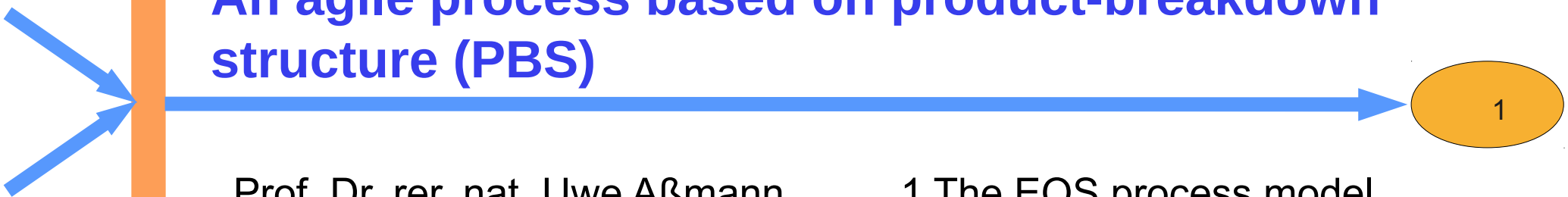
15. Evolutionary Object-Oriented Software Development (EOS)

An agile process based on product-breakdown structure (PBS)



Prof. Dr. rer. nat. Uwe Aßmann
Lehrstuhl Softwaretechnologie
Fakultät Informatik
Technische Universität Dresden
Version 13-1.0, 20.04.12

1 The EOS process model
2 Managing EOS projects

courtesy Prof. Wolfgang Hesse, University of Marburg

Obligatory Literature

- S. Sarferaz: "Methods and tool support for evolutionary, object oriented software development", Ph. D. thesis, Univ. of Marburg
- [Hesse 97a] W. Hesse: From WOON to EOS: New development methods require a new software process model; Bericht Nr. 12, Fachbereich Mathematik, Univ. Marburg; and: Proc. WOON ´96, 1st Int. Conf. on OO technology, St. Petersburg 1997
- [Hesse 97b] W. Hesse: Improving the software process guided by the EOS model. In: Proc. SPI '97 European Conference on Software Process Improvement. Barcelona 1997
- [Hesse, Weltz 94] W. Hesse, F. Weltz: Projektmanagement für evolutionäre Software-Entwicklung; Information Management 3/94, pp. 20-33, (1994)
- [Sarferaz, Hesse 00] S. Sarferaz, W. Hesse: CEOS – A Cost Estimation Method for Evolutionary, Object-Oriented Software Development . In.: R. Dumke, A. Abran (Eds.): New Approaches in Software Measurement. Proc. 10th Int. Workshop, IWSM 2000, Springer LNCS 2006, pp. 29-43

References

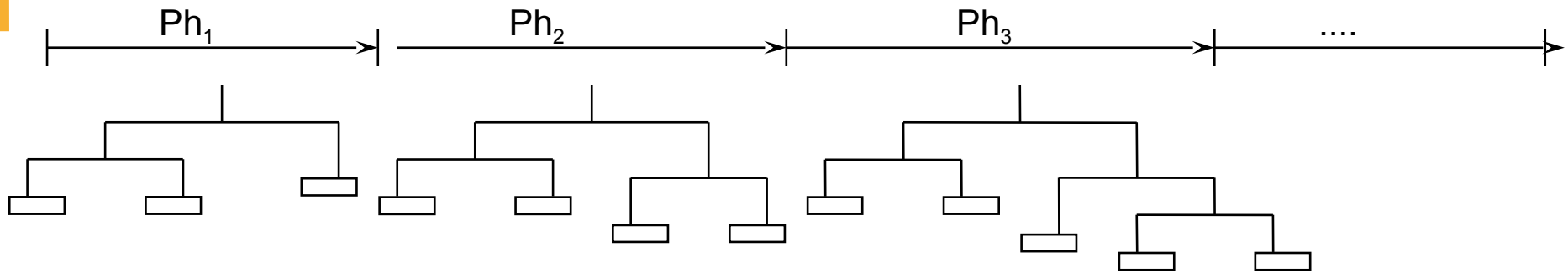
- [Beyer, Hesse 2002] Use of UML for software process modelling. Internal report, Univ. Marburg 2002
- [Bittner, Hesse, Schnath 95] U. Bittner, W. Hesse, J. Schnath: Praxis der Software-Entwicklung, Methoden, Werkzeuge, Projektmanagement - Eine Bestandsaufnahme, Oldenbourg 1995
- [Frese, Hesse 93] M. Frese, W. Hesse: The work situation in software development - Results of an empirical study, ACM SIGSOFT Software Engineering Notes, Vol. 18, No. 3, pp. A-65 - A-72 (1993)
- [Floyd, Reisin, Schmidt 89] Ch. Floyd, F.-M. Reisin, G. schmidt: STEPS to software development with users; in: C. Ghezzi, J. McDermid (eds.): ESEC ‘89, 2nd European Software Engineering Conference; LNCS 387, pp. 48-64, Springer 1989
- [Hesse, Merbeth, Frölich 92] W. Hesse, G. Merbeth, R. Frölich: Softwaretechnik - Vorgehensmodelle, Projektführung und Produktverwaltung, Handbuch der Informatik Bd. 5.2, Oldenbourg 1992
- [Hesse 96] W. Hesse: Theory and practice of the software process - a field study and its implications for project management; in: C. Montangero (ed.): Software Process Tech-nology, 5th Europ. Workshop EWSPT 96; Springer LNCS 1149, pp. 241-256 (1996)

