Karlsruhe Institute of Technology

NOT Y3K COMPLIANT

Softwaretechnik II

Oliver Hummel, IPD

Topic 2
Software Development Processes

SOFTWARE DESIGN AND QUALITY GROUP
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION, FACULTY OF INFORMATICS

sdq.ipd.kit.edu

Further Teasers from Monday

1. Putting more people on a late project makes it _____
2. Errors are more frequent during _____ phases
3. Testing can only show the presence of errors, but not their _____
4. A system that is used will be _____
5. Only what is _____ can be changed without risk

➔ Implications?

Course Schedule

Date	Tentative Content
Mo. 21.10.	Today: Warm-Up
Di. 22.10.	Software Processes
Mo. 28.10.	cont.
Di. 29.10.	Agile Development
Mo. 04.11.	*Guest Lecture by Andrena Objects*
Di. 05.11.	Requirements Elicitation
Mo. 11.11.	cont. + Use Cases
Di. 12.11.	cont.
Mo. 18.11.	Requirements Analysis
Di. 19.11.	cont.
Mo. 25.11.	Software Architecture
Di. 26.11.	cont. + Component-Based Architectures
Mo. 02.12.	cont.
Di. 03.12.	Persistence Patterns

Overview on Today's Lecture

- Content
  - Introduction and Motivation
  - The Waterfall and what is wrong with it
  - Iterative and Incremental Development
  - A few words on the Spiral Model
  - Introduction to the Rational Unified Process
  - Conclusion

- Learning Goals
  - Understand the basic ideas behind well-known process models
  - Being able to recognize and distinguish different types of process models

Code and Fix Approach

- Is it helpful?

- Code and Fix usually leads to:
  - badly structured code
  - non-systematic improvements
  - no real team work, as tasks can not be planned
  - missing design and documentation
  - complications in maintenance
  - …

- Are structured processes the way out?

Software Development Processes

- A way out? → Software Development Processes
  - *Waterfall model*
  - *Prototype Model*
  - *V-Model*
  - *RUP*
  - *…*
  - *Agile Methods*

- Typical advantages of processes:
- _____
- _____
- _____

➔ however, this depends on concrete project and selected process

Process Models

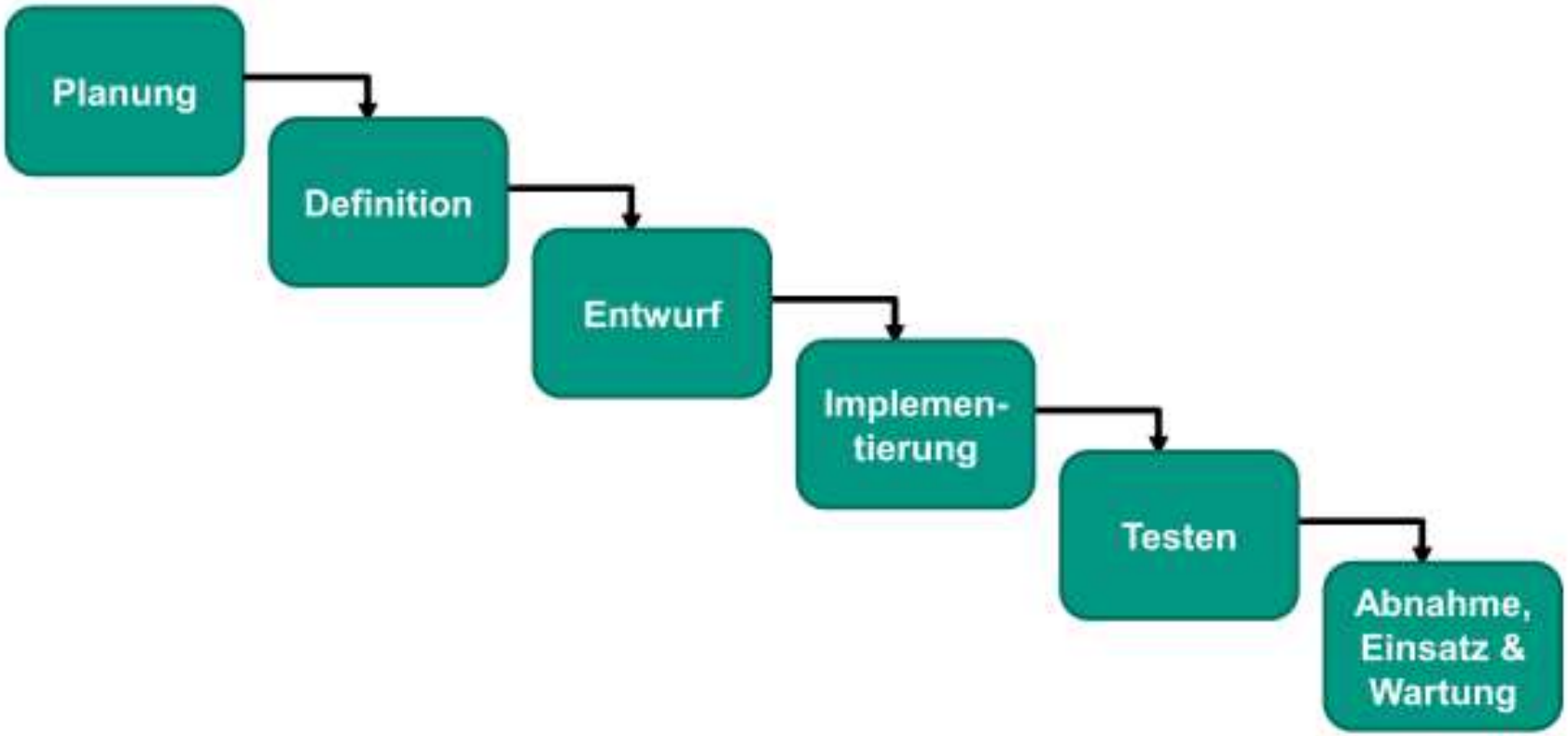
- A software process model ("Vorgehensmodell") is an **abstract representation of a software development process**
  - *it presents a description of a process from some particular perspective*

- I.e. it must recommends guidelines for –
  - which **activities** are to be carried out
    - how and in what order they are to be carried out
    - i.e. phases and milestones are defined
  - who has to carry out what
    - i.e. **roles** and responsibilities are determined
  - which **products** are to be built until when
    - i.e. artifacts, documents, and other work results
  - *and sometimes even which **techniques** and **tools** are to be used*

- *Models that merely define the order of phases and transition criteria between them are sometimes called **life cycle models***
  - *e.g. early Benington's phase model, the waterfall model, old V-Modell*

