Requirements Engineering

Daniel Siahaan
Sept 2020

Content

- Introduction to Requirements Engineering
- Stakeholder Perpective
- Scenario
- Requirements Elicitation
- Requirements Analysis
- Requirements Specification
- SMART
- Requirements Verification and Validation
- Requirements Management (S2)

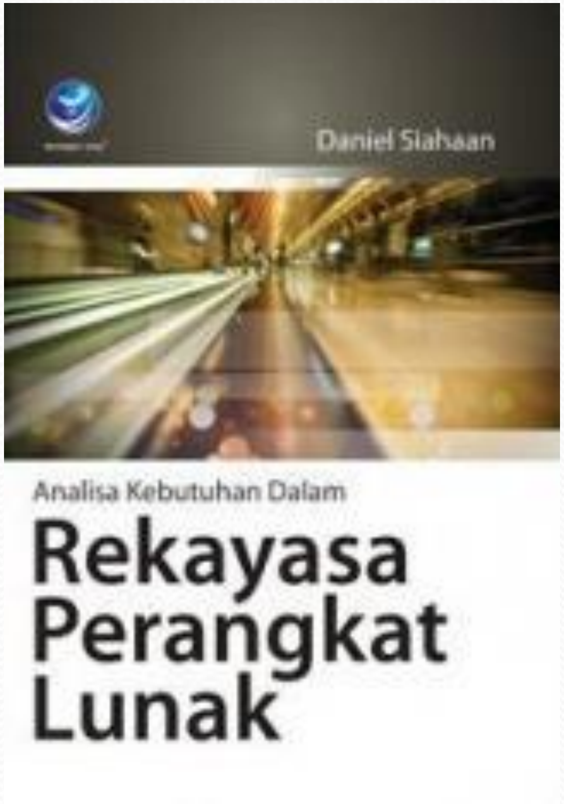
Introduction to Requirements Engineering

Daniel Siahaan

Sept 2020

References

- Daniel Siahaan, “Analisa Kebutuhan dalam Rekayasa Perangat Lunak, “ Penerbit Andi, 2012.

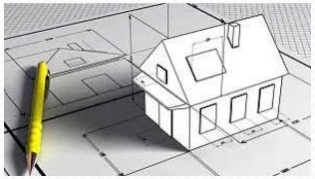


Supporting References

- R.H. Thayer dan M. Dorfman, *Software Requirements Engineering*, *Second Edition*, John Wiley & Sons, 1999.
- Ian K. Bray, *An Introduction to Requirements Engineering*, Addison Wesley, 2002.
- Karl E Wiegers, Software Requirements, Microsoft Press, 2nd Edition, 2003.
- Ian Sommerville and Pete Sawyer, Requirements Engineering: A Good Practice, Chichester England,: John Wiley & Sons, 1997.

What is Engineering?

Deliverable



Design

"The contract is very clear. You're free to go once the project's completed."

