

# Softwaretechnik

## Lecture 03: From Requirements to Definition

Peter Thiemann

University of Freiburg, Germany

SS 2011

# Requirements Engineering

comprises methods, means of description, and tools to discover, analyze, and formulate requirements of software systems

- ▶ **requirements analysis** (*Systemanalyse*)
- ▶ **requirements specification** (*Produktdefinition*)

# Requirements

- ▶ **Functional requirements**
  - ▶ inputs and their constraints
  - ▶ functions of the system
  - ▶ outputs (reactions)
- ▶ **Nonfunctional requirements**
  - ▶ runtime
  - ▶ memory
  - ▶ standards

# Requirements

## ▶ Requirements on realization

- ▶ software / hardware
- ▶ devices
- ▶ interfaces
- ▶ facilities (OS, computers, ...)
- ▶ documentation

## ▶ Requirements on testing, installation, support

## ▶ Requirements on construction of the system

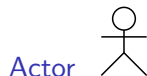
- ▶ approach
- ▶ resources (personal, cost, deadlines)
- ▶ rules, standards

# Systematic Investigation of Functional Requirements

- ▶ Inside-out methods  
modeling starts from product internals  
(rarely applicable for new products)
- ▶ Outside-in methods  
modeling starts from environment of product
  - ▶ actors and use cases (use case diagram, UML)
  - ▶ interfaces and data flows (context diagram)

# Use Cases and Use Case Diagrams

Jacobson, UML



- ▶ participates directly in a process
- ▶ stands for a role
  - ▶ natural person
  - ▶ unit of organization
  - ▶ external system

# Use Cases

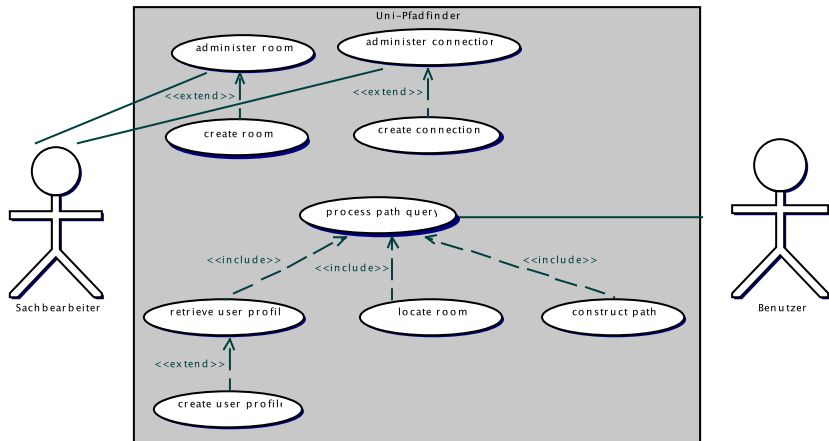
## Use case [Definition]

- ▶ a sequence of actions
- ▶ performed by one actor
- ▶ to achieve a particular goal

two forms:

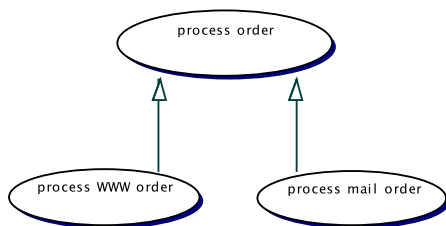
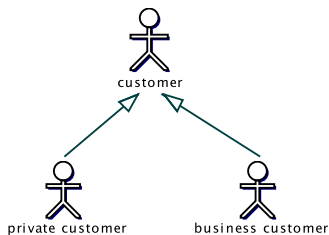
- ▶ graphical (UML diagram)
- ▶ textual (with templates)

# Example Use Case Diagram





# Generalization



- ▶ generalization
- ▶ concrete and abstract use cases
- ▶ concrete and abstract actors

# Use Case Textual Template

Use case: name

Goal: achieved by successful execution

Category: primary, secondary, optional

Precondition:

Postcondition/success:

Postcondition/failure:

Actors:

Trigger:

