Karlsruhe Institute of Technology

Software Engineering II

Prof. Dr. Raffaela Mirandola

Topic 11

Software Development Processes

SASIS – SELF-ADAPTIVE SOFTWARE-INTENSIVE SYSTEMS
KASTEL – INSTITUTE OF INFORMATION SECURITY AND DEPENDABILITY

sasis.kastel.kit.edu

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 Unified Process
- Conclusion

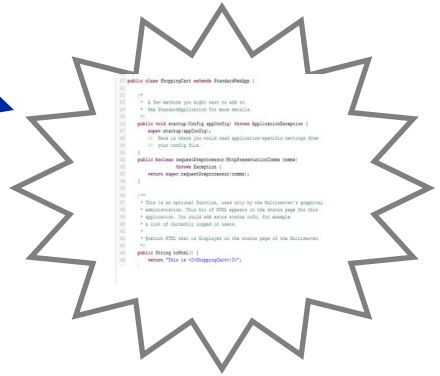
Learning Goals

- 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?

Problem / opportunity in the real world



Code

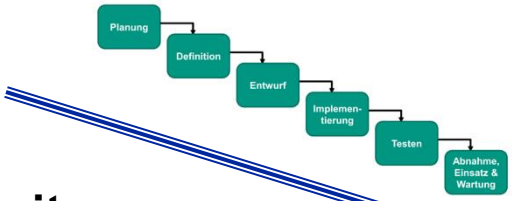
- Code and Fix usually leads to:
  - badly structured code
  - non-systematic improvements
  - no real team work, as tasks can not be planned
  - → Guru needed, no scalability of project size, dependability on one person
  - missing design and documentation
  - complications in maintenance
  - …

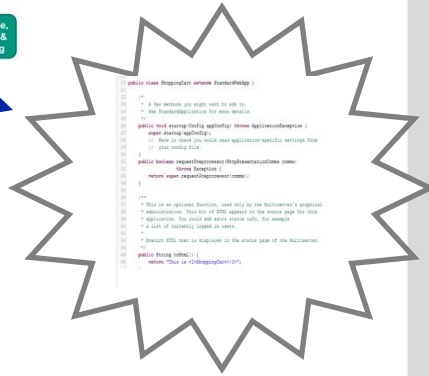
- Are structured processes the way out?

Software Development Processes

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

Problem / opportunity in the real world





Code

- What are typical advantages of processes?

_____

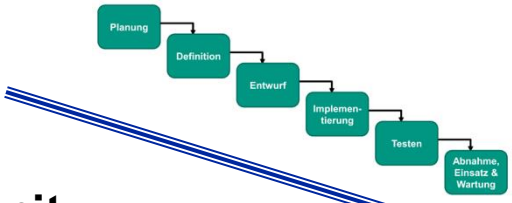
_____

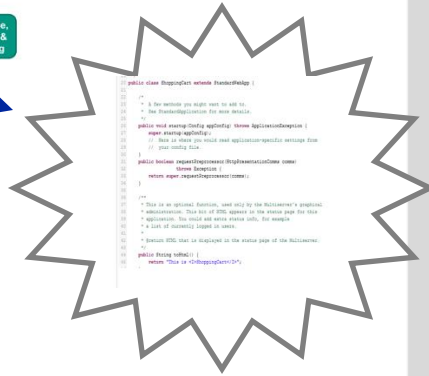
_____

Software Development Processes

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

Problem / opportunity in the real world





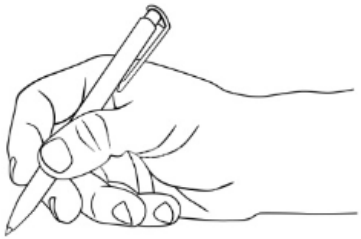
Code

- What are typical advantages of processes?

  lower project risks

  higher product quality

  better maintainability



➔ depends on concrete project and selected process

Process Models

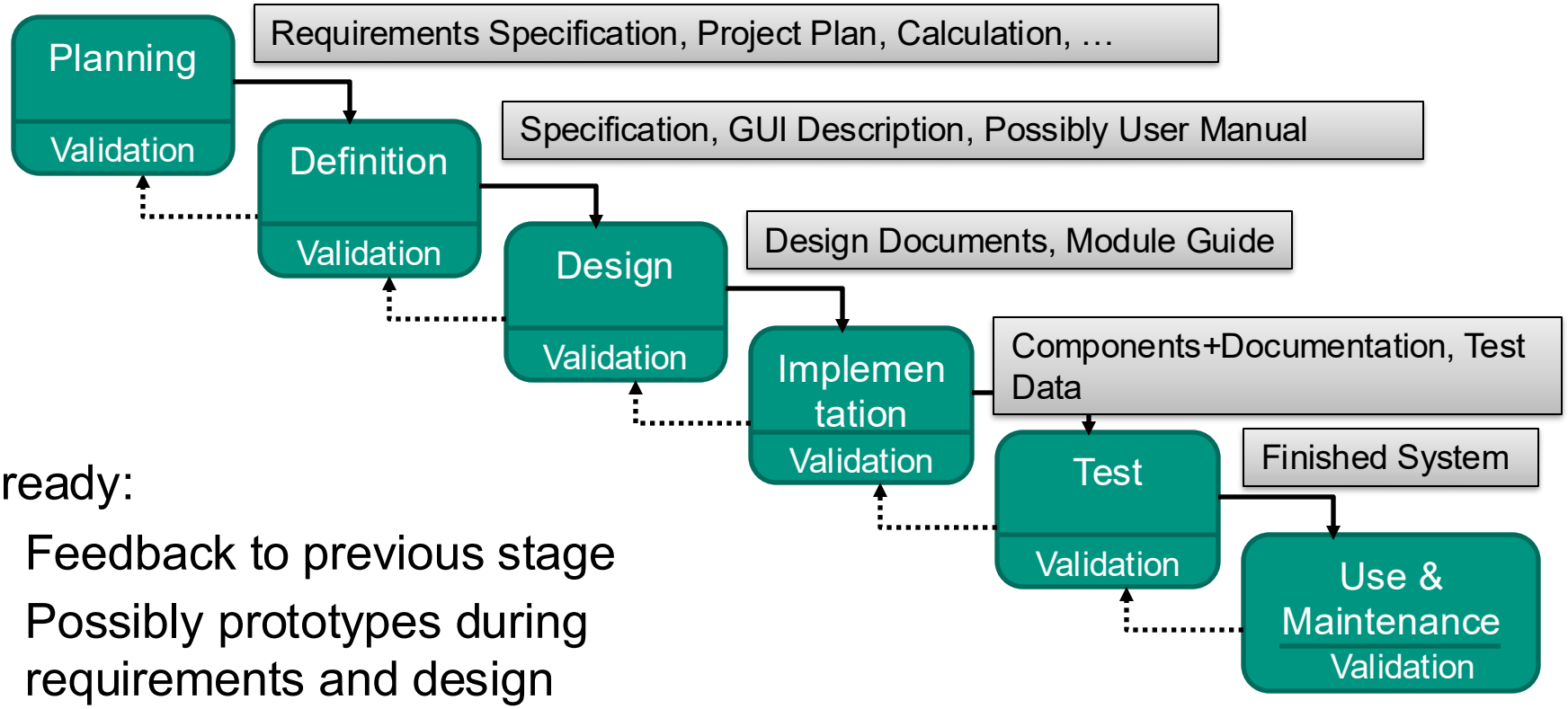
- Software process model (“Vorgehensmodell”) is an **abstract representation of a software development process**
  - *it presents a description of a process from some particular perspective*