Repetition

- The Waterfall Model as a sequential process model …

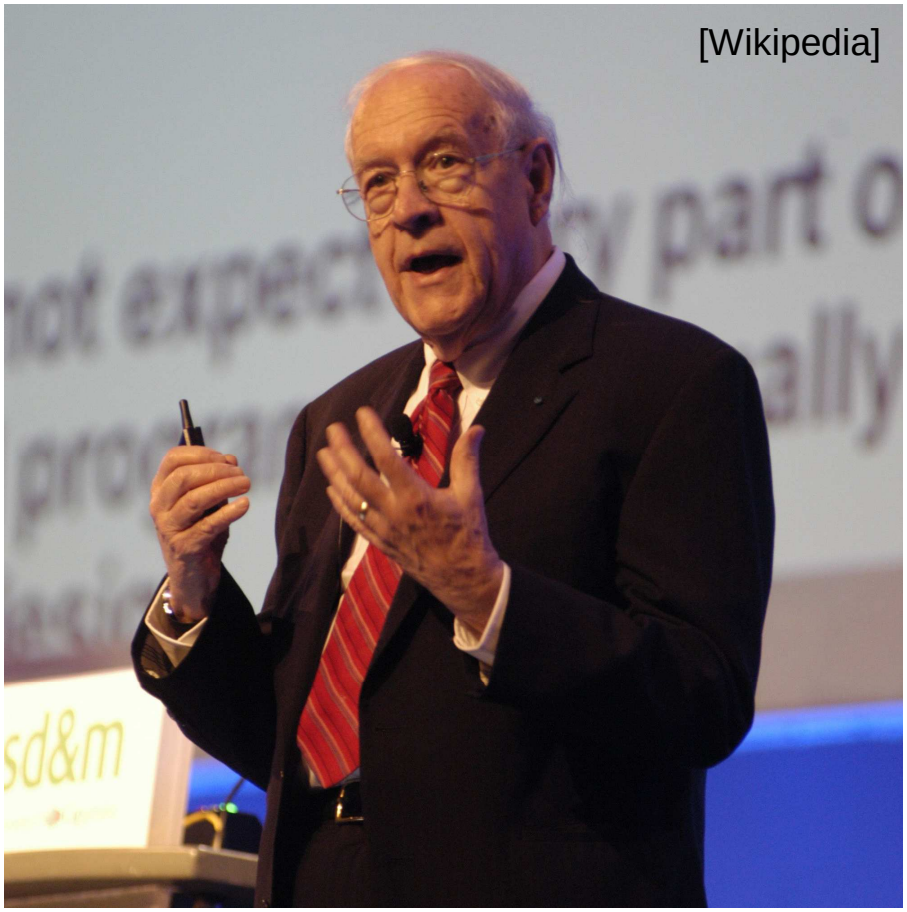


Who‘s that Guy?

“Much of present-day software acquisition procedure rests upon the **assumption that one can specify a satisfactory system in advance**, get bids for its construction, have it built, and install it.

I think this assumption is fundamentally wrong, and that many software acquisition problems spring from that fallacy.”
[No Silver Bullet]

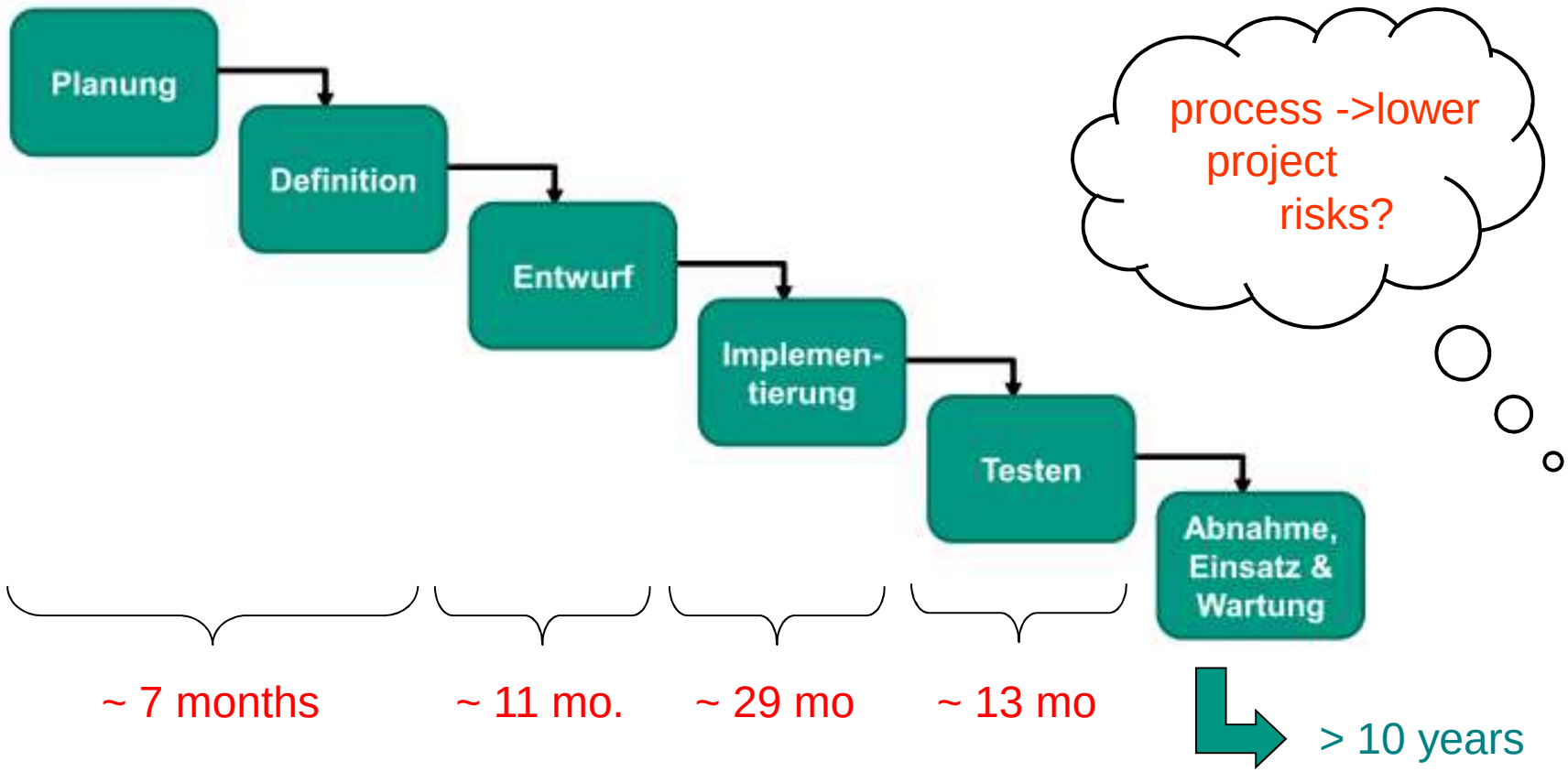
➔ *“The waterfall model is wrong!”*
[ICSE keynote 1995]



Turing Award Winner Frederick Brooks

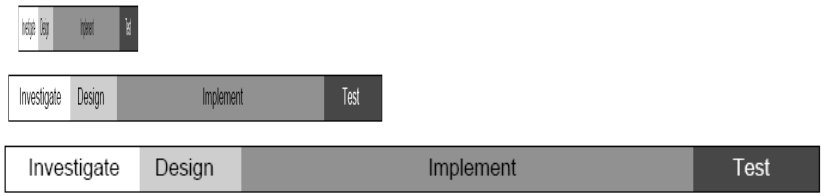
Remember: The Actual Challenge

- Let‘s investigate an example?
  - Airbus cabin software (~800 kLOC executable code)
  - ➔ ~ 4,500 person months and a schedule length of about 60 months



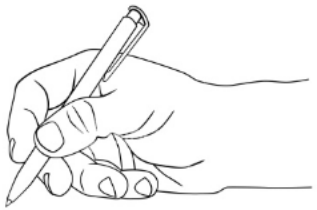
Waterfall Development Life Cycle

- How long can you plan ahead?
  - 5 days?
  - 5 weeks?
  - 5 months?
  - 5 years?

