DAAD
Deutscher Akademischer Austausch Dienst
German Academic Exchange Service

STABILITY PACT
FOR SOUTH EASTERN EUROPE

On the Structure of the Joint Course on Software Engineering

Klaus Bothe

3rd Workshop Software Engineering Education and Reverse Engineering, Ohrid, Macedonia, 2003

Institute of Informatics, Humboldt University – Berlin, Germany,
bothe@informatik.hu-berlin.de

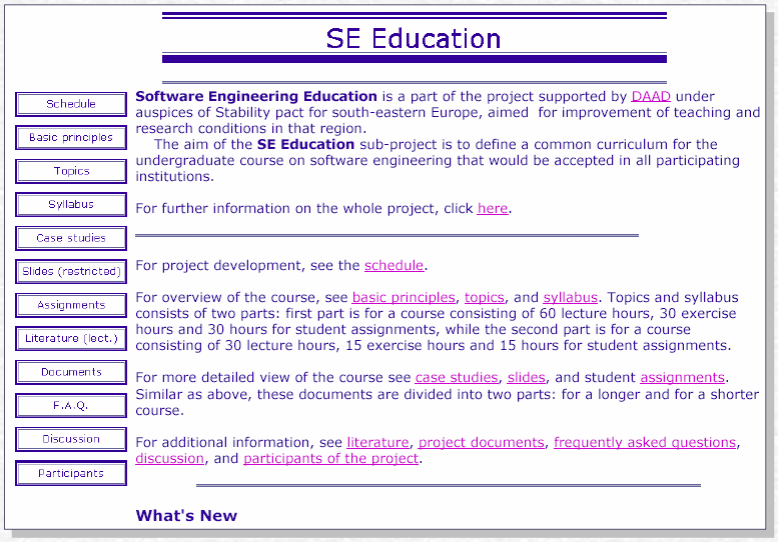
The purpose of this presentation

- To provide an overview of the course for lecturers to approach the course more easily
- Basis of discussion on contents aspects
- To be entered as part of the course materials at the project website

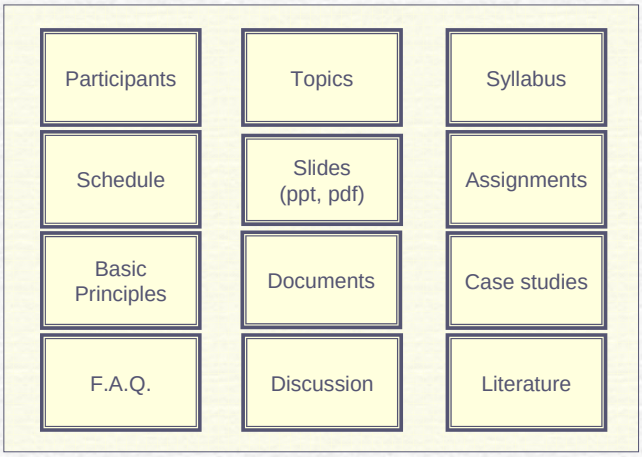
Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

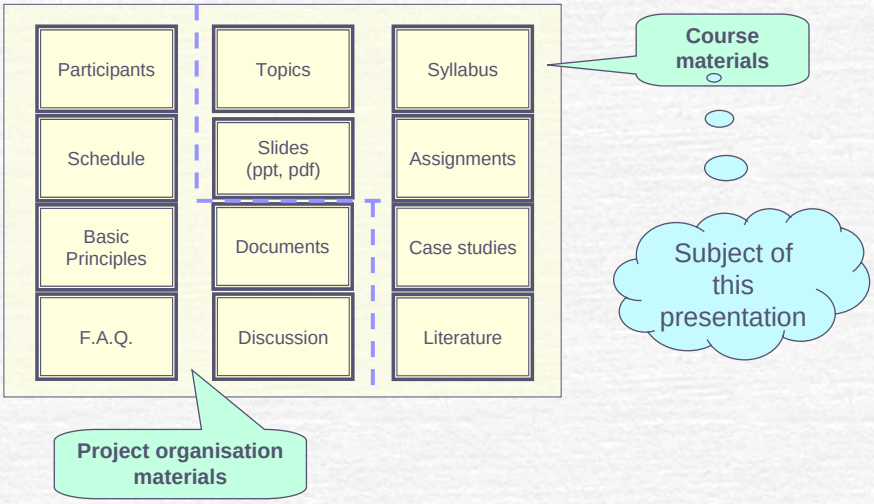
The joint course website



Overview of the project materials



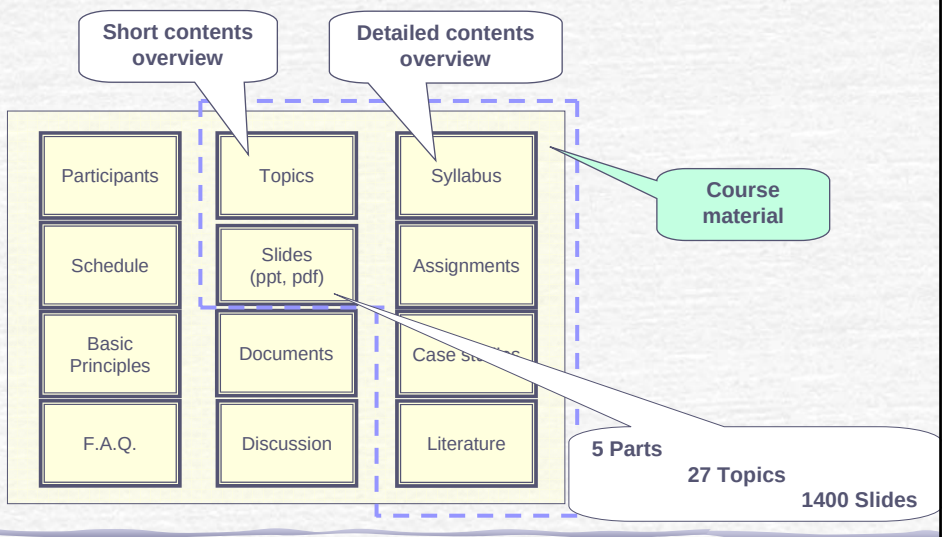
Overview of the project materials: project organisation and course