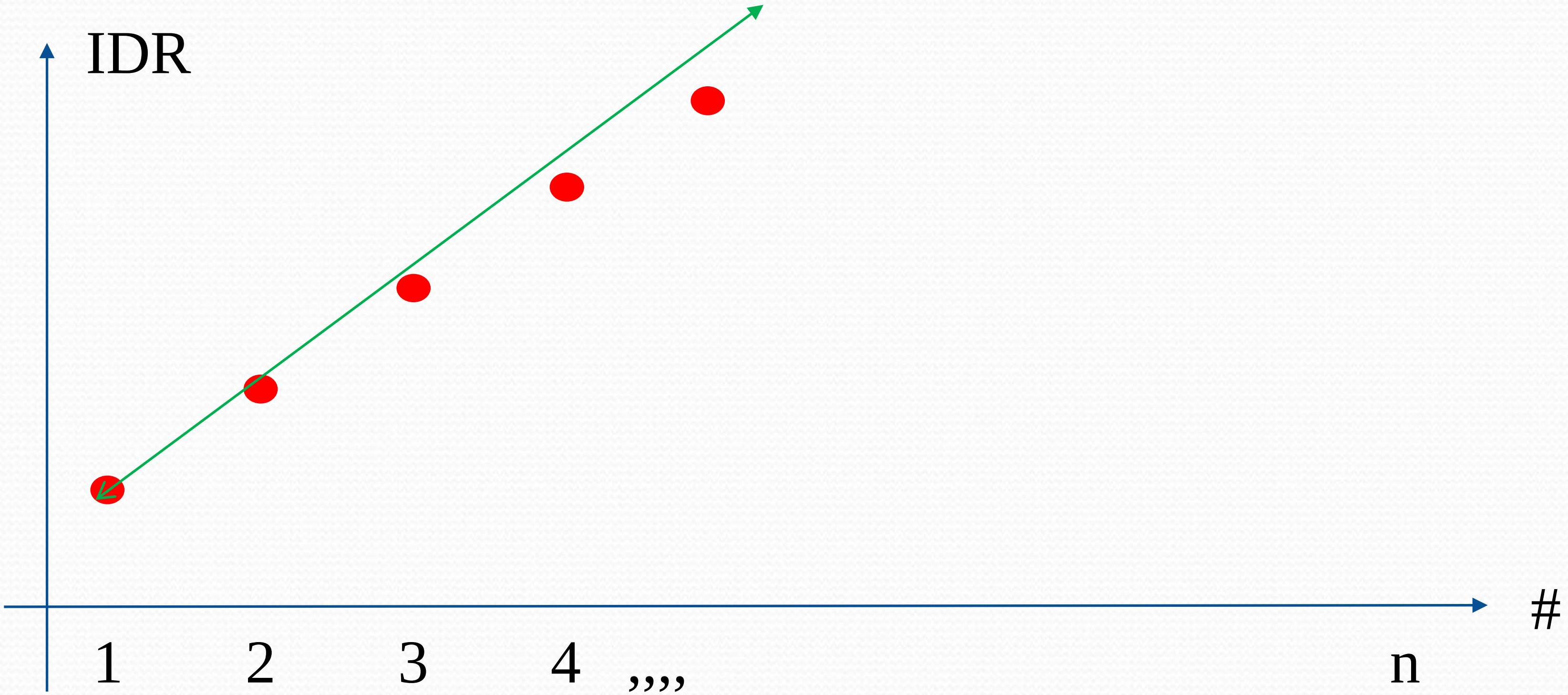
CARTOONSTOCK.COM CARTOONCOLLECTIONS.COM

Deals

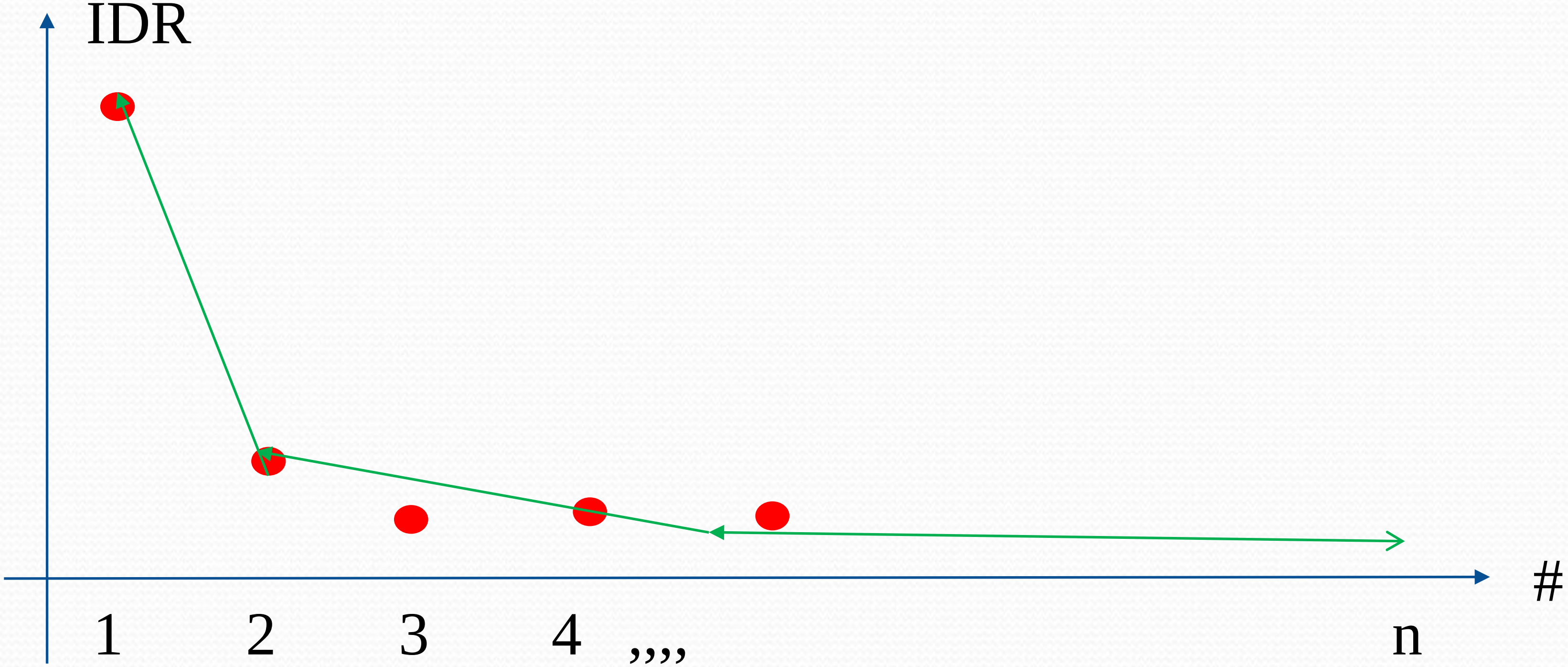
D roles

D techniques/best-practices

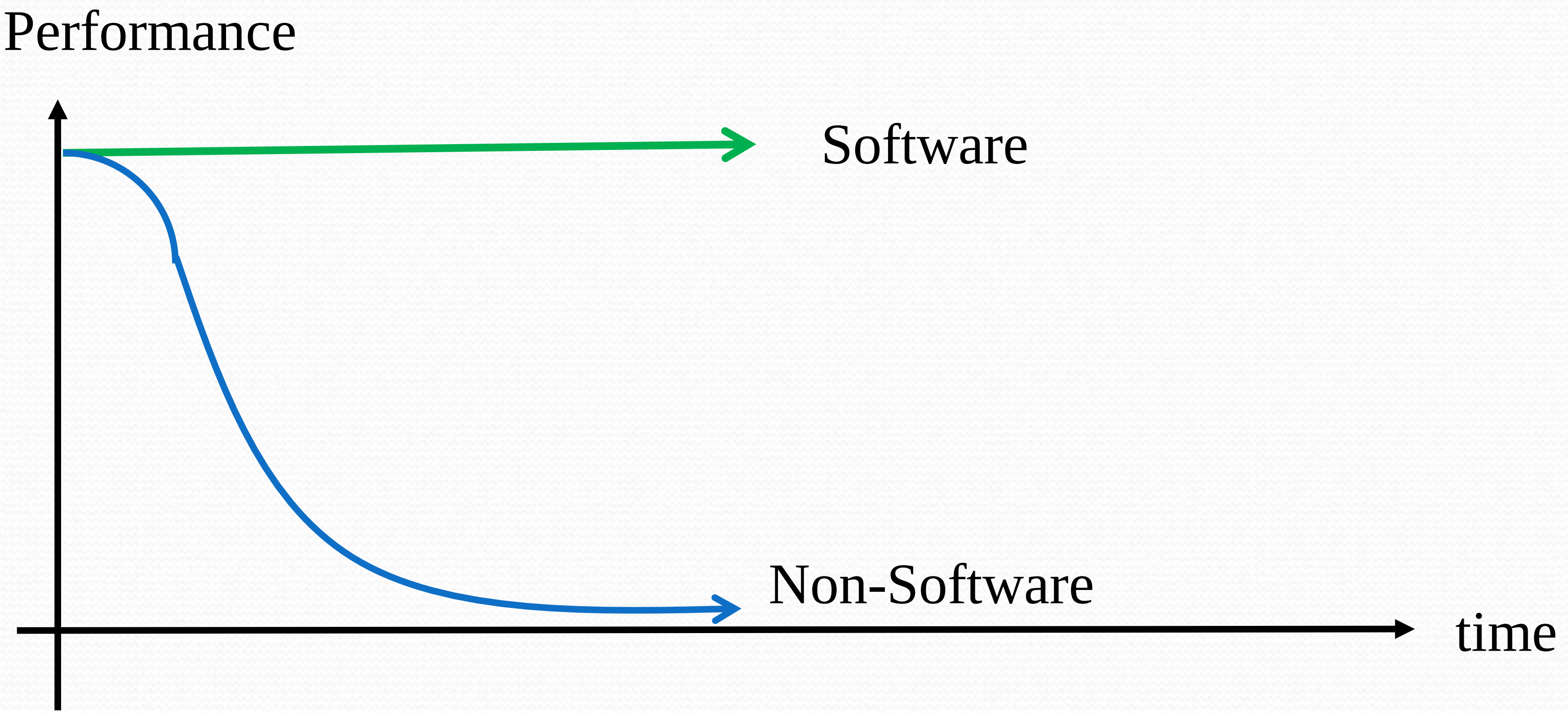
What is Engineering?



What is Engineering?



What is Engineering?



What is Requirements?

Requirements are a specification of what should be implemented.

(Sommerville and Sawyer, 1997)

A set of condition and capabilities required by user to solve a problem or finish a task, or need to be provided or present in a system in order to fulfill a contract, standard, specification, or other formal document.
A documented representation of condictions and capabilities aforementioned

(IEEE Standard Glossaary of Software Engineering Technology, 1977)

A requirement is a singular documented need of what a particular product or service should be or do. *(Wikipedia, August 2009)*

- Necessary Attributes/Properties, Characteristics, Capabilities, Quality , and Constraints
- In order to have value and utility to a user

What is Engineering?

We are not Science

- Best Practices
- Techniques and methods based on experiences

What is Requirements Engineering?

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed (Ian Sommerville, Software Engineering, 5th Edition, 1995)
- **Investigating** and **describing** the problem domain and requirements and designing and **documenting** the characteristics for a solution system that will meet those requirements (Ian K. Bray, An Introduction to Requirements Engineering, 2002)

What is Requirements Engineering?

- Set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. (Steve Eastbrook, Dept. Computer Science, Toronto University)