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

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

Recap: Waterfall Model (from SE/SWT1)

- The Waterfall Model as a sequential process model …



Already:

- Feedback to previous stage
- Possibly prototypes during requirements and design

From SWT (SE) 1 Lecture by Walter F. Tichy

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.

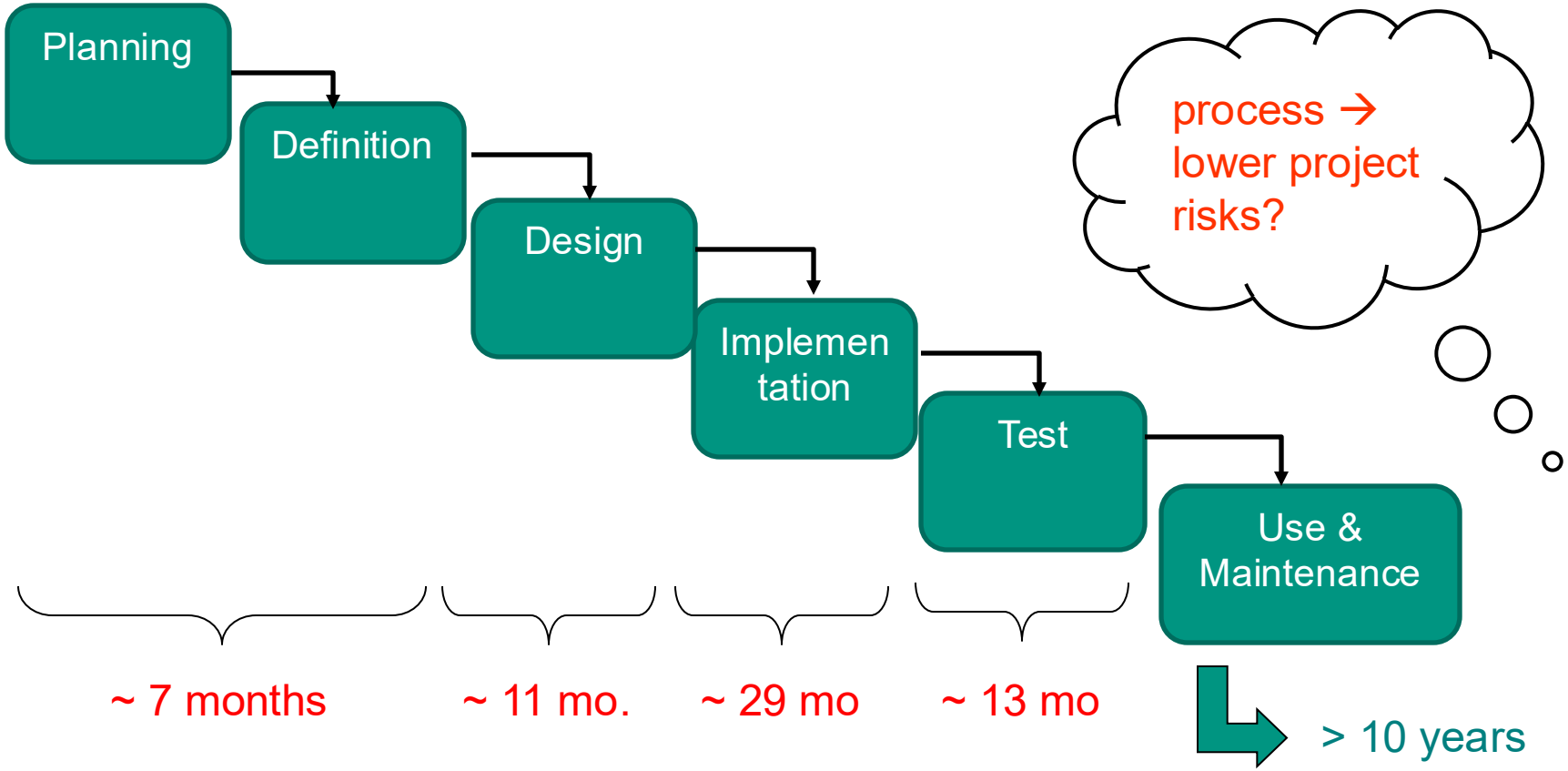
I think this assumption is fundamentally wrong, and that many software acquisition problems spring from that fallacy.”
[No Silver Bullet, 1986]