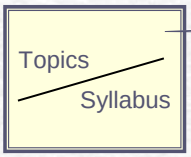
Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

The structure of the course contents



Course contents (1)



5 Parts, 27 Topics, 1400 Slides

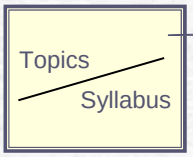
Part I: Introduction to software engineering

- 1. What is software engineering
- 2. Quality criteria for software products
- 3. Software process models
- 4. Basic concepts for software development documents

Part II: Requirements engineering

- 5. Results of the "analysis and definition" phase
- 6. Cost estimation
- 7. Function-oriented view
- 8. Data-oriented view
- 9. Rule-oriented view
- 10. Structured analysis
- 11. State-oriented view
- 12. Scenario-oriented view
- 13. Object-oriented analysis
- 14. Formal software specification and program verification

Course contents (2)



5 Parts, 27 Topics, 1400 Slides

Part III: Software Design

- 15. Overview of design activities
- 16. Structured design
- 17. Object-oriented design

Part IV: Implementation and testing

- 18. Implementation
- 19. Systematic testing
- 20. Functional testing

Part V: Advanced problems

- 21. Software metrics
- 22. Maintenance
- 23. Reverse engineering
- 24. Quality of software development process and its standardisation
- 25. Introduction to software ergonomics
- 26. User manuals
- 27. Project management
- 28. Configuration and version management

The structure of the course: four fundamental influencing factors

A The waterfall model: Fundamental phases of software projects — Topic 3

B Basic concepts: description techniques of software documents — Topic 4

C Fundamental paradigms of SW development:
- imperative (structured)
- object-oriented

Topic 10

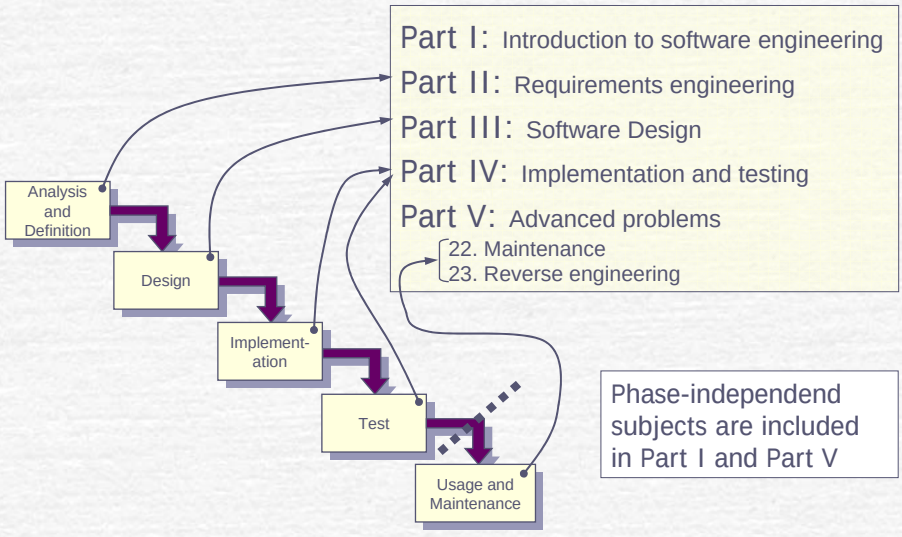
Part I
Part II
Part III
Part IV
Part V

D Case studies:
- Seminar organisation
- XCTL

Topic 5

The waterfall model and the course structure

A



Part I: Introduction to software engineering
Part II: Requirements engineering
Part III: Software Design
Part IV: Implementation and testing
Part V: Advanced problems
- 22. Maintenance
- 23. Reverse engineering

Phase-independend subjects are included in Part I and Part V

In some cases the order of the topics depends on didactical principles

Part II: Requirements engineering

- 5. Results of the "analysis and definition" phase
- 6. Cost estimation
- 7. Function-oriented view
- 8. Data-oriented view
- 9. Rule-oriented view
- 10. Structured analysis
- 11. State-oriented view
- 12. Scenario-oriented view
- 13. Object-oriented analysis
- 14. Formal software specification and program verification

Part V: Advanced problems

- 21. Software metrics
- 22. Maintenance
- 23. Reverse engineering
- 24. Quality of software development process and its standardisation
- 25. Introduction to software ergonomics
- 26. User manuals
- 27. Project management
- 28. Configuration and version management

Problems:
- otherwise Part II is too long and, thus,
- too late for assignments concerning Parts III, IV, V