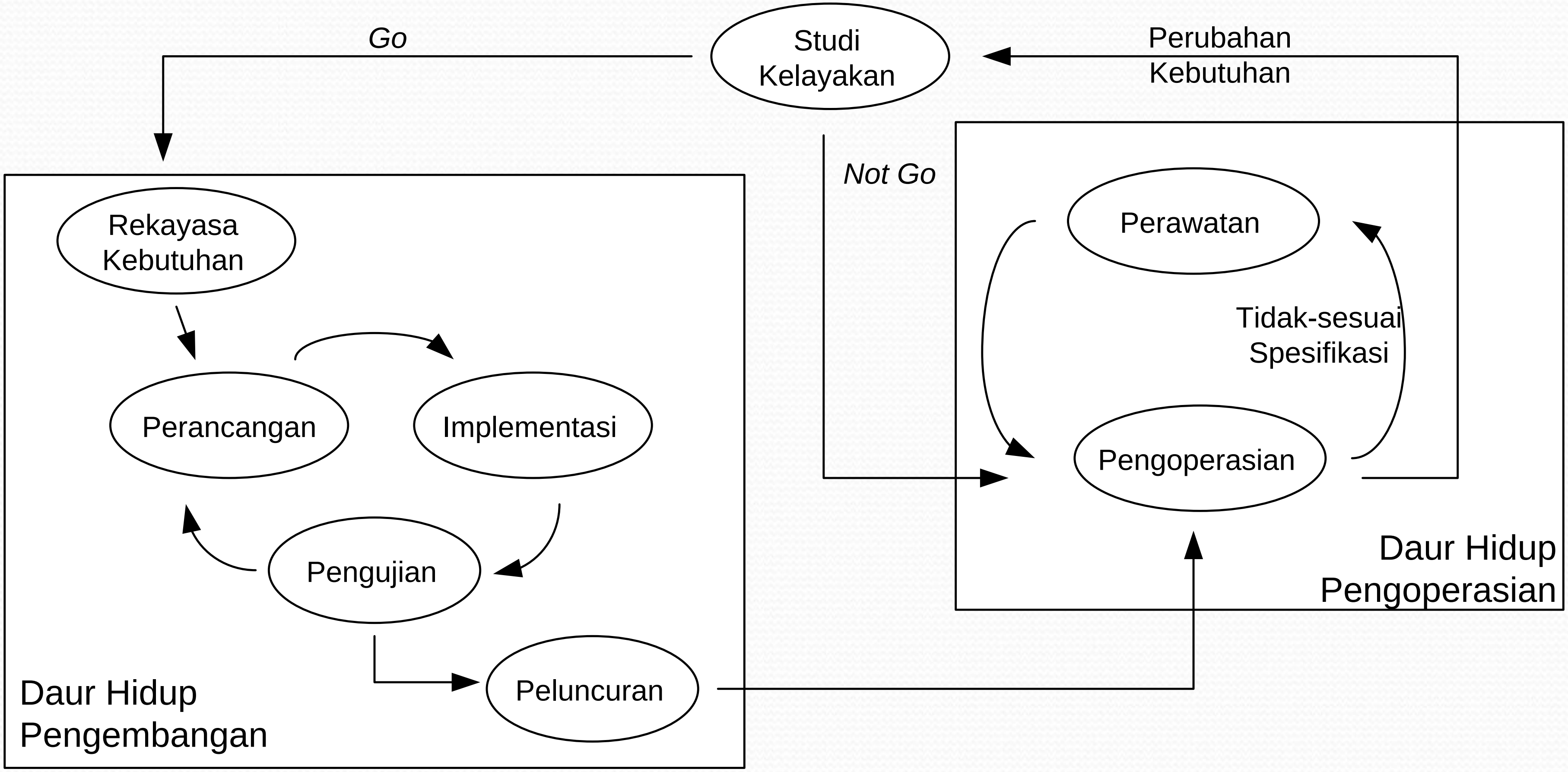
What is Requirements Engineering?

