Universitas Esa Unggul

Smart, Creative and Entrepreneurial

CCJ-123-DASAR PENGEMBANGAN PERANGKAT LUNAK (PERTEMUAN KE 2)

www.esaunggul.ac.id

Dosen Pengampu :

5165-Kundang K Juman

Prodi Teknik Informatika Fakultas Ilmu Komputer

Software Project Management
Requirements Analysis

2010

Theo Schouten

Content

- What is requirements analysis?
- Why is it so difficult?
- Stages
- Methods and tools

Book:

ch 7 Requirements engineering + ch 8 analysis modeling

(version 7: ch 5, 6 and 7)

Requirements Analysis: definitions

‘The process of establishing the services the system should provide and the constraints under which it must operate’

Roger S. Pressman S*oftware Engineering – A practitioner’s Approach European Adaptation, fifth edition*

‘the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system’

Thayer, R.H. and M. Dorfman, *Software requirements engineering*

Embodied knowledge

•“Because software is embodied knowledge, and because that knowledge is initially dispersed, tacit, latent and incomplete in large measure, software development is a social learning process.

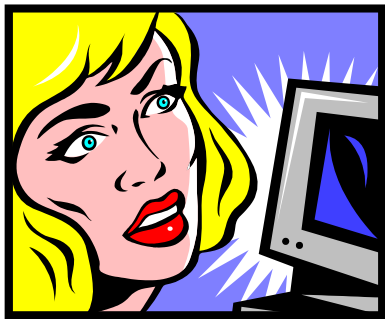
•The process is a dialogue in which the knowledge that must become the software is brought together and embodied in the software.

•The process provides interaction between users and designers and evolving tools (technology).

•It is an iterative process in which the evolving tool itself serves as the medium for communication, with each new round of dialogue eliciting more useful knowledge from the people involved.”

Howard Baetjer, jr.

Why so difficult?



Users/stakeholders

Software designer

- Different “worlds”
  - using vs designing something
  - knowing what should be done vs knowing to let a computer do that
- Users/stakeholders are not an uniform group
  - conflict between cost and usability / performance / features
  - conflicting demands from different departments
- Getting the good (ideal) system vs possibility building it good

other factors

- Expectations, the final solution is difficult to imagine by the users
- Scope of the system
  - need well defined boundaries
- Current vs future system
  - old, rusted demands and wishes
  - resistance to change
- Aiming at a moving target
- ‘Wicked problems’ – more than one good solution
- functional needs versus technical solutions
- Completeness (functional and technical) hard to get
- Difference between ‘nice-to-have’ en critical functionality

Process of negotiation between users and designers

Stages in RE

- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Management

Inception

- ask a set of questions that establish …
  - basic understanding of the problem
  - the people who want a solution
  - the nature of the solution that is desired
  - the effectiveness of preliminary communication and collaboration between the customer and the developer

Elicitation

- elicit requirements from all stakeholders
  - address problems of scope
  - address problems of understanding
    - customers are not sure about what is needed, skip “obvious” issues, have difficulty communicating with the software engineer, have poor grasp of problem domain
  - address problems of volatility (changing requirements)

Elaboration and negotiation

- Elaboration: create an analysis model that identifies data, function, features, constraints and behavioral requirements

- Negotiation: agree on a deliverable system that is realistic for developers and customers
  - rank requirements by priority (conflicts arise here …)
  - identify and analyze risks assoc. with each requirement
  - “guestimate” efforts needed to implement each requirement
  - eliminate, combine and / or modify requirements to make project realistic

Specification

- can be any one (or more) of the following:
  - A written document
  - A set of models
  - A formal mathematical model
  - A collection of user scenarios (use-cases)
  - A prototype

Validation

- a review mechanism that looks for:
  - errors in content or interpretation
  - areas where clarification may be required
  - missing information
  - inconsistencies (a major problem when large products or systems are engineered)
  - conflicting or unrealistic (unachievable) requirements
  - tests for requirements

Management

- involves managing change:
  - **Feature traceability:** how requirements relate to observable system/product features
  - **Source traceability:** identifies source of each requirement
  - **Dependency traceability:** how requirements are related to each other
  - **Subsystem traceability:** categorizes requirements by the sub system (s) they govern
  - **Interface traceability:** how requirements relate to both external and internal system interfaces

Methods and tools

many of them available

- lists
  - elicitation question list
  - checklists for validation
- graphical diagrams, good for communication
- formal methods
  - e.g. UML for elaboration and specification

