10) Requirements Analysis

1. Feasibility Study
2. Requirements Analysis
3. ZOPP
4. SRS
5. Requirements Management

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
WS 2014-0.1, 28.10.2014
Lecturer: Dr. Sebastian Götz

Obligatory Reading

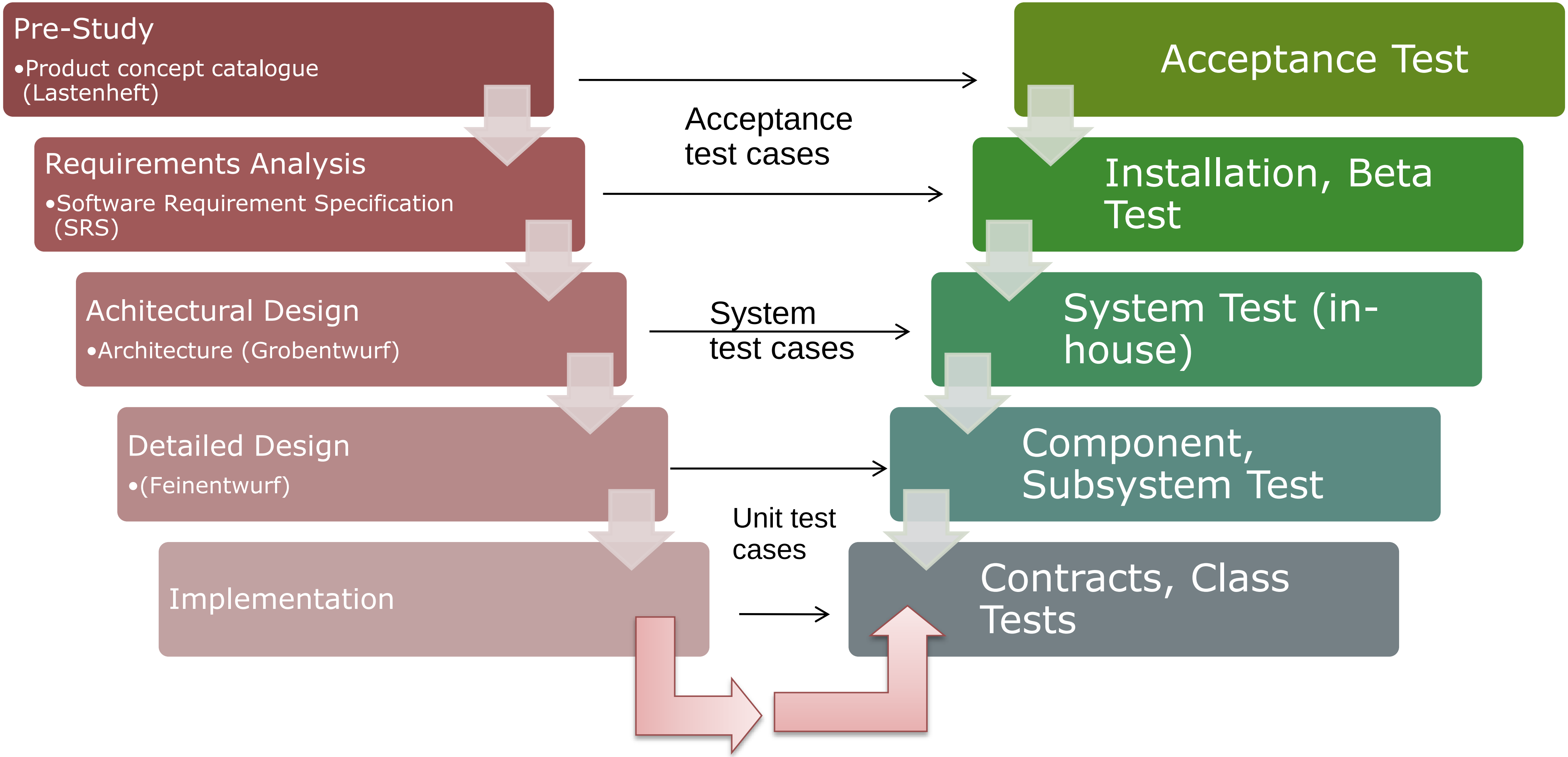
- Balzert Kap. 1 (LE 2), Kap 2 (LE 4)
- Maciaszek Chap 6-8
- ZOPP from GTZ www.gtz.de:
  - Ziel-orientierte Projektplanung. GTZ (Gesellschaft für technische Zusammenarbeit). GTZ is a German society for development. ZOPP is a general-purpose project planning and requirements analysis method. Google for it.....
  - http://portals.wi.wur.nl/files/docs/ppme/ZOPP_project_planning.pdf
  - ZOPP is part of Project Cycle Management (PCM), a more general methodology for project management
    - http://baobab-ct.org/learning/pcm.html
  - http://en.wikipedia.org/wiki/Logical_framework_approach
  - The ZOPP-88 version, on which this material is basedhttp://pmkb.com.br/wp-content/uploads/2013/08/GTZ-ZOPP.pdf

Other References

- B. Schott. Verhandeln. Sicher, kreativ, erfolgreich. Haufe-Verlag.
- W. Winkler. Probleme schnell und einfach lösen. Mvg-Verlag.
- Sophist Group www.sophist.de
  - Gruppe von kompetenten Anforderungsaanlysten, die viele Tipps im Internet veröffentlichen

Software Development in the V-Model

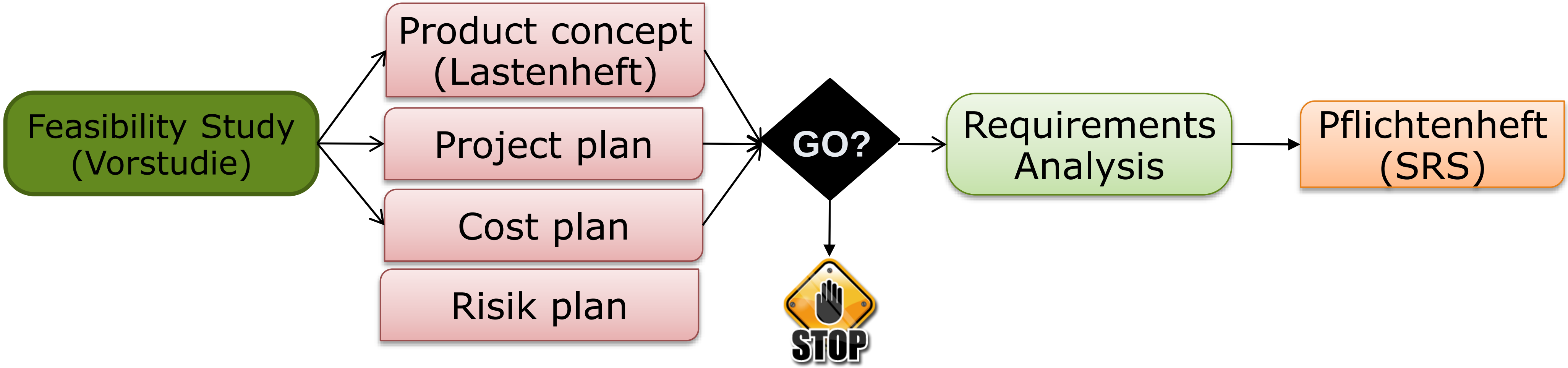
- The most simple software development process is the V-model



10.1. FEASIBILITY STUDY (MACHBARKEITSSTUDIE)

Feasibility Study (Mach-, Durchführbarkeits-, Vorstudie)

- The **feasibility study** is done before the project, permits the customer to decide whether he is interested.
  - Separate contract, separate budget (price)
- Result: **product concept catalogue (Lastenheft**), a summary of all coarse-grain requirements
  - Project plan with parts and steps of the project
  - A cost calculation
- Product concept is refined and extended towards the SRS (software requirements specification, Pflichtenheft)
  - Based on the feasibility study, there is a GO or NO-GO decision
  - And a new contract
  - Part of this contract will be the SRS that is defined in the first phase of the project



Feasibility Study Document (Lastenheft)

- The feasibility study fixes coarse-grain requirements and goals of the project, from the viewpoint of the client.
  - Basis for the real requirements specification
- Contents [Balzert]:
  - Goal of project (problem model, goal model)
  - Application domain of product and target group (stakeholder model)
  - Functions (coarse-grain)
  - Quality requirements (non-functional requirements such as reliability, usability, efficiency,...)
  - Data that are stored permanently
  - Delivery (time, precision)
  - Additions
- Convention: all requirements are numbered with a specific key, specific for the Lastenheft:
  - /LDxx/ for data
  - /LFxx/ for functions
  - /LQxx/ for quality features

Product Concept Catalogue (Lastenheft) - Aufgabe

- Die Daten der an einem Lehrstuhl beschäftigten Hilfsassistenten sollen in einer Datei verwaltet werden.
- Über jeden Hilfsassistenten sind folgende Informationen aufzubewahren:
  - Name, Vorname, Matrikelnummer, Geburtsdatum, Heimatadresse einschließlich
  - Telefon-Nr., Studienadresse einschl. Telefon-Nr., Studienfach,
  - Semesterzahl, Vordiplom, Beschäftigungszeiten, Diplom, Arbeitszeugnis ausgehändigt
- Aus der Hilfsassistenten-Datei ist folgende Liste zu erstellen:
  - Listenkopf:
    - »Alphabetische Liste aller Hilfsassistenten«
  - Listenrumpf:
    - Name, Vorname, Matrikel-Nr., Studienadresse, Geburtsdatum.
- Außerdem müssen pro Hilfsassistent seine sämtlichen Daten ausgeben werden können.

Product Concept Catalogue (Lastenheft)

1 Zielbestimmung

- Das Programm HIWI-Verwaltung soll einen Lehrstuhl in die Lage versetzen, die beschäftigten Hilfsassistenten zu verwalten.

2 Produkteinsatz

- Das Produkt wird im Sekretariat eines Lehrstuhls eingesetzt und von der Sekretärin bedient.

3 Produktfunktionen

- /LF10/
  - Ersterfassung, Änderung und Löschung von Hilfsassistenten
- /LF20/
  - Ausgabe einer alphabetischen Liste aller Hilfsassistenten mit folgenden Daten:
    - Name, Vorname, Matrikel-Nr., Studienadresse, Geburtsdatum
- /LF30/
  - Ausgabe sämtlicher Daten eines Hilfsassistenten
- /LF40/
  - Gezielte Abfragen mit einer Endbenutzersprache

4 Produktdaten

- /LD10/
  - Folgende Daten sind über jeden Hilfsassistenten zu speichern:
  - …………….

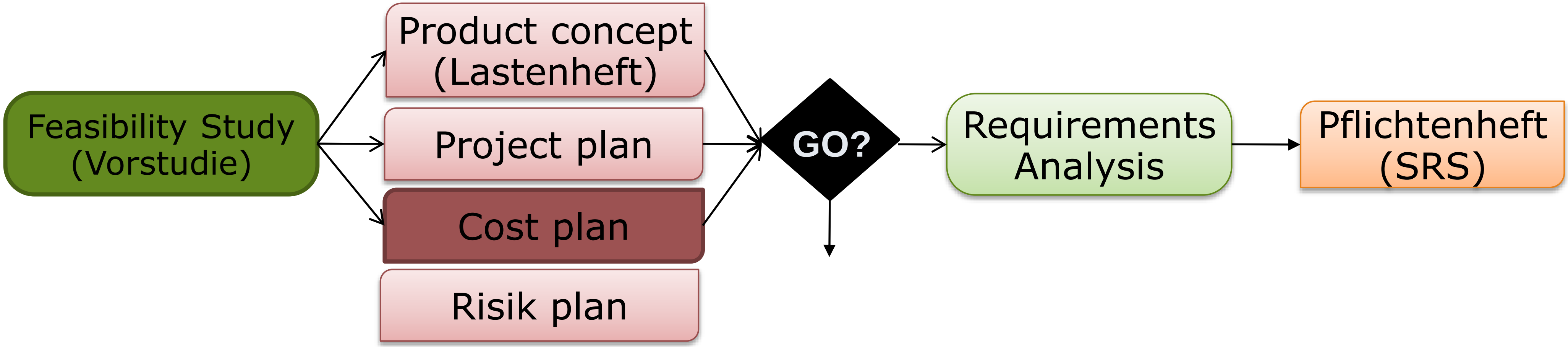
Product Concept Catalogue (Lastenheft) (2)

- 5 Produktleistungen
  - /LL10/
    - Die Funktionen /LF10 / und /LF40/ dürfen nicht mehr als 2 Sekunden Antwortzeit benötigen.
  - /LL20/
    - Es müssen max. 1000 Hilfsassistenten verwaltet werden können.
- 6 Qualitätsanforderungen

	Sehr gut	Gut	Normal	Nicht relevant
Funktionalität		X		
Zuverlässigkeit				X
Benutzbarkeit		X		
Effizienz			X	
Änderbarkeit			X	
Übertragbarkeit				X

Cost Analysis

- Together with the feasibility study, costs have to be analyzed and forecasted.
- Without cost estimation, no price.
  - Analysis methods (Cocomo, Function point analysis) see “Software-Management”.
- Experts that have enough experience to estimate costs are worth a lot of money!

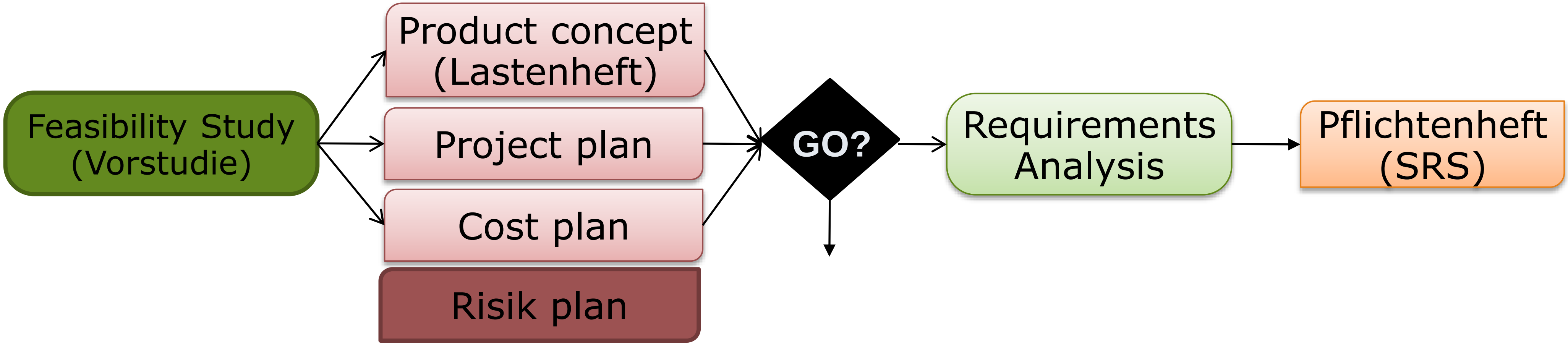


Cost Estimation (Aufwandsschätzung)

- Quantity:
  - LOC (Lines of Code)
  - Size of data
  - Complexity scales (grading 1-5)
- Quality:
  - Quality-of-Service levels (service level agreements (grading 1-5)
- Duration:
  - Person days, months and years
  - Time of project
- Resources
  - Labor cost
  - Equipment
  - Travel costs
- Costs
  - Fix costs
  - Variable costs
- More in course Softwaremanagement

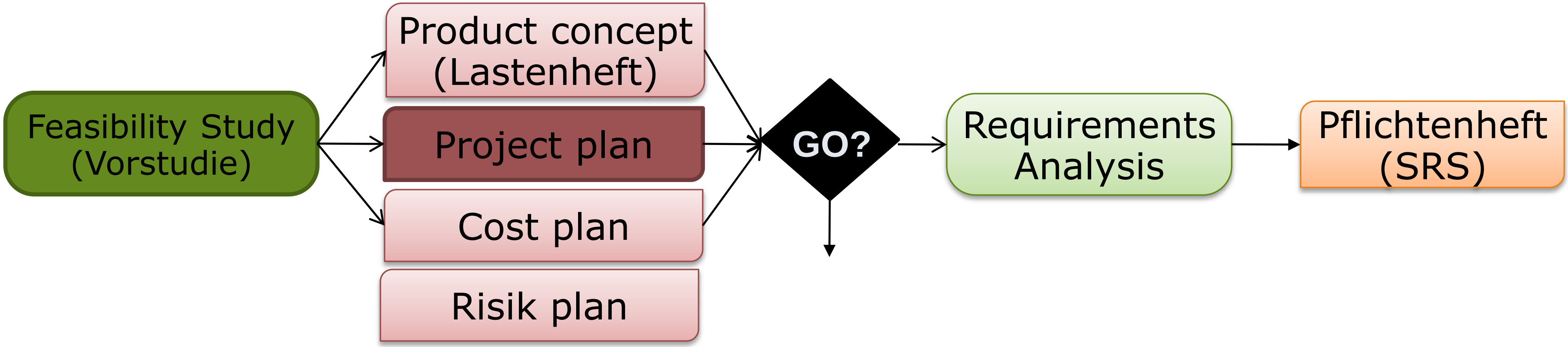
Risk Analysis

- Together with the feasibility study and the requirements analysis documents, the software producer has to perform a *risk analysis*
- Risk analysis protects the producer
  - To underestimate costs
  - To mispredict deadlines
  - To underestimate goal conflicts
- Risk analysis results are preserved and regularly inspected by the management