- Investigating and identifying
- Communicating and Documenting

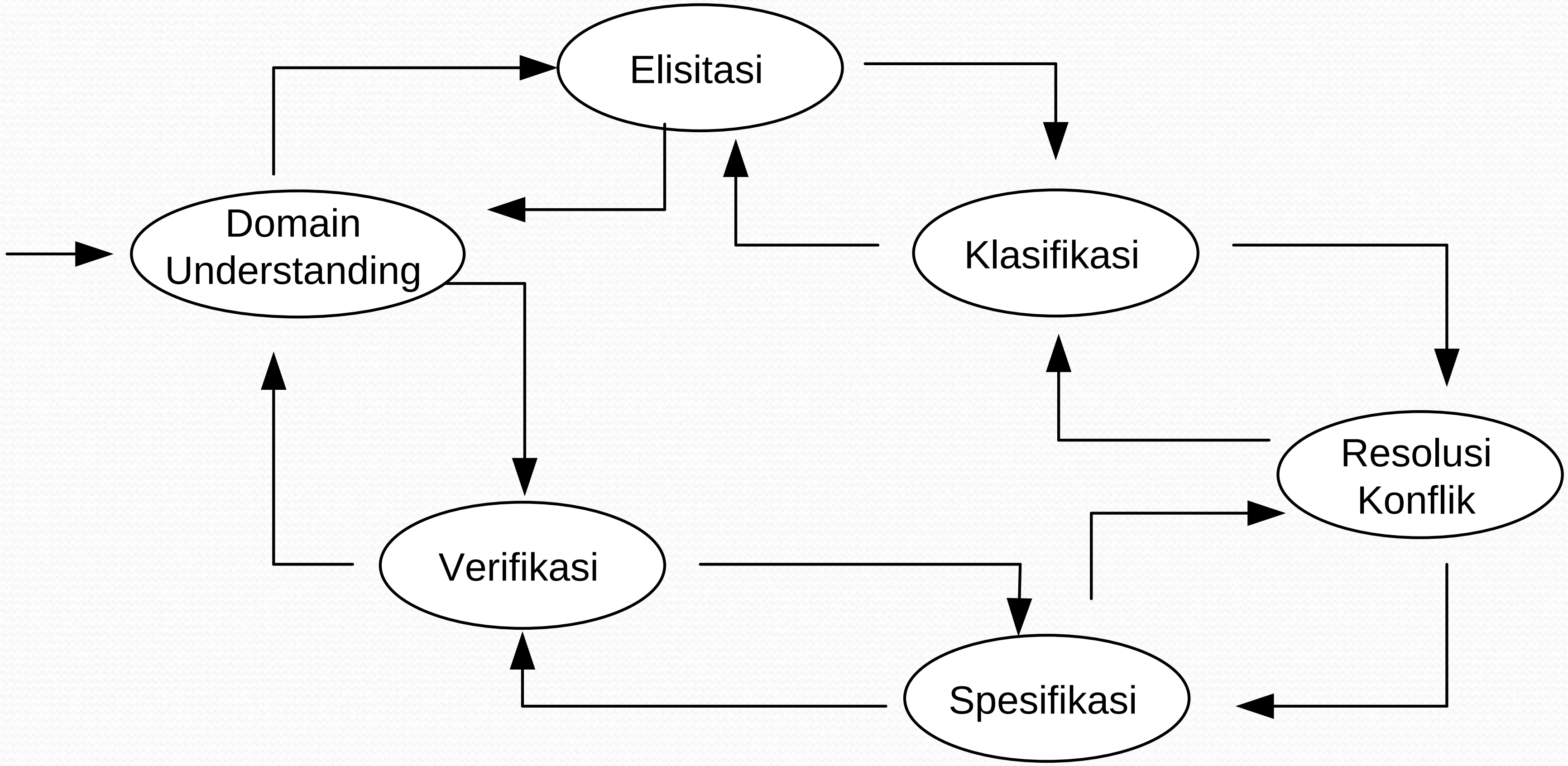
What requirements are not?

- Design and implementation Details
- Project planning information
- Testing information

Software Life Cycle



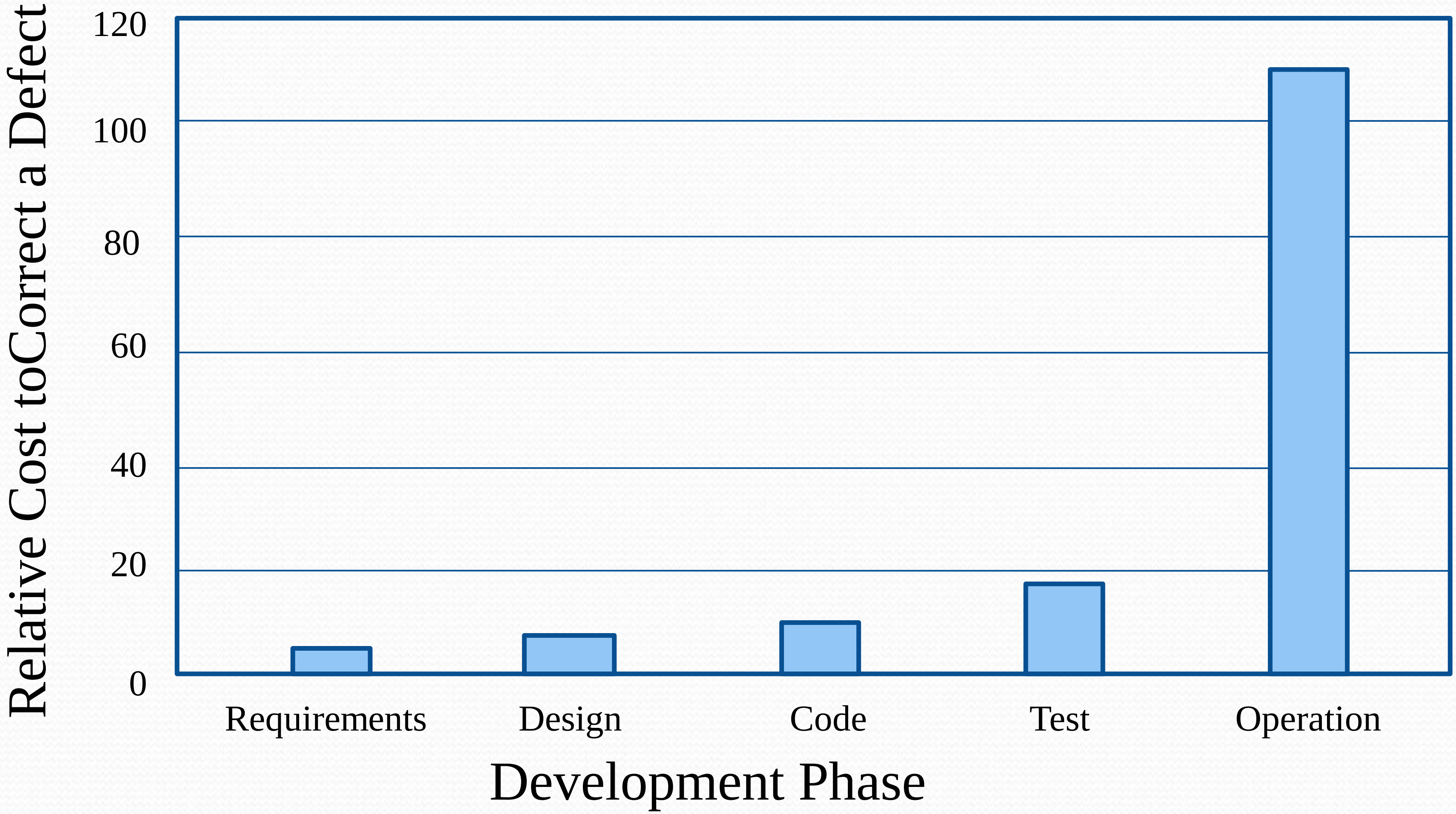
Activities in Requirements Specification