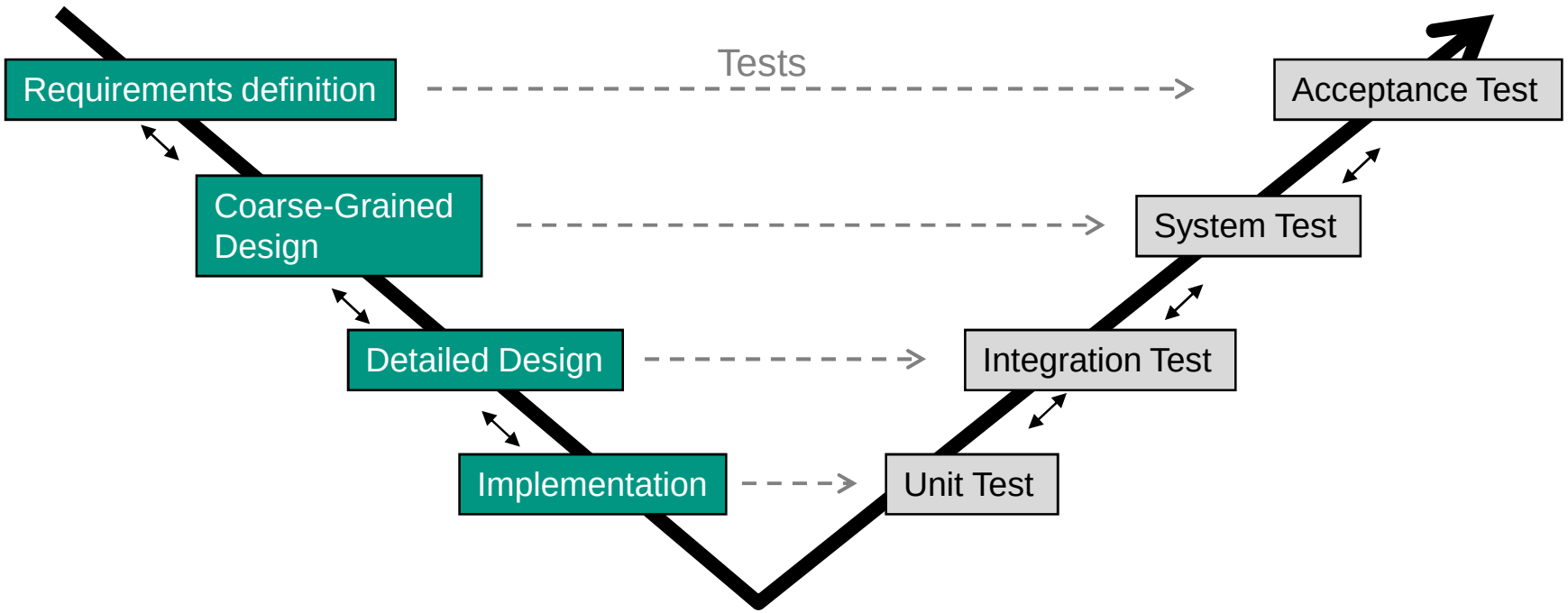


➔ What may happen in these timeframes?

- 5 days: _____
- 5 weeks: _____
- 5 months: _____
- 5 years: _____



V Model



➔ Can be seen as (just) an explanation how to relate testing to basic development phases

Self-Assessment (1)

1) *The six basic phases in every software development project are*

➔ _____, _____, _____

_____, _____, _____

2) *A comprehensive software development process should define –*

➔ _____, _____, _____

3) *Name the central problems of the Waterfall Model*

➔ _____

_____

Alternative Life Cycle?