Basic concepts and the course structure

B

Concepts and Views

Alternative Notations: Often used — Rarely used

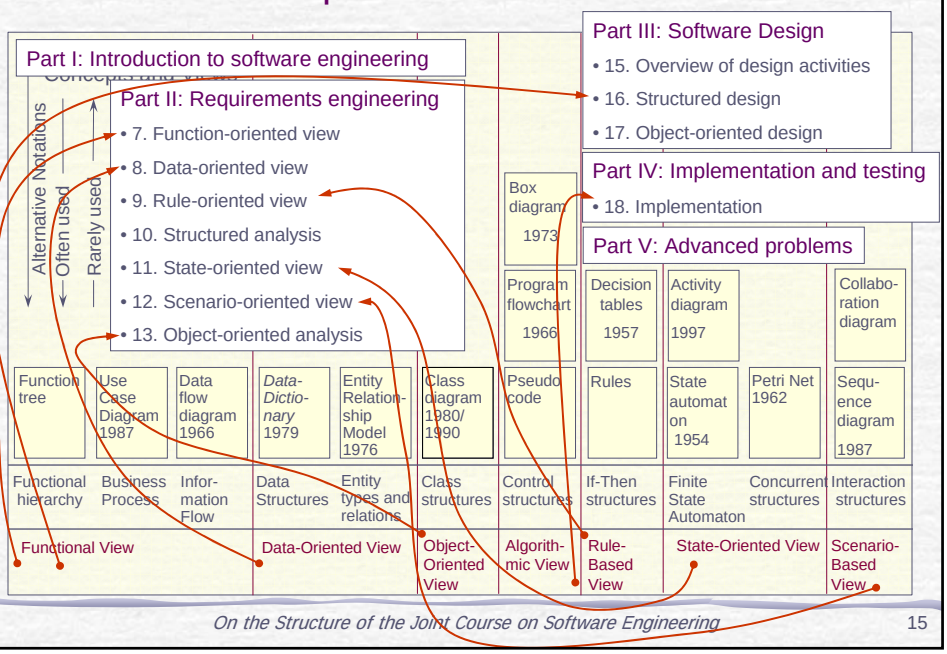
Topic 3 presents an overview of the basic concepts.

These basic concepts will be handled in more detail in other topics.

Basic concepts of ... software development
Balzert vol. 1, 2nd edition 2001

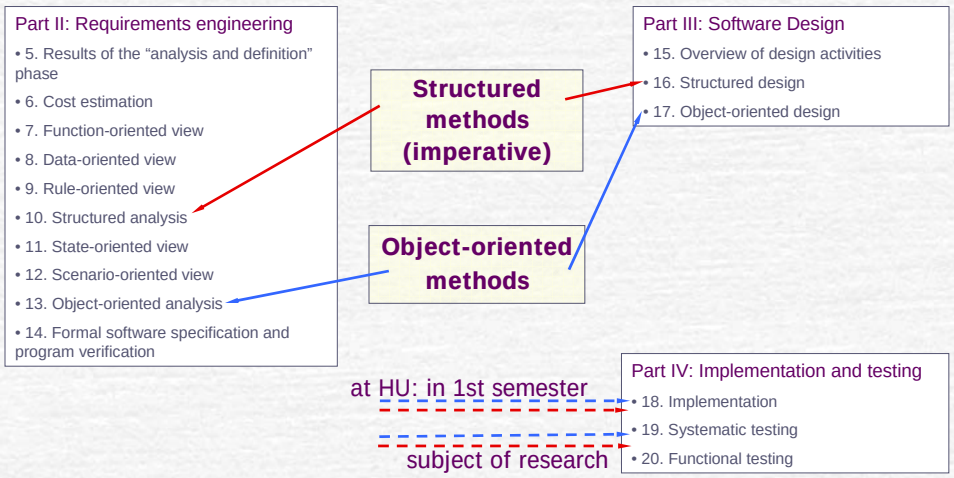
						Box diagram 1973				
						Program flowchart 1966	Decision tables 1957	Activity diagram 1997		Collaboration diagram
Function tree	Use Case Diagram 1987	Data flow diagram 1966	*Data-Dictionary* 1979	Entity Relationship Model 1976	Class diagram 1980/ 1990	Pseudo code	Rules	State automaton 1954	Petri Net 1962	Sequence diagram 1987
Functional hierarchy	Business Process	Information Flow	Data Structures	Entity types and relations	Class structures	Control structures	If-Then structures	Finite State Automaton	Concurrent structures	Interaction structures
Functional View			Data-Oriented View		Object-Oriented View	Algorithmic View	Rule-Based View	State-Oriented View		Scenario-Based View

Basic concepts and the course structure



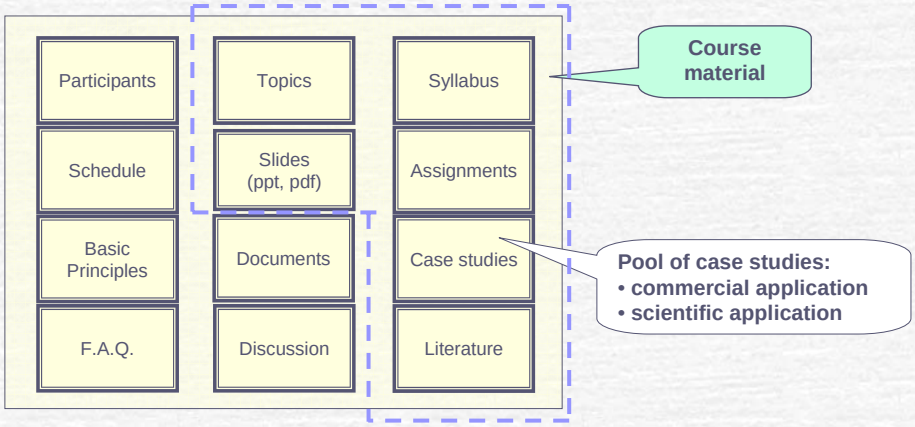
Software development paradigms and the course structure