Why Re is important?

- Any software system has specification
- It's started from requirements
  - Davis (1993) Leffingwell (1997): 40 - 60% defects rooted back to requirements specification
  - Brooks (1987): most of the time over budget, late, contain defects , or not reliable
  - Jones (1991): single major cause is requirements deficiencies
  - Hofmann &Lehner (2001): deficiencies are distributed among domain of process, technology, and human resources.

Why RE is Important?



Source: Wiegers, 2003

Why Re is important?

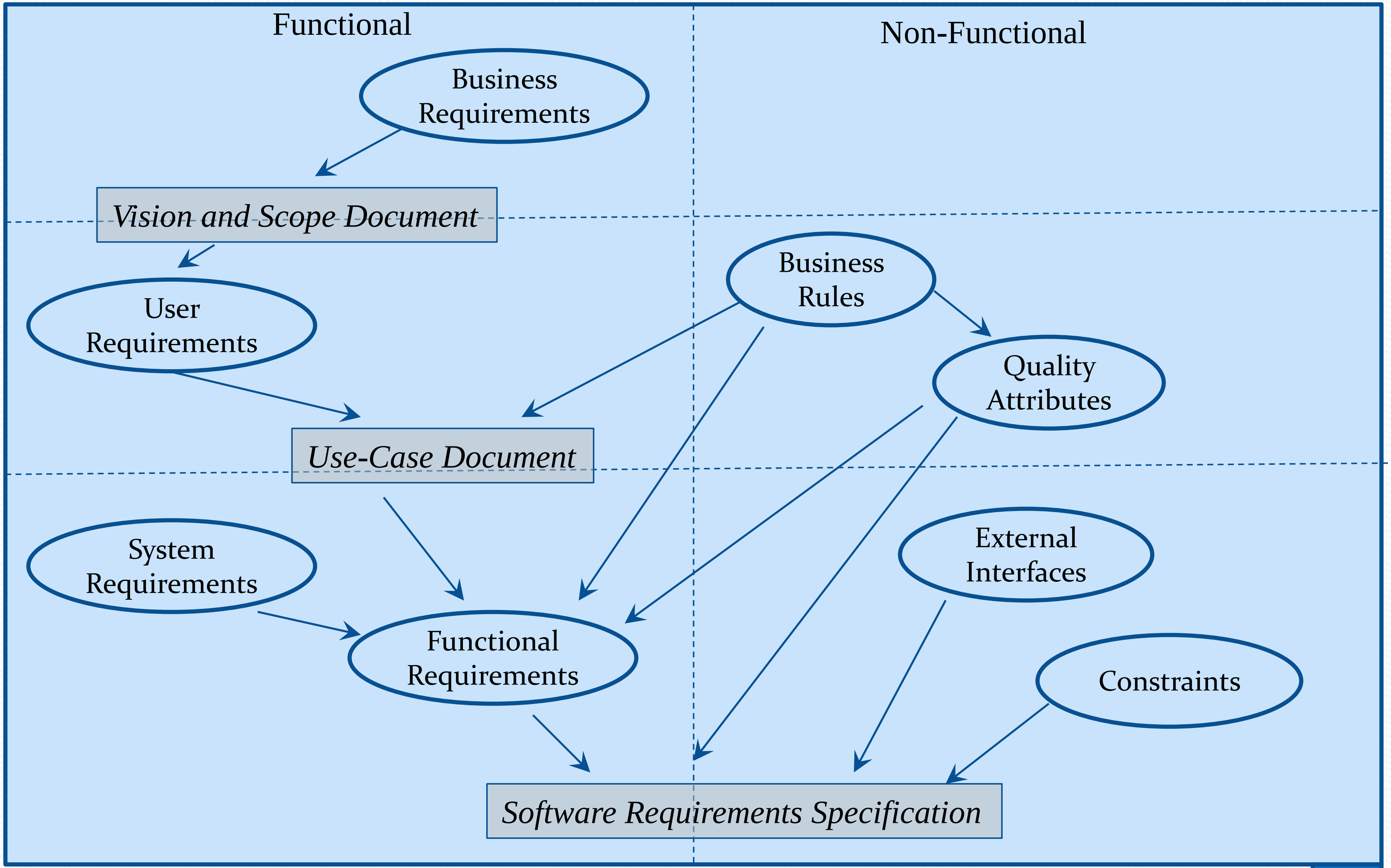
- Lack of awareness
- There is a gap: customer – developer
- Never ending requirements change.

