Object-oriented analysis, design, and programming

OO methodology

- Beginning with a statement of requirements
- the process proceeds through analysis, overall design
- detailed design
- implementation
- test/maintenance plan

Analysis

- use cases and detailing a flow of events for each.
- (initial set of) functional test cases is specified, to serve as a vehicle for checking that the implementation is complete and basically correct.
- identifying classes implied by the use cases,
- documenting classes using an Analysis Class Diagram.

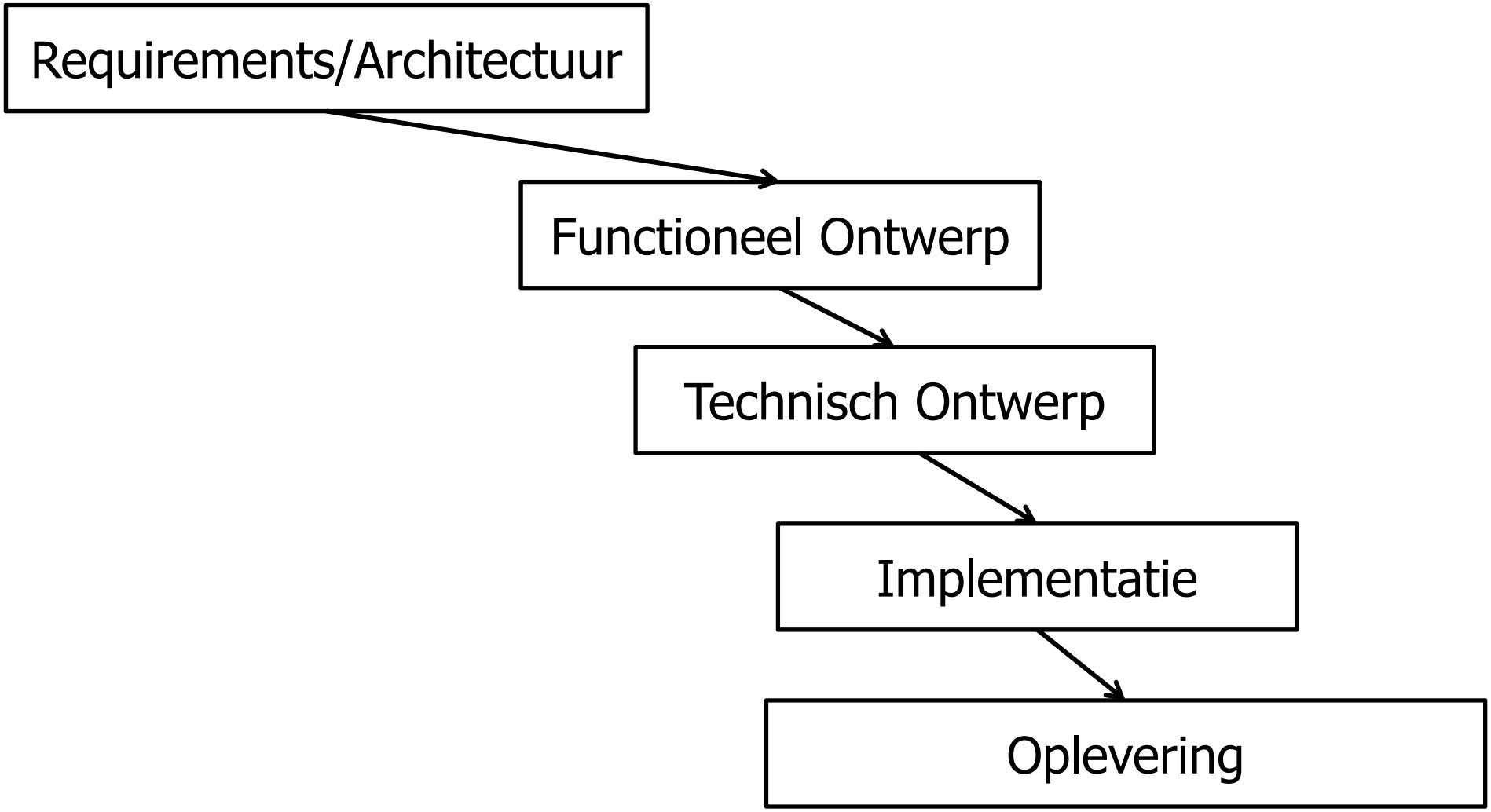
Overall design

- CRC cards to assign responsibilities to the various classes.
- The static structure of the design is summarized by means of an overall Class Diagram.
- Then the dynamic aspects of the design are developed using
  - State Charts for the major controller classes
  - plus an Interaction Diagram for each of the main use cases.