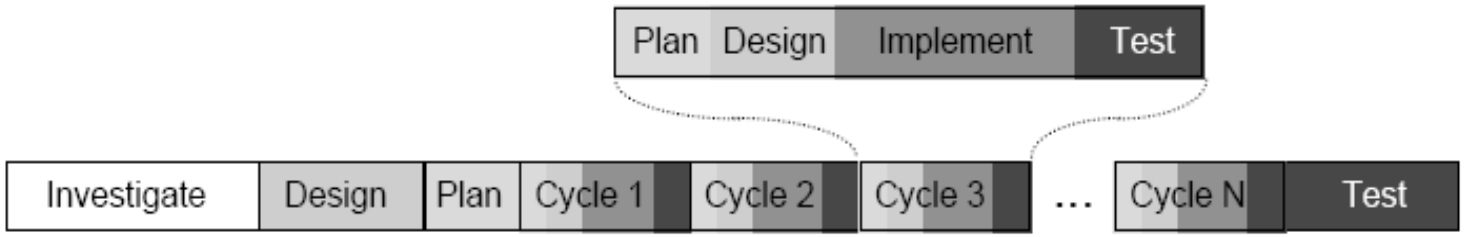
Sequential

e.g. 5 years

Waterfall Development Life Cycle

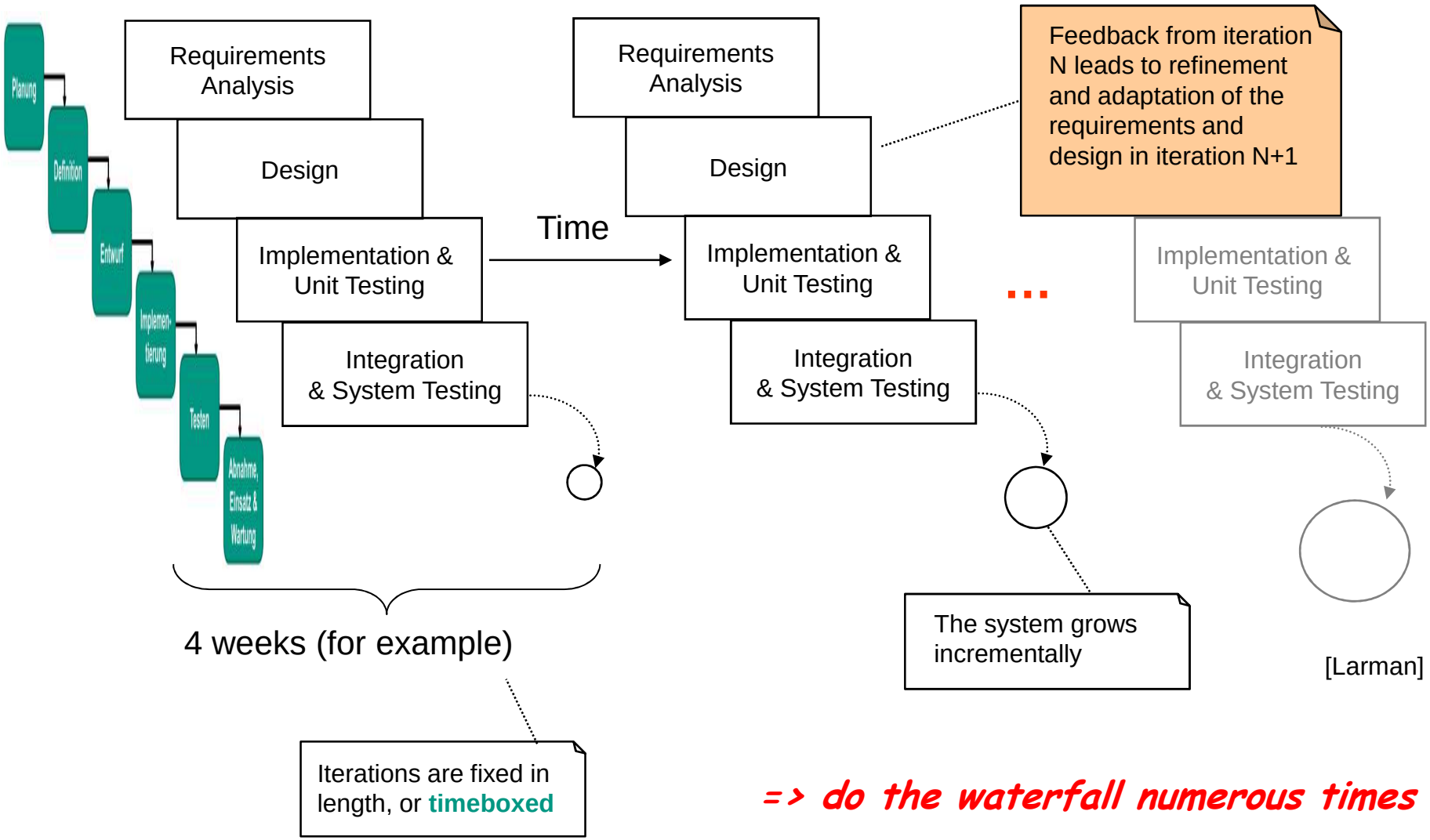
Iterative

usually 2-4 weeks



Incremental Development Life Cycle

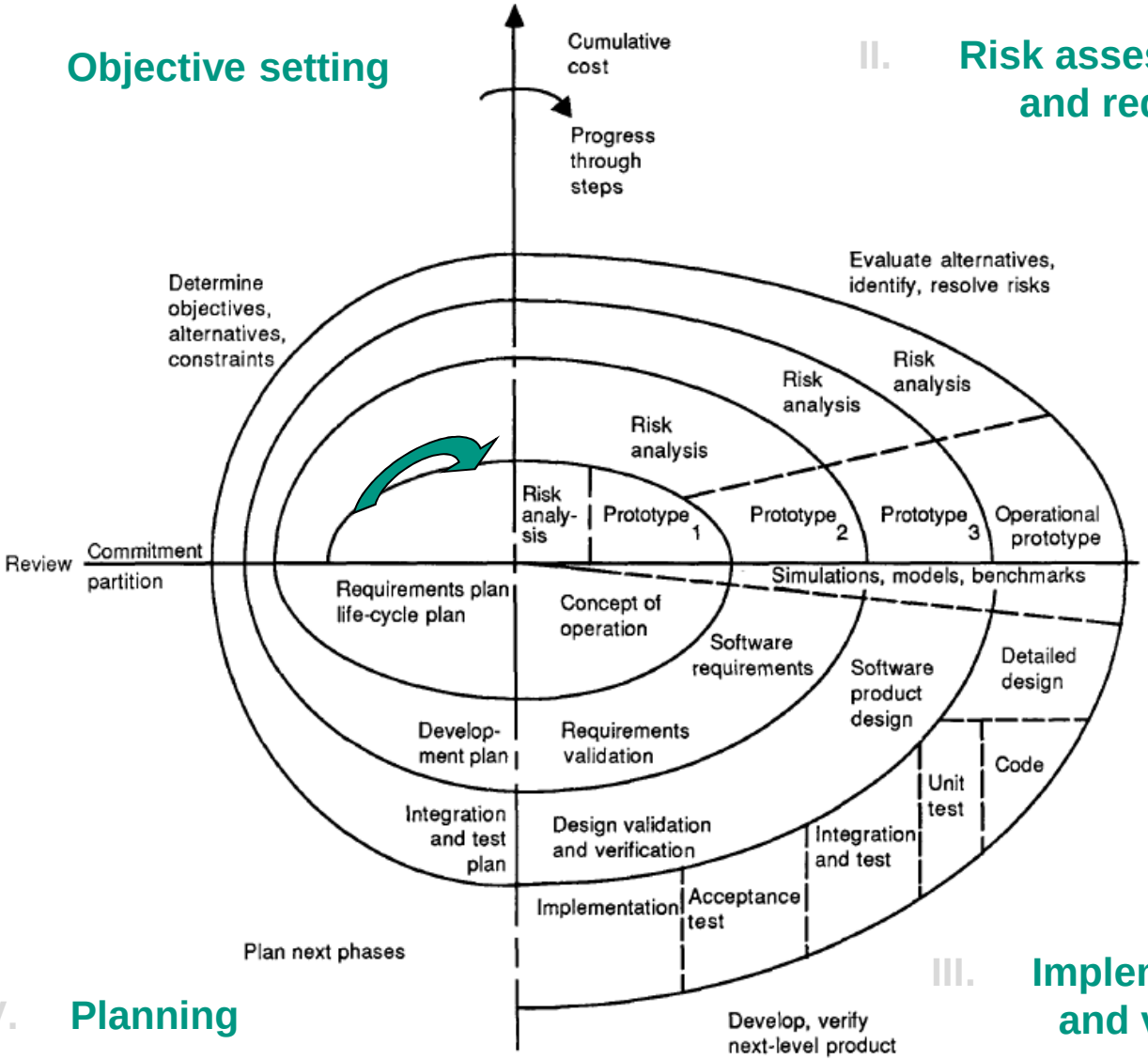
Iterative and Incremental Development



=> do the waterfall numerous times

NOT to be confused with the Spiral Model

[Boehm88]