C



Case studies and the course structure

D



Our two case studies

Seminar organisation

- Origin: textbook (Balzert)
- Commercial case study
- Real-world example adapted to a textbook
- Illustrates all required documents throughout the course

XCTL

- Origin: real customer
- Technical case study (in experimental physics)
- Real-world example
- Demonstrates treatment of unknown software

Case study ‘Seminar organisation’

Rule 1: Before starting with the lecture, try to understand the requirement specification of the fundamental case study ‘Seminar organisation‘

Topic 5:
Introduction to the case study

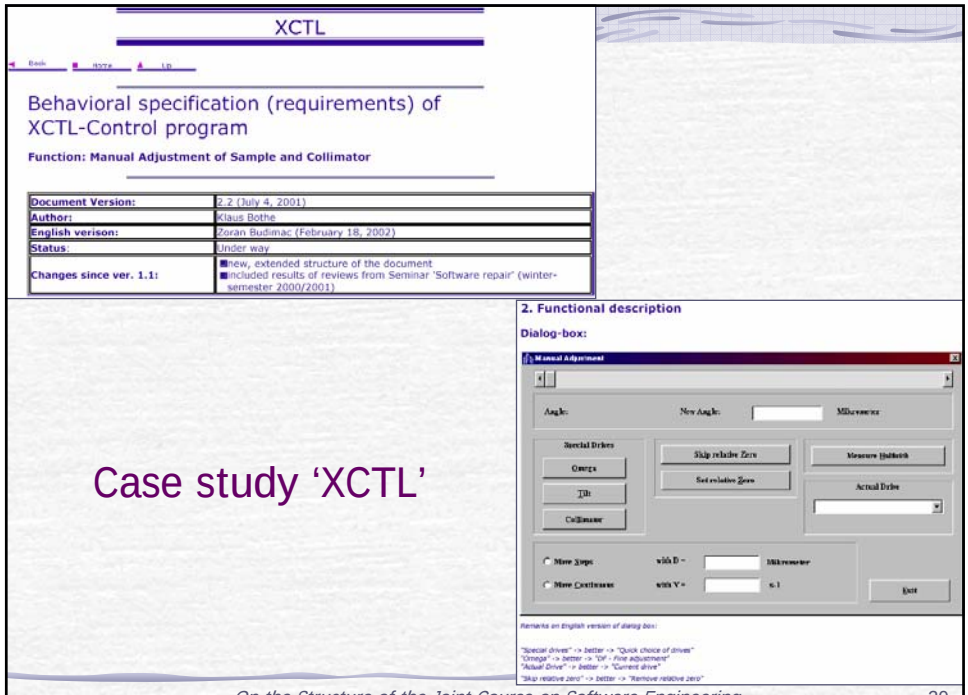
Seminar organisation

version 3.0

Version	Author	Date	State	Comment
2.1	Balzert			
2.2	Balzert			
2.3	Balzert			
3.0	Balzert			

Complete specification files:
Preliminary Requirements Specification v 3.0
Preliminary Requirements Specification v 2.3
Requirements Specification v 3.0
Requirements Specification v 2.3

Case study ‘XCTL’



Case studies Seminar organisation and XCTL: usage in course topics

Part I: Introduction to software engineering
- 1. What is software engineering
- 2. Quality criteria for software products
- 3. Software process models
- 4. Basic concepts for software development documents

Part II: Requirements engineering
- 5. Results of the "analysis and definition" phase
- 6. Cost estimation
- 7. Function-oriented view
- 8. Data-oriented view
- 9. Rule-oriented view
- 10. Structured analysis
- 11. State-oriented view
- 12. Scenario-oriented view
- 13. Object-oriented analysis
- 14. Formal software specification and program verification

Part III: Software Design
- 15. Overview of design activities
- 16. Structured design
- 17. Object-oriented design

Part IV: Implementation and testing
- 18. Implementation
- 19. Systematic testing
- 20. Functional testing

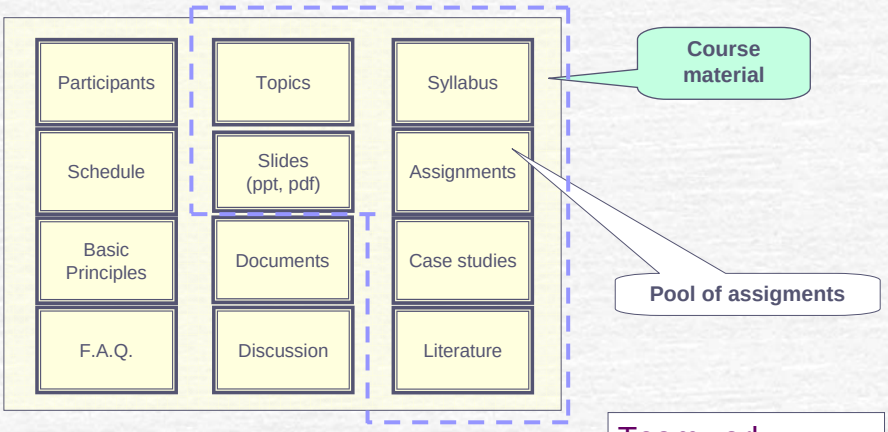
Part V: Advanced problems
- 21. Software metrics
- 22. Maintenance
- 23. Reverse engineering
- 24. Quality of software development process and its standardisation
- 25. Introduction to software ergonomics
- 26. User manuals
- 27. Project management
- 28. Configuration and version management