➔ *“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



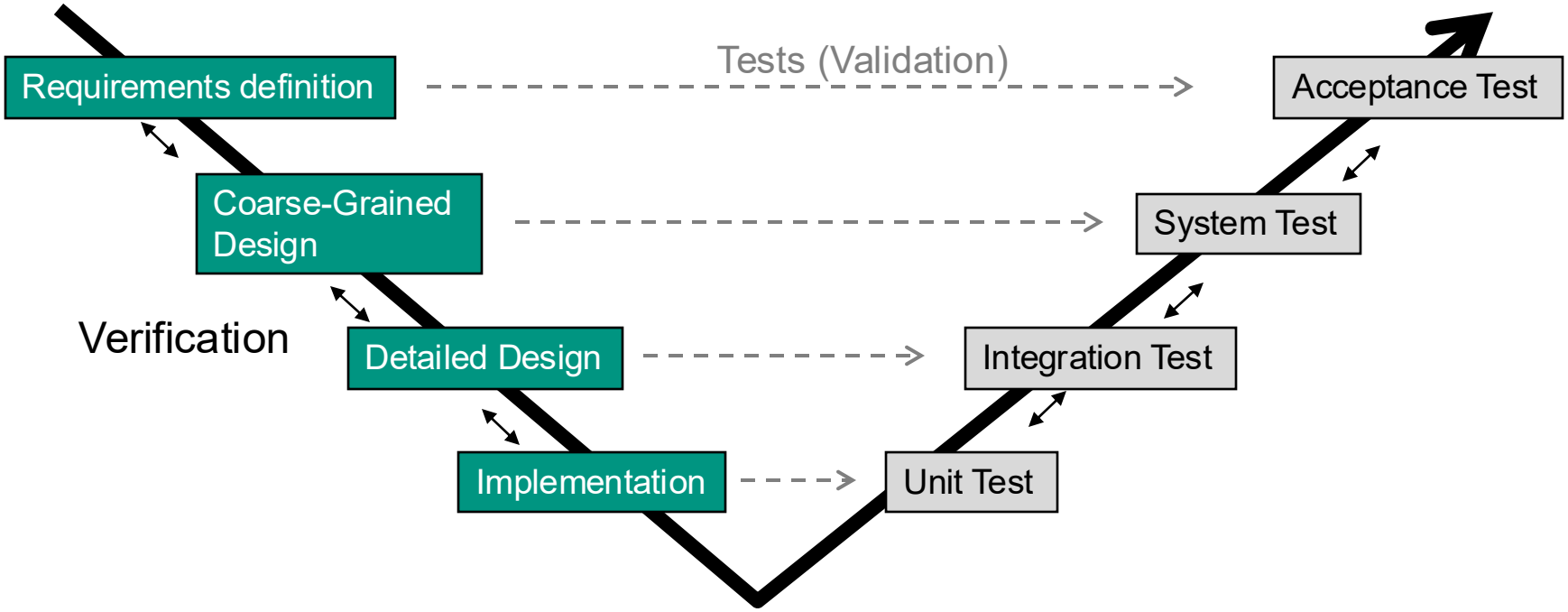
Waterfall Development Life Cycle

- 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 activities relate to each other.

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*

➔ _____

_____

Self-Assessment (2)

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

   ➔ Planning, Specification, Design, Implementation, Test, Deployment

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

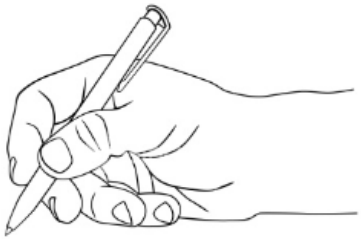
   ➔ Activities, roles, and artifacts

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

   ➔ Users requirements often change

   ➔ Changes invalidate plan, design, and/or implementation

   ➔ Waterfall lacks feedback cycles for replanning, redesign, …



Alternative Life Cycle?