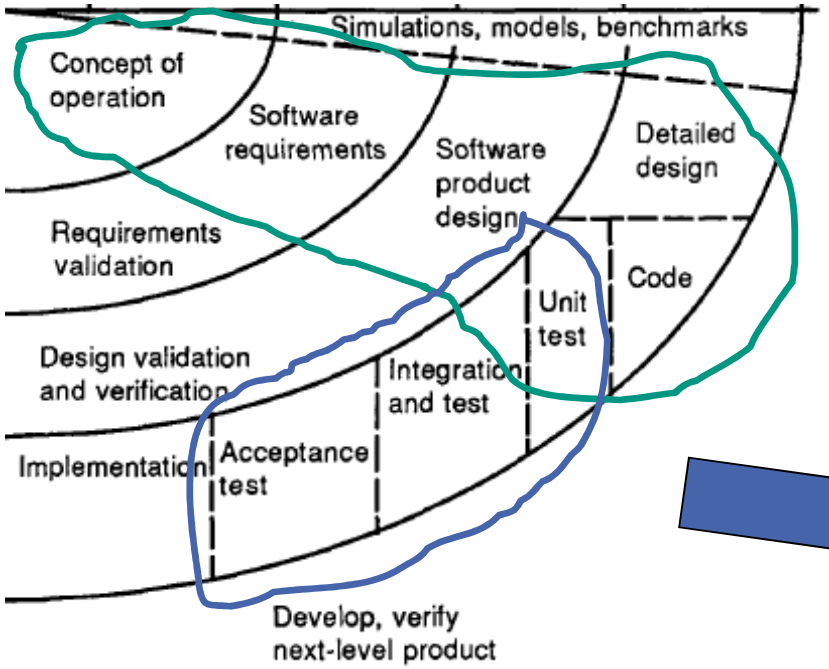


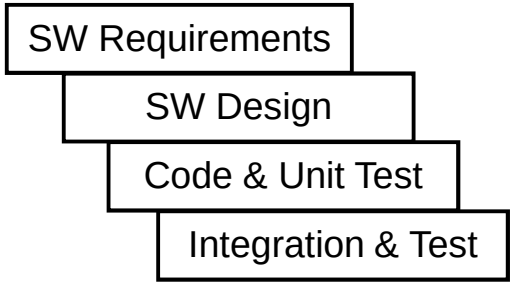
Boehm's Hypothesis:

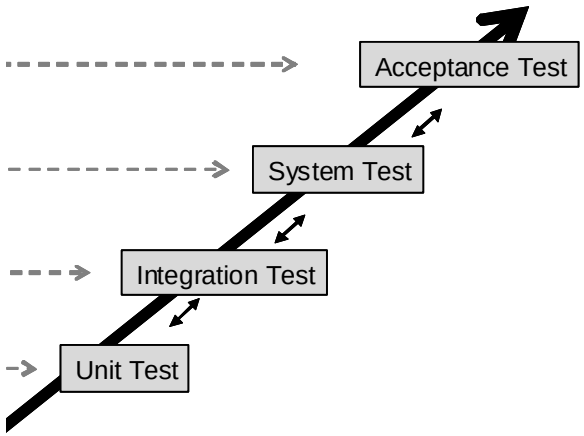
Project risks can be resolved or mitigated by addressing them early.

Spiral Model Revisited

- *Is it genuinely iterative?*
  - *i.e. does it go through the waterfall multiple times?*







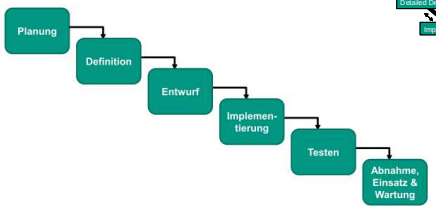
III. Implementation and validation

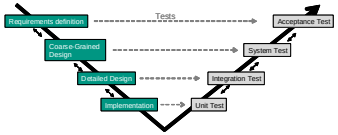
➔ the original spiral model basically incorporates a waterfall and a V model, **but is not a genuine I&I model**

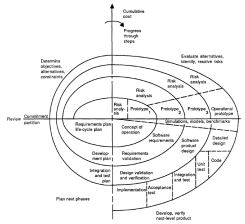
Elementary Models as Building Blocks

- ➔ From today's point of view, the
  1. waterfall model
  2. the original V model
  3. and the spiral model