Seminar organisation

Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

Assignments



Teamwork:
groups of 3 - 4 students

Assignments and course topics

Part I: Introduction to software engineering
- 1. What is software engineering
- 2. Quality criteria for software products
- 3. Software process models
- 4. Basic concepts for software development documents

Part II: Requirements engineering
- 5. Results of the "analysis and definition" phase
- 6. Cost estimation
- 7. Function-oriented view
- 8. Data-oriented view
- 9. Rule-oriented view
- 10. Structured analysis
- 11. State-oriented view
- 12. Scenario-oriented view
- 13. Object-oriented analysis
- 14. Formal software specification and program verification

Kay Schützler, Zoran Budimac:

Assignments in the SE course

Part III: Software Design
- 15. Overview of design activities
- 16. Structured design
- 17. Object-oriented design

Part IV: Implementation and testing
- 18. Implementation
- 19. Systematic testing
- 20. Functional testing

Part V: Advanced problems
- 21. Software metrics
- 22. Maintenance
- 23. Reverse engineering
- 24. Quality of software development process and its standardisation
- 25. Introduction to software ergonomics
- 26. User manuals
- 27. Project management
- 28. Configuration and version management

Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

Assignments and examinations

Examination:
- single
- oral
- inclusion of assigments
- free questions
- use of slide material of the lecture

Practice at Humboldt University Berlin

30 minutes

Oral Examination:

assignment i:
explain the solution

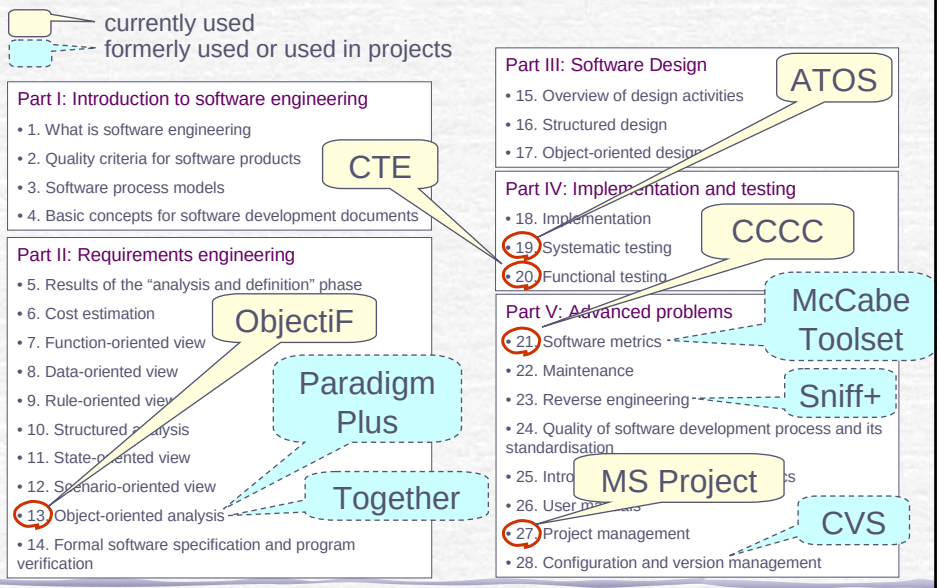
other topic
- free questions
- based on lecture slides

Published examination questions:
- 117 single questions

Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

Tools in the course and in assignments



Contents

1. Overview of the joined course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

Slides have to support the presentation

Rule 2:

Students expect slides also as a textbook. However, slides are designed to support the presentation during the lecture.

Thus, they can not be read as a textbook ...

Rule 3:

Slides are not a handbook of some notation, language, standard ...

Thus, the course will not explain each detail. It rather will explain the idea, the (most) important connections ...

Example: UML reference manual: 700 pages ?

Not each slide should be shown or discussed in detail during the lecture

Some of the slides have the purpose of

- additional information to the students, e.g. checklists
- explanations of other slides

Main purpose of this SE course: knowledge dissemination on SE principles

NOT:

- Practical SE project
- If possible, offer such a project as an extra module (HU: Reverse engineering with XCTL)

Main goals:

- Overview of SE: notions, methods, tools
- Relations between subareas
- Illustrating case studies
- Current research areas

Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

Handouts

Rule 4: Give handouts during the lectures in case of
- complex important contents
- that will be often reused

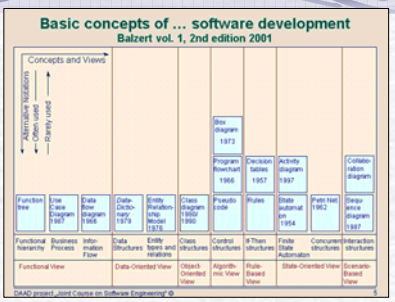
Part I: Introduction to software engineering
- 1. What is software engineering
- 2. Quality criteria for software products
- 3. Software process models
- 4 Basic concepts for software development documents

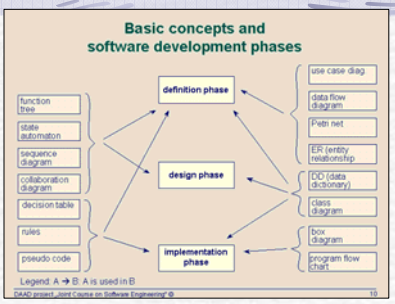
Part II: Requirements engineering
- 5. Results of the "analysis and definition" phase
- 6 Cost estimation
- 7. Function-oriented view
- 8. Data-oriented view
- 9. Rule-oriented view
- 10 Structured analysis
- 11 State-oriented view
- 12. Scenario-oriented view
- 13 Object-oriented analysis
- 14 Formal software specification and program verification