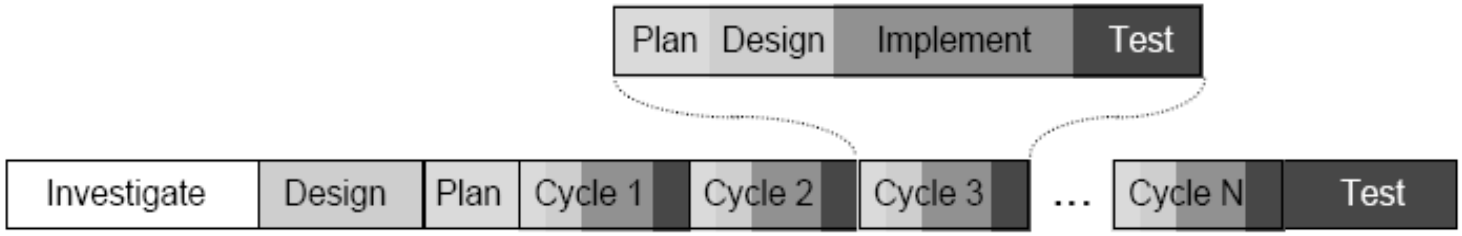
Sequential

e.g. 5 years

Waterfall Development Life Cycle

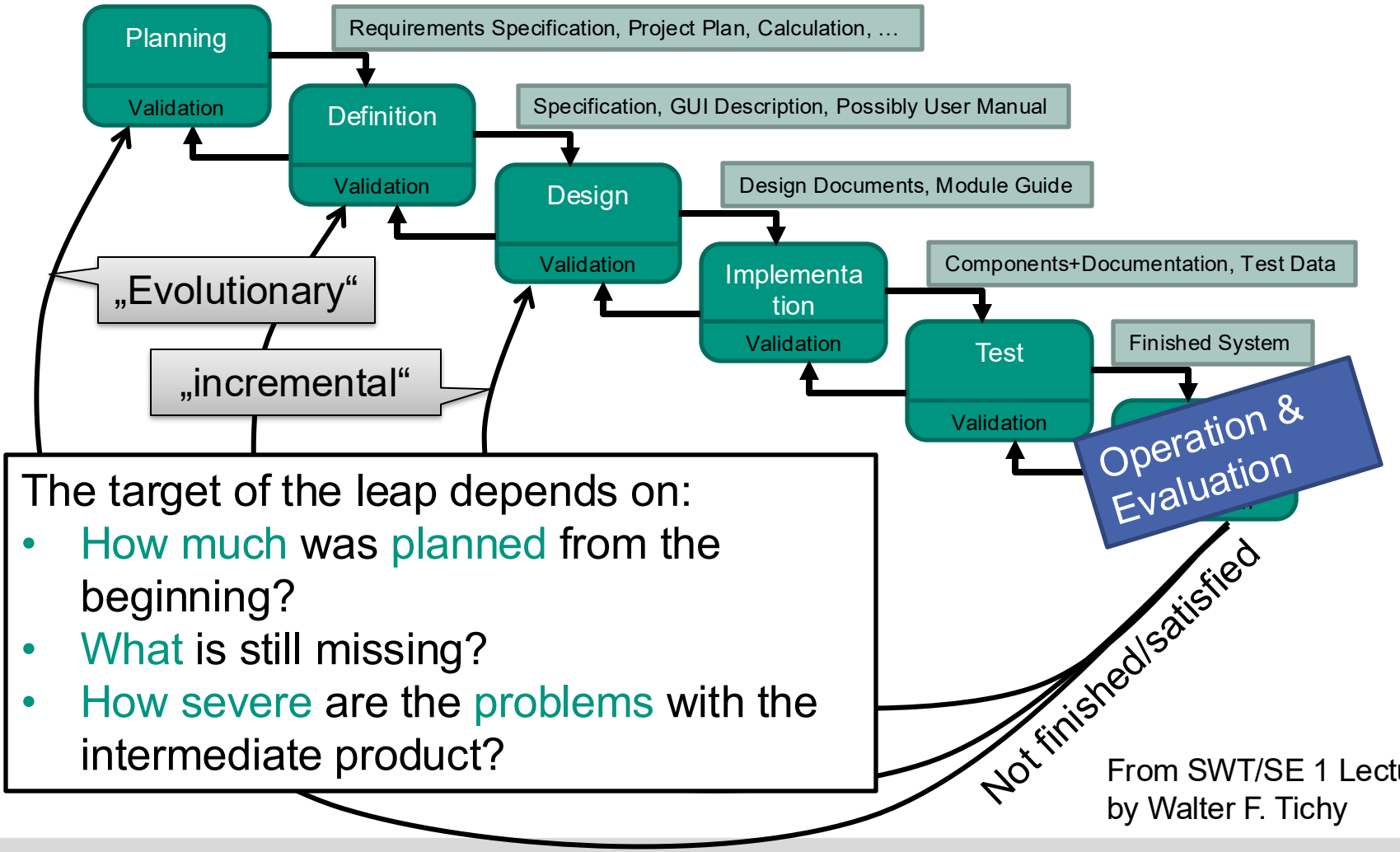
Iterative

usually 2-4 weeks

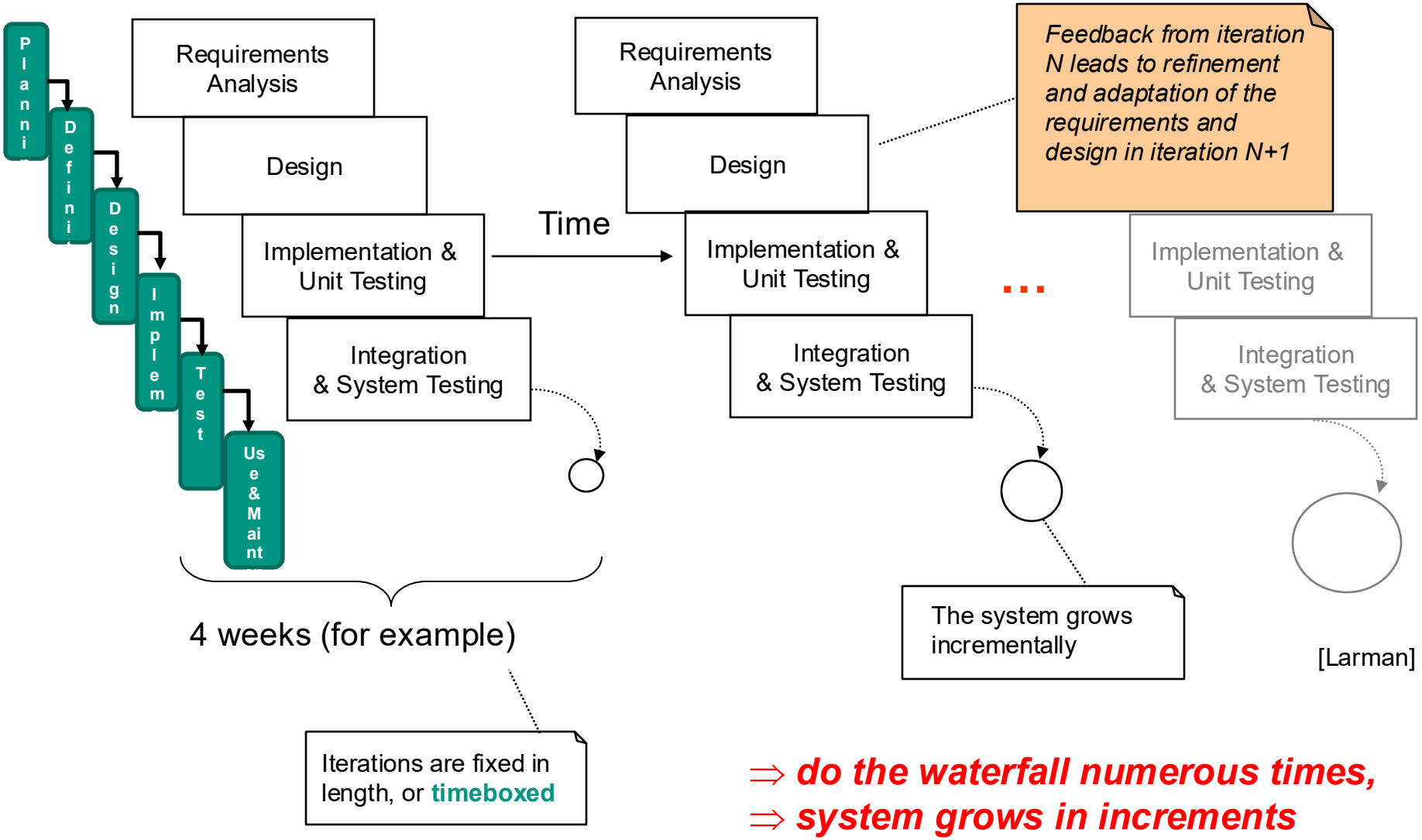


Incremental Development Life Cycle

Recap: Iterative Process (from SE/SWT1)



