Process is allied closely with UML

- Four distinct phases
  - Inception, Elaboration, Construction and Transition
- Considers activity balance across workflows and phases

OPEN

- Process framework
  - process is instantiated from the framework
  - metamodel documents the framework
- Contracts between components
  - process Construction
  - scheduling

Agile Processes

- Embrace change
  - Requirements are never fixed
  - Stop pretending, and get used to it
- Deliver early and deliver often
  - A working system delivers value
    * telephone book scale documentation does not
  - A deployed system generates revenue
  - 80:20 rule

Lean Development

- More a philosophy than a process
  - Think big
  - Act small
  - Fail fast
- Eliminate Waste
- Amplify Learning
- Decide as Late as Possible
- Deliver as Fast as Possible
- Empower the Team
- Build Integrity In
- See the Whole

Formal Processes

- Application of mathematical formality to software development
  - formal specification

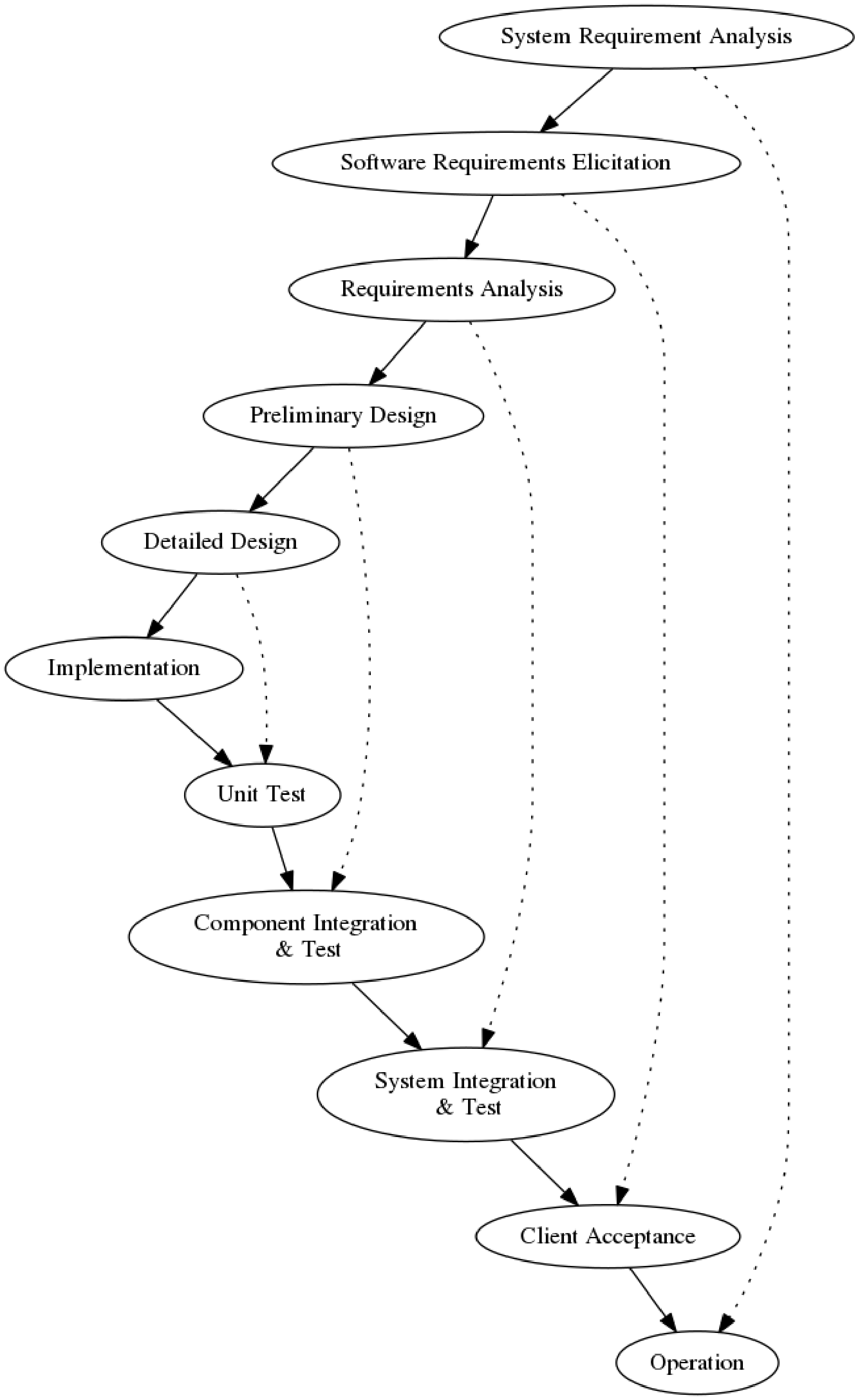
System Requirement Analysis
Software Requirements Elicitation
Requirements Analysis
Preliminary Design
Detailed Design
Implementation
Unit Test
Component Integration & Test
System Integration & Test
Client Acceptance
Operation