Type of Requirements

Requirements may be **functional** or **non-functional**

- Functional requirements describe system services or functions
- Non-functional requirements is a constraint on the system or on the development process

Levels of Requirements



Business Requirements

- High-level objectives of organization
- Usually from funding sponsor or system owner
- Describe Why the organization needs to implement the system.
- Example:
    - University: Improve the efficiency during course registration.
    - Company: Reduce unnecessary cost, Monitor company in-time performance.

User Requirements

- User goals or tasks that must be able to perform with the product.
- Example:
  - FRS-Online: select courses, submit approval, view student background.
  - Online Ticketing: book a ticket, check schedule, reserve a seat.

Functional Requirements

- Software functionality
- Behavioral requirements
- Use the word “shall”
- Example:
  - FRS-Online: “The system shall view a confirmation to the student.”
  - Online Ticketing: “The system shall provide a link to download an softcopy ticket.”

System Requirements

- Top-level requirements for a system that contains multiple sub-system
- System comprise of: Hardware + Software + Brainware

Business Rules

- Include:
  - Corporate policies
  - Government regulations
  - Industry standards
  - Accounting practices
  - Computational algorithm
- Exist outside the system
- Function:
  - Restrict who and how can perform certain use cases
  - Dictate functionality that a system must have to comply with pertinent rules
- Use as quality attributes.
- Examples:
  - Bank System: "All credit card should use smart card."
  - SIAK: "A ID-card should follow KepMen No. 80/2005."

Quality Attributes

- Include performance goals and descriptions
- Examples:
  - Usability: “The system is equipped with user manual.”
  - Portability: “The system shall work in Microsoft-OSs and Unix-OS.”
  - Integrity: “The system shall restrict access for un-authorized user.”
  - Efficiency: “The system shall work with maximum 200VA/hour.”
  - Robustness: “The system shall withstand 5.1 atmoshpere pressure.”

Exercise

- Identify requirements of each level for the following system:
    - ATM Machine
    - Academic Information System of University X

Non-Functional Requirements

IEEE Standard 830 - 1993

- Performance
- Interface
- Operational
- Resource
- Verification
- Acceptance

- Documentation
- Security
- Portability
- Quality
- Reliability
- Maintainability
- Safety

Non-Functional Requirements

ISO 9126 2005

- Functionality
- Reliability
- Usability
- Efficiency
- Maintanability
- Portability

Functionality

- Functionality is a set of attributes that bear on the existence of a set of functions and their specified properties; the functions are those that satisfy stated or implied needs.
- Includes
  - Suitability
  - Accuracy
  - Interoperability
  - Compliance
  - Security

Reliability

- Reliability is a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
- Includes
  - Maturity
  - Fault Tolerance
  - Recoverability

Usability

- Usability is a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
- Includes
  - Understandability
  - Learn-ability
  - Operability

Efficiency

- Efficiency is ‘a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
- Includes
  - Time-based Efficiency
  - Resource-based Efficiency

Maintainability

- Maintainability is a set of attributes that bear on the effort needed to make specified modifications.
- Includes
  - Analyzability
  - Changeability
  - Stability
  - Testability

Portability

- Portability is a set of attributes that bear on the ability of software to be transferred from one environment to another.
- Includes
  - Adaptability
  - Install-ability
  - Conformance
  - Replace-ability

Exercise

- Classify each identified requirements into those classes defined in ISO 9126 2005

Non-Functional Requirements

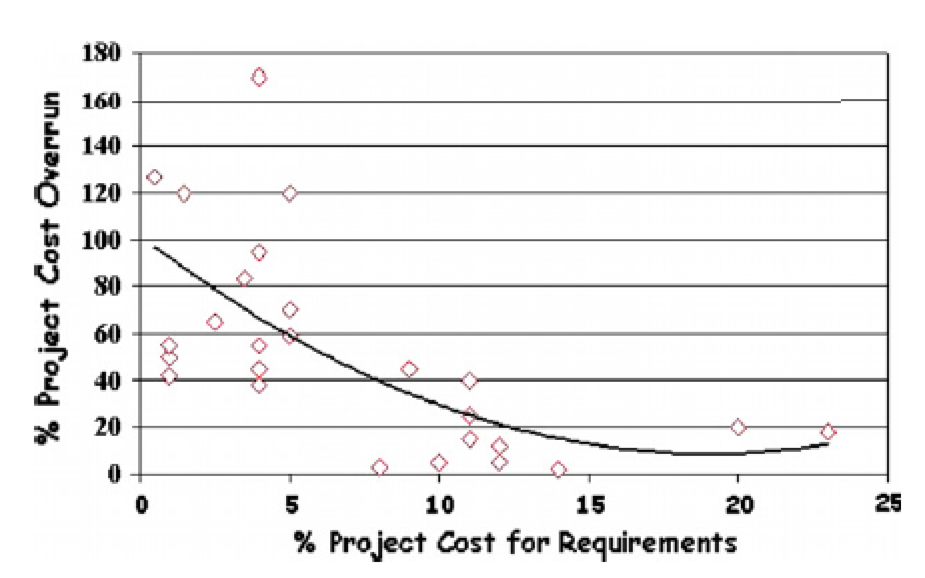
The specifically requested product qualities are expressed directly or indirectly in non-functional requirements