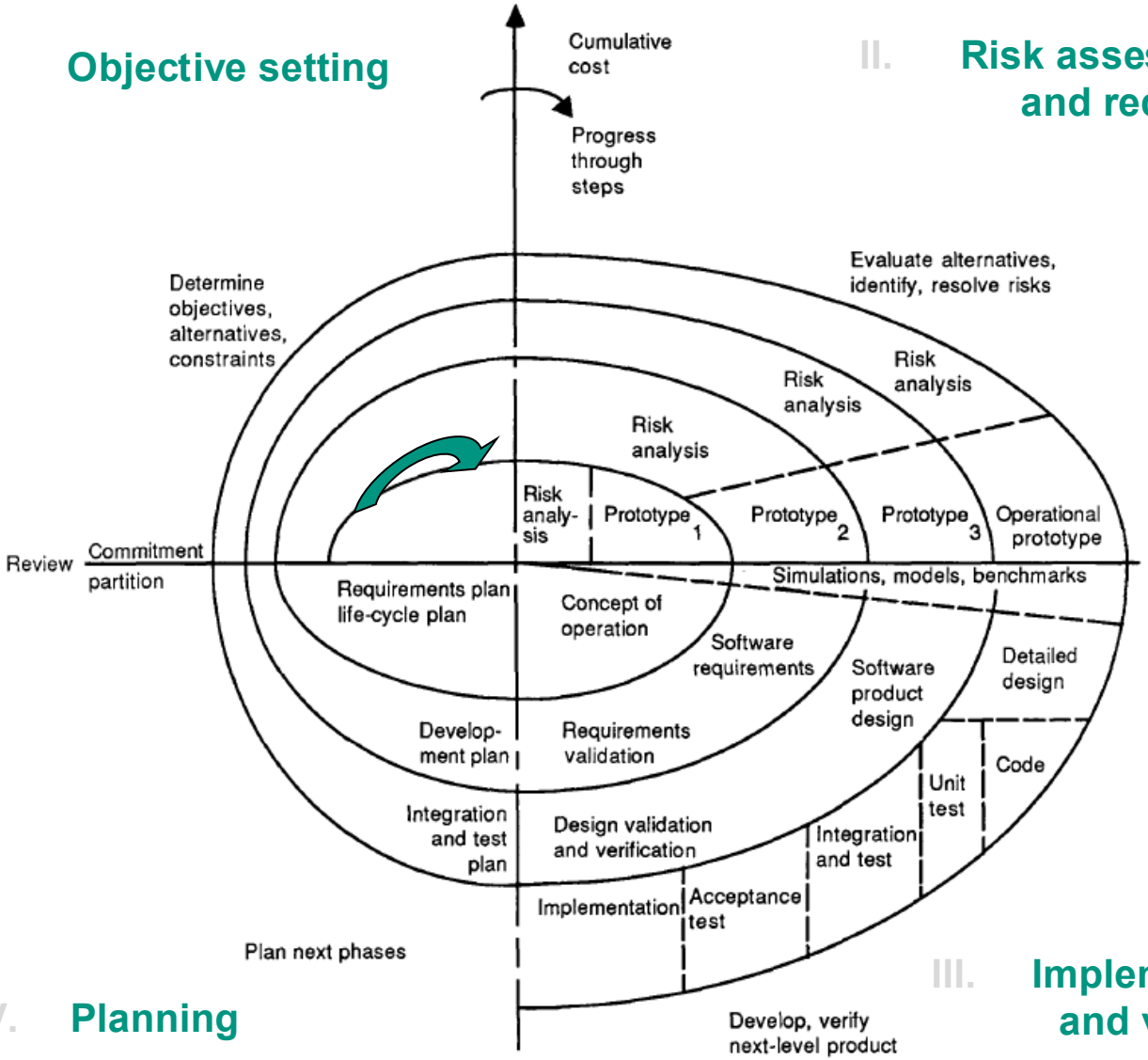
Evolutionary and Incremental Development



⇒ ***do the waterfall numerous times,***
⇒ ***system grows in increments***

NOT to be Confused with *Spiral Model* Picture

[Boehm88]

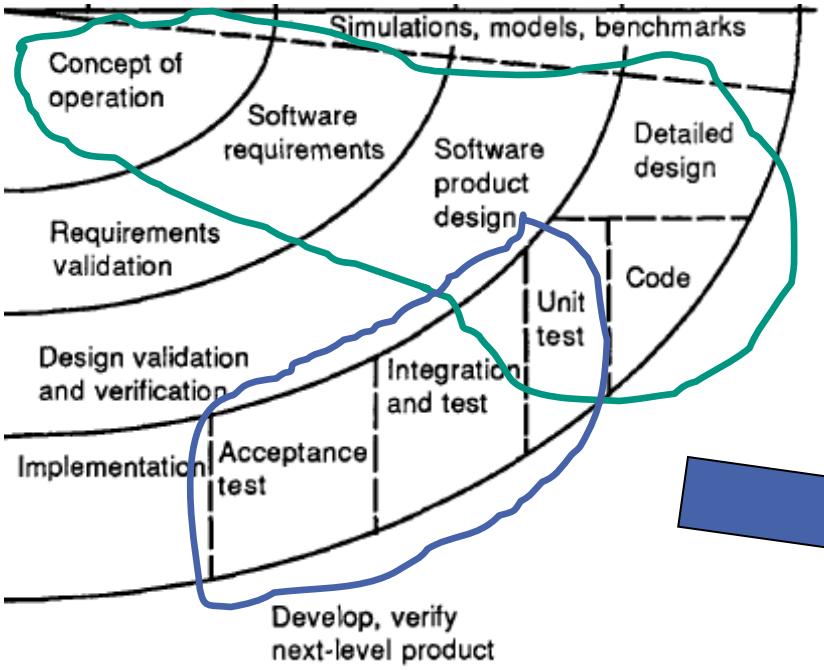


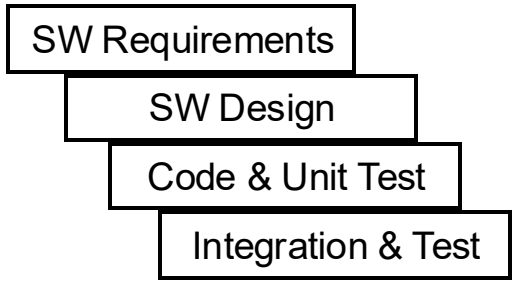
Boehm's Hypothesis:

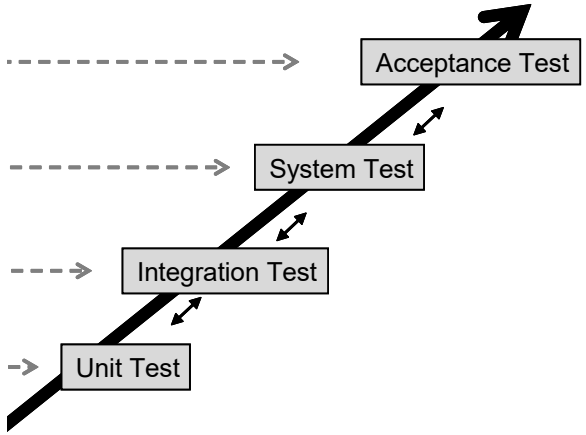
Project risks can be resolved or mitigated by addressing them early.