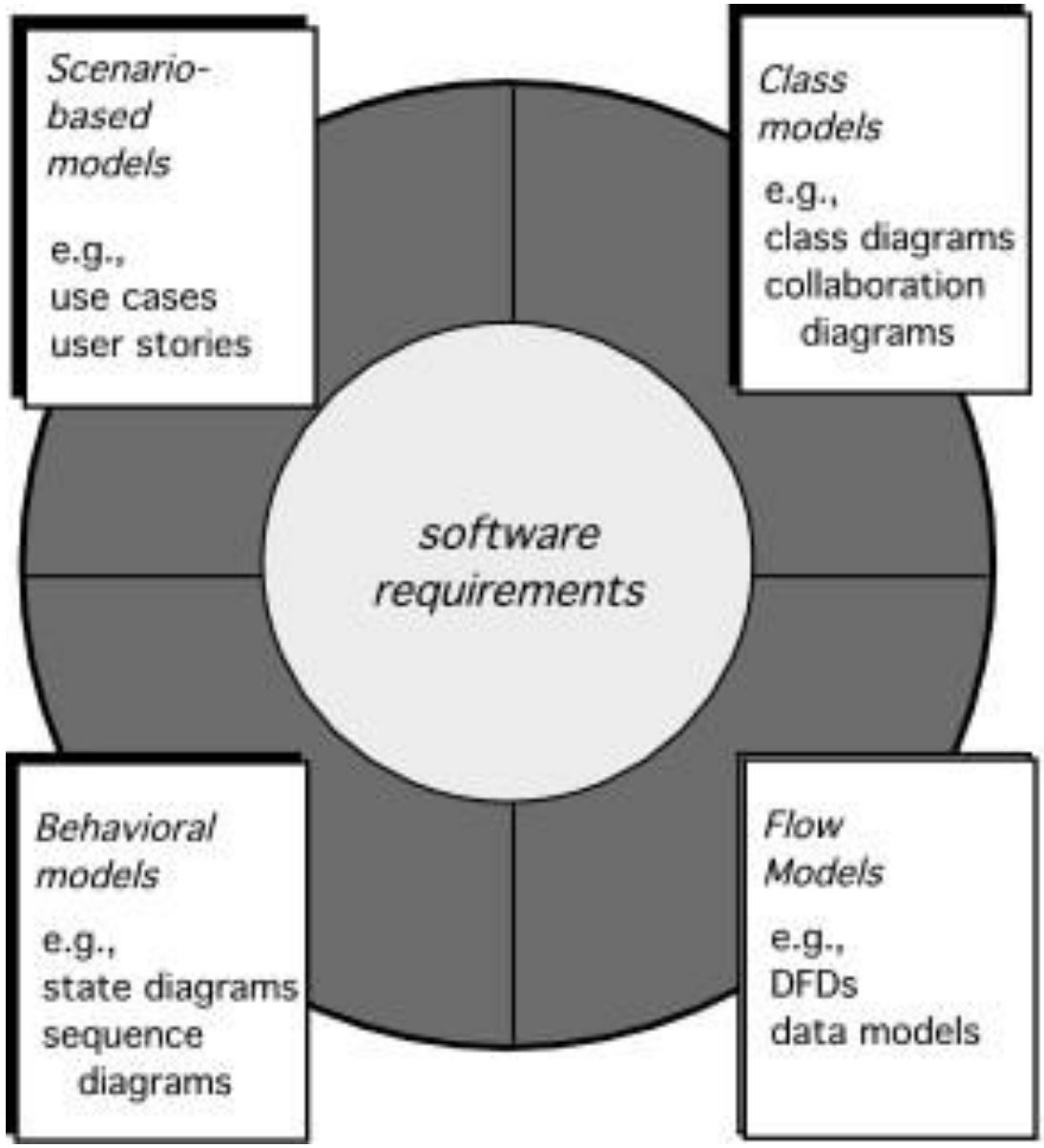
Quality Function Deployment

A technique of translating customer needs into technical system requirements:

- **Normal requirements:** reflect stated customer goals and objectives
- **Expected requirements:** implicit to the product or system; their absence will cause significant customer dissatisfaction
- **Exciting requirements:** featured going beyond customer expectations, causing customer euphoria (;-)
- concentrate on maximizing customer satisfaction

- Function deployment: determines the “value” (as perceived by the customer) of each function required of the system
- Information deployment: identifies data objects and events, ties them to functions
- Task deployment: examines the behavior of the system
- Value analysis: determines the relative priority of requirements

Modeling approaches



- Scenario-based: user interaction with system
  - Use-case: descriptions of the interaction
- Data-based: data objects in system
  - ER (entity-relation) Diagrams
  - Class-based: data+methods
    - OO, Class diagrams, implied by scenarios
- Behavioral-based: how external events change system
  - State, sequence diagram
- Flow-based: information transforms
  - Data flow, control flow diagrams