Figure 2: Diagram explaining V Model

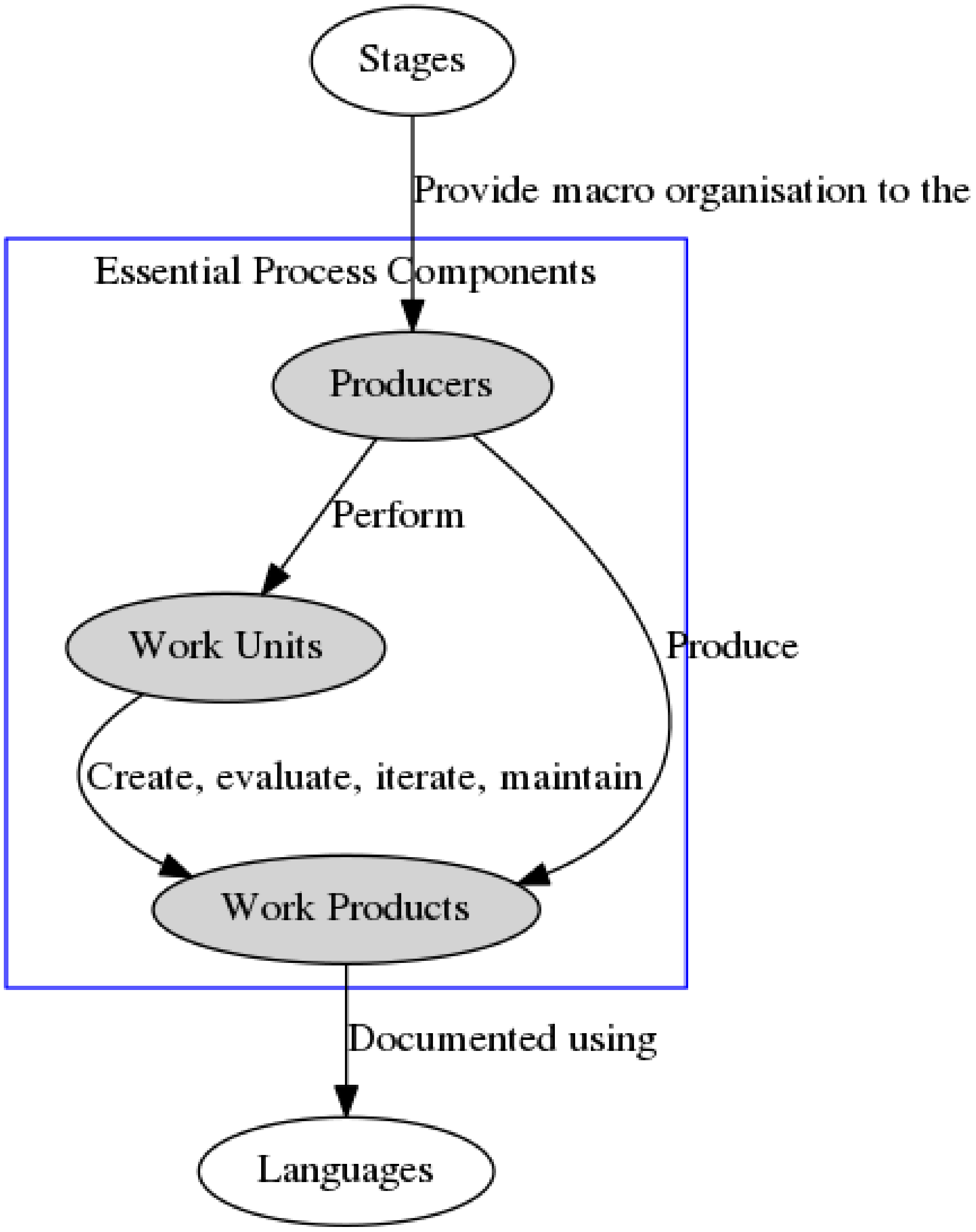
Stages
Provide macro organisation to the
Essential Process Components
Producers
Perform
Work Units
Produce
Create, evaluate, iterate, maintain
Work Products
Documented using
Languages