  can be seen as building blocks for software development processes



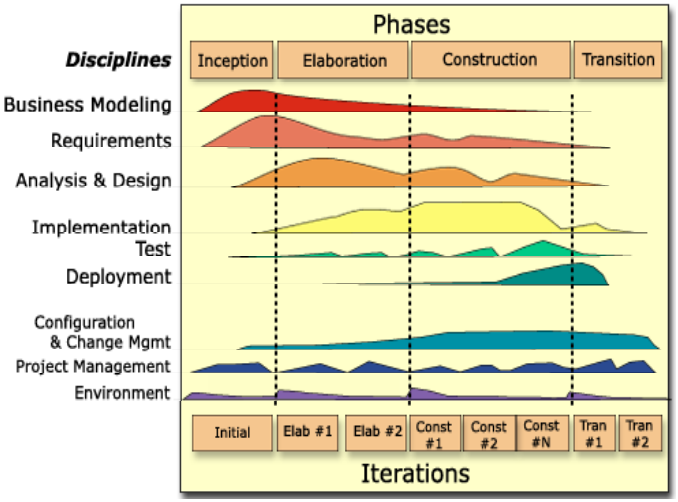




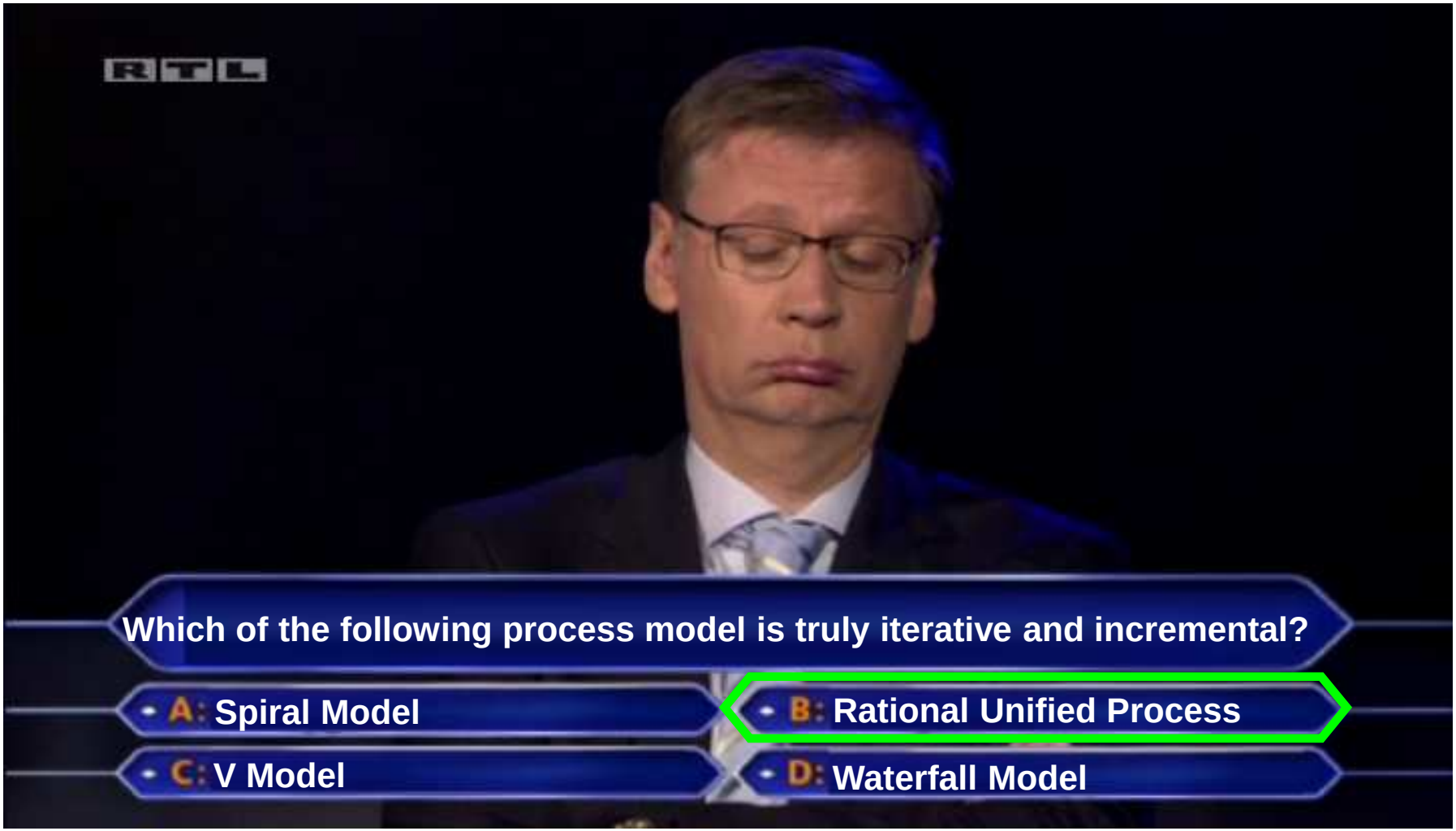
- … explaining how to incorporate
  1. development phases
  2. testing
  3. and risk management

  so that a full grown process can be puzzled together

Rational Unified Process



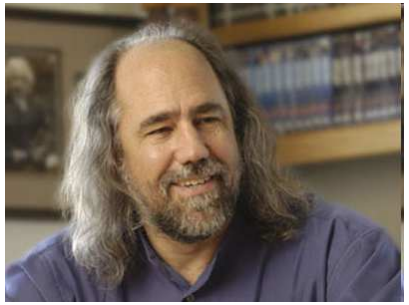
Once again…


RTL
Which of the following process model is truly iterative and incremental?
A: Spiral Model
B: Rational Unified Process
C: V Model
D: Waterfall Model

Living History

- Booch's Hypothesis: [Endres/Rombach03]

  Object model reduces communication problems between analysts and users.



Grady Booch

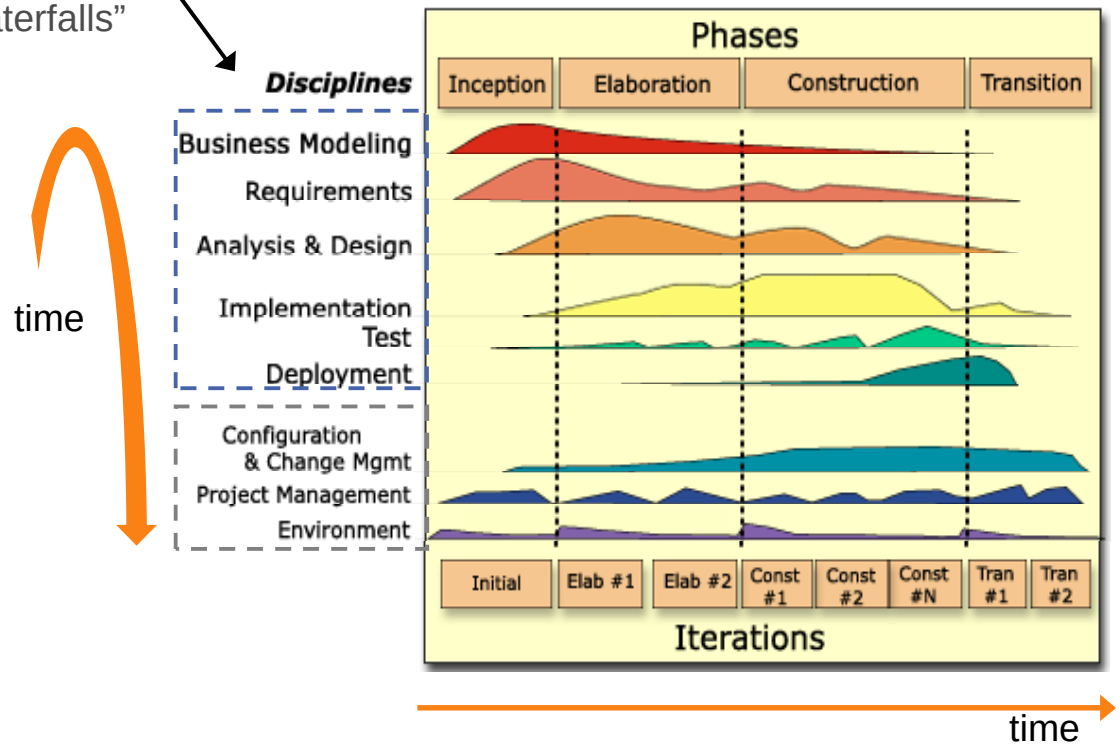
- RUP is the process counterpart of the UML
  - unified from various different approaches
    - OOA/D (Yourdon and Coad)
    - OMT (Rumbaugh)
    - Objectory (Jacobson)
    - Fusion (Coleman et al.)
    - Booch (Booch)
- Rational started the development of RUP and UML in the mid 1990s
  - with the three amigos: Rumbaugh, Jacobson, Booch

The Rational Unified Process

- The RUP is a software development process model
  - provides a disciplined approach for assigning tasks and responsibilities within an organization in order to develop specific products
    - **roles -> who?**
      - defines a set of **skills** and **responsibilities**
    - **activities/tasks -> how?**
      - describes work packages that need to be carried out by a role to achieve a result (i.e. a product)
      - are implemented in disciplines (a.k.a. workflows) in each iteration
    - **artifacts/work products -> what?**
      - results of a task, e.g. models and documents
  - it is supposed to be –
    - **iterative und incremental**
    - **use-case-driven**
    - **architecture-centric**