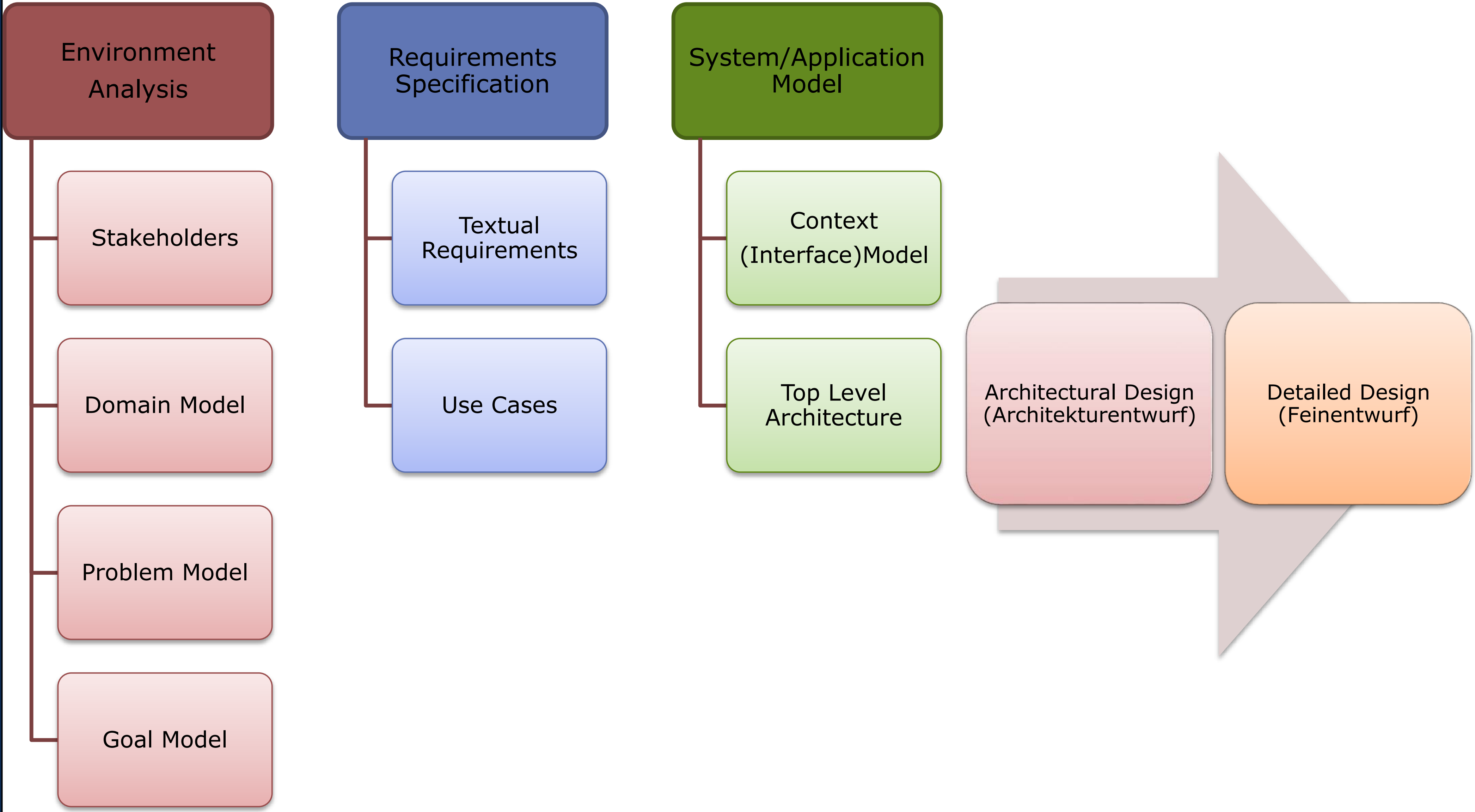
Planning of the Project Management

- In the feasibility study, a preliminary planning of the project is done
  - See Course "Software Management" in SS
- Planning
- Staffing
- Organizing
- Directing
- Control



10.2 REQUIREMENTS ANALYSIS

From the Product Definition to the Design



Environment Analysis

- Stakeholders
- Domain Model
- Problem Model
- Goal Model

Requirements Specification

- Textual Requirements
- Use Cases

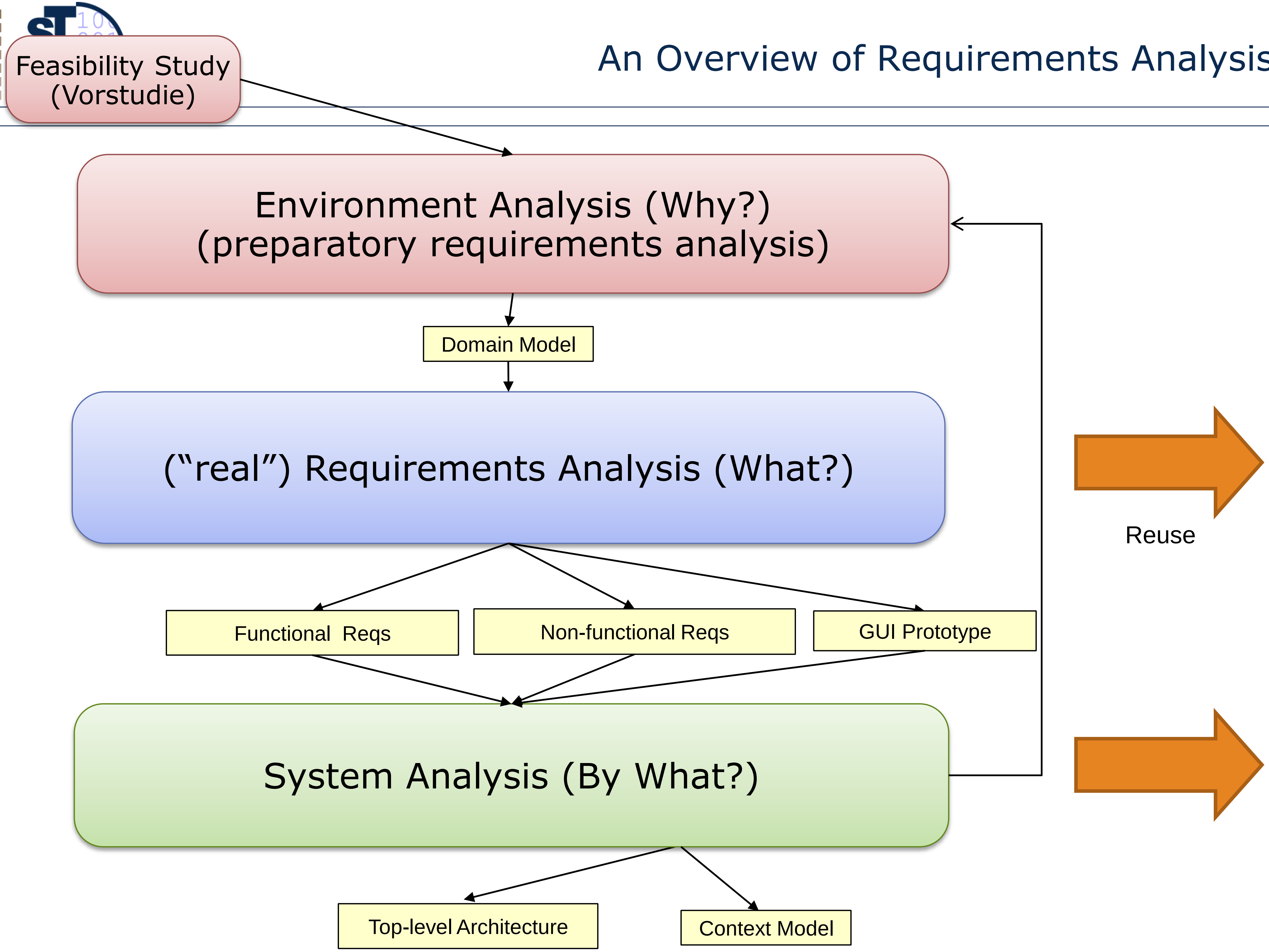
System/Application Model

- Context (Interface)Model
- Top Level Architecture

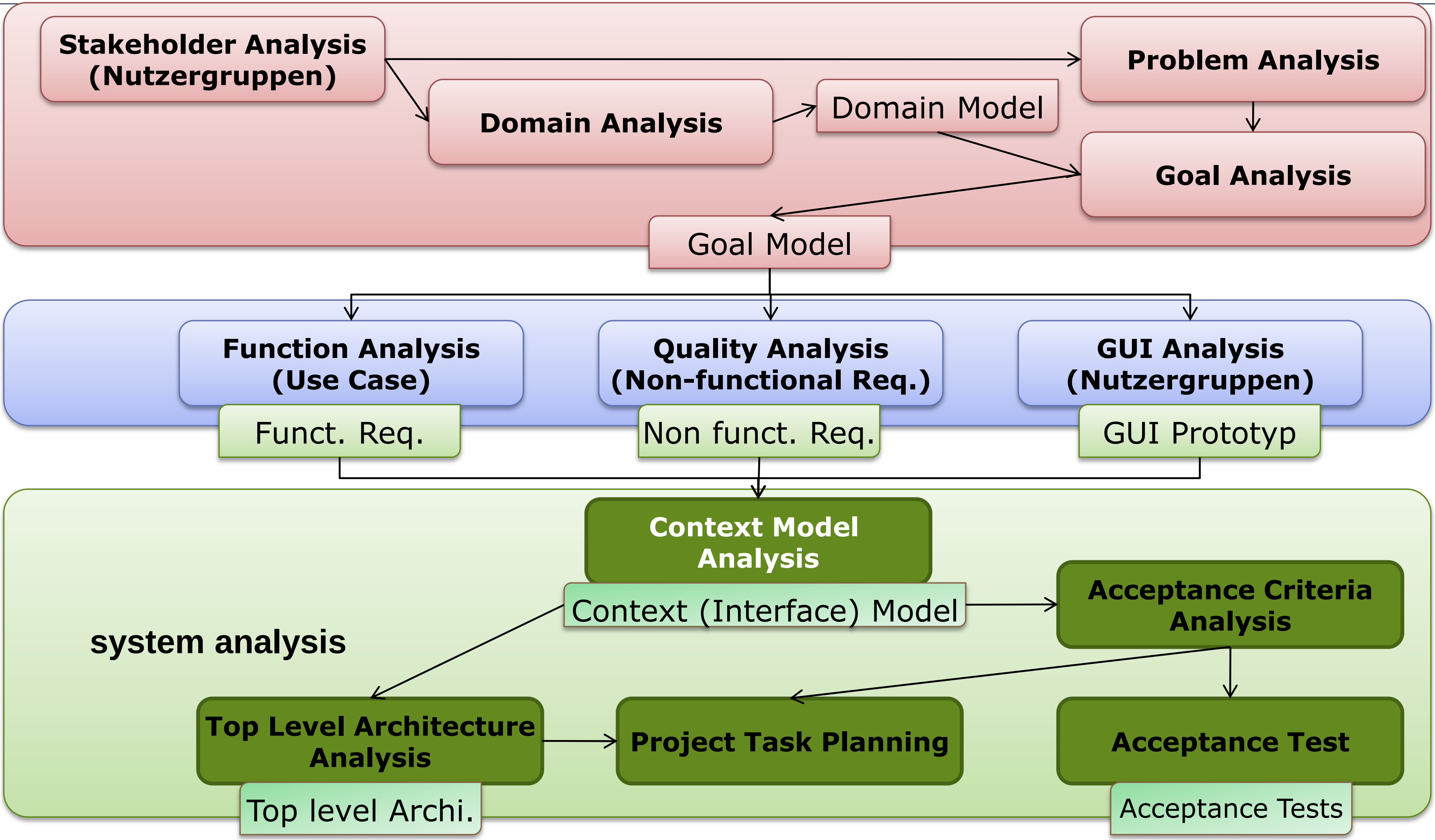
Architectural Design (Architekturentwurf)

Detailed Design (Feinentwurf)

An Overview of Requirements Analysis



Analysis in Detail


Stakeholder Analysis (Nutzergruppen)
Problem Analysis
Domain Analysis
Domain Model
Goal Analysis
Goal Model
Function Analysis (Use Case)
Quality Analysis (Non-functional Req.)
GUI Analysis (Nutzergruppen)
Funct. Req.
Non funct. Req.
GUI Prototyp
Context Model Analysis
Context (Interface) Model
Acceptance Criteria Analysis
system analysis
Top Level Architecture Analysis
Project Task Planning
Acceptance Test
Top level Archi.
Acceptance Tests

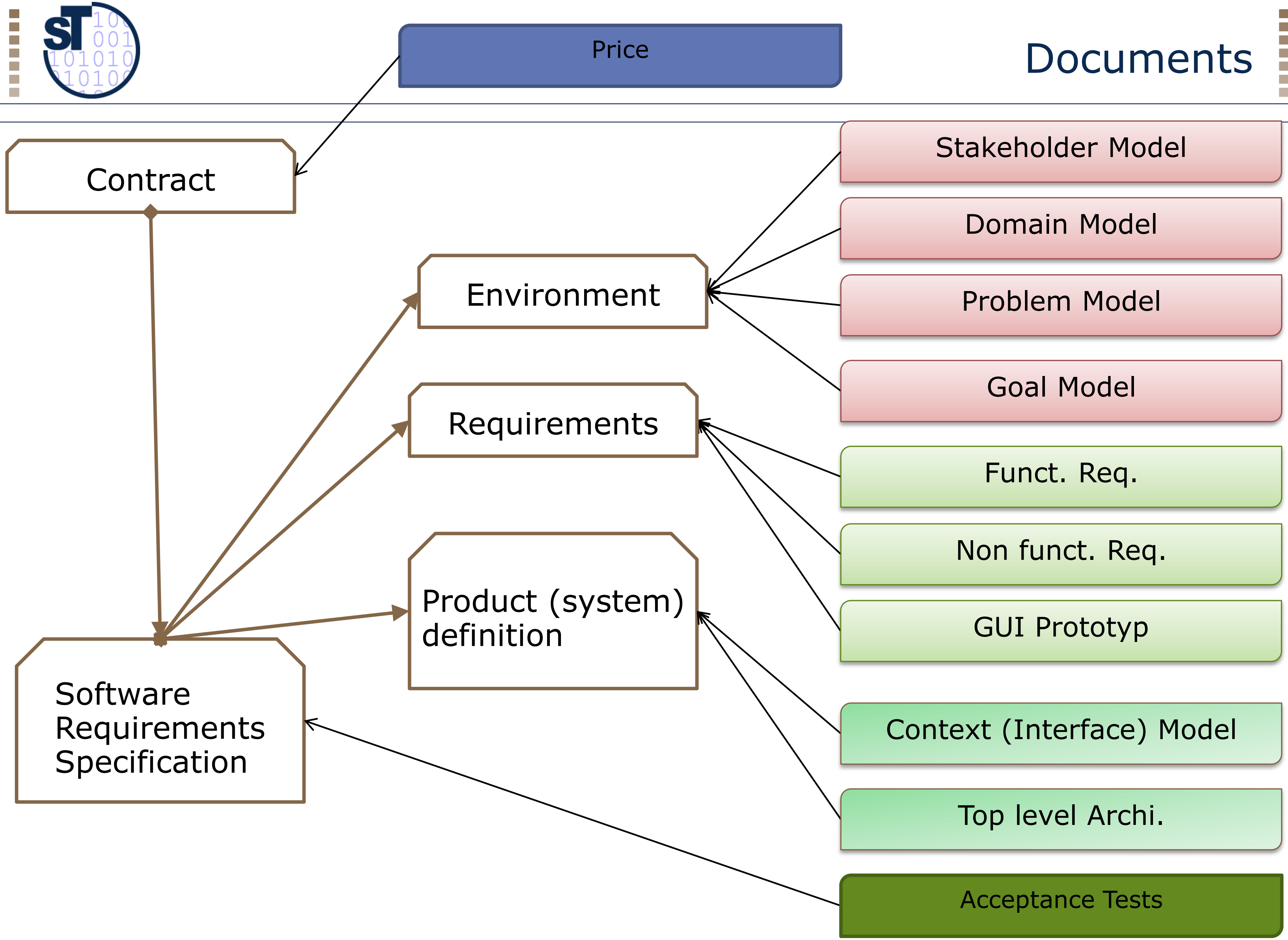
General Rules for Requirements Analysis

- Problems, goals, functional, and non-functional requirements should be distinguished!
  - You should always work problem oriented, start from the problem of the customer
  - And come from problems to goals and requirements.
- Parts of system analysis:

Stakeholder analysis	Who?
Domain analysis	Which language do we speak?
Problem analysis (Is-Analysis)	Why?
Goal analysis (objective)	To which end?
Requirements analysis	What? Through What?
Acceptance criteria analysis	When ok?

- "The goal of system analysis is to fix the desires and requirements of the client, to analyse and to model them." [Heide Balzert]

Documents

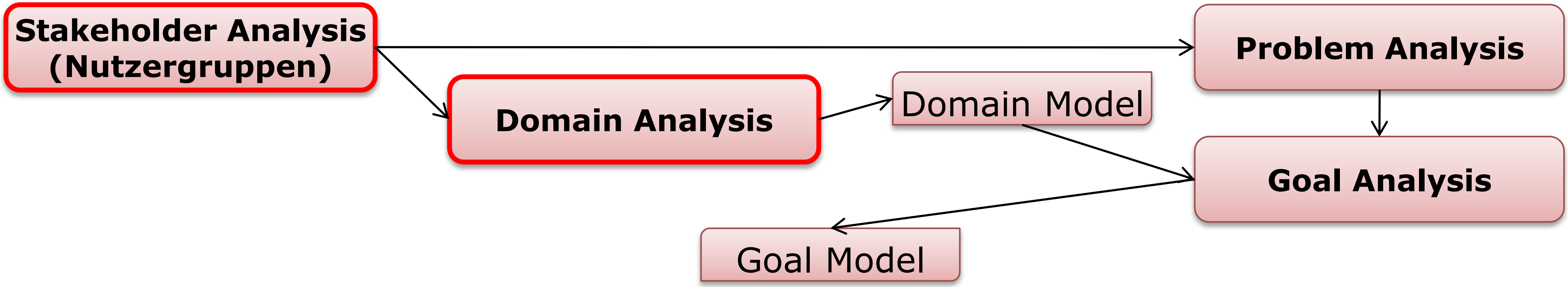

Price
Contract
Stakeholder Model
Domain Model
Environment
Problem Model
Goal Model
Requirements
Funct. Req.
Non funct. Req.
Product (system) definition
GUI Prototyp
Software Requirements Specification
Context (Interface) Model
Top level Archi.
Acceptance Tests

Inhalte eines Vertrags

- Pflichtenheft
  - Produktdefinition
    - Anforderungsspezifikation (das WAS)
      - Nutzermodell (stakeholders)
      - Domänenmodell
      - Funktionale Anforderungen
      - Problemmodell, Zielmodell, Nicht-funktionale Anforderungen
    - Fachliches Modell (das WIE, das der Kunde wissen muss)
      - Kontextmodell (Schnittstellenmodell)
      - GUI-Prototyp
      - Top-level-Architektur
  - Akzeptanztestfälle:
    - Messbare Akzeptanzkriterien, die bei der Abnahme vom Kunden abgehakt werden können. Ohne bestandenen Akzeptanztest keine Bezahlung!
- Preisliche Regelung

- *Achtung: In der Literatur wird der Begriff "Analysemodell" sowohl für die Produktdefinition als auch nur für das fachliche Modell verwendet!*

Results of Environment Analysis (Why?)



- *Nutzermodell (stakeholder model):* Liste oder UML-Klassendiagramm aller am System Interessierten
  - Enthält die Benutzer des Systems (die *stakeholder* oder *Aktoren*)
  - Ziele der Nutzer
- *Domänenmodell (domain model):*
  - Termini, Struktur und Grundkonzepte des Aufgabengebiets
  - Schaffung einheitlicher Terminologie für die Anforderungsspezifikation
  - Aus der Sicht des Kunden
  - Zusammenhang mit Anforderungsspezifikation sichern
  - Implementierungsaspekte ausklammern: Annahme *perfekter Technologie*
- *Problemmodell:* Spezifikation der Problemwelten der Nutzer
- *Zielmodell:* Spezifikation der Ziele der Nutzer

Results of Requirements Analysis (What?)

