

# Software Life Cycle

Mark van den Brand



**TU/e** Technische Universiteit  
Eindhoven  
University of Technology

Where innovation starts

## Activities Common to Software Projects

- Requirements and specification
  - Domain analysis
  - Defining the problem
  - Requirements gathering
    - Obtaining input from as many sources as possible
  - Requirements analysis
    - Organizing the information
  - Requirements specification
    - Writing detailed instructions about how the software should behave

**TU/e** Technische Universiteit  
Eindhoven  
University of Technology

2-2-2009 PAGE 1

/ Faculteit Wiskunde en Informatica

## Activities Common to Software Projects

- Design
  - Deciding how the requirements should be implemented, using the available technology
  - Includes:
    - *Systems engineering*: Deciding what should be in hardware and what in software
    - *Software architecture*: Dividing the system into subsystems and deciding how the subsystems will interact
    - *Detailed design* of the internals of a subsystem
    - *User interface design*
    - *Design of databases*

**TU/e** Technische Universiteit  
Eindhoven  
University of Technology

2-2-2009 PAGE 2

/ Faculteit Wiskunde en Informatica

## Activities Common to Software Projects

- Modeling
  - Creating representations of the domain or the software
    - Use case modeling
    - Structural modeling
    - Dynamic and behavioral modeling
- Programming
- Quality assurance
  - Reviews and inspections
  - Testing
- Deployment & maintenance
- Managing the process

**TU/e** Technische Universiteit  
Eindhoven  
University of Technology

2-2-2009 PAGE 3

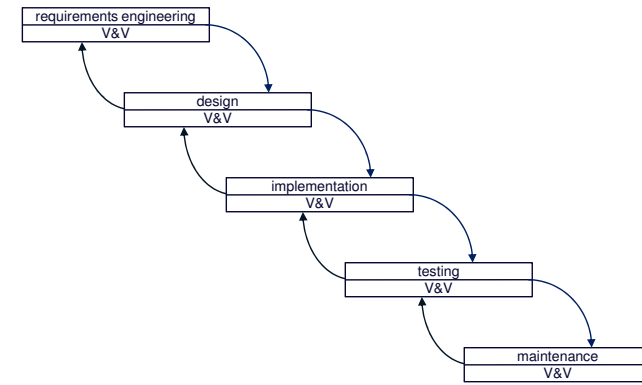
/ Faculteit Wiskunde en Informatica

# Software Engineering Projects

- Most projects are *evolutionary* or *maintenance* projects, involving work on *legacy* systems
- **Corrective** projects: fixing defects
- **Adaptive** projects: changing the system in response to changes in
  - Operating system
  - Database
  - Rules and regulations
- **Enhancement** projects: adding new features for users
- **Reengineering** or **perfective** projects: changing the system internally so it is more maintainable

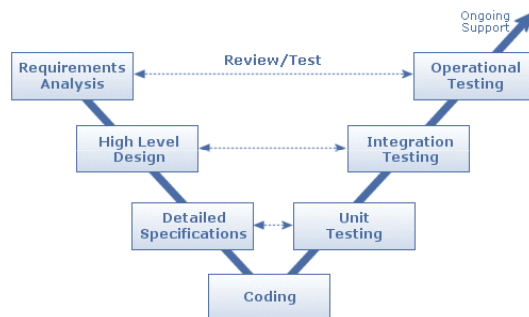
# Software development model

Waterfall model



# Software development model

V-model



# Software development model

- **Waterfall model**
  - Document oriented
  - Suited for (very) large projects (> 50 people)
  - Too many design activities during coding and testing

## Software development model

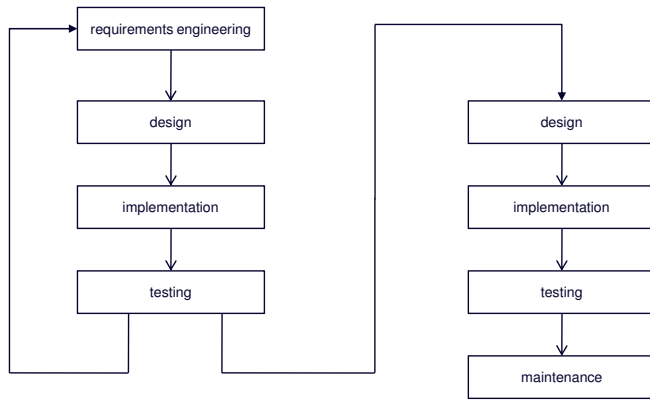
- **Agile methods**
  - **Individuals and interactions are more important than processes and tools**
  - **Working software is more important than comprehensive documents**
  - **Customer collaboration is more important than contract negotiation**
  - **Responding to change is more important than following a plan**

## Software development model

- **Agile methods**
  - **No extensive architectural or design phase**
  - **No energy spend on documentation**

## Software development model

### Prototyping



## Software development model

- **Advantages of prototyping**
  - **Resulting system is easier to use**
  - **Resulting system has less features**
  - **User needs are better accommodated**
  - **Design is of higher quality**
  - **Problems are detected earlier**
  - **Resulting system is easier to maintain**
  - **Development costs less effort**



## Software development model

- **Rapid application development (RAD)**
  - **Similar to iterative development process models**
    - User involvement
    - Prototyping
    - Reuse
    - Automated tools
    - Small development teams
  - **Time boxing**

## Software development model

- **RAD has four phases:**
  - Requirements planning
  - Application design
  - Construction
  - Cutover (testing, training, installation)
- **MoSCoW:**
  - Must haves
  - Should haves
  - Could haves
  - Won't haves

## Software development model

- **Dynamic systems development method (DSDM)**
- **Builds on RAD**
- **5 phases:**
  - Feasibility study
  - Business study
  - Functional model iteration
  - Design and build iteration
  - implementation

## Software development model

- **Extreme programming (XP) principles:**
  - Rapid feedback
  - Simplicity
  - Incremental change
  - Embracing change
  - Quality work

# Software development model

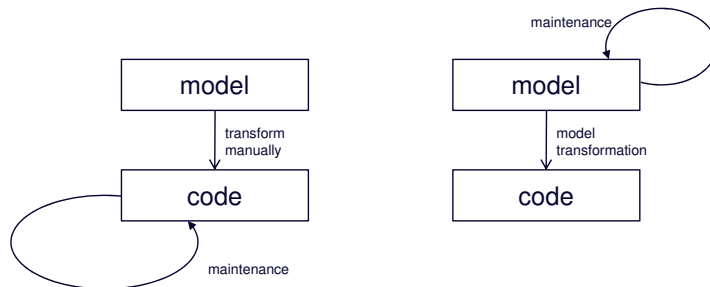
- **Extreme programming in practice:**
  - Planning game
  - Small releases
  - Metaphor
  - Simple design
  - Testing
  - Refactoring
  - Pair programming
  - Collective ownership
  - Continuous integration
  - 40-hour week
  - On-site customer
  - Coding standards

# Software development model

- **Rational Unified Process (RUP) in practice:**
  - Iterative development
  - Requirements management
  - Architecture and use of components
  - Modeling and UML
  - Quality of process and product
  - Configuration and change management
  - Use-case driven development
  - Process configuration
  - Tool support

# Software development model

- **Model-driven architecture (MDA)**
  - Traditionally models are manually transformed into code
  - MDA advocates model transformation maintenance



# Software development model

- **Several models**
  - CIM (computation independent model)
  - PIM (platform independent model)
  - PSM (platform specific model)
  - Code