Modeling for WebApps

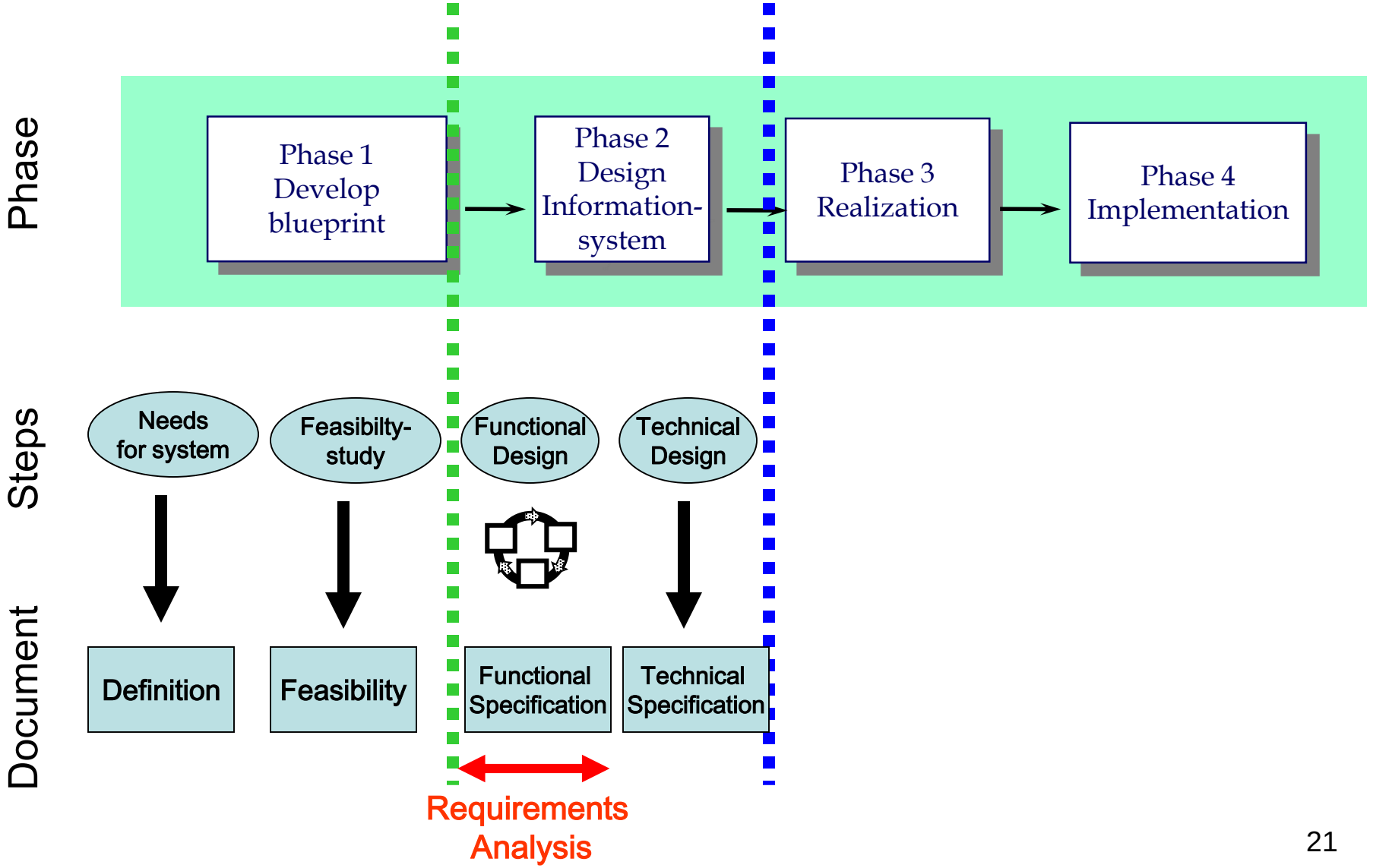
Content Analysis. the content is identified, including text, graphics and images, video, and audio data.

Interaction Analysis. The manner in which the user interacts with the WebApp is described in detail.

Functional Analysis. The usage scenarios (use-cases) created as part of interaction analysis define the operations that will be applied to WebApp content and imply other processing functions.

Configuration Analysis. The environment and infrastructure in which the WebApp resides are described in detail. Server-client side.

In which phase of the whole process?



Attention points

- Focus on external performance of the system towards the users
- Limitations in the environment should be well described (technical, number of users and usage, use etc.)
- Ease of adaptation
- References for maintaining the system
- Vision on the life cycle of the system
- How to deal with unexpected events (fault-proof)