- *Funktionale Anforderungen: Funktionale Essenz* des Systems. Was muss das System können?
  - Nicht das Wie, sondern nur das Was
  - möglichst quantitativ (z.B. Tabellenform)
  - eindeutig identifizierbar (Nummern)
  - Notation: Anwendungsfalldiagramme (Nutzfalldiagramme), Funktionsbäume
    - Textuelle Spezifikation mit nummerierten Anforderungen /FXX/
    - Manchmal auch mathematische Spezifikation
- *Nicht-funktionale Anforderungen:* Qualitätsanforderungen
  - Nutzungsmerkmale:
    - z.B. Antwortzeit, Speicherbedarf, HW/SW-Plattform
  - Entwicklungsmerkmale:
    - Maintainability
    - Extensibility
  - Managmentmerkmale
- *Datenformate*, die gespeichert werden müssen

Results of System Analysis: (fachliches Modell, Fachkonzept) (By What? What the Customer Should Know)

- *Kontextmodell*: äußere Schnittstellen des Systems
  - Ein- und Ausgabekanäle, Masken, Abfragen
  - Daten, die ein und aus fließen, im Domänenmodell typisiert

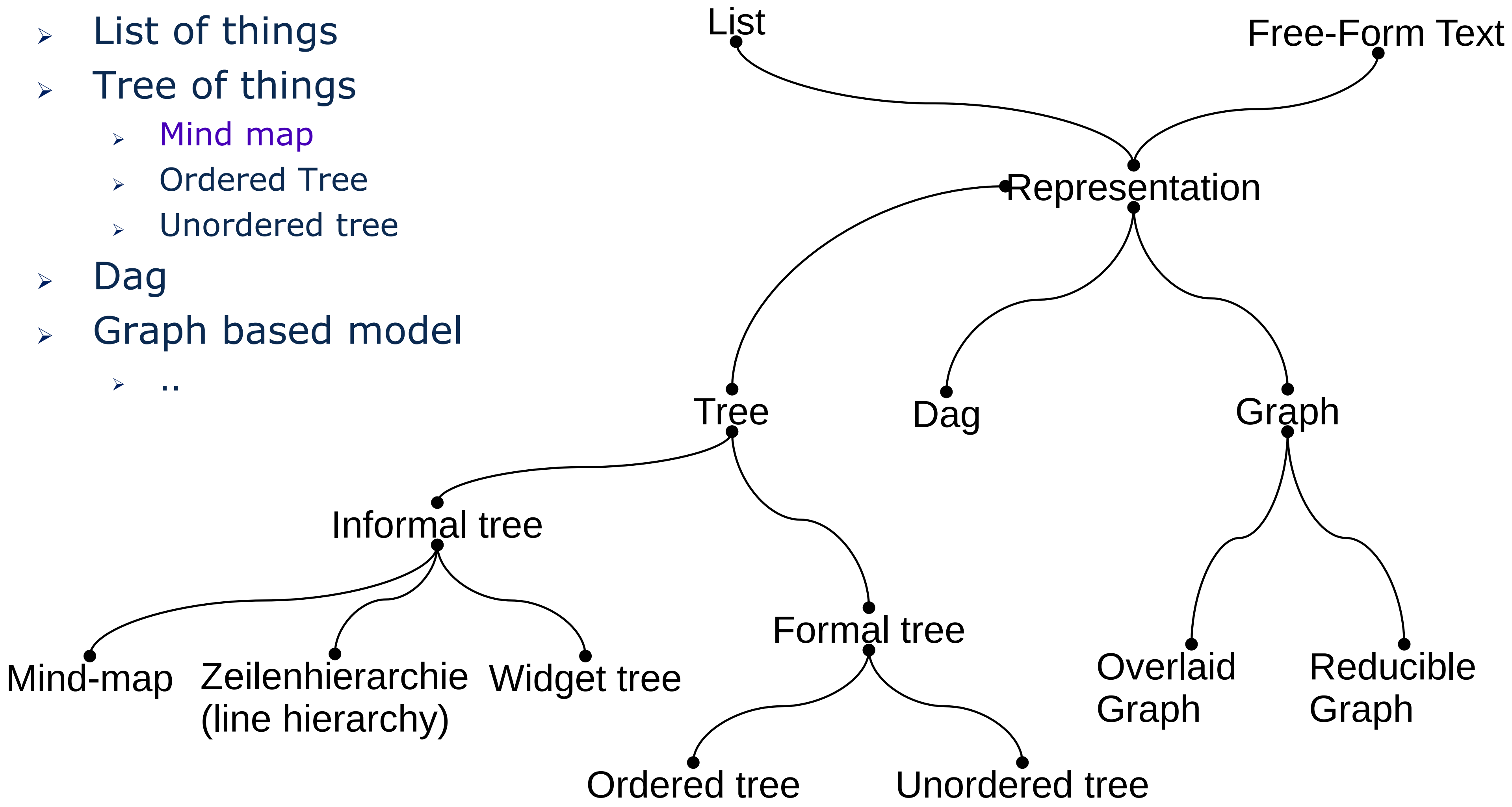
- *GUI Prototyp:* Prototypische Masken, Formulare, Bildschirme, die den GUI ausmachen: Wie sieht das Programm aus?

- *Top-level Architektur (Initiale Architektur, Facharchitektur):* Bestimmt die Hauptkomponenten des Systems und ihre Interaktionen, ohne auf Details einzugehen
  - Verfeinert das Kontextmodell um eine Stufe, d.h. die ***Top-Level Architektur***
  - Stellt das dar, was der Kunde von der Systemarchitektur wissen muss

Stepwise Formalization of Textual Requirements

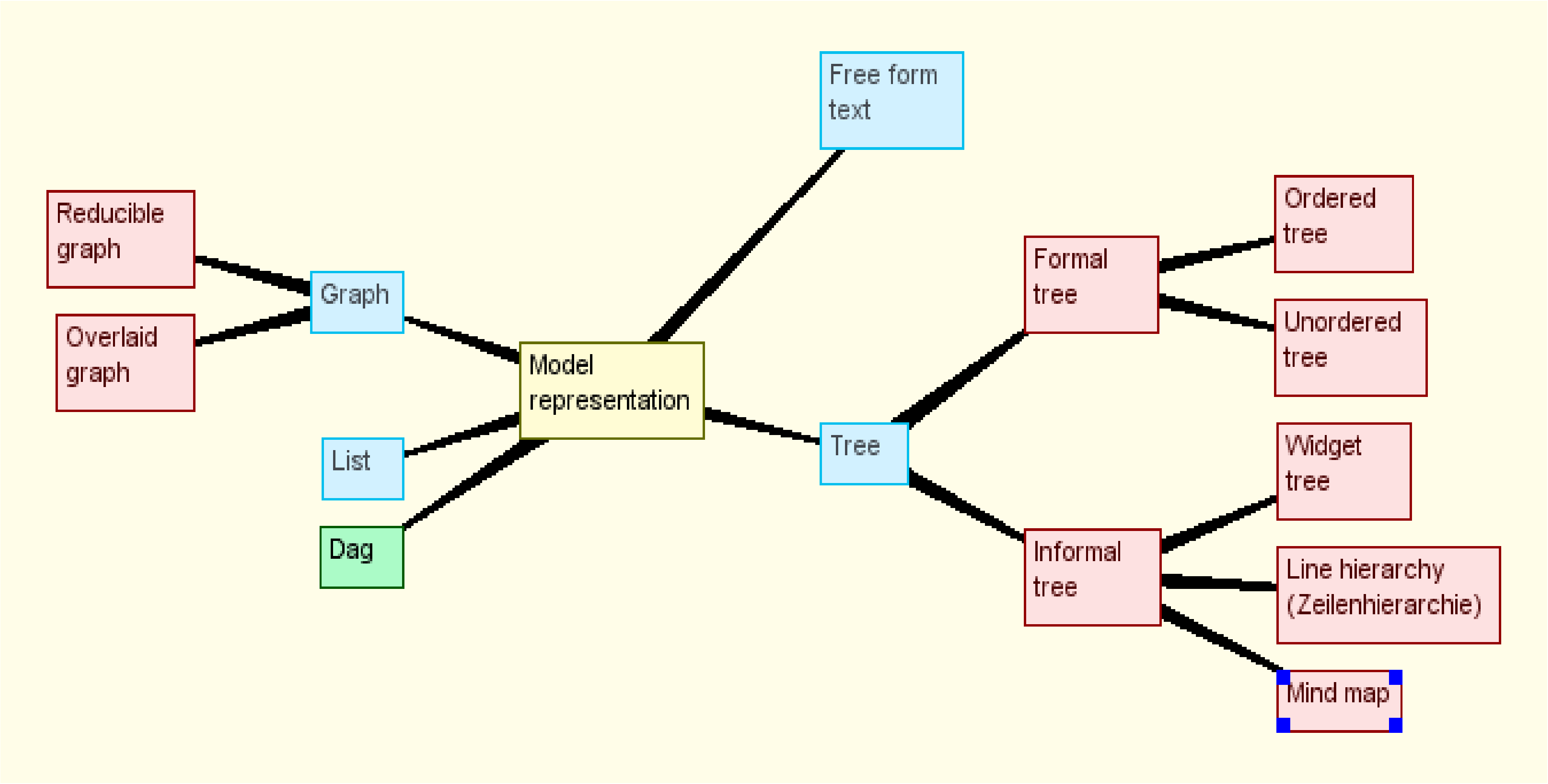
Specifications of requirements can be less or more formal (degree of formalization):

- Free form text
- List of things
- Tree of things
  - Mind map
  - Ordered Tree
  - Unordered tree
- Dag
- Graph based model
  - ..



Mind-Maps

- Tools offers simple brainstorming, easy structuring into trees
- Tool kdissert from KDE:



Stakeholder Analysis

- It is important to find all stakeholders of the system (Nutzer, Beteiligte, Involvierte, Betroffene)
- Stakeholders have different problems and also goals for the system, which can be quite different
  - Analyzing problems and goals for the different stakeholders separately is very important to discover *goal conflicts*
  - Goal conflicts will sabotage all project efforts (stakeholders work against each other)
  - Compromises have to be found
- Distinguish *actors (users), cooperating systems,* and other stakeholders
- Example: Fiscus project of German finance authorities
  - Stakeholders (tree):
    - End-users: tax payers
    - In all Deutsche Länder
      - Finance minister
      - Finance department
      - IT-departments

Mediation between Stakeholders' Goals

- Often, goals of different stakeholders are in conflict. We need *goal negotiations*
- *Show* to the other stakeholder*:*
  - Consequences of goals for the other stakeholder
    - Advantages and disadvantages
  - Consequences for you
- Try to find a *compromise*

“Es ist unmöglich, jemanden von etwas zu überzeugen. Man kann ihm nur helfen, sich selbst zu überzeugen.”

“Es ist unmöglich, jemanden von einer Idee zu überzeugen. Man kann ihm nur helfen, selbst die Idee zu bekommen.”

Example: Stakeholders of a Fictitious Course Management System

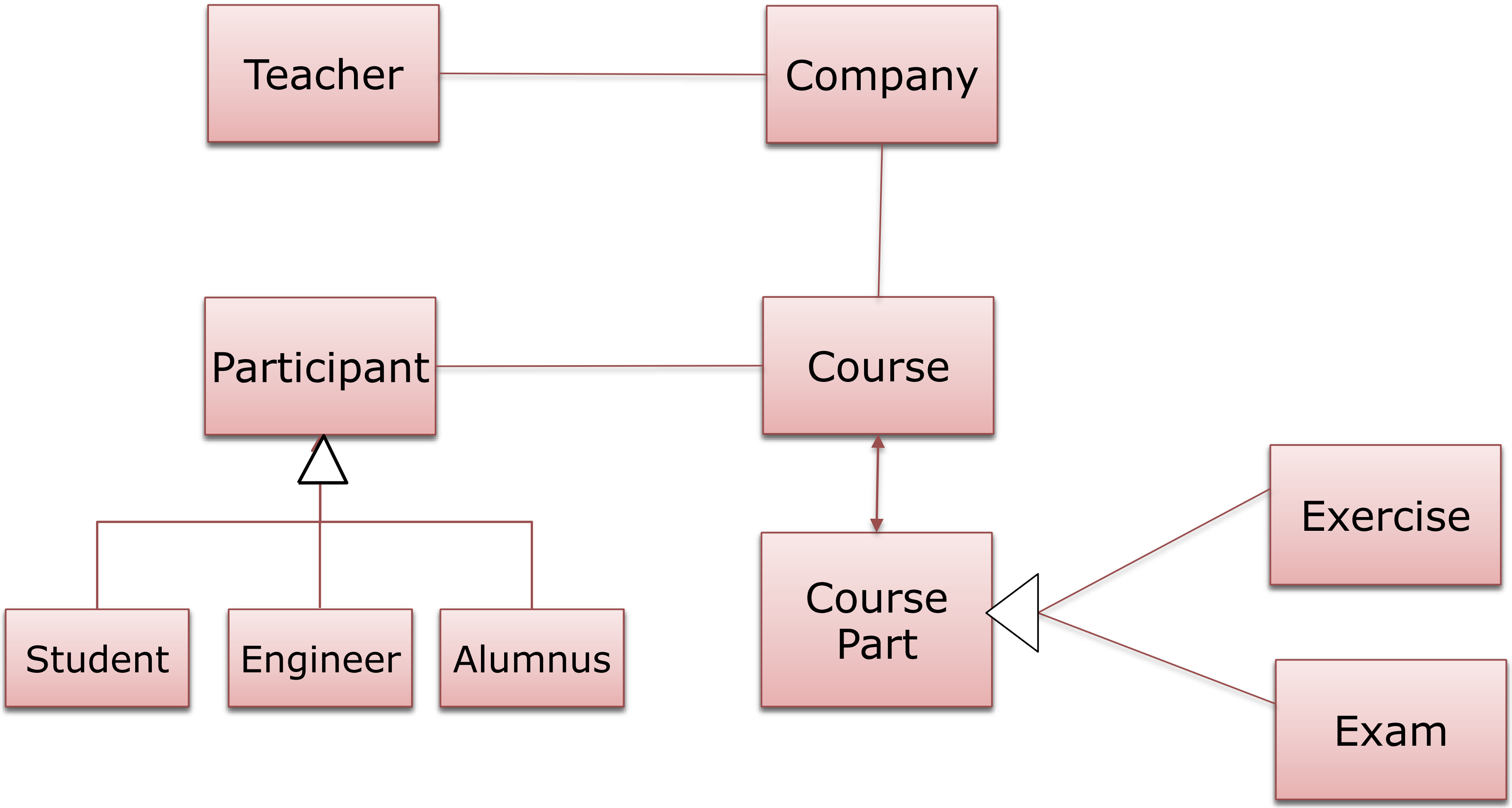
- Fictitious scenario:
  - CMT (Gesellschaft für **C**ourse **M**anagemen**T**) is a university-owned company responsible for industrial courses. It wants to construct a course management system for courses, given to the industry, alumni, and to students
- Users (actors):
  - Student: wants to get a diploma. Doesn't want to pay.
  - Engineer from industry: wants to learn, but also "escape" the company. Payment doesn't matter, company pays.
  - Alumnus, wants to learn something new and to meet old acquaintances again
  - Professors: teach courses, want to learn or be inspired from discussions
- University
  - Chancellor: wants the success of the CMT
  - Professors: similar
  - Assistants: dislike the course management system because they want to give the courses at the university (not over CMT)
- University administration:
  - Might not want the CMT, because it is a competing organization
- CMT:
  - Wants to earn money with the course system

Domain Analysis

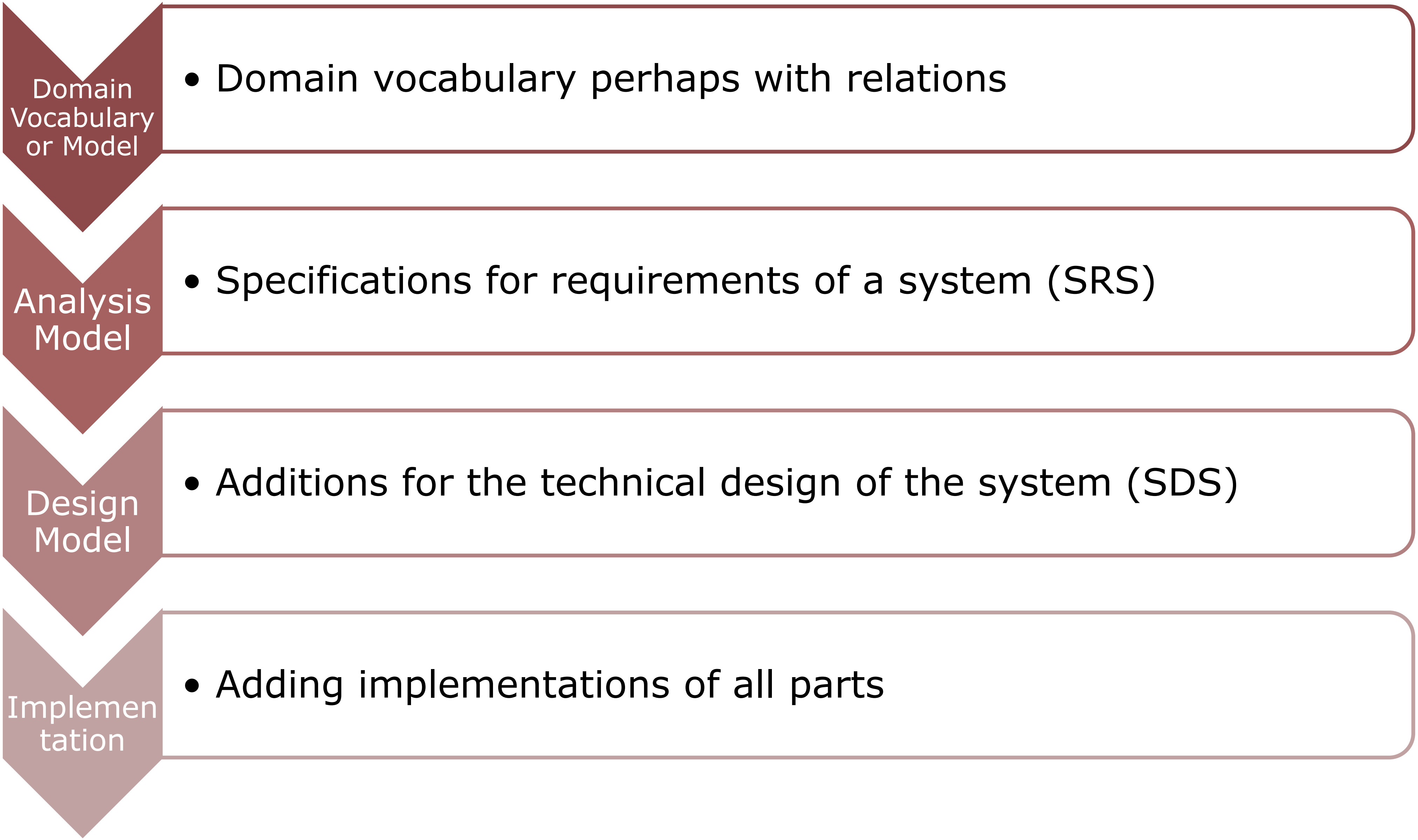
- The *domain analysis* creates a *domain model* of the domain of the application:
  - A *glossary* (vocabulary of concepts, terms)
  - A *taxonomy* (hierarchical organisation of terms in an inheritance hierarchy: Begriffshierarchie)
  - A vocabulary with relations (a *graph-based domain model*)
  - A vocabulary with constraints (an *ontology*)
- The domain analysis is important
  - For understanding the customer: requirements are phrased in the vocabulary of the domain
  - The domain model captures everything the customer will understand about the system. Hence, it is an important interface between customer and engineer
  - In particular for product lines (see later). Domain models