Spiral Model Picture Revisited

- *Is it evolutionary and incremental?*
  - *i.e. does it go through the waterfall multiple times?*
  - *i.e. does the system grow in increments?*







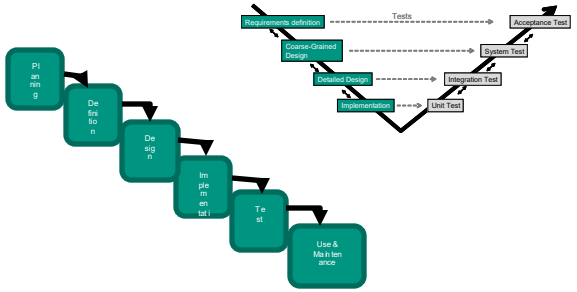
III. Implementation and validation

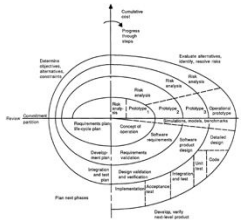
➔ the spiral model picture basically incorporates a waterfall and a V model, **but is not a genuine I&I model**

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

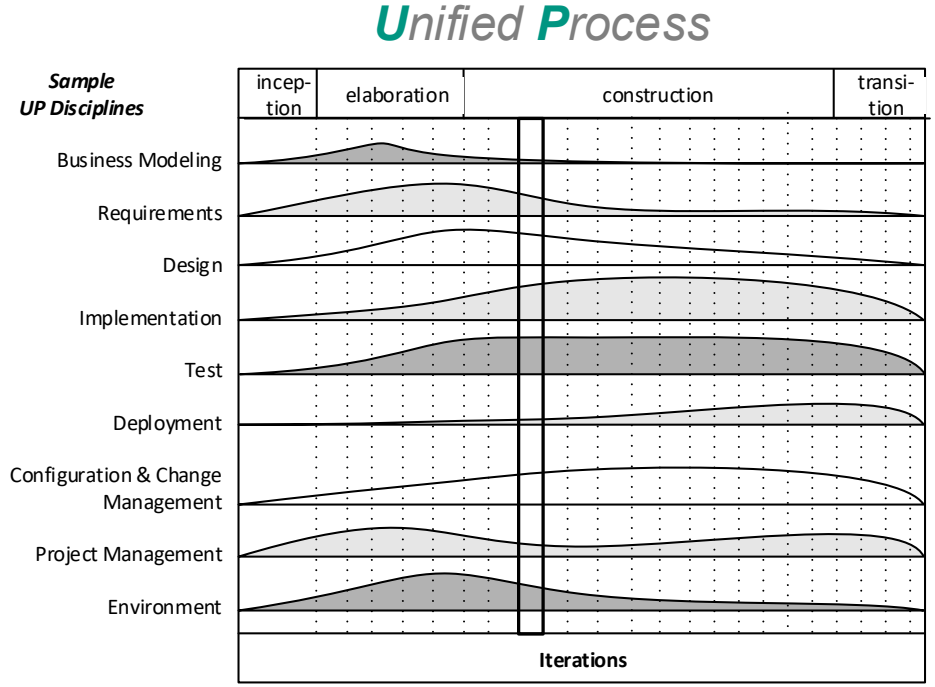
  can be seen as building blocks for software development processes





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

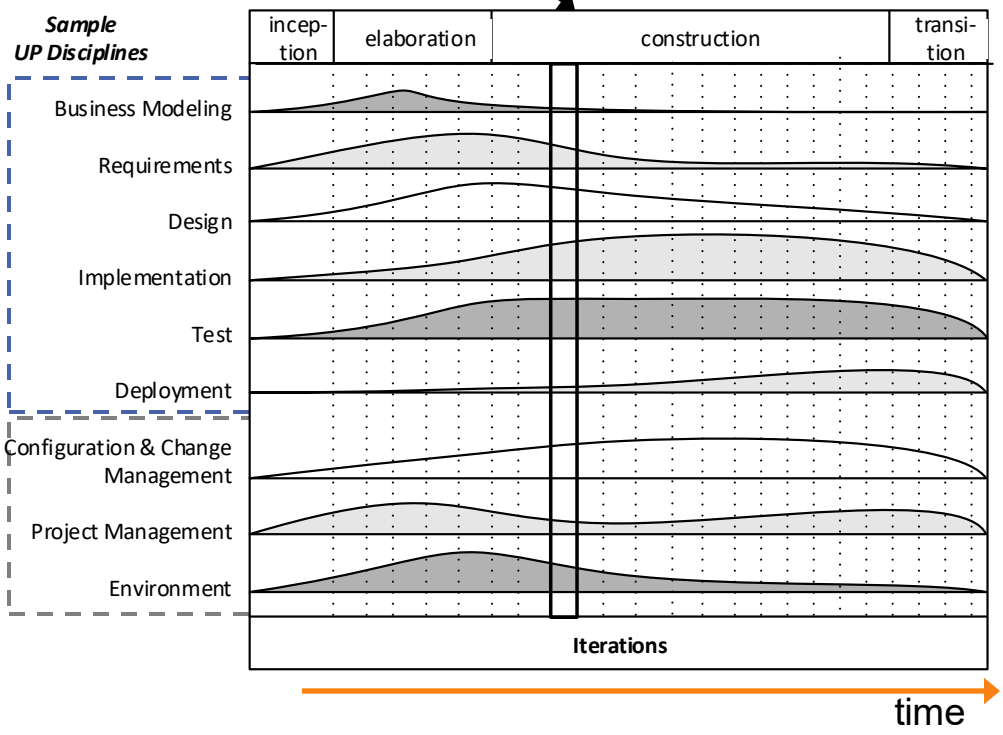
  such that a full grown process can be pieced together



UP Overview

- The UP 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”
- UP is supposed to be –
  - **iterative und incremental**
  - **risk-driven**
  - **client-driven**
  - **architecture-centric**

Adapted from: [Larman]



UP Phases

Inception (Anfangsphase)

- *"Envision the project scope, vision, and business case"* [Larman]
- Explore following kind of questions:
  - *„What is the vision and business case for this project?*
  - *Feasible?*
  - *Build and/or buy?*
  - *Rough unreliable cost range: is it $10K – 100K or in the millions?*
  - *Should we proceed or stop?"* [Larman]

Elaboration (Ausarbeitungsphase)

- *"is the initial series of iterations during which, on a normal project,*
  - *the core, risky software architecture is programmed and tested*
  - *the majority of requirements are discovered and stabilized*
  - *the major risks are mitigated or retired"* [Larman]
- *"Build the core architecture, resolve the high risk elements, define most requirements, and estimate the overall schedule and resources"* [Larman]

Construction (Konstruktionsphase)

- *"Iterative implementation of the remaining lower risk and easier elements, and preparation for deployment"* [Larman]

Transition (Übergabephase)

- *"beta tests, deployment"* [Larman]

→ All disciplines relevant in all phases, but usually relative effort shifts across phases

UP Disciplines

- **Business Modelling**
  - *"to understand and communicate the structure and the dynamics of the organization in which a system is to be deployed"* [Larman]
  - Mainly domain concepts and business processes
- **Requirements**
  - Requirements engineering activities: Elicitation, analysis, documentation, validation, management
  - Possible artefacts: Use-Case models, other requirements, glossary, vision statement, business rules
- **Design**
  - Plan how to realize system in software, object-oriented analysis
  - Model software classes as opposed to real world concepts
  - Main artefacts: Software classes, behaviour models, architecture
- **Implementation, Test, Deployment**
- **Configuration & Change Management, Project Management, Environment**

But: No strict rules what to do!