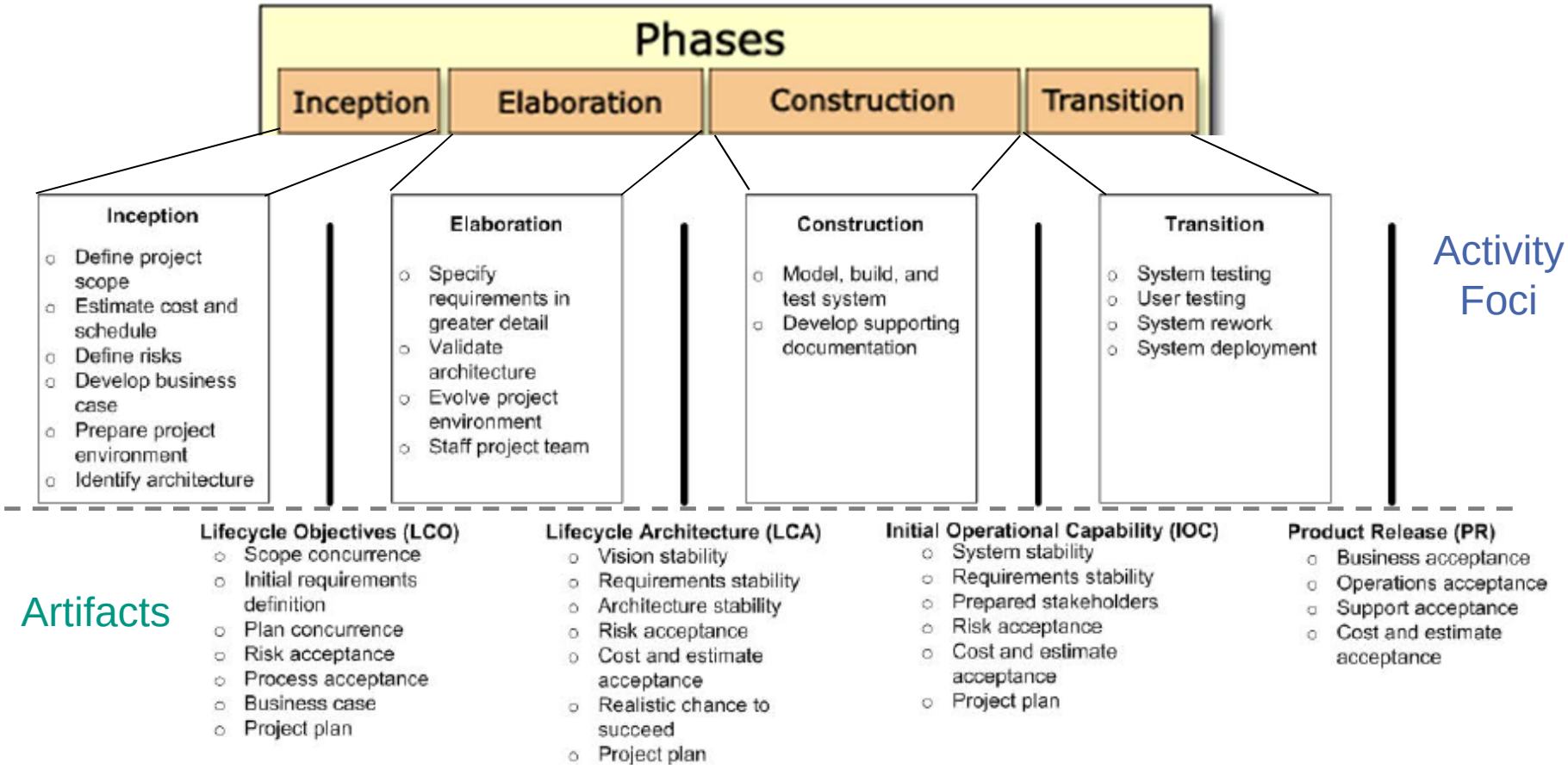
RUP Overview

- The RUP defines –
  - 4 abstract phases
    - to be concluded with a milestone
    - ***not equivalent to waterfall phases!!!***
  - 9 disciplines
    - 6 engineering + 3 supporting
    - here we find our "mini waterfalls"
  - 6 central best practices
    1. *develop software iteratively*
    2. *manage requirements*
    3. *use component-based architectures*
    4. *model software visually*
    5. *verify software quality*
    6. *control changes to software*



Phases at a Glance

- The RUP phases, their objectives and their concluding milestones [Ambler05]



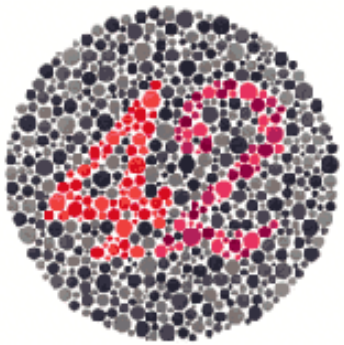
If you can‘t read the following…

- … slides
  - ➔ never mind

- The following two slides are just to illustrate the complexity of RUP
  - With its –
    - 57 activities
    - 117 artifacts
    - 38 roles

➔ ***tailor it to your needs in practice!***

- if you see a 48 here, go get your eyes checked ;-)