Example: Domain Model of a Course Management System

- Constraints:
  - Teacher != Student
  - Company == CMT
  - Alumnus doesn't have exams



Domain Model as First Part of the Software Product



Domain Vocabulary or Model

- Domain vocabulary perhaps with relations

Analysis Model

- Specifications for requirements of a system (SRS)

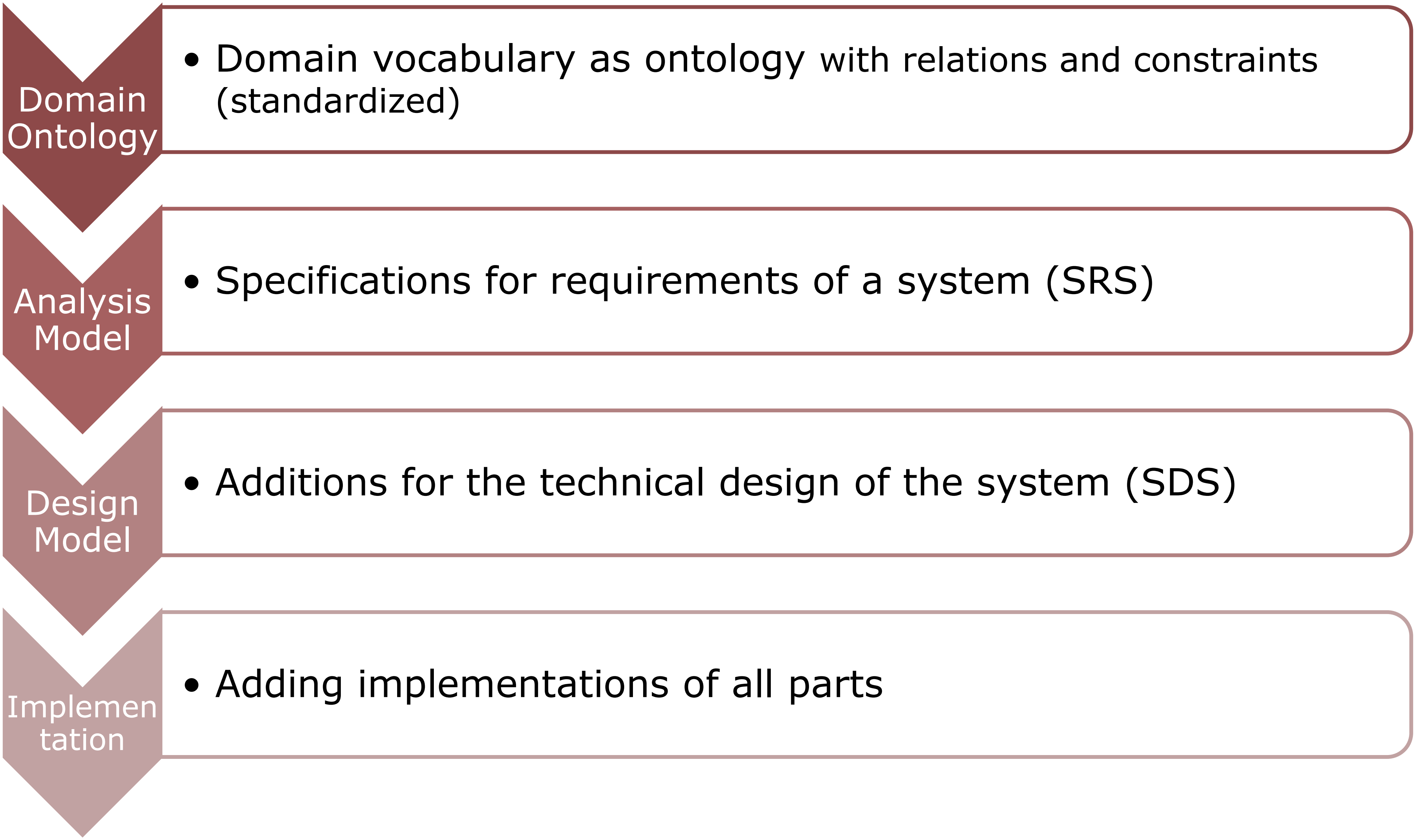
Design Model

- Additions for the technical design of the system (SDS)

Implementation

- Adding implementations of all parts

Domain Models with Ontology Reuse



Domain Ontology

- Domain vocabulary as ontology with relations and constraints (standardized)

Analysis Model

- Specifications for requirements of a system (SRS)

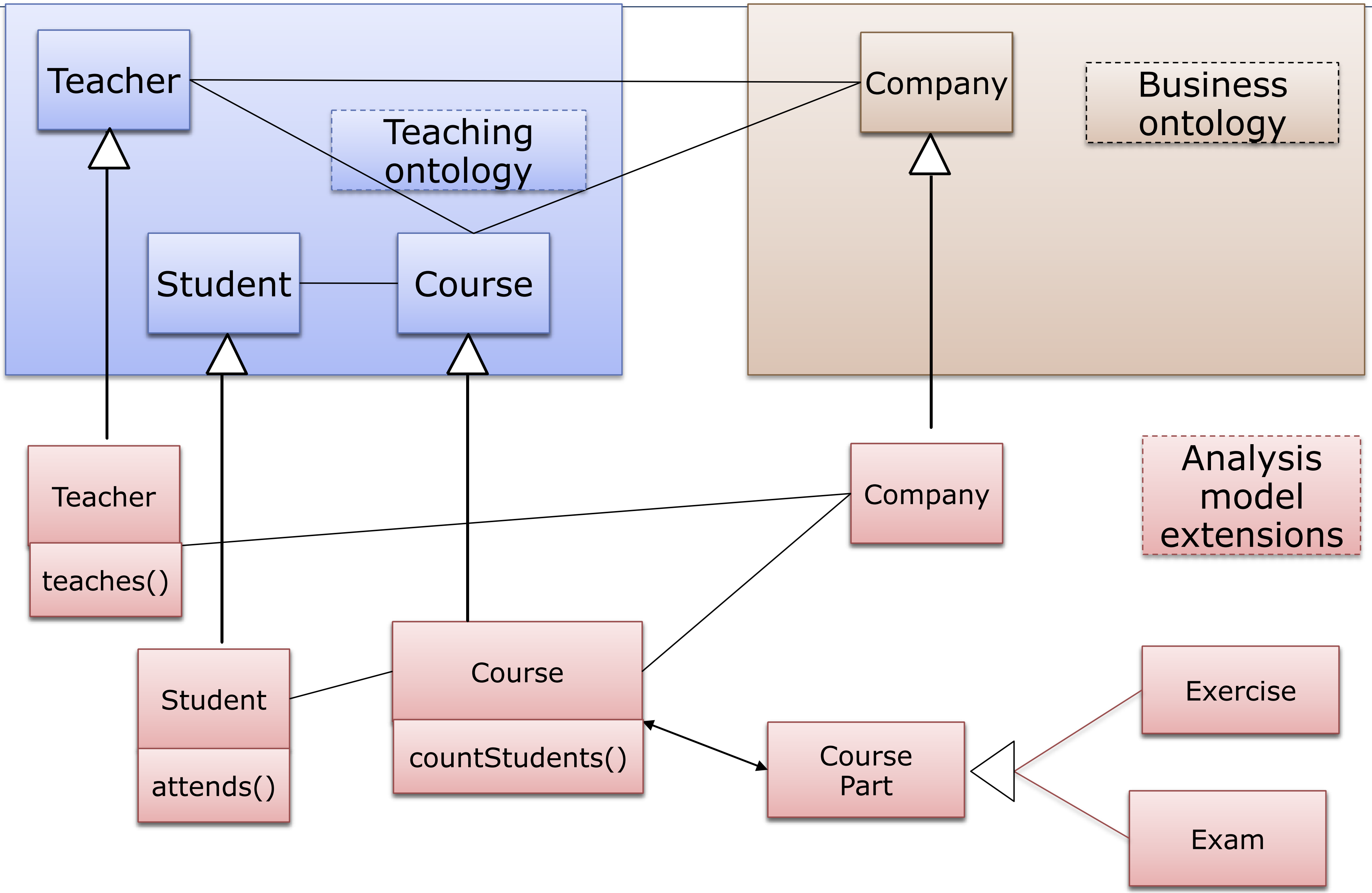
Design Model

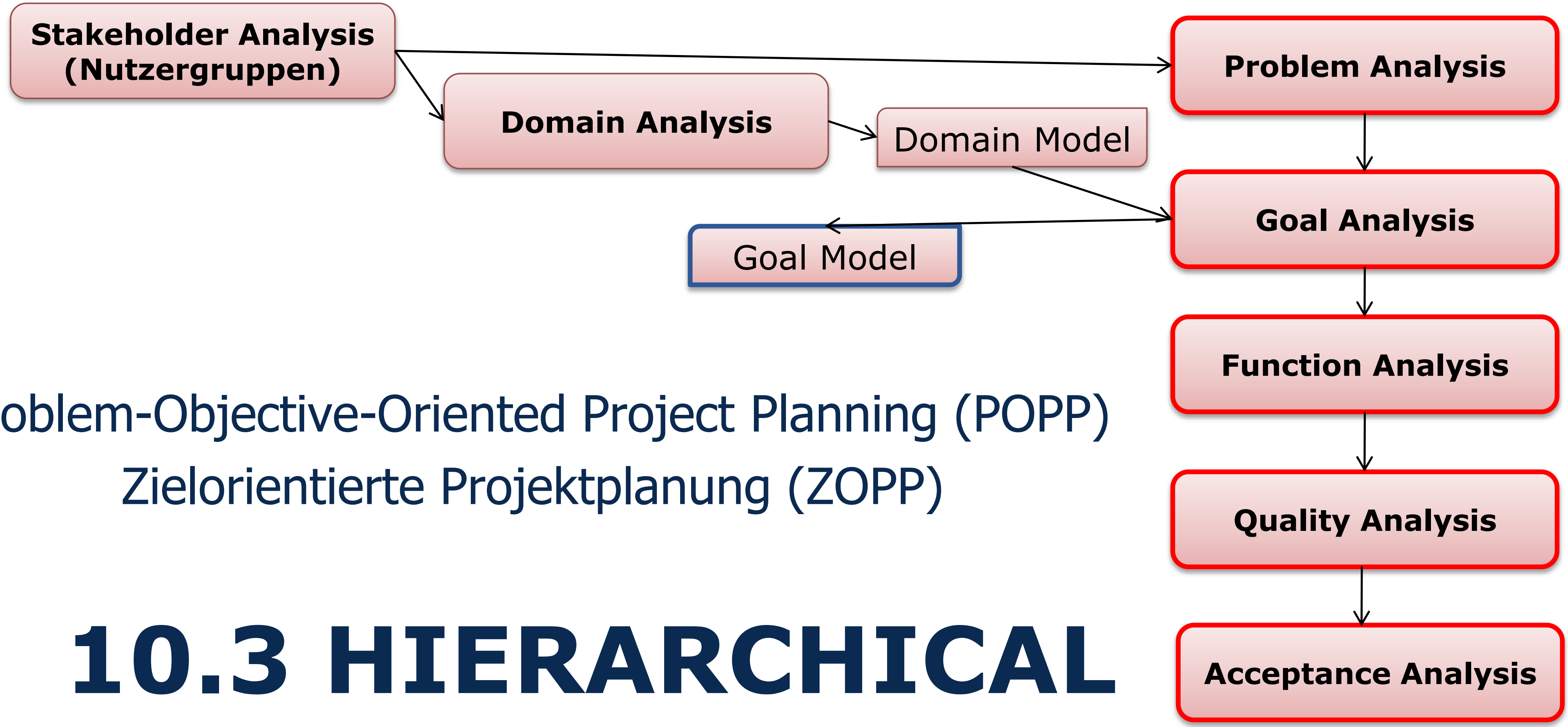
- Additions for the technical design of the system (SDS)

Implementation

- Adding implementations of all parts

Course Management System: Reuse of Ontological Terms





Problem-Objective-Oriented Project Planning (POPP)

Zielorientierte Projektplanung (ZOPP)

10.3 HIERARCHICAL PROBLEM-GOAL-FUNCTION ANALYSIS

The Hierarchical Problem-Goal-Function Analysis – A Simple Development Process

ZOPP origins from the GTZ (German development society), here adapted to software development

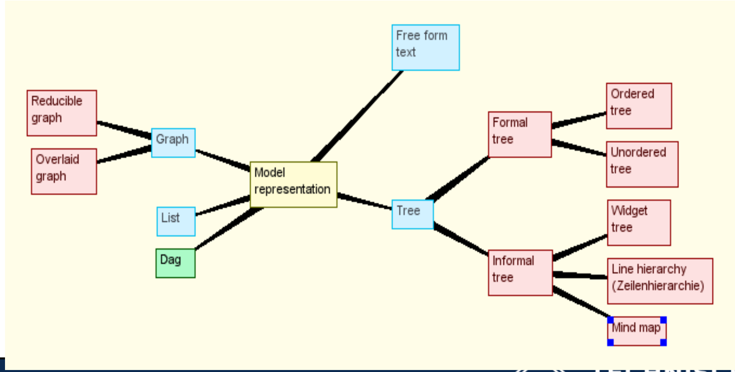
1. Problem analysis (Situation analysis, Ist-Analyse)
2. Goal analysis (objectives analysis, Soll-Analyse)
3. Functional requirements analysis
4. Non-functional requirements analysis (quality analysis)
5. Success criteria analysis
6. Construction of acceptance tests (Acceptance tests analysis)

General Rules for Problem Analysis

- What is the problem? Why do we need the system?
  - Problem analysis (IS analysis, IST-Analyse) is necessary, since the problem is not understood (and hence the requirements)
- Without problem no goal
- Without goal no requirements
- Without requirements no success criteria
- Without success criteria no measurable success
- We need to know the kernel problem
  - And distinguish from sub problems
  - ➔ Problem hierarchy

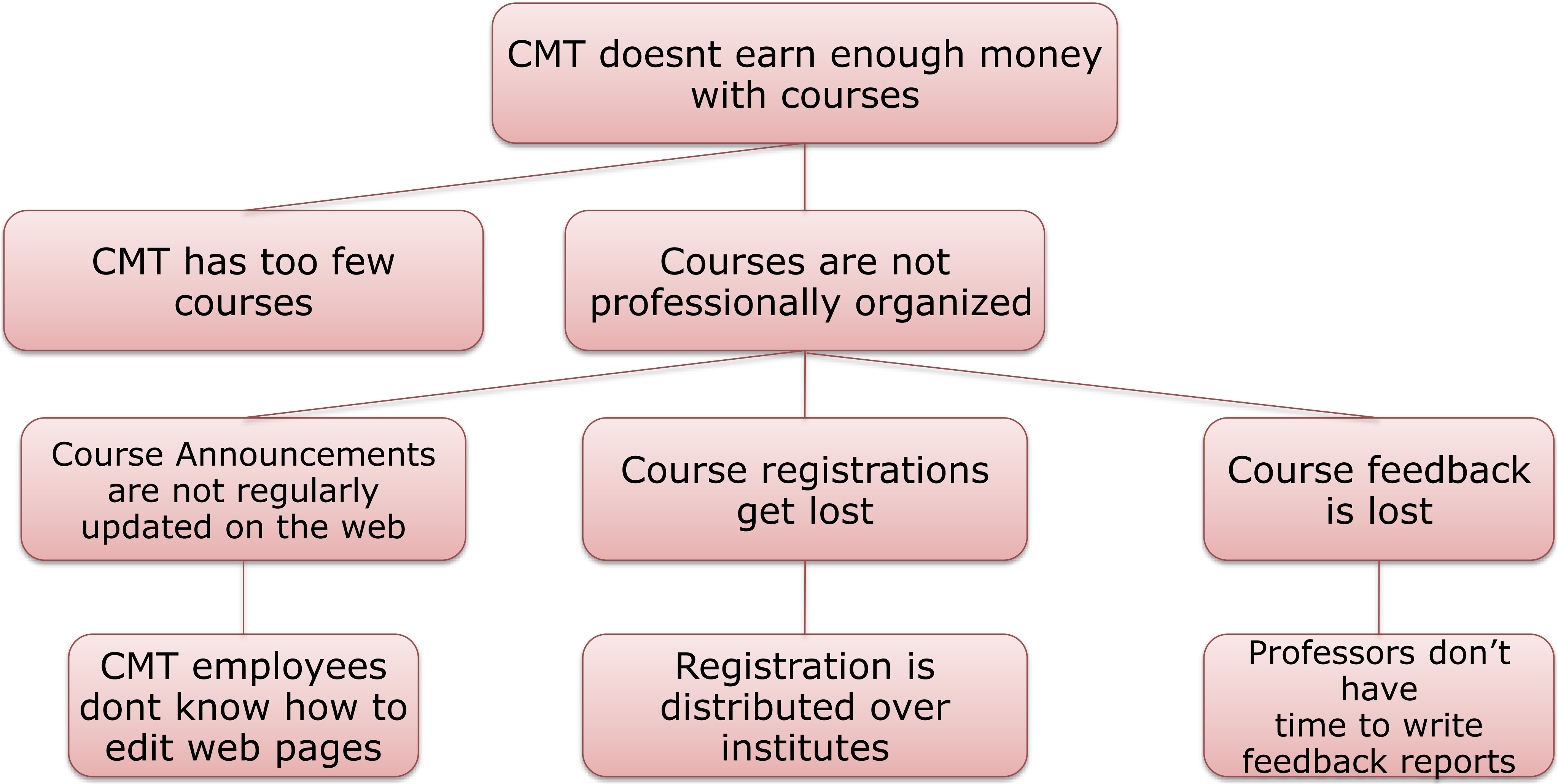
10.3.1. Problem Analysis: How To Develop a Problem Hierarchy

- Problem hierarchy: The main problem should be decomposed into smaller ones
  - Resulting in a problem tree
- **Steps:**
  - Phase 1: **problem hierarchy**
    - Brainstorming to collect problems and problem domains (collection phase)
    - Using a mind map or a tree tool is possible
    - Identification of the main problem
    - Elaborate a problem tree: arrange main and sub problems.
  - Phase 2: **cause-effect analysis**
    - Identify the *reasons* for the main problem
    - Identify the *consequences* of the main problem
  - Phase 3: **prioritization**
    - Prioritize sub problems
    - Check whether the hierarchy is complete and consistent