Part III: Software Design
- 15. Overview of design activities
- 16. Structured design
- 17. Object-oriented design

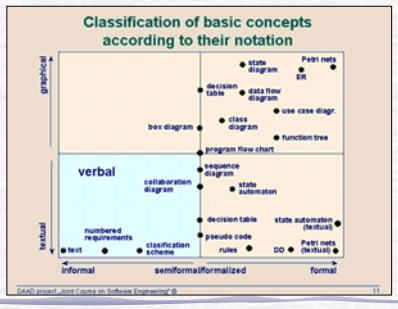
Part IV: Implementation and testing
- 18. Implementation
- 19 Systematic testing
- 20. Functional testing

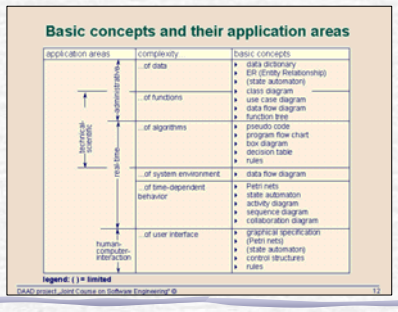
Part V: Advanced problems
- 21 Software metrics
- 22. Maintenance
- 23 Reverse engineering
- 24. Quality of software development process and its standardisation
- 25. Introduction to software ergonomics
- 26. User manuals
- 27. Project management
- 28. Configuration and version management





Handouts: topic 4





Handout: topic 6

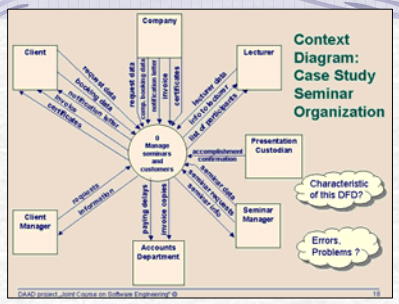
Category	Number	Classification	Weighting	Sums
Input data		simple	x 3	=
		middle	x 4	=
		complex	x 6	=
Queries		simple	x 3	=
		middle	x 4	=
		complex	x 6	=
Output data		simple	x 4	=
		middle	x 5	=
		complex	x 7	=
Data		simple	x 7	=
		middle	x 10	=
		complex	x 15	=
Reference data		simple	x 5	=
		middle	x 7	=
		complex	x 10	=
Sum			E1	=
Influencing factors (Function Point value can be changed by +/- 30 %)		1 Integration with other applications (0-5)		=
		2 Decentralized data/ processing (0-5)		=
		3 Transaction rate (0-5)		=
		4 Processing logic a Arithmetic operations (0-10) b Control procedures (0-5) c Exception handling (0-10) d Logic (0-5)		= = = =
		5 Reusability (0-5)		=
		6 Data stock conversions (0-5)		=
		7 Adaptability (0-5)		=
Sum of the 7 influences		E2		=
Evaluation of infl. factors = E2 / 100 + 0,7		E3 =		=
Weighted Function Points: E1 * E3				=

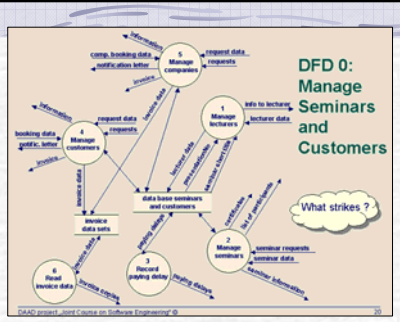
Quelle: IBM 85, S. 12

Calculation form IBM 1985

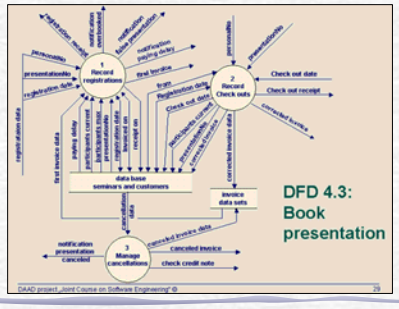
Source: Balzert, Vol. 1, p. 84

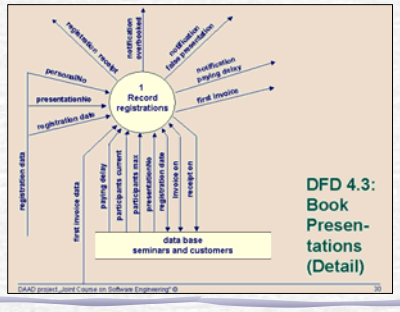
DAAD project „Joint Course on Software Engineering" © 29