15.1 The EOS Process Model

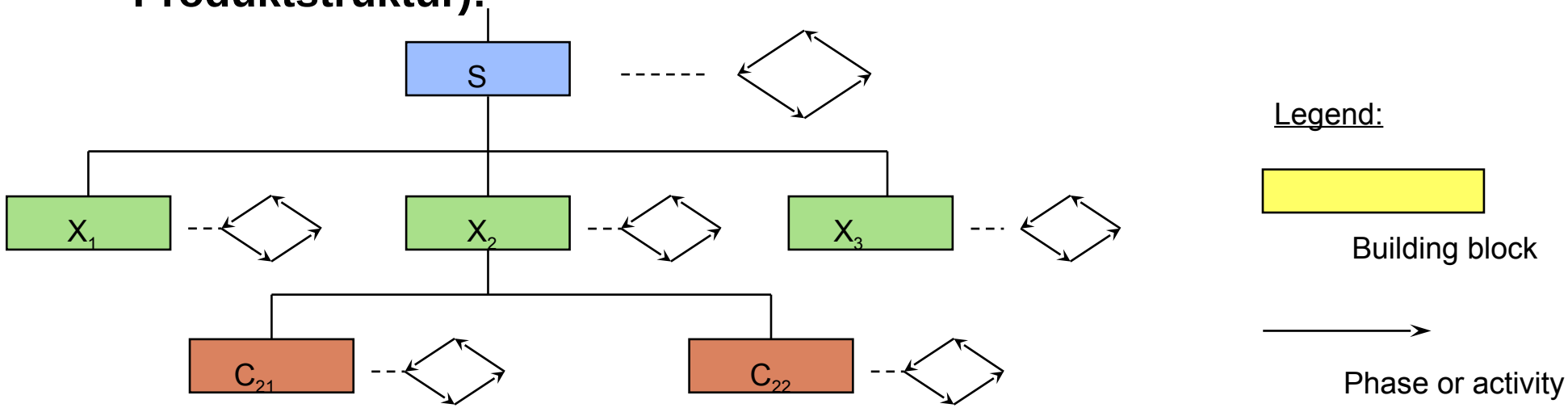
- Heavy-weight process models are often too bureaucratic and not (or hardly) scalable
  - The aspect of software evolution is hardly reflected
  - Planning relies on assumptions and may go wrong
  - Unforeseen descoveries change the planning
- Component-oriented, distributed and web-based SW development requires flexible and well-adaptable processes
- EOS works if the architecture of the system is clear (standard architecture, well-known domain, low innovation)
  - But it treats unforeseen dependencies between the components
  - Different availabilities of resources
  - Parallel work

Phase-oriented vs. component-oriented process

- Process in phases (Phasenmodell):



- EOS is a process structured along **product breakdown strukture (PBS, Produktstruktur):**



Objects and features of the software process

- The **product breakdown structure (PBS, Produktstruktur)** is a decomposition of the software product into components
- In EOS, it is assumed that the PBS is organised in a hierarchy with three level system development structure with three forms of components:
  - S System level
  - X Subsystem level
  - C Class level
- What are the features of those objects?
  - Attributes: Size, Responsible_person, Start_date_of_work, Delivery_date, ...
  - Operations: Development activities: Analysis, Design, Implementation, Operational_Use
  - State: active, interrupted, completed