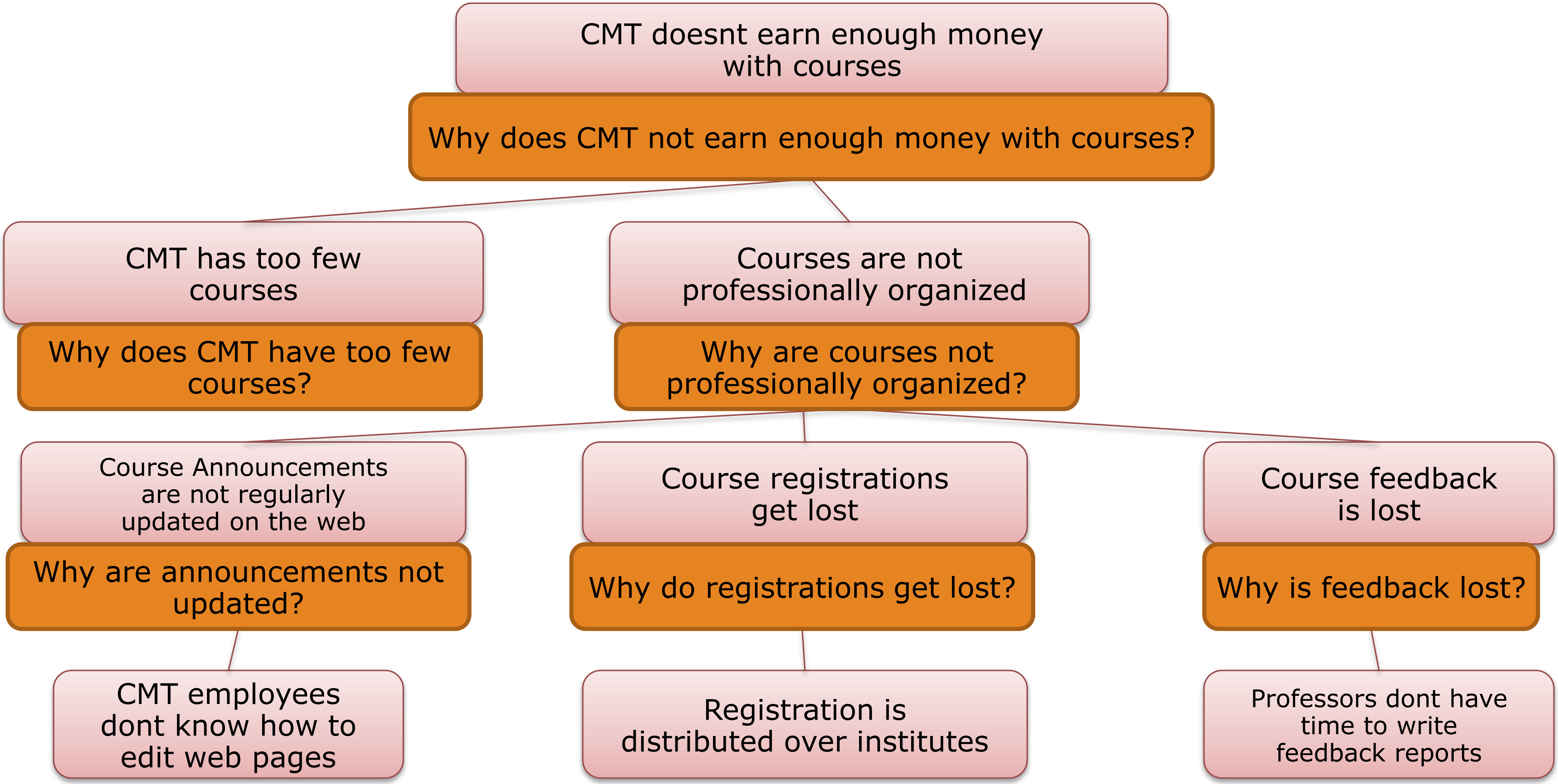
Course Management System: Problem Analysis for User CMT

- Problems must be arranged in super- and sub problems, decomposing on *reasons (reason hierarchy)*


CMT doesnt earn enough money with courses
CMT has too few courses
Courses are not professionally organized
Course Announcements are not regularly updated on the web
Course registrations get lost
Course feedback is lost
CMT employees dont know how to edit web pages
Registration is distributed over institutes
Professors don’t have
time to write feedback reports

Root Cause Analysis

- Problems can be decomposed by asking "Why"-questions


CMT doesnt earn enough money with courses
Why does CMT not earn enough money with courses?
CMT has too few courses
Why does CMT have too few courses?
Courses are not professionally organized
Why are courses not professionally organized?
Course Announcements are not regularly updated on the web
Why are announcements not updated?
Course registrations get lost
Why do registrations get lost?
Course feedback is lost
Why is feedback lost?
CMT employees dont know how to edit web pages
Registration is distributed over institutes
Professors dont have time to write feedback reports

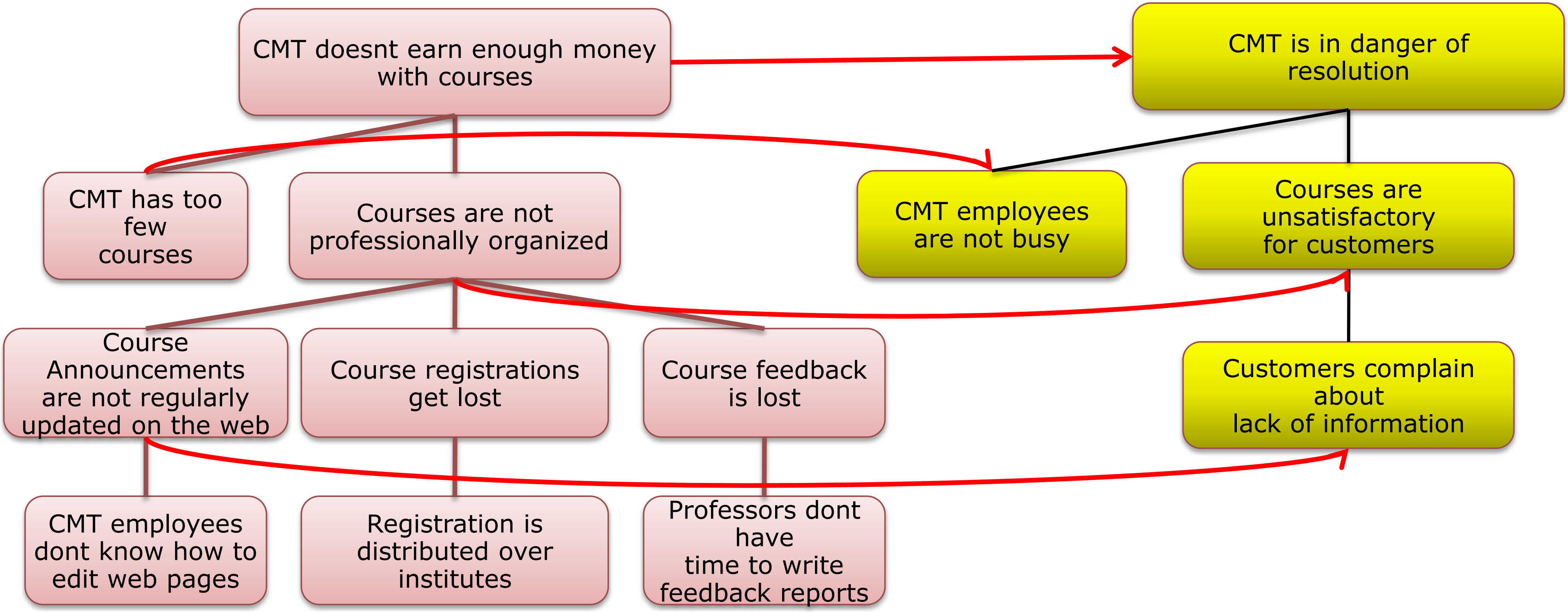
http://en.wikipedia.org/wiki/Root_cause_analysis
http://eliminatethemuda.com/2010/02/why-cant-i-save-money/

Problem Reasons and Problem Consequences

- It can also be distinguished between a hierarchy of “reasons” and a hierarchy of “consequences” (cause-effect, Ursache vs. Wirkung)
- Cause Hierarchy (in Root-Cause Analysis)
  - Sub problem is reason for super problem
- Consequence Hierarchy (in Root-Effect Analysis)
  - Subproblem is consequence of super problem
- Cause-and-Consequence model form together the *cause-effect graph*

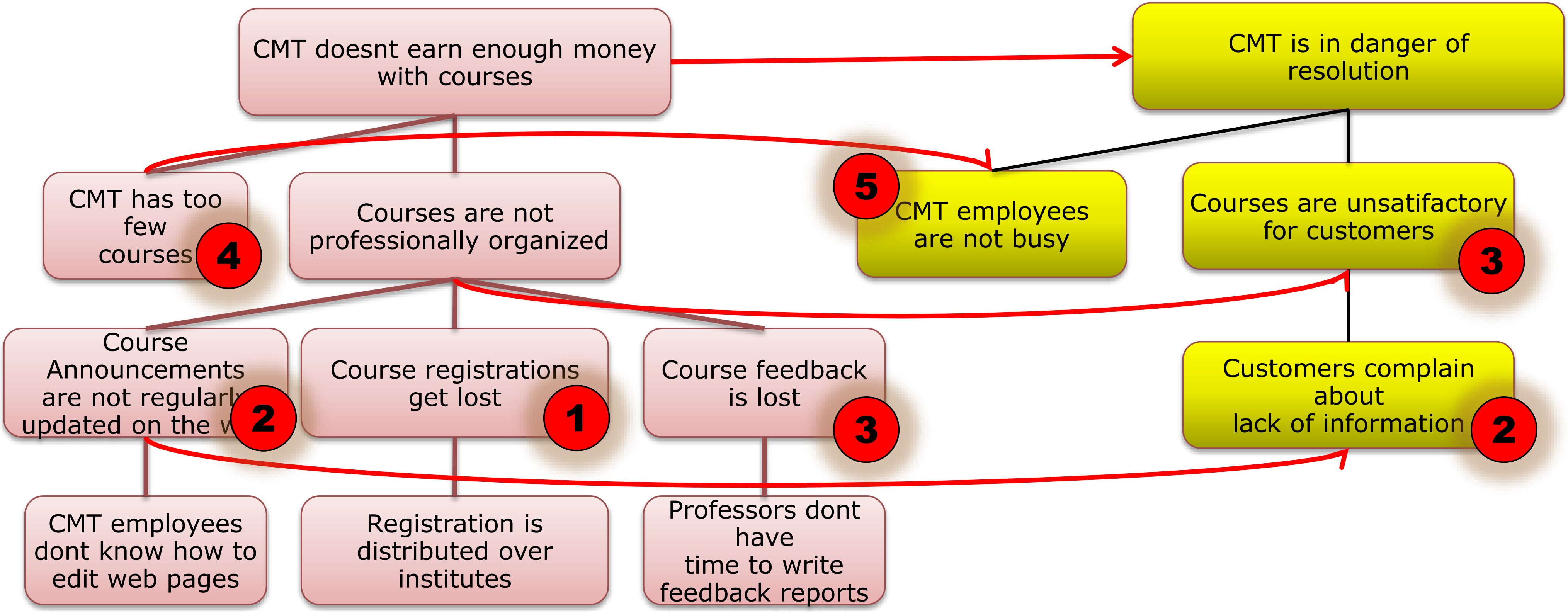
Course Management System: Cause-Effect Analysis for User CMT

- Separate between reasons and consequences:
  - Build separate reason and consequence hierarchies
  - Build up **cause-effect graph**
  - Focus later on causes, because effects are secondary


CMT doesnt earn enough money with courses
CMT is in danger of resolution
CMT has too few courses
Courses are not professionally organized
CMT employees are not busy
Courses are unsatisfactory for customers
Course Announcements are not regularly updated on the web
Course registrations get lost
Course feedback is lost
Customers complain about lack of information
CMT employees dont know how to edit web pages
Registration is distributed over institutes
Professors dont have time to write feedback reports

Course Management System: Problem Prioritization

- Go over the problem trees, the cause-effect graph and **prioritize**
- Importance is different from containment and problem decomposition



CMT doesnt earn enough money with courses
CMT is in danger of resolution
CMT has too few courses
4
Courses are not professionally organized
5
CMT employees are not busy
Courses are unsatifactory for customers
3
Course Announcements are not regularly updated on the w
2
Course registrations get lost
1
Course feedback is lost
3
Customers complain about lack of information
2
CMT employees dont know how to edit web pages
Registration is distributed over institutes
Professors dont have time to write feedback reports

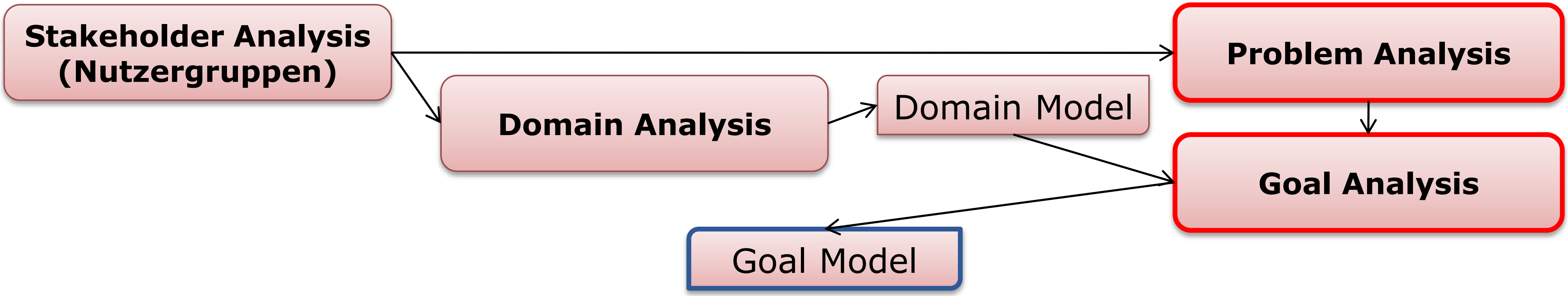
How to Phrase Problems

- Problems should be phrased negatively
  - Problems should not be intermixed with goals, requirements, nor solutions
- A problem is characterized
  - Not by the nonexistence of a solution, but by a negative state
  - By a bad feeling
- Try to separate reasons and consequences (causes and effects)

- A good problem analysis is half of the solution!
  - Problems generate money

10.3.2 Goal Analysis (Objectives Analysis)

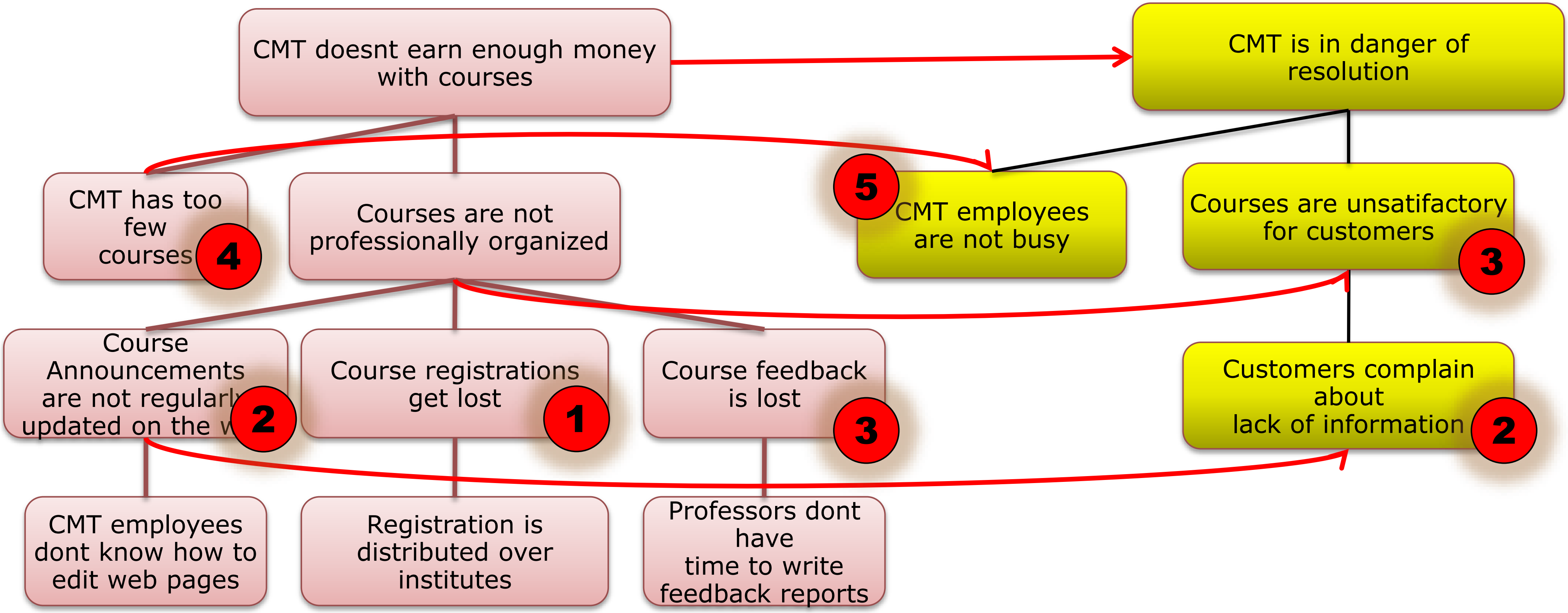
- Starting from the problems and sub problems of the problem hierarchy, the goal hierarchy is developed (Sollanalyse, Zielanalyse)
  - All negative statements of the problem hierarchy should be transformed into a goal statement
  - Every sub problem must relate to a sub goal (isomorphic mapping)
  - 4-6 sub-goals
  - All sub-goals of a goal are equally important
  - But can be prioritized (goal weight)
- Check completeness and correctness
- Create a dependency graph of goals



Important for Goals

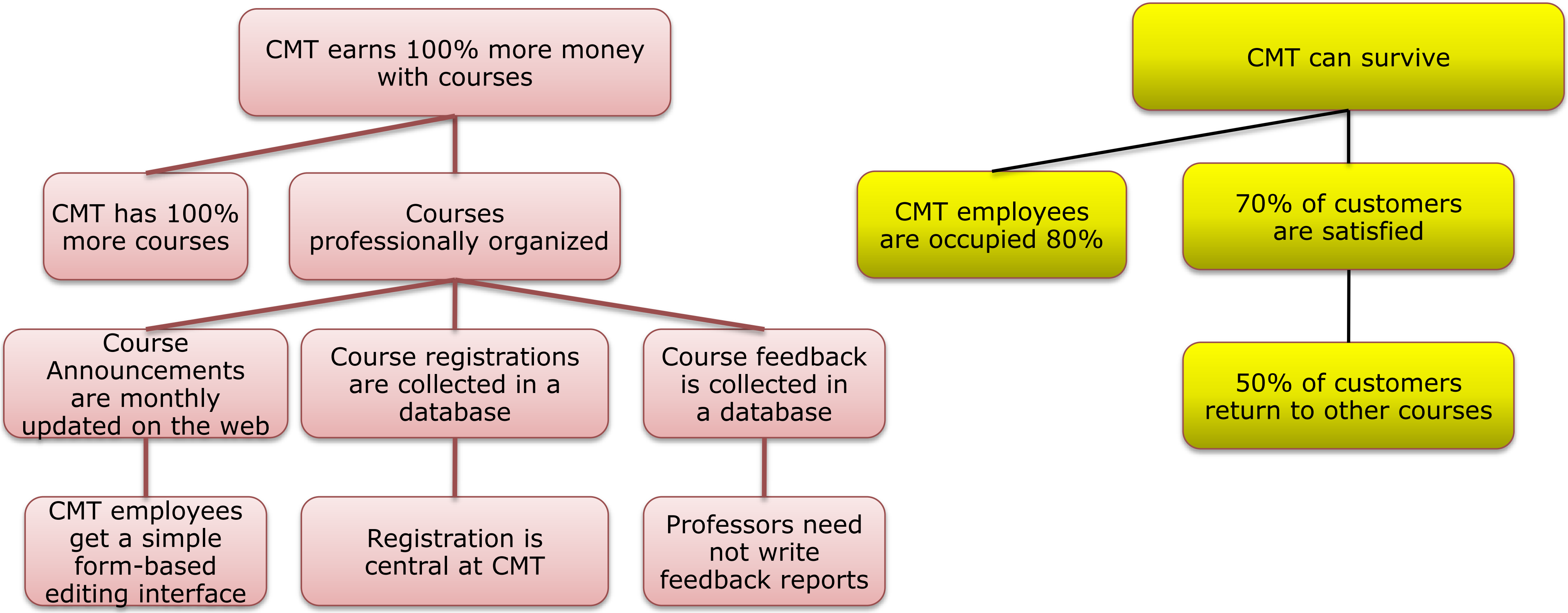
- Goals have dependencies:
  - Conflicting, independent, or complementary
- Goals underly a *part-of* relation
  - Main goals: the goal that the stakeholder wants to achieve
    - Often, but not always: the goal the system should achieve
  - Sub goals: partial goals that serve the main goal
- Goals have a time dimension
  - Intermediate goals: describe the features that must be achieved on the way to the result goals
  - *Result goals:* final goals
- Goals differ in their *obligation*
  - Obligatory goals (MUST)
  - Additional goals (MAY)
  - Non-goals (NEVER)
- Goal prioritization should comply to problem prioritization
  - Goal priority order can be derived from problem prioritization