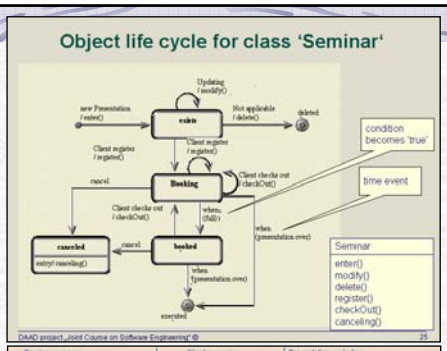




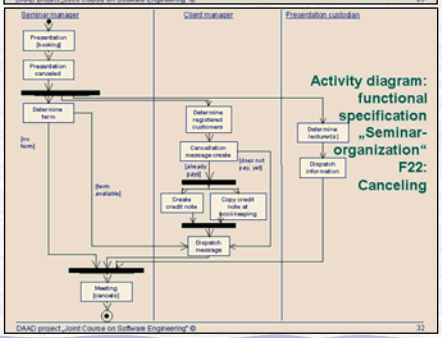
Handouts: topic 10

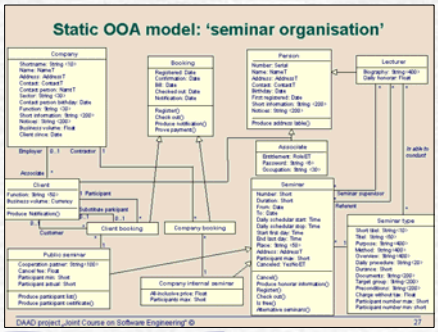






Handouts: topic 11 and 13





Class Stack: algebraic specification

Types
Integer, Boolean, Stack

Signature
new: → Stack
push: Integer x Stack → Stack
pop: Stack → Stack
top: Stack → Integer
is_empty: Stack → Boolean

Axioms (equations)
top(push(e,s)) = e
pop(push(e,s)) = s
is_empty(new) = true
is_empty(push(e,s)) = false

With erroneous situations:
top(new) = ERROR
pop(new) = new (oder: ERROR)

Correctness proof: quotient-rest example

Line	Formal Proof	Justification
1	**true** ⊃ x = x + y • 0	Lemma 1
2	x = x + y • 0 ⟨r := x⟩ x = r + y • 0	D0
3	x = r + y • 0 ⟨q := 0⟩ x = r + y • q	D0
4	**true** ⟨r := x⟩ x = r + y • 0	D1 (1, 2)
5	**true** ⟨r := x, q := 0⟩ x = r + y • q	D2 (4, 3)
6	x = r + y • q ∧ ¬ y ≤ r ⊃ x = (r − y) + y • (1 + q)	Lemma 2
7	x = (r − y) + y • (1 + q) ⟨r := r − y⟩ x = r + y • (1 + q)	D0
8	x = r + y • (1 + q) ⟨q := 1 + q⟩ x = r + y • q	D0
9	x = (r − y) + y • (1 + q) ⟨r := r − y, q := 1 + q⟩ x = r + y • q	D2 (7, 8)
10	x = r + y • q ∧ ¬ y ≤ r ⟨r := r − y, q := 1 + q⟩ x = r + y • q	D1 (6, 9)
11	x = r + y • q ⟨**while** y ≤ r **do** ⟨r := r − y, q := 1 + q⟩⟩ ¬ y ≤ r ∧ x = r + y • q	D3 (10)
12	**true** ⟨⟨r := x, q := 0⟩; **while** y ≤ r **do** ⟨r := r − y, q := 1 + q⟩⟩ ¬ y ≤ r ∧ x = r + y • q	D2 (5, 11)

source: Hoare, CACM, 1969

Handouts: topic 14

Example „petrol tank": data

Container
contents: N
capacity: N
contents ≤ capacity

Indicator
light: {off, on}
reading: N
danger_level: N
light = on ⇔ reading ≤ danger_level

Storage_tank
Container
Indicator
combination of 2 schema
reading = contents
capacity = 5000
danger_level = 50

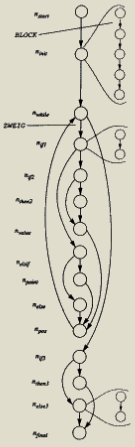
Example „petrol tank": operations

Fill-OK
Δ Storage_tank
amount?: N
contents + amount? ≤ capacity
contents' = contents + amount?

OverFill
Ξ Storage_tank
amount?: N
rf: seq CHAR
capacity < contents + amount?
rf = „Insufficient tank capacity – Fill cancelled"

Handouts: topic 19

Example of control-flow graph



```
Wert = 0.0;
Genauigkeit = 1.0;
WoBinIch = VorDemKomma;
Fehlerfrei = true;
Position = 0;

while ( (Position < inZiffernString.length()) && Fehlerfrei) {
        Zchn = inZiffernString.charAt(Position);
        if Ziffer(Zchn) {
           if (WoBinIch == NachDemKomma) {
              Genauigkeit = Genauigkeit / 10.0;
           }
           Wert = 10.0 * ZchnZuWert(Zchn);
        }
        else
        if ( (Zchn == '.') && (WoBinIch == VorDemKomma)) {
            WoBinIch = NachDemKomma;
        }
        else {
            Fehlerfrei = false;
        }

        Position++;
}

if ( !Fehlerfrei ||  (inZiffernString.length() == 0) ||
     (WoBinIch == NachDemKomma) && (inZiffernString.length() == 1) ) {
         return FehlerCode;
}
else {
    Wert = Wert * Genauigkeit;
    return Wert;
}
```