Development Cycles

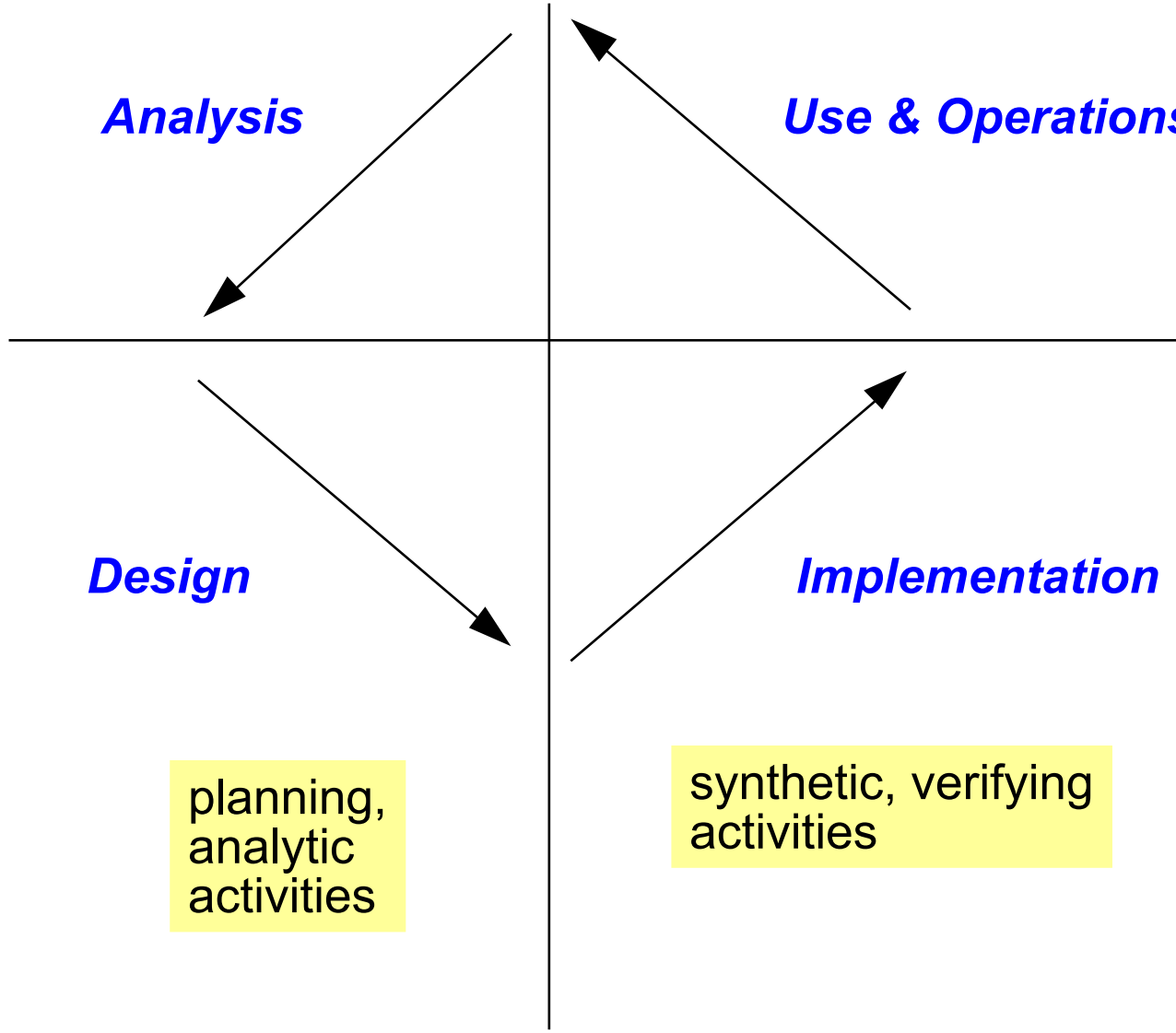
- Each development cycle, for every component on every level, has the *same structure* and consists of
  - *(.A) Analysis:* Define requirements, build model, consult building block (BB) library
  - *(.D) Design:* Specify and construct BB's
  - *(.I) Implementation:* Transform designed BB's to code, test, integrate
  - *(.O) Operational use:* installation, acceptance test, usage, revision