Problem Trees with Causes and Effects (Rpt)


CMT doesnt earn enough money with courses
CMT is in danger of resolution
CMT has too few courses
4
Courses are not professionally organized
5
CMT employees are not busy
Courses are unsatifactory for customers
3
Course Announcements are not regularly updated on the w
2
Course registrations get lost
1
Course feedback is lost
3
Customers complain about lack of information
2
CMT employees dont know how to edit web pages
Registration is distributed over institutes
Professors dont have time to write feedback reports

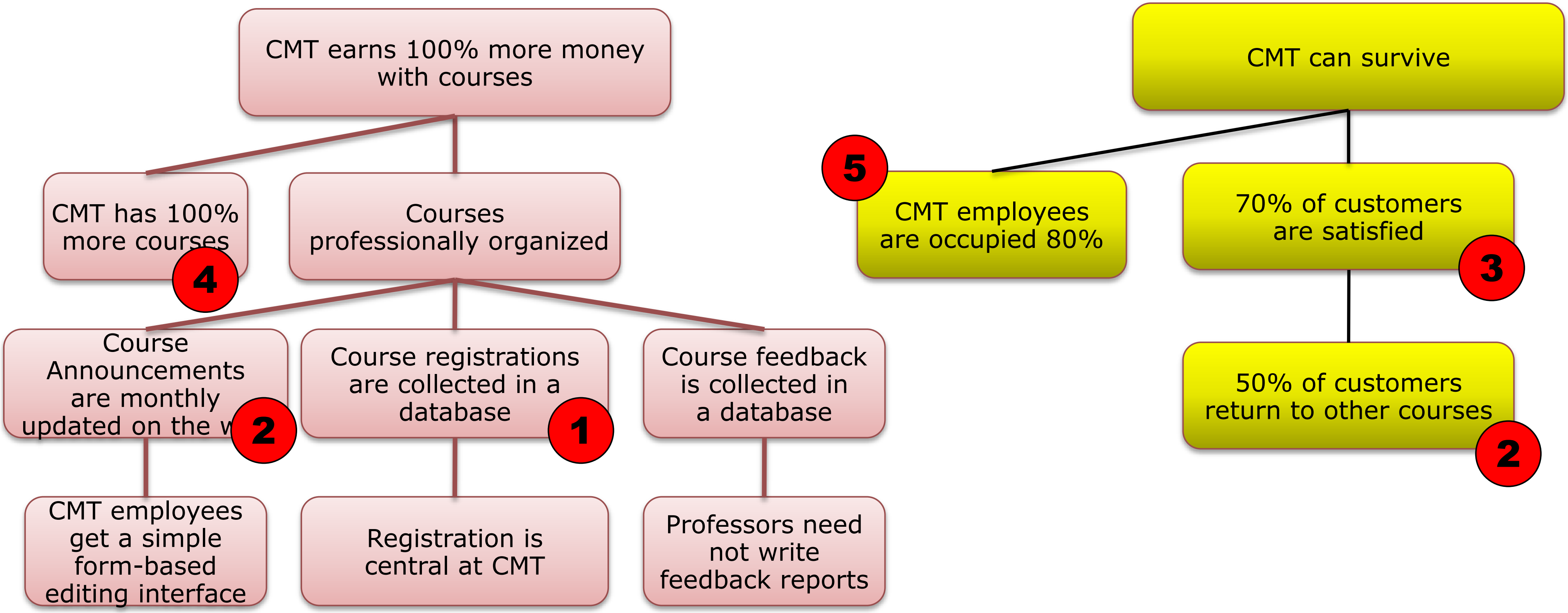
Course Management System: Goal Analysis for User CMT

- From reasons and consequences, goals can be derived


CMT earns 100% more money with courses
CMT has 100% more courses
Courses professionally organized
Course Announcements are monthly updated on the web
Course registrations are collected in a database
Course feedback is collected in a database
CMT employees get a simple form-based editing interface
Registration is central at CMT
Professors need not write feedback reports
CMT can survive
CMT employees are occupied 80%
70% of customers are satisfied
50% of customers return to other courses

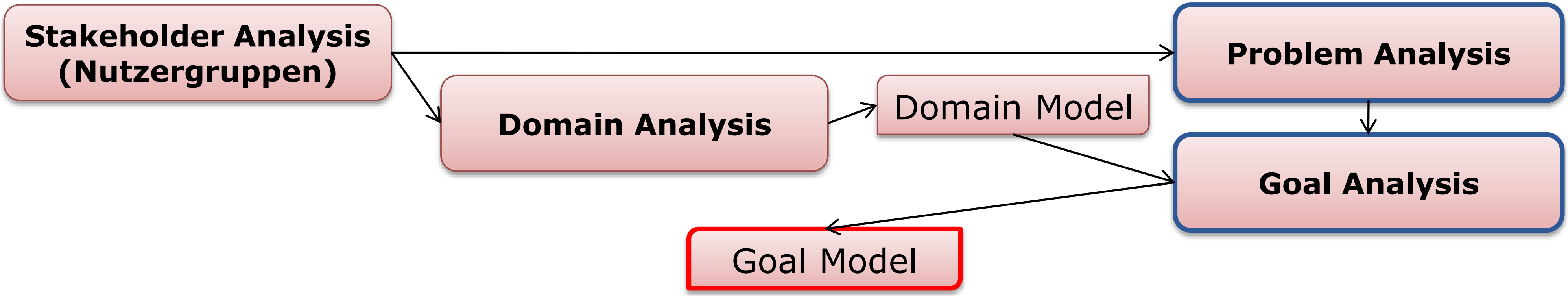
Course Management System: Goal Prioritization

- Goal priorities are derived from problem priorities

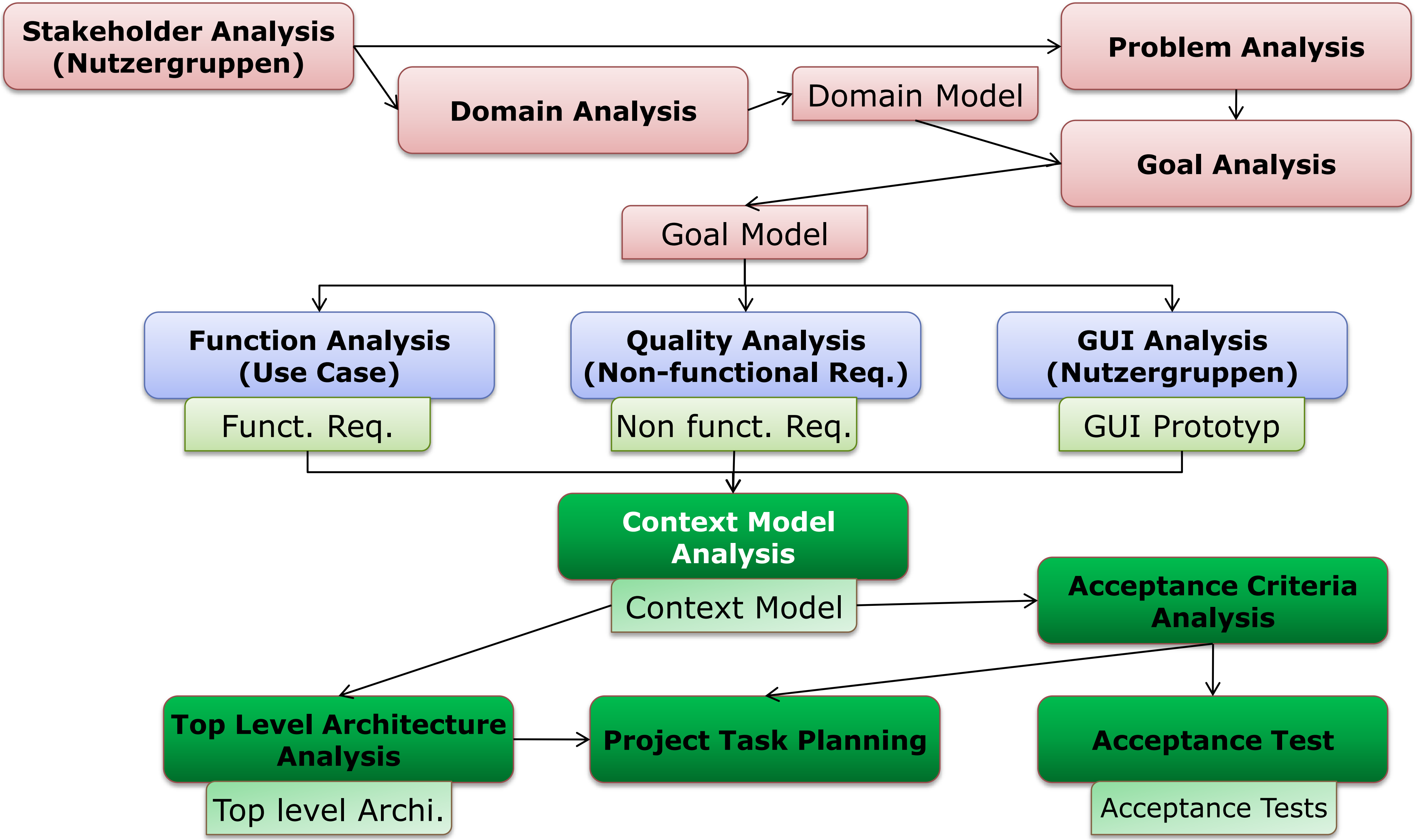

CMT earns 100% more money with courses
CMT has 100% more courses
4
Courses professionally organized
Course Announcements are monthly updated on the w
2
Course registrations are collected in a database
1
Course feedback is collected in a database
CMT employees get a simple form-based editing interface
Registration is central at CMT
Professors need not write feedback reports
CMT can survive
5
CMT employees are occupied 80%
70% of customers are satisfied
3
50% of customers return to other courses
2

How To Phrase a Goal

- Positive:
  - A goal should not revive sentiments. Don't phrase it negatively
- Attractive or motivating:
  - The goal should *drive* the stakeholders and developers, not *demotivate* them
  - [Merkel 2011: Germany lives of natural energy in 2022]
- Comprehensible:
  - A misunderstood goal is worse than none
- Easy-to-overlook:
  - Too complex goals make life difficult
- Objectively:
  - Try to think about other stakeholders, what they might be interested in. Try to phrase goals objectively
- Realistically:
  - The stakeholder should be able to reach the goal



Analysis in Detail

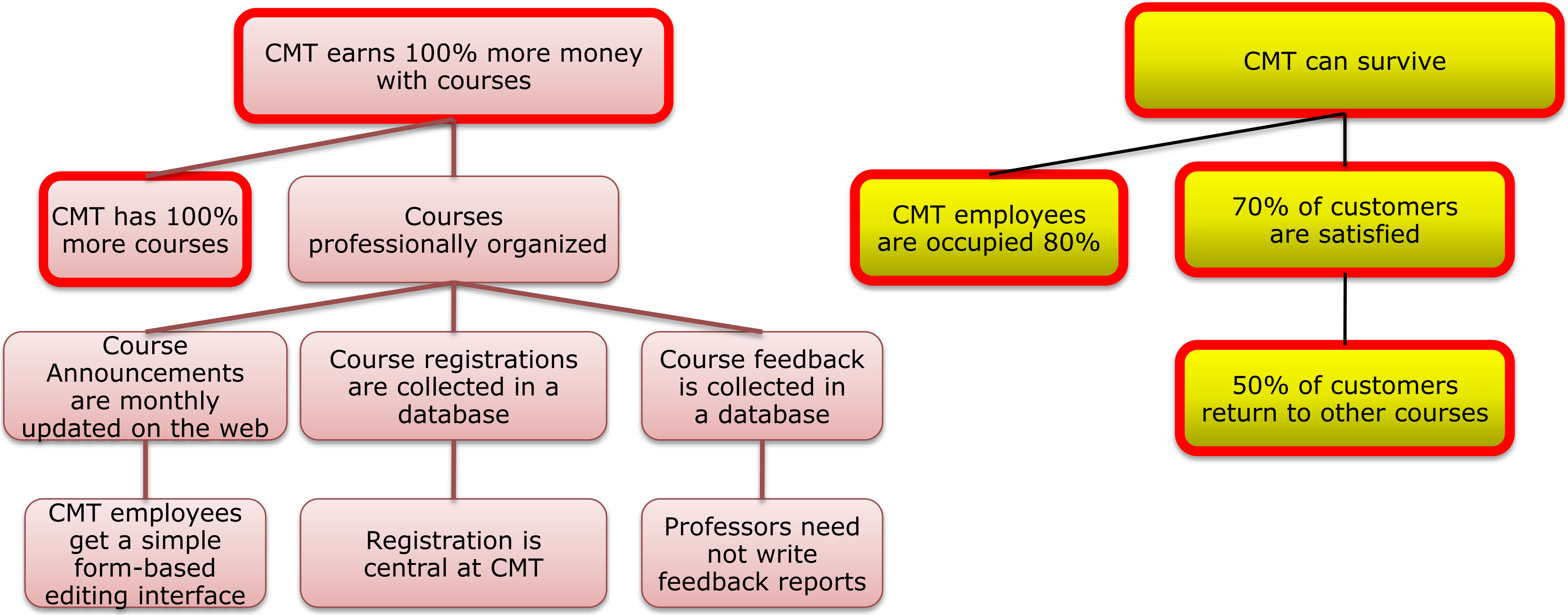


10.3.3 Functional Requirement Analysis

- From the goal analysis result the **functional**, **non-functional** and **semi-functional** requirements
  - If all requirements are fulfilled, goals are reached and the problems are solved
  - Construct a requirements tree from the goal tree
- Goals are unlike requirements:
  - Requirements are desired features of the system to solve the problems and to achieve the goals
  - Not all goals can be transformed into functions of the system. A *manifestable goal* is a goal that can be manifested in the functional requirements

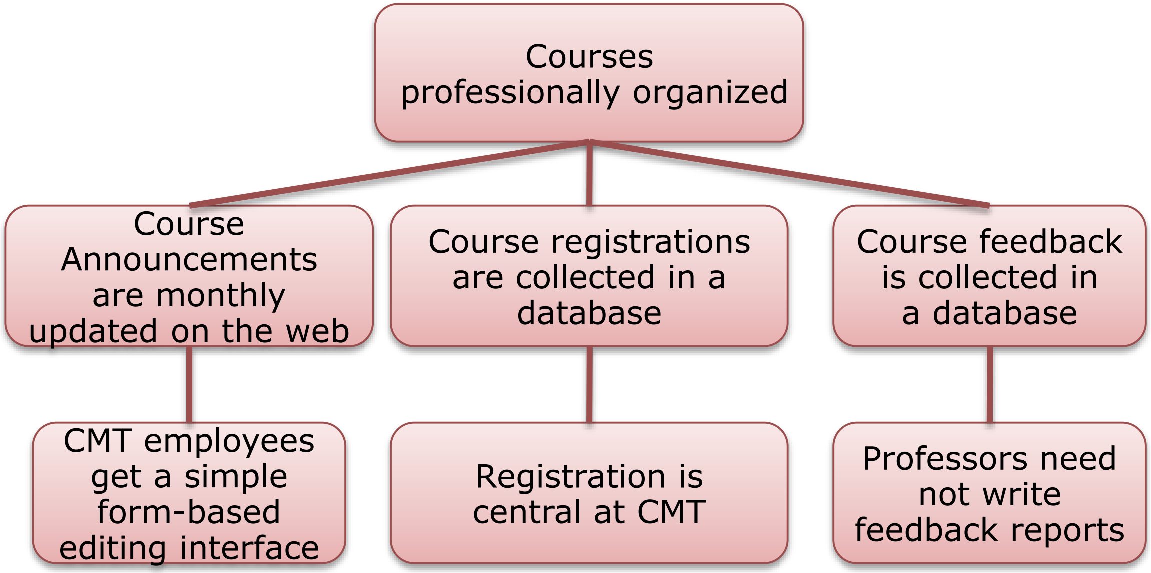
Course Management System: Function Analysis for User CMT

- From goals, some preliminary functions can be derived
  - Separate quantifiable goals from others
  - Quantifiable goals are arranged in a *function tree*
- Question: do we have non-functional requirements for the CMS?