Figure 3: Diagram of OPEN Process

    - transformation of specification to code

Process

- All Software Engineering Processes involve phases
    - Requirements
    - Design
    - Development (implementation, coding)
    - Testing (Verification)
    - Delivery and Maintence
- These are never disjoint, never just sequential
- We iterate between them, and we blur the distinctions because we want to get it right
- Software Engineering cannot work without a defined development process
    - anything else is randomised hacking
- Processes cannot work if they are not usable
    - people don't read telephone books cover to cover
- Good processes should engage the team
    - support technical excellence and innovation
    - embed a culture of trust and reponsibility
        - no place to hide

Standards

- Rules, guidelines and heuristics
    - assist in achieving "good" practice
        - not best practice
- De facto - implicit agreement
    - easily changed
- De jure - formal agreement
    - usually debated & documented

Standard Adoption

- Voluntary
    - achieving good practice
    - safety net
- Required
    - demands of clients
    - certification requirements
    - follow on from other standards
    - process improvement activity

SE Standards

- Normative and informative
- Document centred
- Adaptable

Main SE Standards

- ISO/IEC 122207:2008
    - *Systems and software engineering – Software life cycle processes*
    - Framework for lifecycle modelling

    - Focus on bespoke software
        * including product and services
    - Includes process for defining, controlling and improving software processes

  (Last reviewed in 2013)
- ISO/IEC/IEEE 15288:2015
    - *Systems and software engineering – Software life cycle processes*
    - Framework for process descriptions
    - Focus on system engineering
        * software as a component of system
    - Focus on bespoke system development
    - Includes process for defining, controlling and improving processes

  (Ratified in 2015)
- ISO/IEC/IEEE 15289:2015
    - *Systems and software engineering – Content of life-cycle information items (documentations)*
    - Standard project documentation
    - Focus on purpose and content
        * not necessarily a formal document
            · e.g. central data repository

  (Ratified in 2015)

Compare 12207 and 15288

- 15288 focusses on systems
    - hardware, software, people, facilities, material, . . .
- 12207 focusses on software
    - intended to be used for software component of 15288

IEEE Standards

- Terminology
- QA Plans
- Configuration Management
- Requirements Specification
- Unit Testing
- V&V
- Reviews & Audits
- Productivity Metrics
- Quality Metrics
- Project Management Plans
- User Documentation
- Maintence

Ethics

Code of Ethics

- Agreed standard of behaviour
- Mark of professionalism
    - most professional bodies have one
- Enforceable?

ACS

- Primary of Public Interest
    - place interests of public above personal, business or sectional interests
- Enhancement of Quality of Life
    - strive to enhance quality of life of those affected

- Honesty
  - honest representation of skills, knowledge, services and products
- Competence
  - work competently and dilidently for stakeholders
- Professional Development
  - enhance your own development and your colleagues and staff
- Professionalism
  - enhance integrity of the ACS and respect of members for each other

Requirements Project

- Software Requirements Specification
  - template will be provided
  - contents
    - functional & non-functional requirements
    - requirements & analysis models
    - risk management plan
    - size, time & cost estimates
    - UAT
- Stakeholder’s Project Outline

eVisa Issues

- Security
- Privacy
- Translation
  - Auto and Manual
- Interfaces to other departments & authorities
- Data Mining
  - Applications
  - Applicants

eVehicle

- Battery swap for electric vehicles
- Batteries owned separately to vehicles
- Swap done at a service station
- Charged for energy used
- Battery can be charged in vehicle
  - Vehicle owner pays maintence fee to battery owner

eVehicle Issues

- Tracking batteries
- Billing system
  - Vehicle owners
  - Battery owners
  - Service owners
  - Energy suppliers
- Security
- Privacy

Project Charter

- Goal & Objectives

    - reason for doing work
- Summarises project strategy
    - from a business perspective
- Set direction for project
- Scope
- Generate “buy-in”
- Usually produced by sponsor

Goals

- High-level
- What project will accomplish
- One
    - sometimes more

Objectives

- Specific
- Supports a goal
    - think “how” it does this
- Describe with an action verb
    - measurable
    - address project end result

Goals & Objects Example

- Data mining of insurance claims
- Goal
    - Increase identification of fraudulent insurance claims by 20%
- Objective
    - Identify common characteristics of false theft claims

SMART

Specific

- what is to be accomplished
- only essential aspects

Measurable

- need success/completion criteria

Agreed-upon

- common understanding amongst stakeholders

Realistic

- achievable with available resources

Time-based

- realistic deadline

What is Requirements Engineering?