- *Evolutionary development* is supported by:
  - Integration of *operational use* (incl. "maintenance" and revision) into development cycles
  - Further development and re-use of components
  - Dynamic project planning and control based on cycles and activities

Phases of a Simple Object-Oriented Development Cycle

Use environment

Development environment

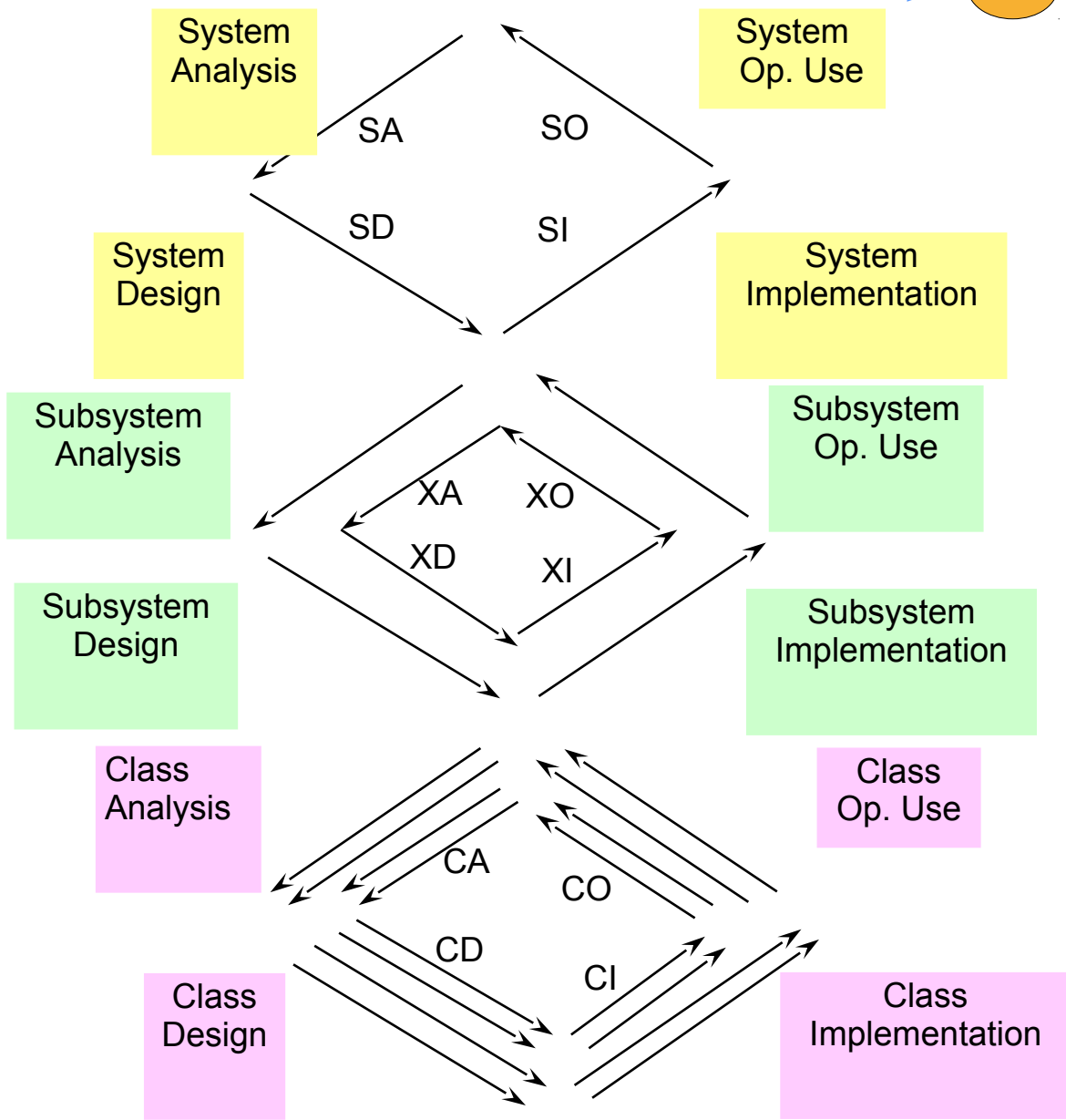


planning, analytic activities

synthetic, verifying activities

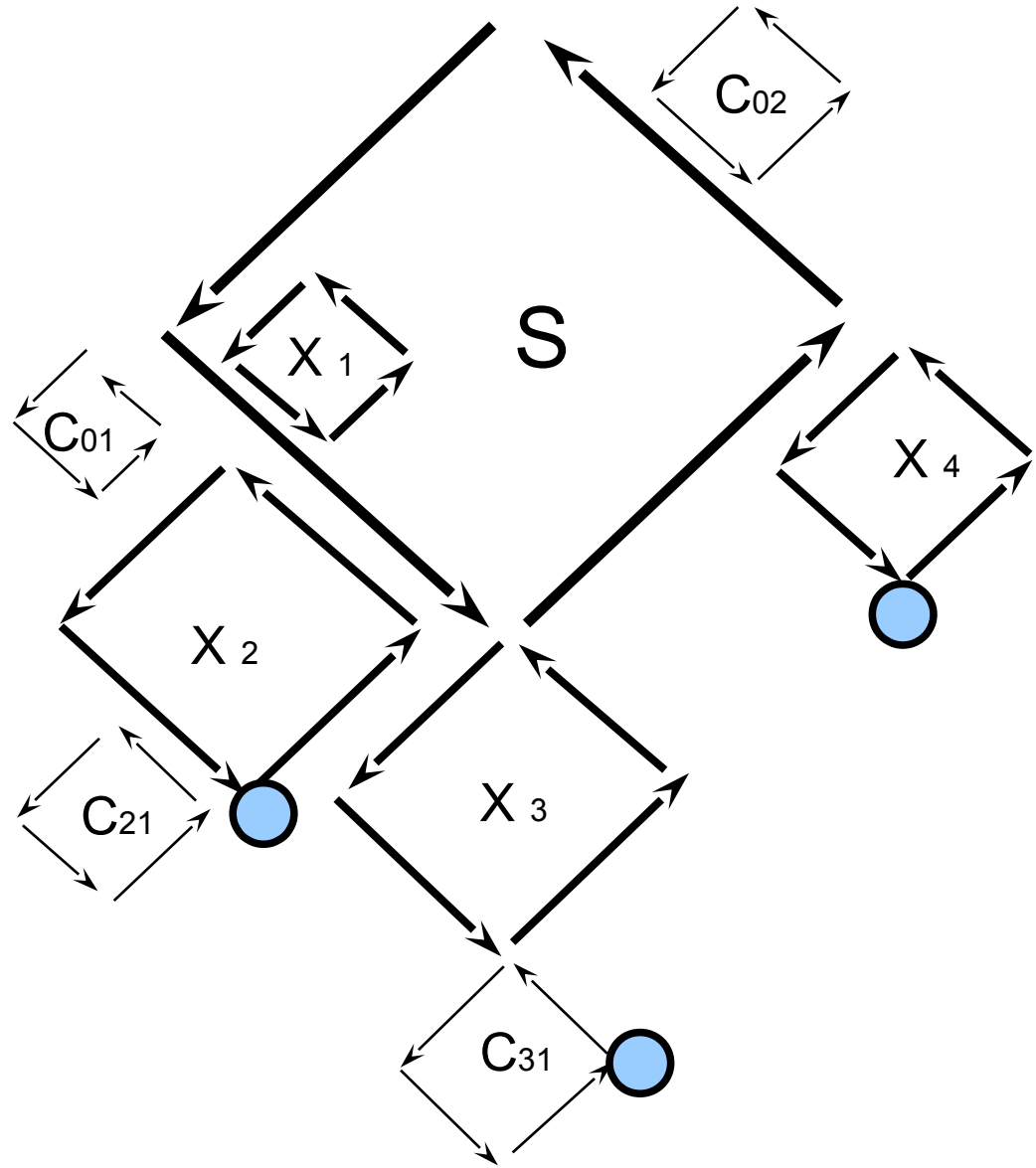
Combining development cycles in a traditional way

- Development phases for the components overlap
- System S has n subsystems X_i
- Subsystem X_i has m classes C_{ij}