Detailed design

- attributes and methods for each class, using a class diagram for each class
- A package diagram is used to show how the various classes are groupled into packages.

Waterval-based development



Requirements

Meetbare uitspraken

over de diensten die een systeem verwacht wordt aan te bieden,

en de condities waaronder deze moeten worden uitgevoerd

Requirements zijn SMART afspraken

(**S**pecifiek, **M**eetbaar, **A**cceptabel, **R**ealistisch, **T**ijdgebonden)

Soorten requirements

- Functionele requirements
  - wat het systeem moet doen

- Niet-functionele requirements
  - Bijv. performance, security, …
  - Randvoorwaarden t.a.v. ontwikkeling/beheer
  - Gebruik van standaarden

Requirement beschrijving

- Functionele requirement beschrijft alleen extern gedrag
- Meetbaar, testbaar
- Rationale
- Bron

Moeilijk!

- Scope – vage afbakening van grenzen, of te veel detail
- Begrip
  - Uiteindelijk oplossing is moeilijk voor te stellen
  - Taalverschil gebruiker - ontwikkelaar
  - Huidig vs toekomstig systeem ('vastgeroeste' eisen en wensen)
  - Denken in behoeften versus oplossingen
  - "Voor-de-hand-liggende" zaken worden verzwegen
- Volledigheid
- Vluchtigheid – requirements veranderen in de tijd
- Conflicterende eisen bij gebruikers
- Verschil tussen opdrachtgever en gebruiker (bijv. budget)
- Verschil tussen 'nice-to-have' en kritische functionaliteit

Van geheel naar detail

Begin met begrip opbouwen van
het bedrijf of organisatie als geheel,

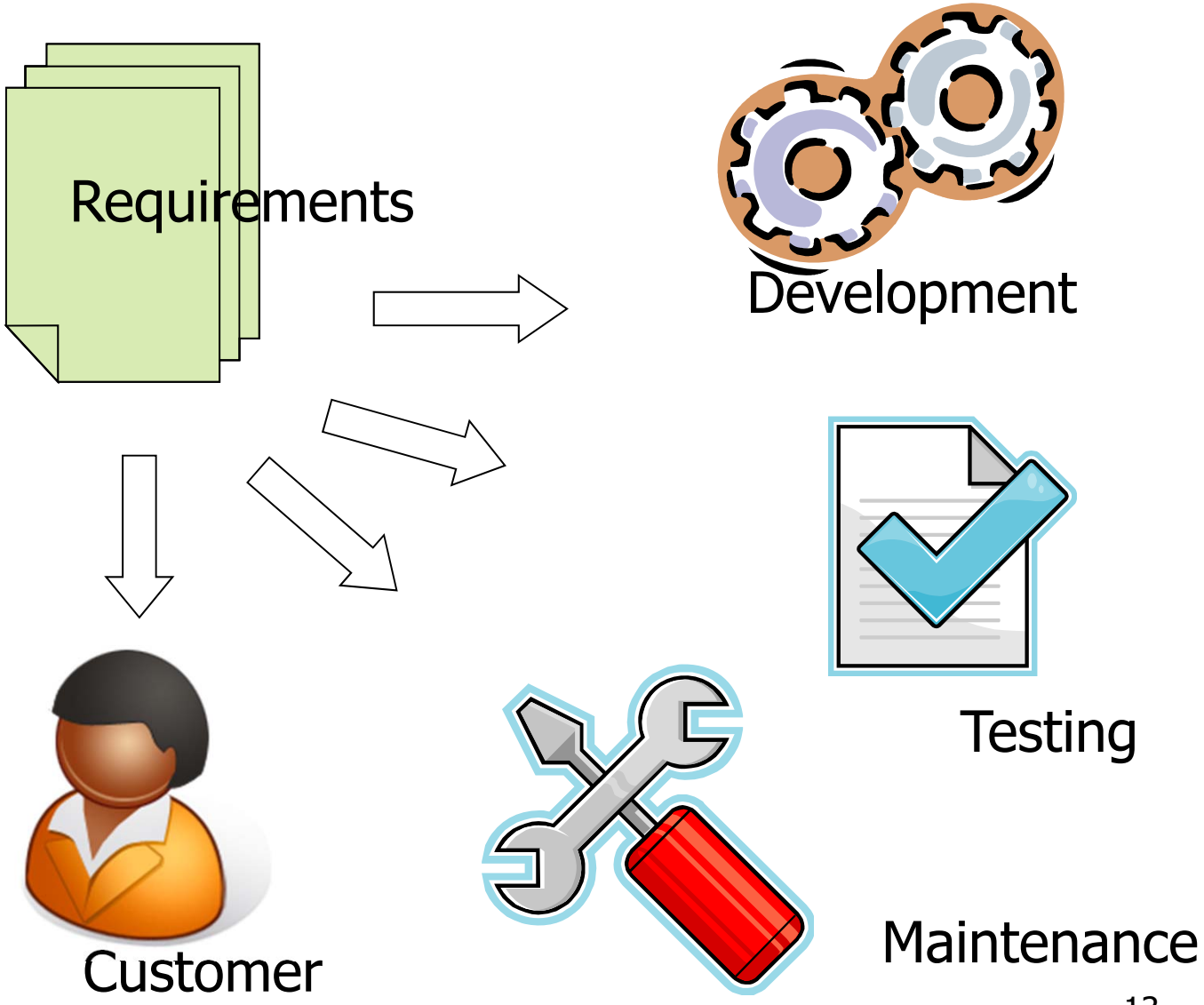
en zoom dan in op bepaalde delen ervan!