CMT earns 100% more money with courses
CMT has 100% more courses
Courses professionally organized
Course Announcements are monthly updated on the web
Course registrations are collected in a database
Course feedback is collected in a database
CMT employees get a simple form-based editing interface
Registration is central at CMT
Professors need not write feedback reports
CMT can survive
CMT employees are occupied 80%
70% of customers are satisfied
50% of customers return to other courses

Function Trees of the Course Management System

- Function trees result



Courses professionally organized

Course Announcements are monthly updated on the web

Course registrations are collected in a database

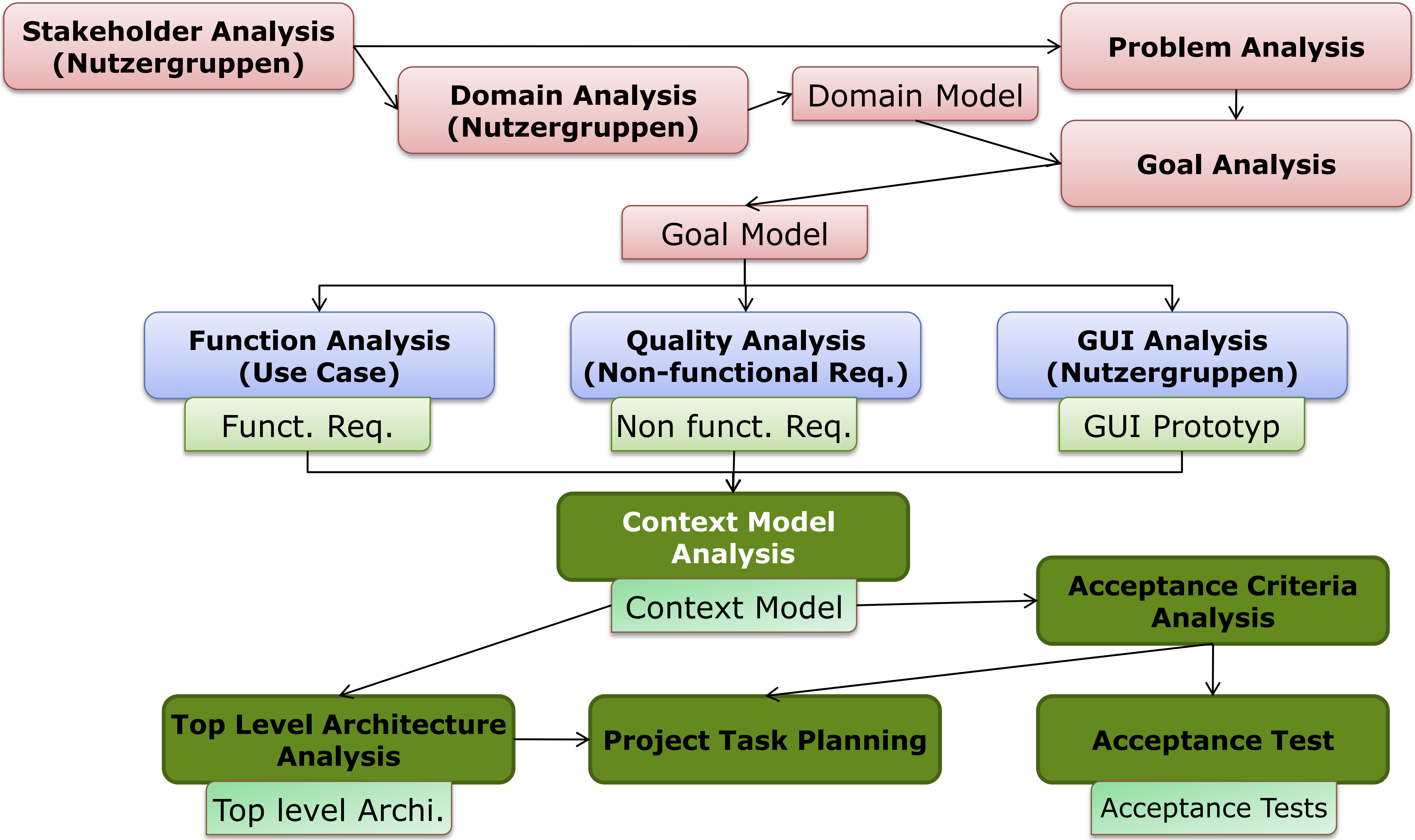
Course feedback is collected in a database

CMT employees get a simple form-based editing interface

Registration is central at CMT

Professors need not write feedback reports

Analysis in Detail

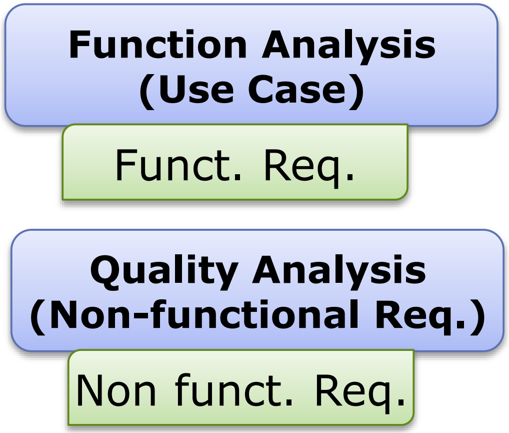

Stakeholder Analysis (Nutzergruppen)
Problem Analysis
Domain Analysis (Nutzergruppen)
Domain Model
Goal Analysis
Goal Model
Function Analysis (Use Case)
Funct. Req.
Quality Analysis (Non-functional Req.)
Non funct. Req.
GUI Analysis (Nutzergruppen)
GUI Prototyp
Context Model Analysis
Context Model
Acceptance Criteria Analysis
Top Level Architecture Analysis
Top level Archi.
Project Task Planning
Acceptance Test
Acceptance Tests

10.3.4 ANALYSIS OF NON-FUNCTIONAL REQUIREMENTS (QUALITY REQUIREMENTS)

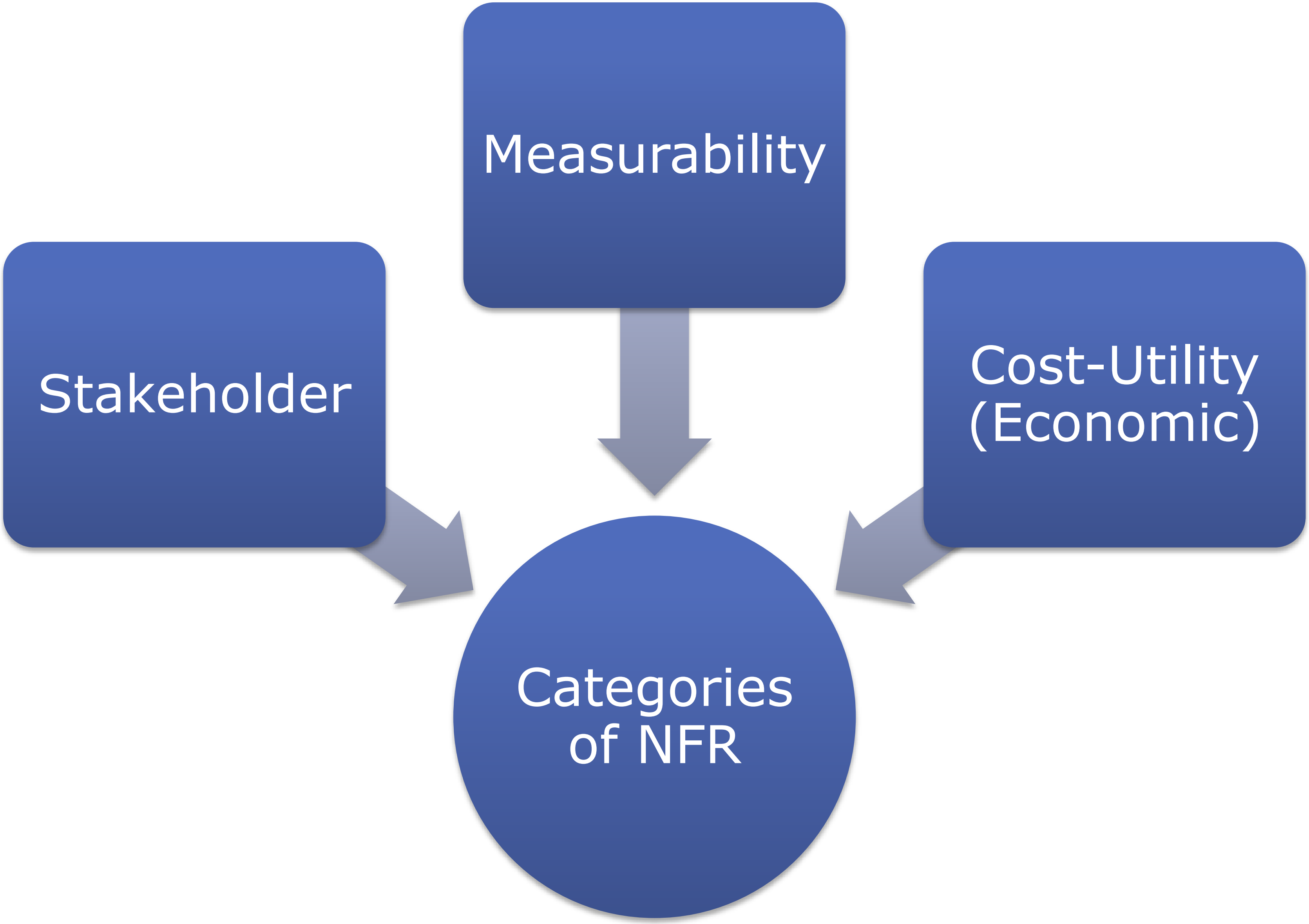
Functional, Semi- and Non-Functional Requirements (Quality Analysis)

- Functional requirements make the software system "fit for purpose"
  - Are boolean k.o.-criteria
- **Non-functional requirements** (NFR, “ilities”) describe quality features and can be multi-classified
  - NFR are related to a stakeholder (developer, user, maintainer,..): **stakeholder facet**
    - Development qualities
    - Usage qualities (for users)
    - Manager qualities
  - NFR should be *measured* with *measurements (quantitative scales)* i.e, have a **measurability facet**
    - Measurable, Empirical, Paperware
  - **Efficiency Reuirements** are NFR which relate *utility* with *cost* (cost-utility function)

- **Semi-functional requirements** are usually non-functional, but are vital, i.e., *turn into* functional ones
  - Reliability
  - Robustness: system doesn't break
  - Efficiency: system shows sufficient throughput
  - Usability
  - Resource constraints can be met


Function Analysis (Use Case)
Funct. Req.
Quality Analysis (Non-functional Req.)
Non funct. Req.

Categories (Facets) of Non-Functional Requirements



10.3.4.a Stakeholder Facet for Non-Functional Requirements

- **Development qualities** (for developer, Entwicklerqualitäten)
  - Reusability (Evolution quality)
    - **Variability**: from the system, easily other variants can be produced
    - **Extensibility**: the system can be extended easily
    - **Portability**: the system can be ported to new platforms easily
    - **Evolvability**: the system can evolve easily (well documented, stable concepts)
    - **Middleware scalability**: easy to use in parallel, persistent, distributed, changing contexts
- **Business qualities** (for manager, Geschäftsqualität)
  - Market attractivity
  - Good return on investment (ROI)
  - Sustainability (Nachhaltigkeit)
  - Good markets
- **Usage qualities** (for user, at run time, Benutzungsqualitäten)
  - **Usability**: it is easy to use the system
  - **Invisibility**: user can't notice the (embedded) system
  - **Security**: system resists against attacks
  - **Safety**: system doesn't destroy things (safety-critical systems)
  - **Personalizability**: system can be adapted to users or contexts
  - **Resource constraints**: real-time conditions are met, memory or energy consumption conditions

10.3.4.b Measurability Facet for Non-functional Requirements

- **Non-measurable quality**
  - **Paperware quality:** paper persuades that quality exists
  - **Slideware quality:** material works for a presentation, but not for more
- **Verifiable quality: a specification exists, against which the quality feature can be verified**
  - A **metric scale** exists
    - E.g., time deadline in real-time behavior
  - A **threshold** decides whether the requirement is met
    - Ex.: “system will react within 5ms”
- **Measurable quality:** a quantitative scale exists
  - Cardinal scale (Kardinalskala, metric): ex.
  - Inverval scale (Intervallskala, metric, but in pairs)
  - Ratio scale (Verhältnisskala): Interval scale with origin point (Nullpunkt). Verhältnisse dürfen gebildet werden.
  - Absolute scale (Absolutskala): Absolute values
  - http://de.wikipedia.org/wiki/Skalenniveau
- **Categorial quality**
  - Nominal scale (non-metric, Notenskala mit Kategorien, Güteklassen, Noten, Enumeration)
  - Ordinal scale (Ordinalskala, Rangordnung), for prioritizations
- **Empirical quality:** empirical experiments can show the quality

10.3.4.c Economic Facet: Olympic vs. Efficiency NF-Requirements

- A measurable NFR has an *economic facet,* it can be *olympic* or *efficiency-based:*
- An **olympic (utility) requirement** measures the utility of a system feature
- An **efficiency requirement** measures the utility-cost ratio of a system feature (cost-utility relation or function)

Efficiency NFR (Efficiency Quality)

- **Efficiency** describes the relation between cost and utility (resource consumption and produced utility) by a **cost-utility function**
- **Efficiency** in general describes the extent to which time or effort is well used for the intended task or purpose.
  - It is often used with the specific purpose of relaying the capability of a specific application of effort to produce a specific outcome effectively with a minimum amount or quantity of waste, expense, or unnecessary effort. (Wikipedia)
- Ex.:

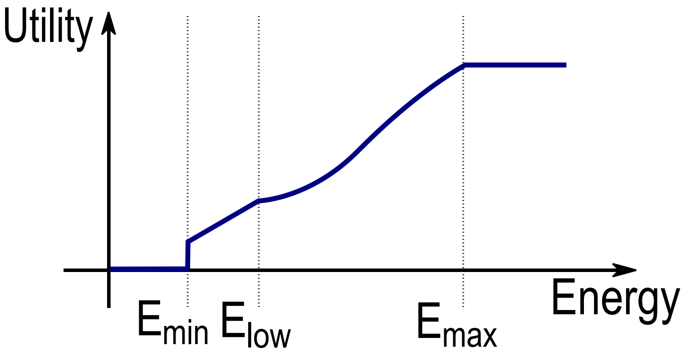
1000 frames per second = utility

1 Ghz clock rate and 10ms access time for disk

Ct. Measurable Efficiency NFR: „Efficiency Qualities"

- An **efficiency quality (efficiency requirement)** is a measurable non-functional requirement of a system that obeys a *cost-utility function (CUF)*
- Examples:
  - Performance: cost: processor price, utility: Gigahz
  - Soft real-time: cost: number of cores, utility: time
  - Throughput: cost: number of servers, utility: transactions per second
- Cost-utility functions can be continuous:
  - linear,
- point-wise defined
- piecewise defined (streckenweise definiert)

Ex:.: Energy-Utility-Function
is usually piecewise defined:
Depends on Applications, Resources, Context



10.3.5 The Context (Interface) Model

- The **context model (interface model)** describes the interfaces of the system to the outer world
  - Input and output streams
  - Input forms
  - Output queries
  - Typed by the domain model (data dictionary)

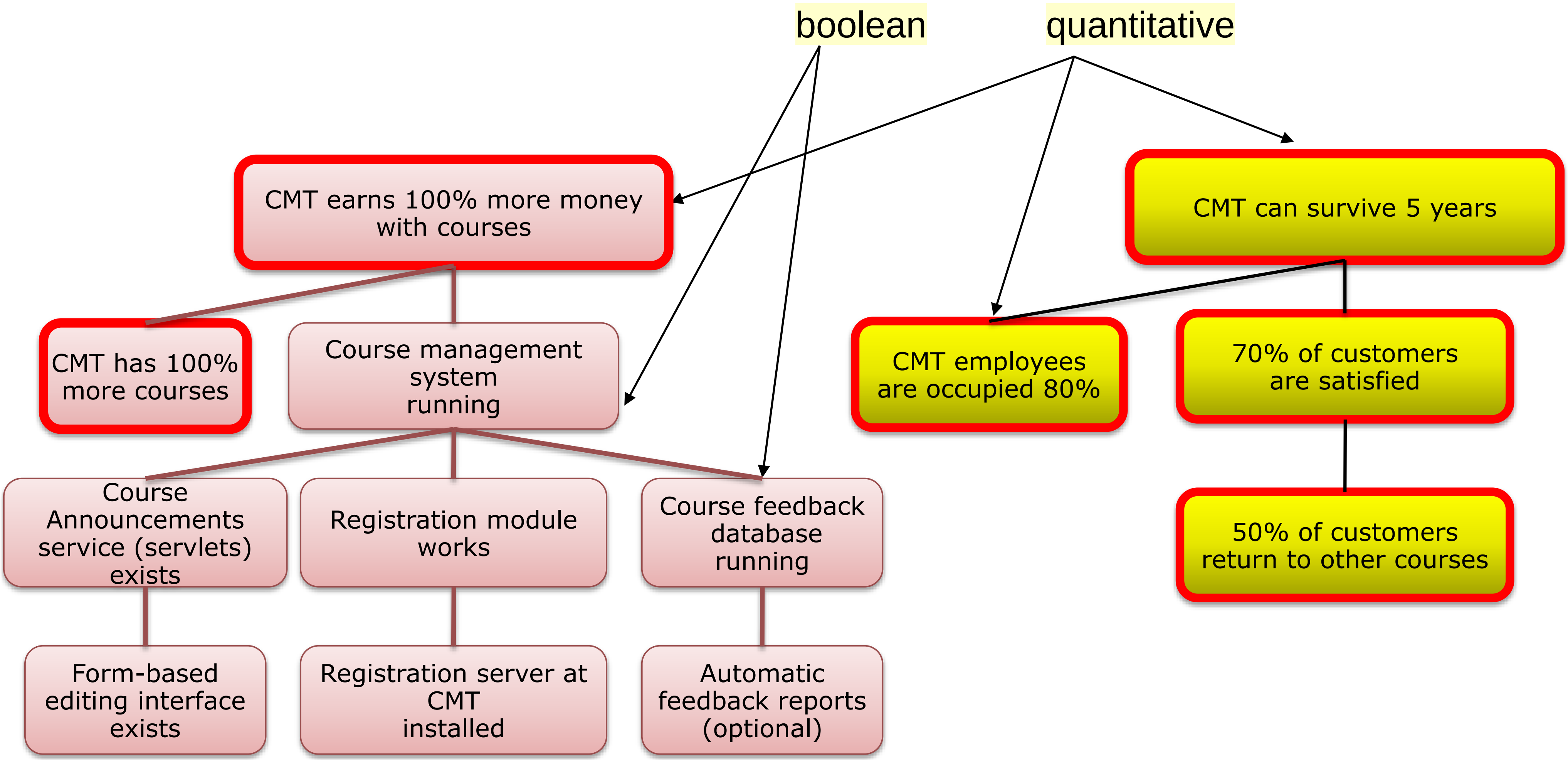
- Representations:
  - Class diagrams with functions and their contracts
  - Data flow diagrams
  - Colored Petri nets
  - Function trees and modules

10.3.6 Success Criteria Analysis

- The **success criteria analysis** finds out when a project will be accepted.
  - It is basis for selling the product, or the acceptance test of the product
- The success criteria list contains a *mandatory list* of *measurable* functional, non-functional, and semi-functional requirements
  - A success criterion can be *measured* (boolean, metric, ...)
    - Functional requirements are boolean
      - "Does the system print all GUI?"
    - Non-functional requirements are metric
      - Performance criteria: "the user should not wait more than 1 sec"
      - Stress criteria "500 users should be possible"
  - Mandatory test cases, often derived from mandatory use cases or function trees (*acceptance test cases*)
  - Usability considerations (Can the system be used easily?)
  - Architectural considerations (Does the client want to evolve the system himself? Then the architecture must be easy to comprehend)