The Rational Unified Process

- The RUP is a specific implementation of the UP
  - provides a disciplined approach for assigning tasks and responsibilities within an organization 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

- 6 central best practices

1. develop software iteratively	4. model software visually
2. manage requirements	5. verify software quality
3. use component-based architectures	6. control changes to software

If you can‘t read the following…

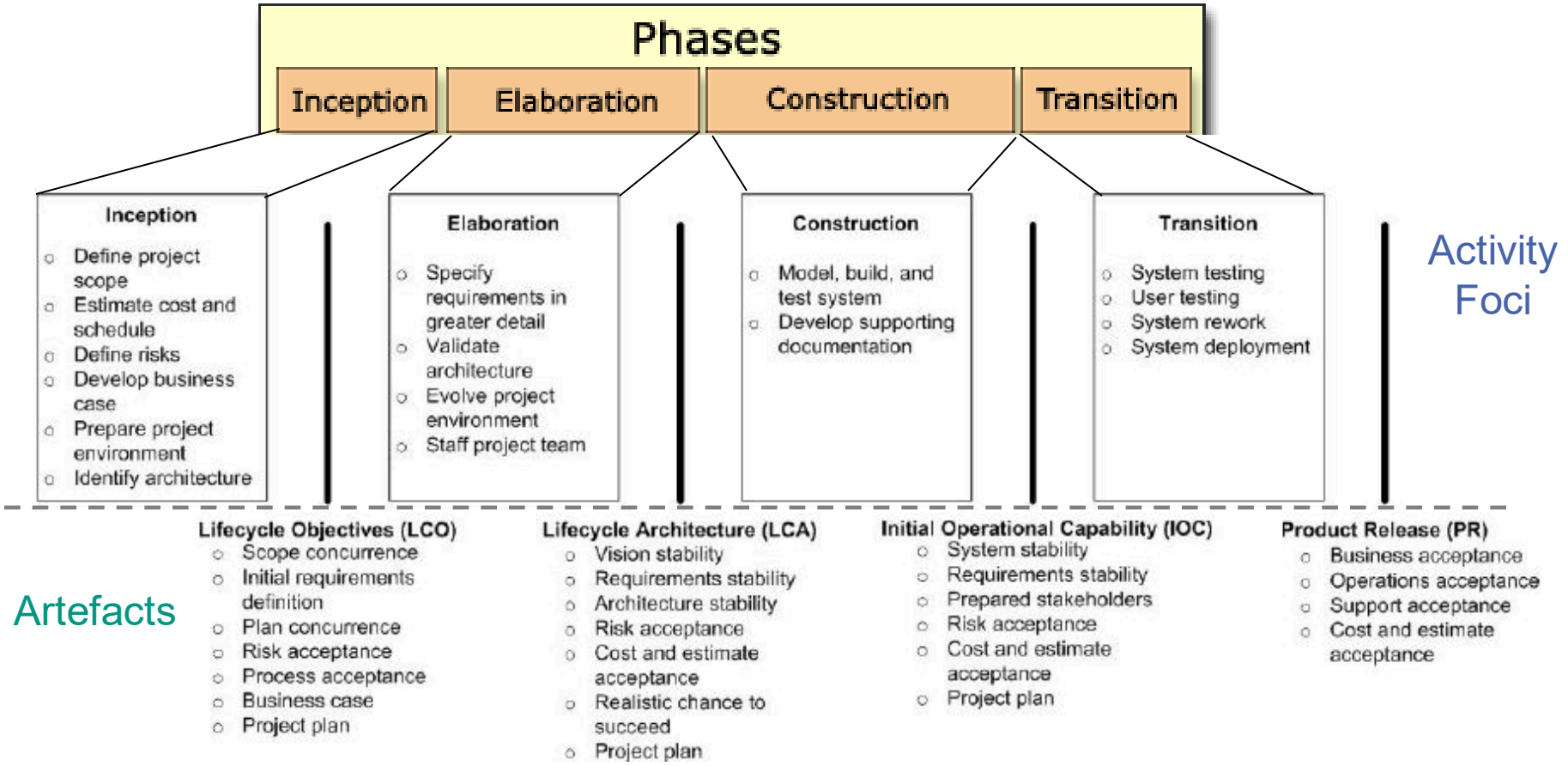
- … slides
  - ➔ never mind
  - ➔ just know UP phases, UP disciplines, RUP best practices, and that RUP additionally defines roles, activities/tasks, and artefacts/work products (previous slides)

- Following two slides are just to illustrate the complexity of RUP
  - With its –
    - 57 activities
    - 117 artefacts
    - 38 roles
  - To know possible activities, artefacts, and roles. Never use them all!

➔ ***In practice, must be tailored to your needs!***

Phases at a Glance

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



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 ☺

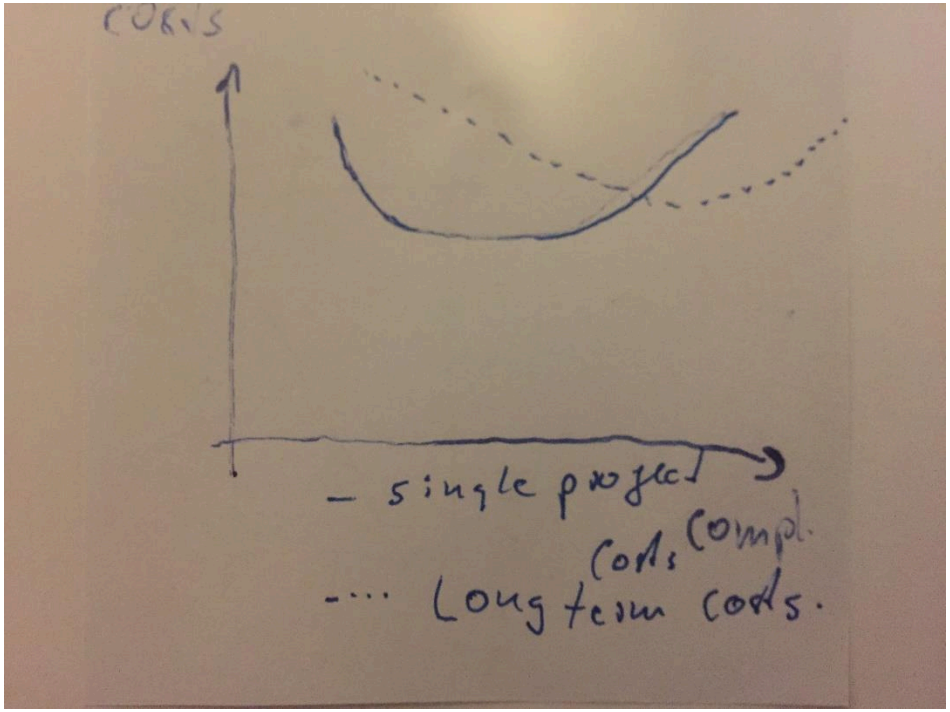
General Reflection on Processes

Good:

- Phases help to structure project
- Divide and conqer of big task into smaller steps
- Roles support specialisation, help to define focus and responsibilities
- Good to have an overview on usually important activities / tasks.