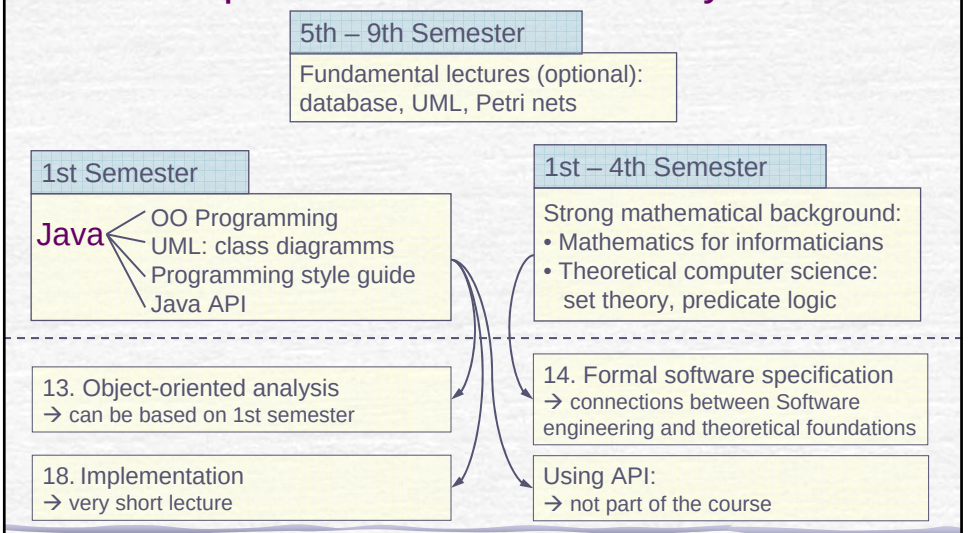
nach: Pagel, Six, S. 436

Prof. Klaus Bothe, HU-Berlin, HK Software Engineering 29

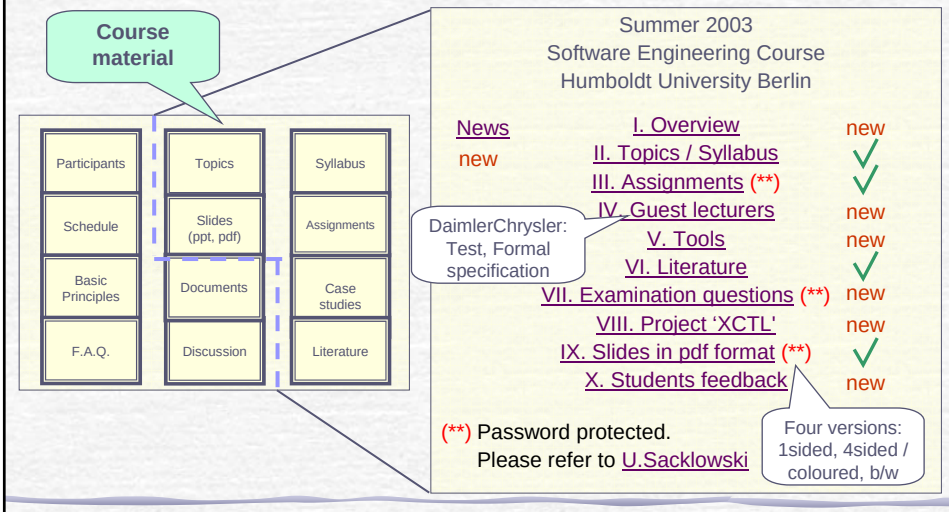
Contents

1. Overview of the joint course website
2. The structure of the course contents
3. Assignments and the course contents
4. Examinations
5. Tools
6. The purpose of the slide material
7. Handouts
8. Adaptation of the material to the needs of a particular university

University-dependence: example of Humboldt University Berlin



Different website structures: Joint course website – Local HU students website



Duration: How long will a lecture take?

in minutes	Z.B. (W 02)	K.B. (S 03)
Part I: Introduction		
• 1. What is software engineering	80	120
• 2. Quality criteria ...	40	45
• 3. Software process models	120	90
• 4. Basic concepts ...	60	40
Part II: Requirements engineering		
• 5. Results of the ... phase	(70)	100
• 6. Cost estimation	60	100
• 7. Function-oriented view	60	50
• 8. Data-oriented view	50	35
• 9. Rule-oriented view	50	40
• 10. Structured analysis	80	65
• 11. State-oriented view	(45)	80
• 12. Scenario-oriented view	30	25
• 13. Object-oriented analysis	(60)	210
• 14. Formal software specification ...	--	190

Part III: Software Design		
• 15. Overview of design activities	--	90
• 16. Structured design	--	15
• 17. Object-oriented design	--	45
Part IV: Implementation and testing		
• 18. Implementation	--	90
• 19. Systematic testing	--	180
• 20. Functional testing	--	150
Part V: Advanced problems		
• 21. Software metrics	--	180
• 22. Maintenance	--	-
• 23. Reverse engineering	--	90
• 24. Quality of software development ...	--	90
• 25. Software ergonomics	--	180
• 26. User manuals	-	-
• 27. Project management	?	90
• 28. Configuration ... management	-	45

() short version **2425 : 45 = 53** lecture hours practical **58** lh Sum: **2425**

Software engineering in special courses at other sites

Part I: Introduction to SE
Part II: Requirements engineering
11. Basic concepts of state-oriented view
12. Basic concepts of scenario-based view
13. Object-oriented analysis
14. Formal specification and program verification
Part III: Design
17. Object-oriented design
Part IV: Implementation and testing
19. Systematic testing
Part VI: Advanced problems
21. Software metrics
23. Reverse engineering
24. Quality of software development process and its standardization
24. Software ergonomics
25. Project management

- **Requirements Engineering** (SEI, Pittsburgh, USA)
- **Petri nets** (Reisig, HU)
- **Objectoriented SW development (with UML)** (Fischer, HU)
- **Formal methods of SW development** (SEI, Pittsburgh, USA)
- **Software architecture** (SEI)
- **Software test methods** (Spillner, Bremen)
- **CAME-Tools** (Dumke, Magdeburg)
- **Software reengineering** (Koschke, Stuttgart)
- **PSP** (SEI)
- **CMM** (SEI)
- **Software ergonomics** (Wandtke, Inst. of psychology, HU)
- **Project Management** (SEI, Pittsburgh, USA)

Parts of this presentation on the structure of the course could also be presented to the students to provide them with an overview.