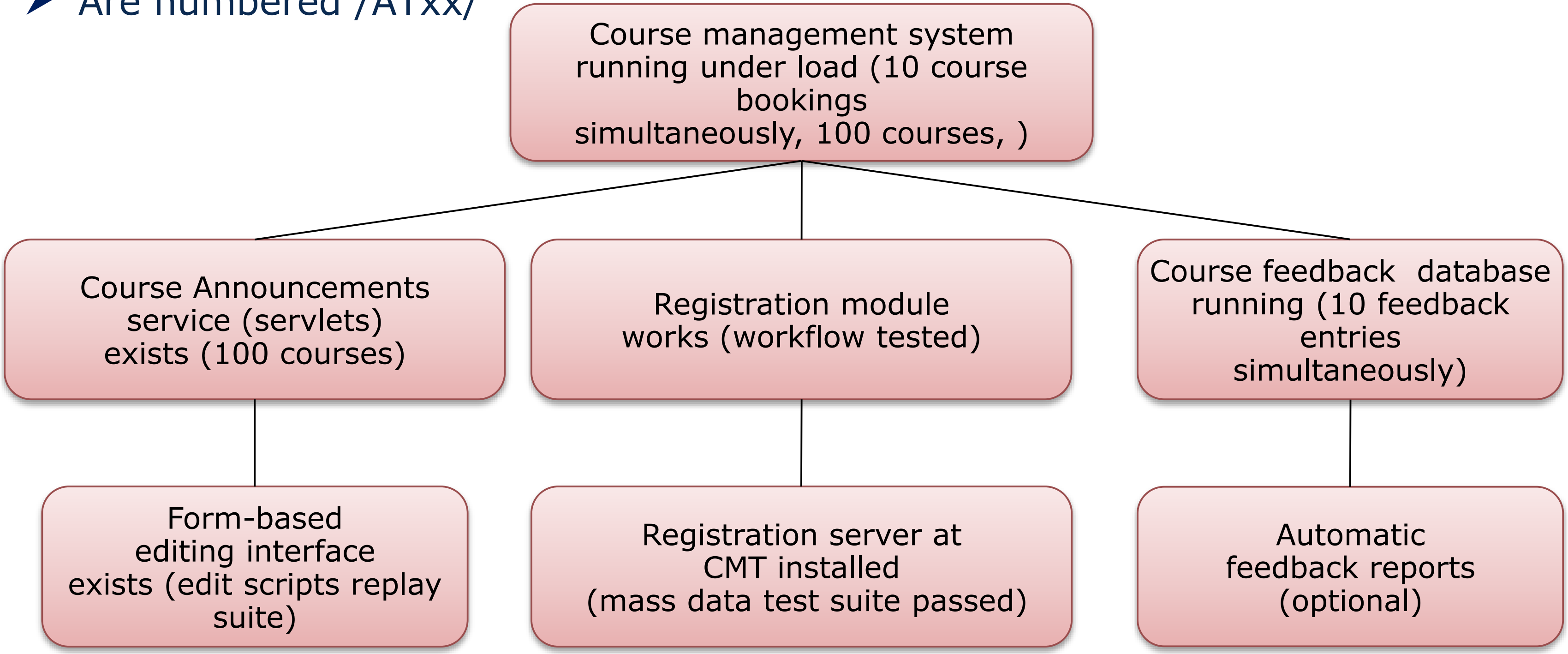
Course Management System: Success Criteria Analysis for User CMT

- Success criteria must be measurable: Boolean or Quantitative


boolean
quantitative
CMT earns 100% more money with courses
CMT has 100% more courses
Course management system running
Course Announcements service (servlets) exists
Registration module works
Course feedback database running
Form-based editing interface exists
Registration server at CMT installed
Automatic feedback reports (optional)
CMT can survive 5 years
CMT employees are occupied 80%
70% of customers are satisfied
50% of customers return to other courses

10.3.6 Course Management System: Acceptance Tests

- **Acceptance tests** are developed out of the success criteria, showing that they are fulfilled.
  - Not all success criteria are acceptance test criteria; success criteria *lead to* acceptance test criteria
- They are conducted after installation of the system at the client. Passing them means that the contract is fulfilled
  - Hence, they must be measurable: Boolean or Quantitative, with size of test data suites
  - Are numbered /ATxx/



Course management system running under load (10 course bookings simultaneously, 100 courses,)

Course Announcements service (servlets) exists (100 courses)

Registration module works (workflow tested)

Course feedback database running (10 feedback entries simultaneously)

Form-based editing interface exists (edit scripts replay suite)

Registration server at CMT installed (mass data test suite passed)

Automatic feedback reports (optional)

10.4 SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Software Requirements Specification (SRS)

- The result of the problem, goal, and requirements analysis is the software requirements specification (SRS, Anforderungs-spezifikation).
  - It replaces the feasibility study
  - It is part of the contract and binds legally
  - It describes desired features of the system
- Numbering of requirements, e.g, with
  - Functional /Fxx/
  - Quality /NFxx/, /Qxx/ with scales and thresholds
  - Semi-functional /Sfxx/
  - Success criteria /Sxx/
  - Acceptance test criteria /ATxx/
- The SRS should be
  - Correct, Complete, Consistent (CCC). This should be validated (validation or consistency checking)
  - Verifiable (measurable, clear success criteria)
  - Functional (what, not how)
  - Tractable in design and evolution
  - Simple to change and evolve

Contents of the SRS

- The Software Requirement Specification (SRS) contains a list of things the system has to fulfill.
  - Example [Richard Fairley, Software Engineering] The structure can be different, e.g., Balzert has some other components

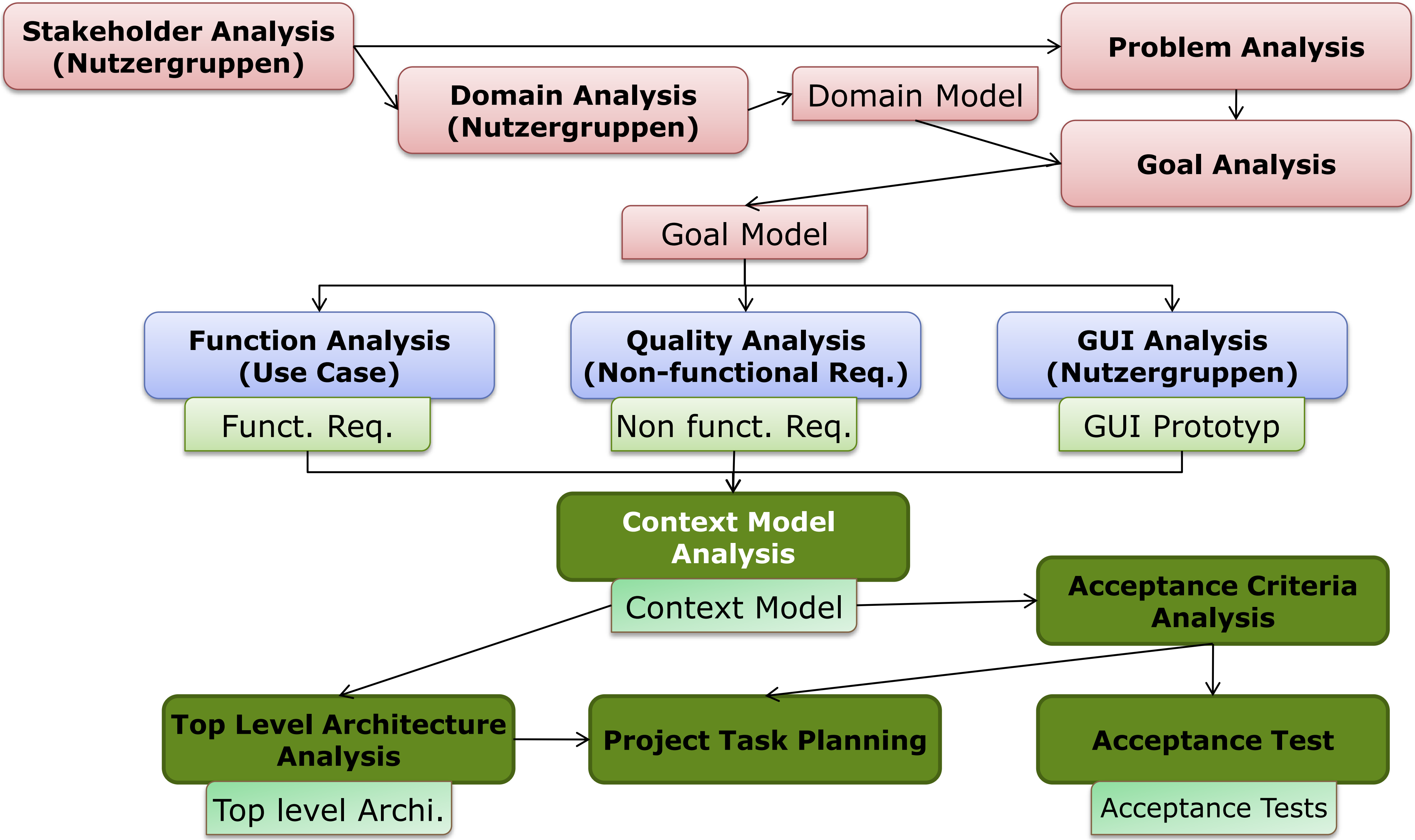
- Content of the SRS:
- Overview of Product
  - Background, environment, platforms
- Environment model
  - Stakeholders model
  - Domain model (at least glossary)
  - Problem model
  - Goal model
- Requirements
  - Functional requirements
    - Use cases, function trees, with priorities
  - Non-functional requirements
  - Semi-functional requirements
  - Error handling

- Application model (Fachliches Modell)
  - Context model (interfaces of the system)
    - I/O interfaces, data formats (screens, protocols, etc.), commands
  - Data dictionary
  - Top-level Architecture
    - Overview of data flow through systemPossible extensions of the system
- Success criteria
  - Acceptance test criteria
  - Acceptance test cases
  - Other success criteria
- Documentation guidelines
- Literature

Beware

- Requirement errors cost the most!
  - Because all actions depend on them, and must be undone if requirement is wrong
- Software engineers often mismodel domains, since they are no domain experts
- Customers will only pay if all acceptance tests are passed!

Analysis in Detail



Stakeholder Analysis (Nutzergruppen)

Domain Analysis (Nutzergruppen)

Domain Model

Problem Analysis

Goal Analysis

Goal Model

Function Analysis (Use Case)

Funct. Req.

Quality Analysis (Non-functional Req.)

Non funct. Req.

GUI Analysis (Nutzergruppen)

GUI Prototyp

Context Model Analysis

Context Model

Acceptance Criteria Analysis

Top Level Architecture Analysis

Top level Archi.

Project Task Planning

Acceptance Test

Acceptance Tests

10.5 REQUIREMENTS MANAGEMENT

Requirements Must Be Managed

- Once we got the specifications, they will change
  - Customers discover new things
  - More details are discovered
- The SRS should stay “fixed” once and forever, but in collaboration with the customer be maintained and evolved.
- How to manage the requirements in the SRS?

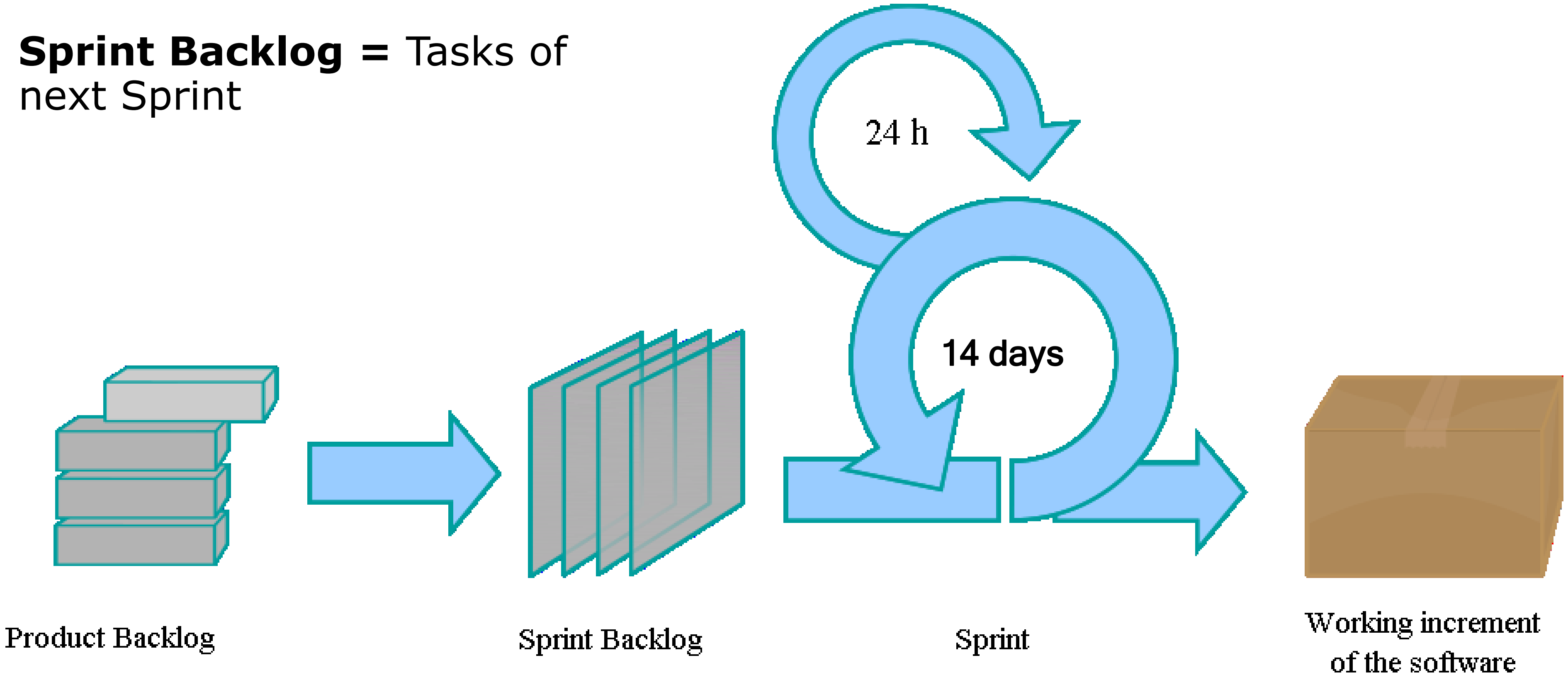
Answer of Extreme Programming

- Manage requirements as “story” cards
  - Muddy cards, with one requirement of the customer in plain text
  - Sort them along priorities
  - Realize one requirement at a time, then pick up the next one
  - Story cards are kept until the end of the project
- Evaluation
  - Works only for small teams and project sizes
  - Does not work if a product runs over years, and the developers change

Scrum Manages Requirements with Backlogs

Product Backlog = Features of product to be developed

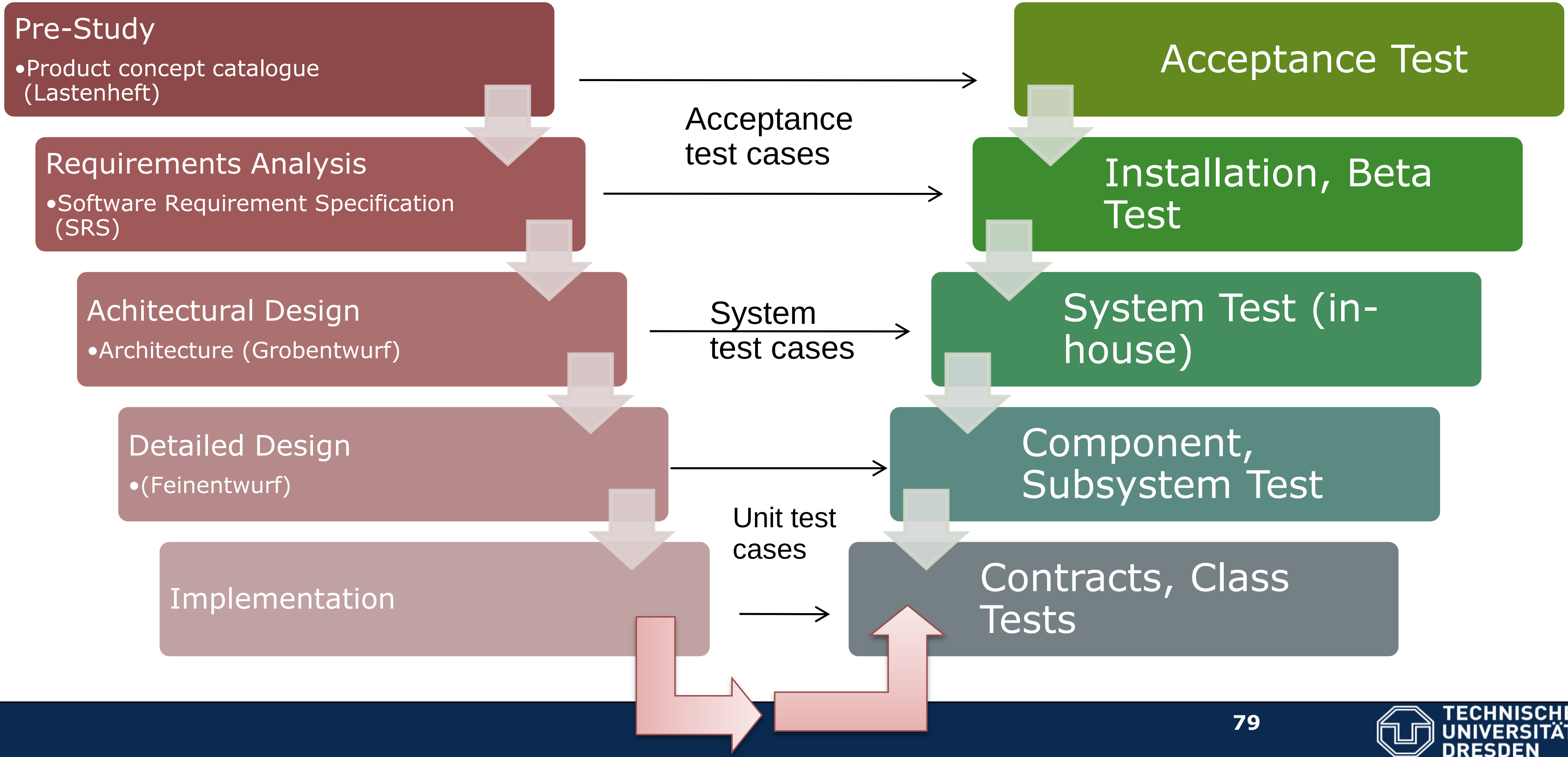
Sprint Backlog = Tasks of next Sprint



[Lakeworks]

Answer in the V-Model

- Update the SRS every time another document is produced along the V, e.g., at the architectural design
- Publish it on a intranet web site, visible for all developers and the customer
- Maintain a cross-relationship between requirement lists and trees and passed acceptance tests (to show the progress of the project)



Professional Answer: Requirement management system (RMS)

- Maintains requirements in a database (repository) company-wide
- Customers can see the requirement database and add new entries
- Plan when the new requirements will be realized: build road maps
  - In which version (major, minor)

What Have We Learned?

- Problem analysis is not goal analysis is not requirements analysis
- Stakeholders should be investigated separately
- Hierarchical grouping of problems, goals, requirements, and success criteria is useful
  - Helps to sort the problems and to make stakeholder's goals clear
- Requirements stem from goals stem from problems
- Success criteria must be defined to show when the project was completed
- Requirement errors are very costly
- Requirements must be managed professionally

The End