Description: numbered tasks

Extensions: wrt previous tasks

Alternatives: wrt tasks

# Use Case Guidelines

- ▶ Outside view — System as black box
- ▶ No implementation specifics
- ▶ No UI specifics
- ▶ **Primarily text**

# Tools

- ▶ <http://www.umlet.com/>  
UML diagram drawing — standalone and in Eclipse
- ▶ <http://yum1.me/>  
online drawing of use case and class diagrams (UML)
- ▶ <http://www.gliffy.com/flowchart-software/>  
flowcharts and DFD

# Related Approaches

## User Stories

A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it. [Scott Ambler

<http://www.agilemodeling.com/artifacts/userStory.htm>]

- ▶ Very slim, very high-level, often just one sentence.
- ▶ Informal, but proposed formal style [Mike Cohn]:  
**As a (role) I want (something) so that (benefit).**

## Example User Stories

- ▶ Students can purchase monthly parking passes online.
- ▶ Parking passes can be paid via credit cards.
- ▶ Professors can input student marks.
- ▶ Students can obtain their current seminar schedule.
- ▶ Students can order official transcripts.
- ▶ Students can only enroll in seminars for which they have prerequisites.
  
- ▶ As a student I want to purchase a monthly parking pass so that I can drive to school.
- ▶ As a student I want to obtain my current seminar schedule so that I can follow my classes.

# User Stories Guidelines

- ▶ Authors
- ▶ Tools
- ▶ Size
- ▶ Priority
- ▶ Traceability

# Related Approaches

## Usage Scenarios

A usage scenario, or scenario for short, describes a real-world example of how one or more people or organizations interact with a system. They describe the steps, events, and/or actions which occur during the interaction. Usage scenarios can be very detailed, indicating exactly how someone works with the user interface, or reasonably high-level describing the critical business actions but not the indicating how they are performed.

[Scott Ambler

<http://www.agilemodeling.com/artifacts/usageScenario.htm>]

- ▶ Further elaboration of a use case.
- ▶ Scenario  $\sim$  path through a use case.



# Example High-Level Scenario

# Example Detailed Scenario

# Interfaces and Data Flows

interfaces:

- ▶ information sources
- ▶ information sinks
- ▶ should specify origin of information

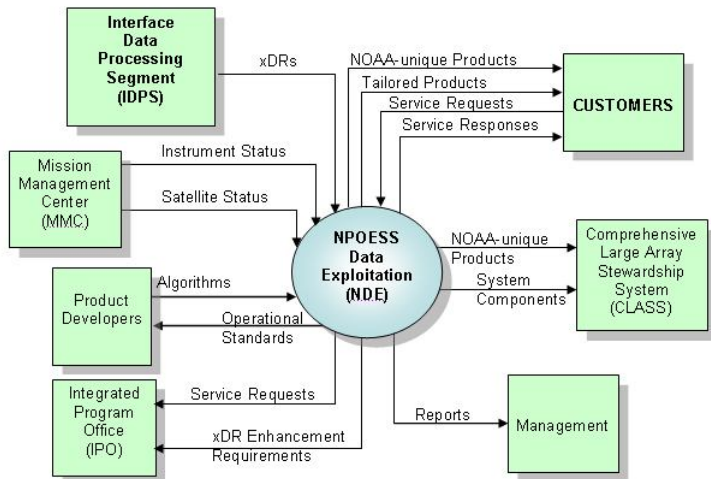
data flow:

- ▶ all input and output data
- ▶ arrows with markings
- ▶ markings should be informative

representation by **context diagram**

origin: Tom DeMarco's structured analysis

# Example Context Diagram



[http://upload.wikimedia.org/wikipedia/commons/8/8e/NDE\\_Context\\_Diagram.jpg](http://upload.wikimedia.org/wikipedia/commons/8/8e/NDE_Context_Diagram.jpg)

# Perspective on Changing Requirements

- ▶ Produce high quality requirements (see checklist in CC2)
- ▶ Advertise the cost of requirements changes
- ▶ Establish a change-control procedure
- ▶ Anticipate changes
- ▶ Consider the business value of requirements
- ▶ Cancel a project with bad or frequently changing requirements