Heavyweight (1): Engineering Disciplines

	Business Modeling	Requirements	Analysis and Design	Implementation	Test	Deployment
Activities (57)	1. Assess business status 2. Describe current business 3. Identify business process 4. Refine business process definitions 5. Design business process realizations 6. Refine roles and responsibilities 7. Explore process automation 8. Develop a domain modeling	1. Analyse the problem 2. Understand stakeholder needs 3. Define the system 4. Manage the scope of the system 5. Refine the system definition 6. Manage changing requirements	1. Define a candidate architecture 2. Refine the architecture 3. Analyse behaviour 4. Design components 5. Design real time components 6. Design the database 7. Perform architectural synthesis	1. Structure the implementation model 2. Plan the integration 3. Implement components 4. Integrate each subsystem 5. Integrate the system	1. Plan test 2. Design test 3. Implement test 4. Execute tests in integration test stage 5. Execute tests in system test stage 6. Evaluate test	1. Plan deployment 2. Develop support material 3. Manage acceptance tests 4. Produce deployment unit 5. Package product 6. Provide access to download site 7. Beta test product
Artifacts (117)	1. Support specifications 2. Business glossary 3. Business rules 4. Business use case model 5. Business object model 6. Target organization assessment 7. Business vision 8. Business architecture document 9. Supplemantary business specification 10. Business use case 11. Business use case realization 12. Organization unit 13. Business entity 14. Business worker 15. Business modelling guidelines 16. Review record 17. Analysis model	1. Software architecture document 2. Requirements measurement plan 3. Stakeholder requests 4. Glossary 5. Vision 6. Use case model 7. Supplementary specifications 8. Use case 9. Software requirements specification 10. User interface prototype 11. Use case storyboard	1. Component 2. Reference architecture 3. Software architecture document 4. Use case realization 5. Analysysis model 6. Design model 7. Design subsystem 8. Design package 9. Design class 10. Interface 11. Capsule 12. Protocol 13. Data model 14. Deployment model 15. Integration build plan 16. Test component	1. Integration build plan 2. Component 3. Implement subsystem 4. Software architecture document 5. Integration build plan 6. Test component	1. Change requests 2. Test plan 3. Test model 4. Test case 5. Test procedure 6. Test script 7. Test class 8. Test packages 9. Test component 10. Test subsystem 11. Test results 12. Test evaluation summary 13. Workload analysis document	1. Installation component 2. End-user artifacts 3. Support material 4. Deployment plan 5. Release notes 6. Bill of materials 7. Training material 8. Test results 9. Change request 10. Development infrastructure 11. Development unit 12. Product
Roles (38)	1. Business process analyst 2. Business designer 3. Stakeholders 4. Business reviewer	1. System analyst 2. Use case specifyer 3. User interface designer	1. Architect 2. Designer 3. Database designer 4. Capsule designer	1. Architect system integrator 2. System integrator 3. Code reviewer 4. Implementer	1. Test designer 2. Designer 3. Implementer 4. Tester	1. Implementer 2. Technical writer 3. Deployment manager 4. Graphic artist 5. Course developer

➔ *tailor it to the needs of your project!*

Heavyweight (2): Supporting Disciplines

	Project Management	Environment	Configuration and Change management
Activities (57)	1. Conceive new project 2. Evaluate project scope and risk 3. Develop software development plan 4. Monitor and control project 5. Plan for next iteration 6. Manage iteration 7. Close out phase 8. Close out project	1. Prepare environment for project 2. Prepare environment for an iteration 3. Prepare guidelines for an iteration 4. Support environment during an iteration	1. Plan project configuration and change control 2. Create a project configuration management enviror 3. Change and deliver configuration items 4. Manage baselines and releases 5. Monitor and report configuration status 6. Manage change requests
Artifacts (117)	1. Test plan 2. Software architecture document 3. Iteration assessment 4. Business case 5. Software development plan 6. Iteration plan 7. Problem resolution plan 8. Risk management plan 9. Product acceptance plan 10. Measurement plan 11. Work order 12. Status assessment 13. Project measurements 14. Review record 15. Requirements Attributes 16.Vision 17. Risk list 18. Change requests	1. Development case 2. Development organization assessment 3. Project specific templates 4. Manual style guide 5. Use case modeling guidelines 6. Requrements management plan 7. Business modeling guidelines 8. User interface guidelines 9. Test guidelines 10. Design guidelines 11. Programming guidelines 12. Tools 13. Tool support assessment 14. Tool guidelines 15. Support environment	1. Project measurements 2. Deployment unit 3. Configuration audit fundings 4. Configuration management plan 5. Project repository 6. Change request 7. Workspace 8. Work order (integration) 9. Work order (completed) 10. Workspace (development)
Roles (38)	1. Project manager	1. Process engineer 2. Technical writer 3. System analyst 4. Business process analyst 5. User interface designer 6. Test designer 7. Architect 8. Tool specialist 9. System administrator	1. Configuration manager 2. System integrator 3. Change control manager 4. Project member

Roles in a Project ☺

Why is Software Development* so difficult?

- "In short, automatic programming always has been a euphemism for programming with a higher-level language than was then available to the programmer.

  Research in automatic programming is simply research in the implementation of higher-level programming languages."

  Dave Parnas, 1985

** or Reuse, or Modelling or any other technique intended to simplify programming*

[http://klabs.org/richcontent/software_content/papers/parnas_acm_85.pdf]

Lust auf eine ExpedITion?

Stuttgart
28.11.2013

Rightshore? One Team? – Neue Perspektiven!

Erleben Sie, wie Capgemini lokale Projekte mit globaler Delivery zum Erfolg führt. Wir zeigen, wie die Zusammenarbeit in interkulturellen Teams bei Capgemini funktioniert und Sie dürfen es dann in unserem Workshop selbst ausprobieren

In einem interkulturellen Training erhalten Sie von erfahrenen Capgemini Kollegen viele hilfreiche Tipps für die Zusammenarbeit mit indischen Kollegen. Diese können Sie dann auch direkt in der Praxis anwenden:

In einer Case Study bereiten Sie ein Meeting mit den indischen Capgemini-Kollegen vor und führen dieses per Live-Videokonferenz durch.

Im Anschluss erhalten Sie dann von den indischen und deutschen Kollegen ein Feedback.

expedITion Workshop Stuttgart	
12:00	Ankommen und Begrüßung
12:15	Einführung Case Study: Offshore Project Kick-Off
12:30	Gruppenarbeit Case Study
14:00	Kaffeepause
14:30	Abschluss Case Study: Live Kick-Off Meeting
16:00	Pause
16:15	Intercultural Session: Incredible India
17:30	Abschluss des Workshops
18:00	Abendbuffet und Topic Tables
ca. 20:00	Ende

Weitere Infos und Anmeldung unter:
www.capgemini-expedITion.de

Conclusion

- The best practice in industry today is a iterative approach to software development
  - usually based on a process such as the RUP defining –
    - activities (how)
    - artefacts (what)
    - roles (who)

- Thank you for your attention!

References (1)

- [Ambler] S. Ambler
  The Object Primer: Agile Model-Driven Development with UML 2.0
  Cambridge University Press, 2004

- [Ambler05] S. Ambler
  A Manager's Introduction to the Rational Unified Process, 2005
  http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

- [Endres/Rombach03] A. Endres, D. Rombach
  A Handbook of Software and Systems Engineering, Addison-Wesley, 2003

- [Larman] C. Larman
  Applying UML and Patterns (3rd ed.)
  Prentice Hall, 2004

- [Larman und Basili] C. Larman, V. Basili: Iterative and incremental developments. a brief history, IEEE Computer 36/6, 2003.

- [Booch] G. Booch et al.
  Object Oriented Analysis and Design
  Addison-Wesley, 1993

References (2)

- [Fusion] D. Coleman et al.
  Object-Oriented Development: The Fusion Method
  Prentice Hall, 1995
- [Objectory] I. Jacobson
  Object-Oriented Software Engineering – A Use Case Driven Approach
  Addison Wesley, 1992
- [OOA] P. Coad & E. Yourdon
  Object Oriented Analysis
  Yourdon Press, 1990
- [OOD] P. Coad & E. Yourdon
  Object-Oriented Design
  Prentice Hall, 1991
- [OMT] J. Rumbaugh et al.
  Object-Oriented Modeling and Design
  Prentice Hall

Appendix: History of MLs

[Wikipedia]