Haalbaarheidsstudie

Beantwoord in ieder geval de volgende vragen:

- Draagt het systeem bij aan de doelstellingen van het bedrijf of de organisatie?
- Kan het systeem worden geïmplementeerd met de beoogde technologie en binnen het opgegeven budget en tijdsframe?
- Kan het systeem worden geïntegreerd binnen de bestaande omgeving?

User Requirements



Use-cases

- Scenario's over typische interacties met het systeem beschreven vanuit actor-perspectief
- Actor is bijv. gebruiker of device
- Beschrijft
  - Taken/functies van een actor
  - Informatie die wordt verkregen, gegeven, of gewijzigd door een actor
  - Toestandsveranderingen van het systeem
  - (Onverwachte) events waarvan de actor op de hoogte moet zijn

Structured Use Cases?

UC1: Buy Ticket
Actors: Customer
Preconditions: • Vending machine is turned on…
Main Scenario: 1. Customer selects destination station on machine 2. Customer selects single or return trip ... 6. Machine outputs ticket when amount is reached 7. Customer takes ticket from machine
Post-conditions: • Customer owns ticket to start traveling with
Variations: 1. Customer makes a mistake in step 1 till 3. After the mistake he pushes the reset button and starts over at step 1 …
Exceptions: 1. Change money is not present in machine …
Extra Requirements: 1. The machine displays a message “Use exact change” when any of the coins has less than 10 in the cash register …

Is there a glossary?

Buyer =

A person that buys a house from a seller

Buyer.Name

Buyer.SocialSecurityNumber

Buyer.Address

Social security numbers always have 9 digits

Validating specific requirements

A. Consistent
B. Complete
C. Criteria
    1. Identifiable
    2. Atomic
    3. Unambiguous
    4. Traceable
    5. Testable

Architecture?

- A building has an architecture
  - But what is that exactly?
- An ICT system is exactly defined
  - But what is its architecture?

It reflects both the implicit and the explicit choices!

- What
- How
- Why

Different Kinds of ICT Architectures

- The Enterprise Architecture
  - principles, standards
- The Project architecture
  - collection of applications, interconnection, hardware components
- **The Application architecture**
  - Global technical design, patterns, structure (classes)
- The Platform architecture
  - Garbage Collection and Type Safety Java,
  - Multilanguage .Net,
  - LAMP-stack

Key Architectural Aspects

- Standards
  - Windows, Linux
- Hardware and Software
  - Sensors, Actuators, Robots, Web Servers, Virtualisation Servers
- Structure and Interaction
  - Servers, clients, services
- Dependency relations
  - call-graph
- Communication
  - interfaces, synchronisation

Why Application Architecture?

The application architecture is not the operational software.

Rather, it is a representation that enables a software engineer to:

(1) analyze the effectiveness of the design in meeting its stated requirements,

(2) consider architectural alternatives at a stage when making design changes is still relatively easy, and

(3) reduce the risks associated with the construction of the software.

Global Styles of Application Architectures

Each style describes

a system category that encompasses:

(1) a set of components (e.g., a database, computational modules) that perform a function required by a system,

(2) a set of connectors that enable "communication, coordination and cooperation" among components,

(3) a set of constraints that define how components can be integrated to form the system, and

(4) one or more semantic models that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

- Data-centered architectures
- Data flow architectures
- Call and return architectures
- Layered architectures
- Object-oriented architectures
- Service oriented architectures

Or a combination of them….