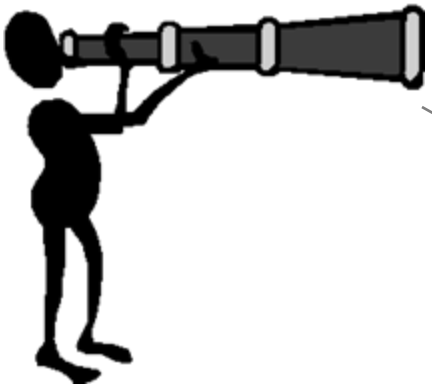
Problematic:

- If planning and realisation are strictly separated, it assumes a stable world. This assumption is most often wrong.
- Roles can lower productivity, as not every role is equally needed at any time.
- There is „optimum“ in the relation between (long term) costs and process complexity (including creating non code artifacts)



Conclusion

- The best practice in industry today is a iterative approach to software development
  - Usually based on a process such as the UP defining
    - Phases, disciplines
  - Possibly even based on a rich process such as RUP adding
    - activities (how), artefacts (what), roles (who)
  - Or leaner and more agile: *next lecture*



Let‘s get agile!

thetable.wordpress.com

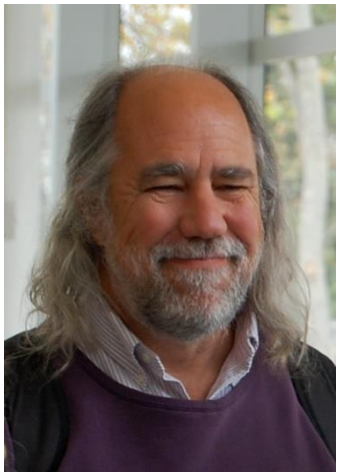
References

- [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

Appendix: Living History

- Booch's Hypothesis: [Endres/Rombach03]

  Object model reduces communication problems between analysts and users.



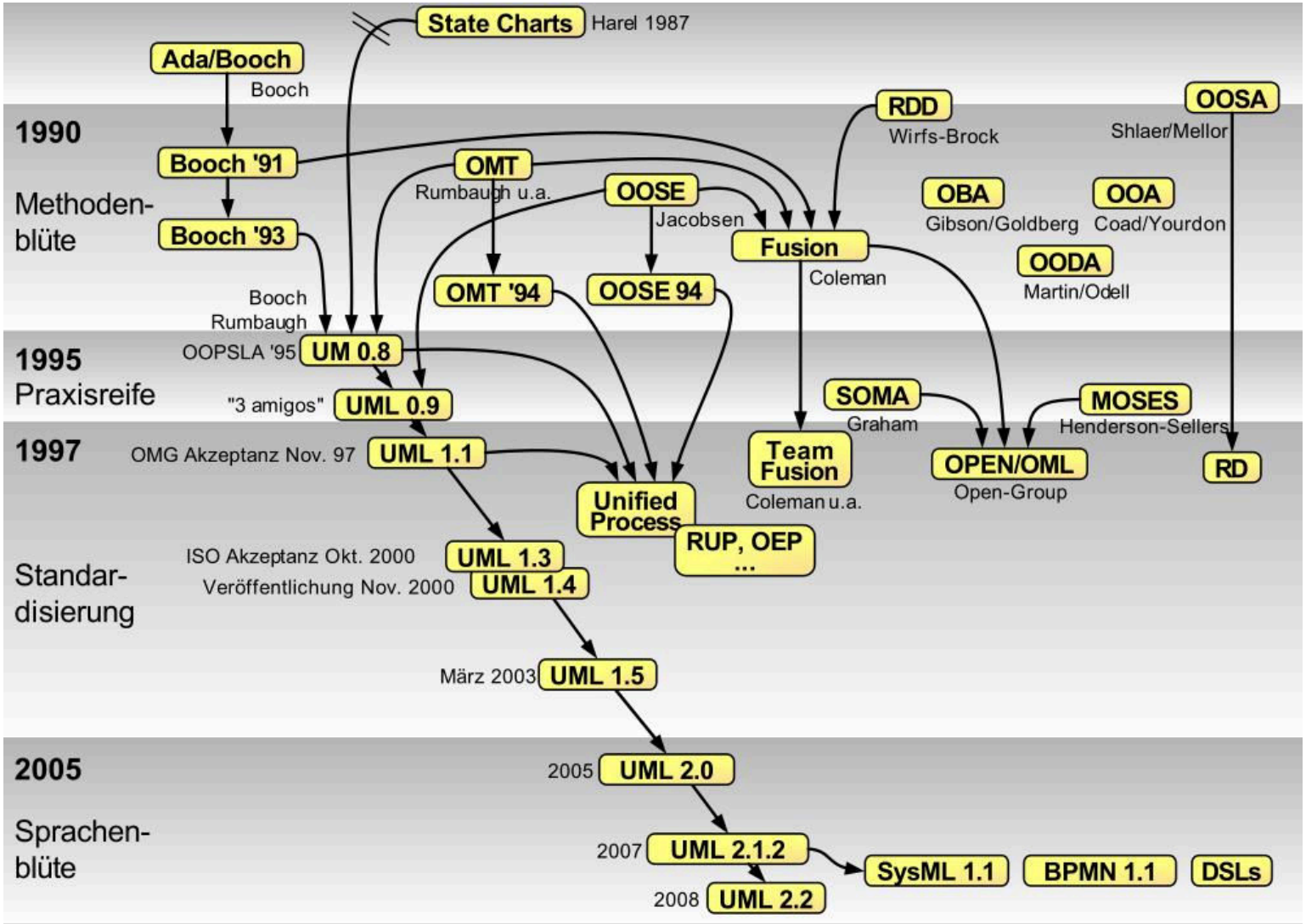
Grady Booch
[Image: Wikipedia]

- UP 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 Rational UP and UML in the mid 1990s
  - with the three amigos: Rumbaugh, Jacobson, Booch

Appendix: History of MLs

[Wikipedia]



Appendix: References

- [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