- External quality factors (factors observable by the stakeholders)
  - correctness
  - robustness
  - performance
- Internal quality factors (factors observable by the sw engineer)
  - readability
  - testability

Non-Functional Requirements

- Better software: accuracy, adaptability, completeness, comprehensibility, configurability, flexibility, maintainability, modularity, performance, portability, reliability, reusability, safety, security, testability, traceability, user-friendliness, usability, etc.
- Cheaper software: cost
- Faster production: timeliness, project stability, etc
- Happier customer: supportability, trainability, etc.

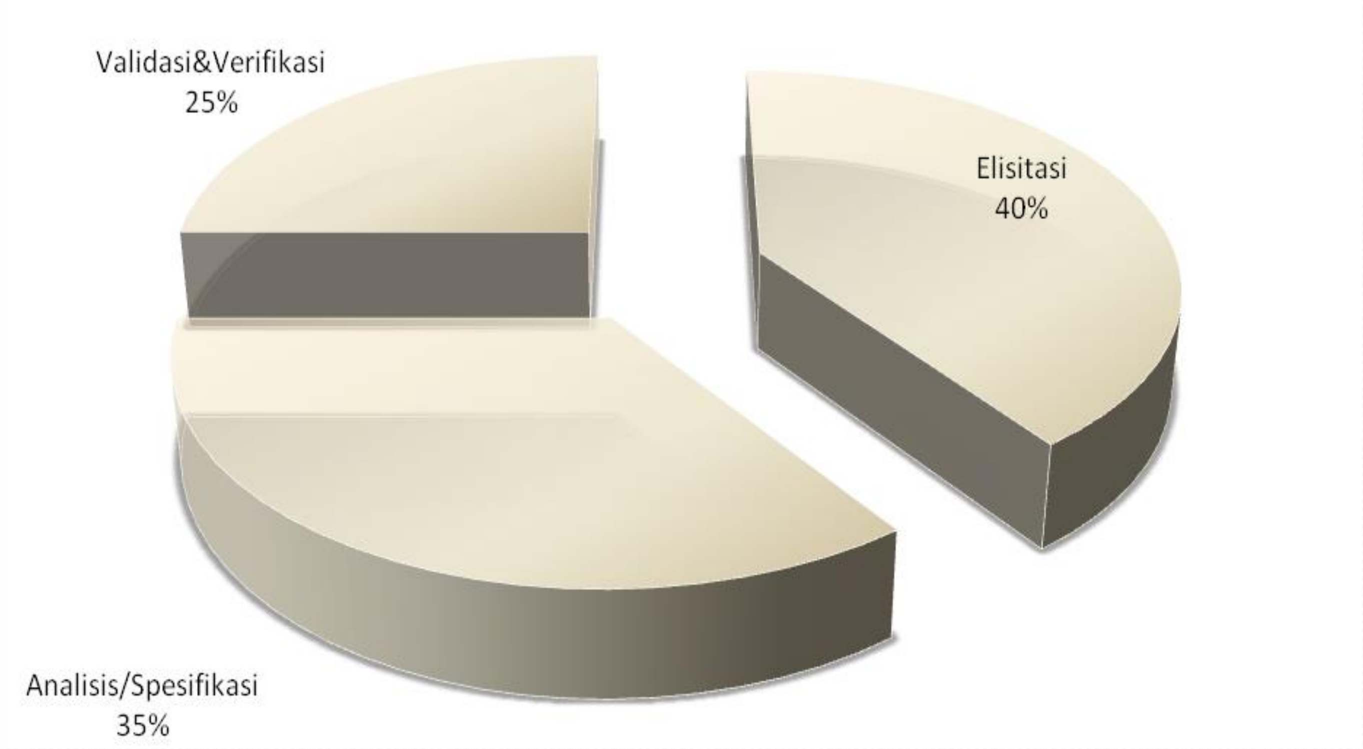
Project Management



Source: Forsberg, 1997

- Successful projects usually use 7 to 15% of project resources (Forsberg, 1997)
- On average, projects use 8% of total project resources, during the period of time averaging 20% (Boehm, 2000)

Time Allocation for each Activities



Guidelines for Better Requirements Management (Hofmann and Lehner, 2001)

- Conduct requirements specification related activities through out the lifecyle
  - Teams with the activities only carried out in the beginning of the project tend to have bad performance.
- Some part-timers support full-time member.
- Combine prototyping and model-based processes to help stakeholders grap the picture of the proposed solution.
- Frequent feedback from stakeholder is a must
- Use modern approach (OO or AO) combined with basic models (ERD, state transition diagram, or Petri-Net).
- Apply evolutionary approach for managing requirements changes, using mock-ups, prototype, peer reviews, walkthroughs, and scenario
  - The effective average number of iterations: 3 iterations