- Requirements engineering is a term often used for a systematic approach to acquire, analyse, validate, document and manage requirements
- Typically implemented as a cyclic or iterative process
- Requirements validation may include prototype Construction and evaluation
- Applied at both system and software levels, often with interleaved system architecture design

What is a 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 document

3) A documented representation of a condition or capability as in 1 or 2

- From IEEE Standard Glossary of Software Engineering Terminology

Requirements Engineering Products

- Primary outcome is a requirements specification
  - Essentially a contract between user and developer
  - Basis for all subsequent development and verification processes
- Secondary outcome is usually system and software acceptance test criteria

Why is RE important?

- Most faults observed in a software project are from incorrect, incomplete, or misinterpreted functional specifications or requirements.
  - Glass' Law *It is expensive to fix an error the later it occurs in a project*
- Helps earlier detection of mistakes, which are much more costly to correct if discovered later
- Forces clients to articulate and review requirements
- Enhances communications between participants
- Helps record and refine requirements
- All this is about producing good requirements

What happens if the Requirements are Wrong?

- System may be delivered late and cost more than originally expected
- Customer and end-users are not satisfied with the system
  - May not use its facilities or may even decide to scrap it altogether
- System may be unreliable in use with regular system errors and crashes disrupting normal operation
- If the system continues in use, the costs of maintaining and evolving the system are very high

Benefits of Good Requirements

- Agreement among developers, customers, and users on the job to be done and the acceptance criteria for the delivered system
- A sound basis for resource estimation
  - cost, personnel quantity and skills, equipment, and time
- Improved system usability, maintainability, and other quality attributes
- The achievement of goals with minimum resources
  - less rework, fewer omissions and misunderstandings

Advice / Perspective

- … a systematic approach to finding, documenting, organizing, and tracking the changing requirements of a system [RUP]
- In practice it is impossible to produce a complete and consistent requirements document [Somerville]
- Getting the requirements right is critical for success
- Requirements are rarely right at the start of a large project
  - Expect change
  - Manage it
  - Agile mantra "Embrace Change"

Functional Requirements

- Requirements (or capabilities) for functions (specific behaviour) that must be performed by the system
  - e.g. read a bar code, change a username, maintain a temperature
- Primary focus of most requirements activities

Non-Functional Requirements

- Constraints on performance or quality
- **Product Properties**
  - Requirements on the behaviour of the product
    - System shall process a minimum of 8 transactions per second
    - User credit card details shall be secured
- **Process Properties**
  - Requirements on the practices used to develop / produce the system
    - Control software shall be verified in accordance with IEEE STD 1012-1998

Classification of Non-Functional Requirements

According to the ISO standard - Safety requirements - Security requirements - Interface requirements - Human engineering requirements - Qualification requirements - Operational requirements - Maintence requirements - Design constraints

Non-Functional Requirements Examples

- Safety Requirements
  - The system shall not permit operation unless the operator guard is in place
- Security Requirements
  - Only the system administrator can change system data
  - All system data must be backed up every 24 hours
- Interface Requirements
  - The system's interaction with other existing or proposed systems
    - e.g. specific databases, API's for other systems
- Human Engineering Requirements (usability)
  - Adequacy requirements – system does what is required
  - Learning requirements – time needed to learn the facilities of the system
  - (Error) handling requirements – error rate of the end-users
  - Recovering requirements – time to restart after system failure
- Qualification Requirements (set V&V targets)
  - The Queensland government requires certification to the quality standard ISO 9001 (Quality management systems) for its major software suppliers
- Operational Requirements
  - (Components of) an industrial or military control system may have to withstand extreme heat or cold, to survive vibration, movement, sudden impact, etc
  - System efficiency or performance
    - limits on memory and processor speed are often consequences of the operational environment
- Maintence Requirements
  - Software readability, flexibility/portability and testability
- Design Constraints
  - Use database X because of user familiarity
  - Use algorithm Y for function Z because of user preference

Requirements

- There is a relationship between the quality / cost / timeliness / etc. of the product and the quality of the process
- Both functional and non-functional requirements are essential for successful software
  - Both must be verified
    - Consequently they must be testable
      - Non-functional Requirements should be measurable

The Requirements Engineering Process

See Figure 4

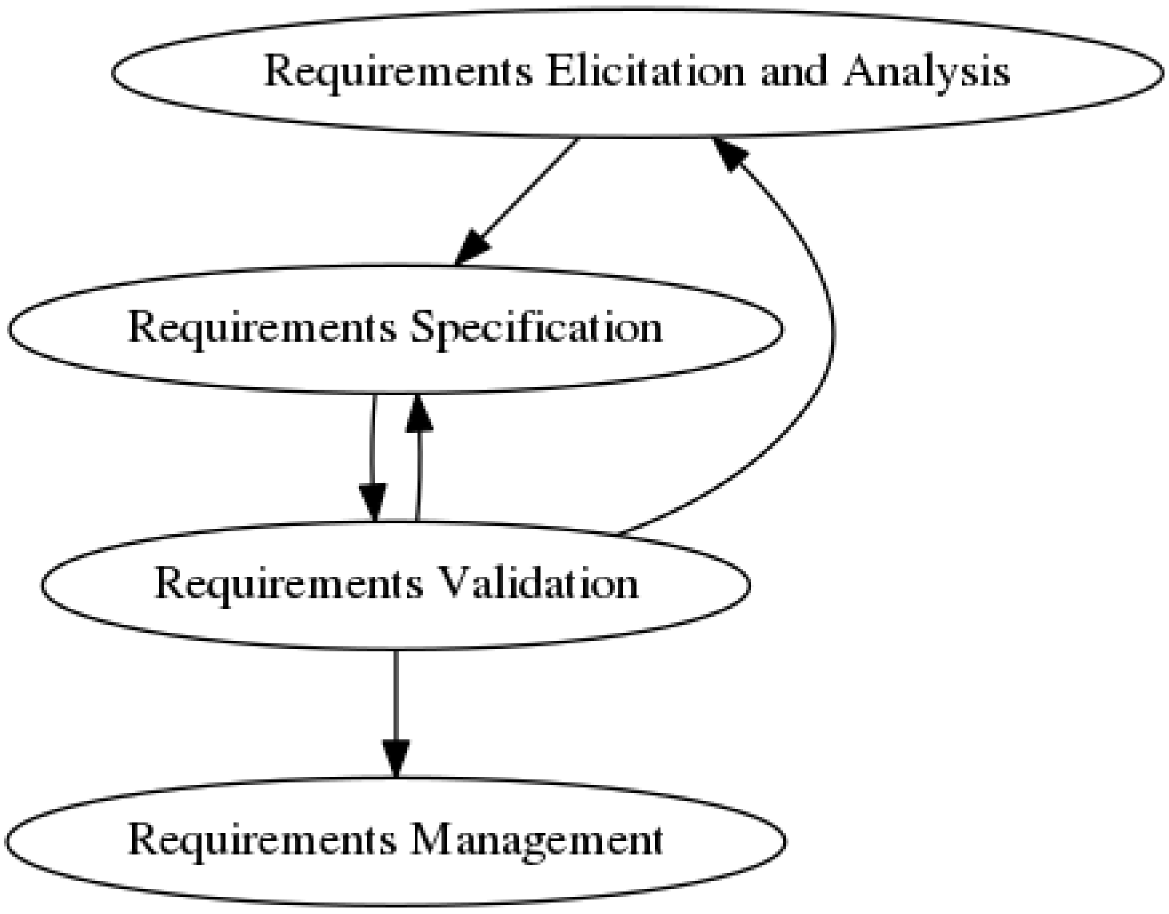
Requirements Elicitation and Analysis
Requirements Specification
Requirements Validation
Requirements Management

Figure 4: The Requirements Engineering Process

Who does it?

User organization (Domain competent)

Universities prepare requirements specifications for student enrolment systems (Si-net)

Developer organisation (IT competent)

IT companies (e.g. Technology 1) prepare requirements specifications for their clients

Third party organisation (Domain and IT competent)

Consulting companies (e.g. Accenture) prepare requirements specifications for their clients

Sources of Requirements

Users

e.g. customers or end-users – user requirements

Non-Human Sources

e.g. other devices or systems in the environment

Other Stakeholders

e.g. marketing experts, regulators, managers, business owners, developers