Typical EOS-like Process Structure

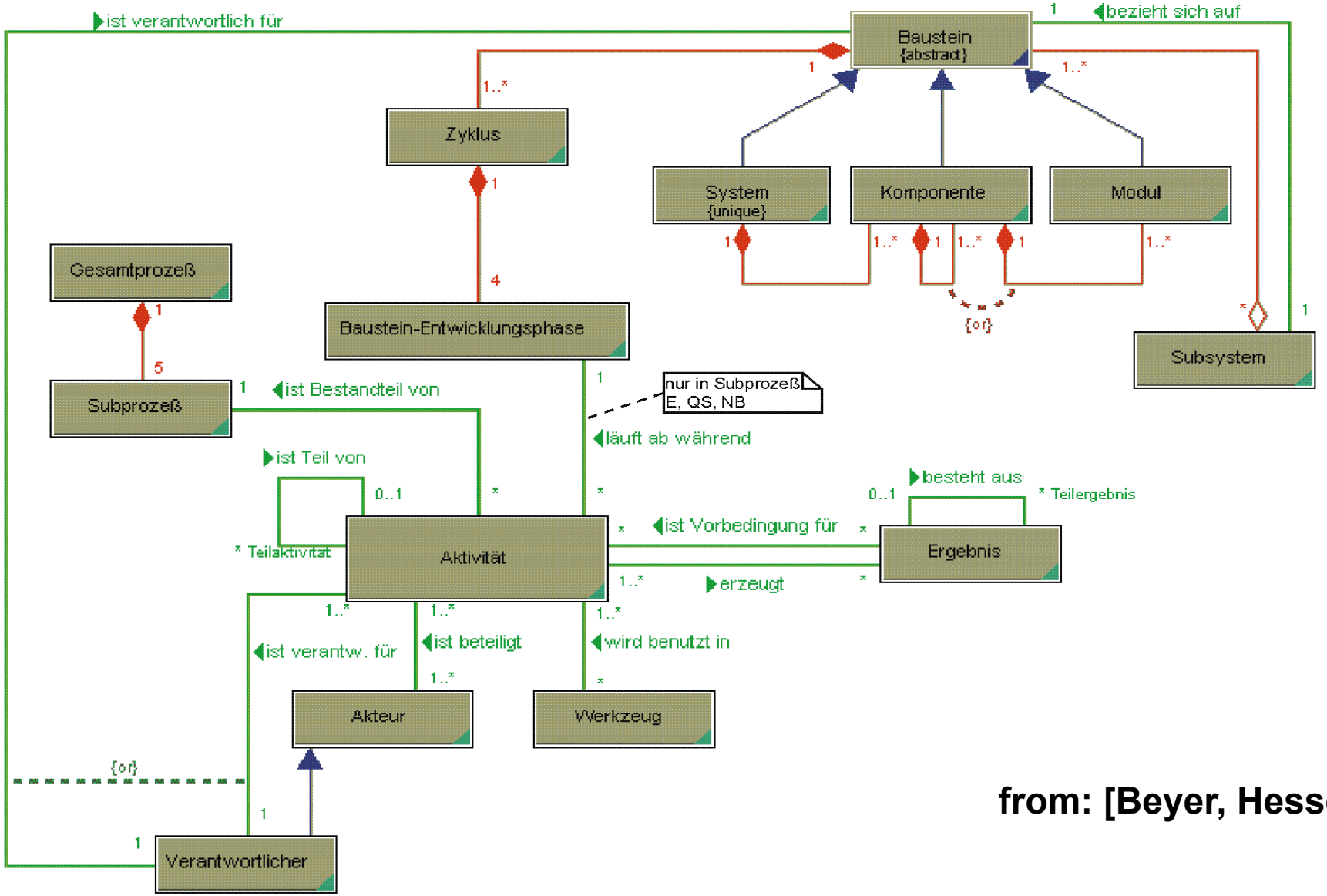
- EOS blends the phases
- k parallel development threads, resp. state tokens
- Development cycles intertwined in time
- If an obstacle appears, thread continues elsewhere
  - E.g., when dependencies to other components appear which were not known beforehand
- Parallel wavefront algorithm over the 3-level tree (bush)



EOS is Agile with Backlogs

- As in SCRUM, there is a backlog of prioritized next activities
- At the completion of an activity (small or large),
  - EOS allows for replanning and reprioritization of the activities to perform (agile development)
  - Costs can be estimated anew (agile cost estimation)
- k parallel development threads
- Very flexible
- Customer can be involved, but need not

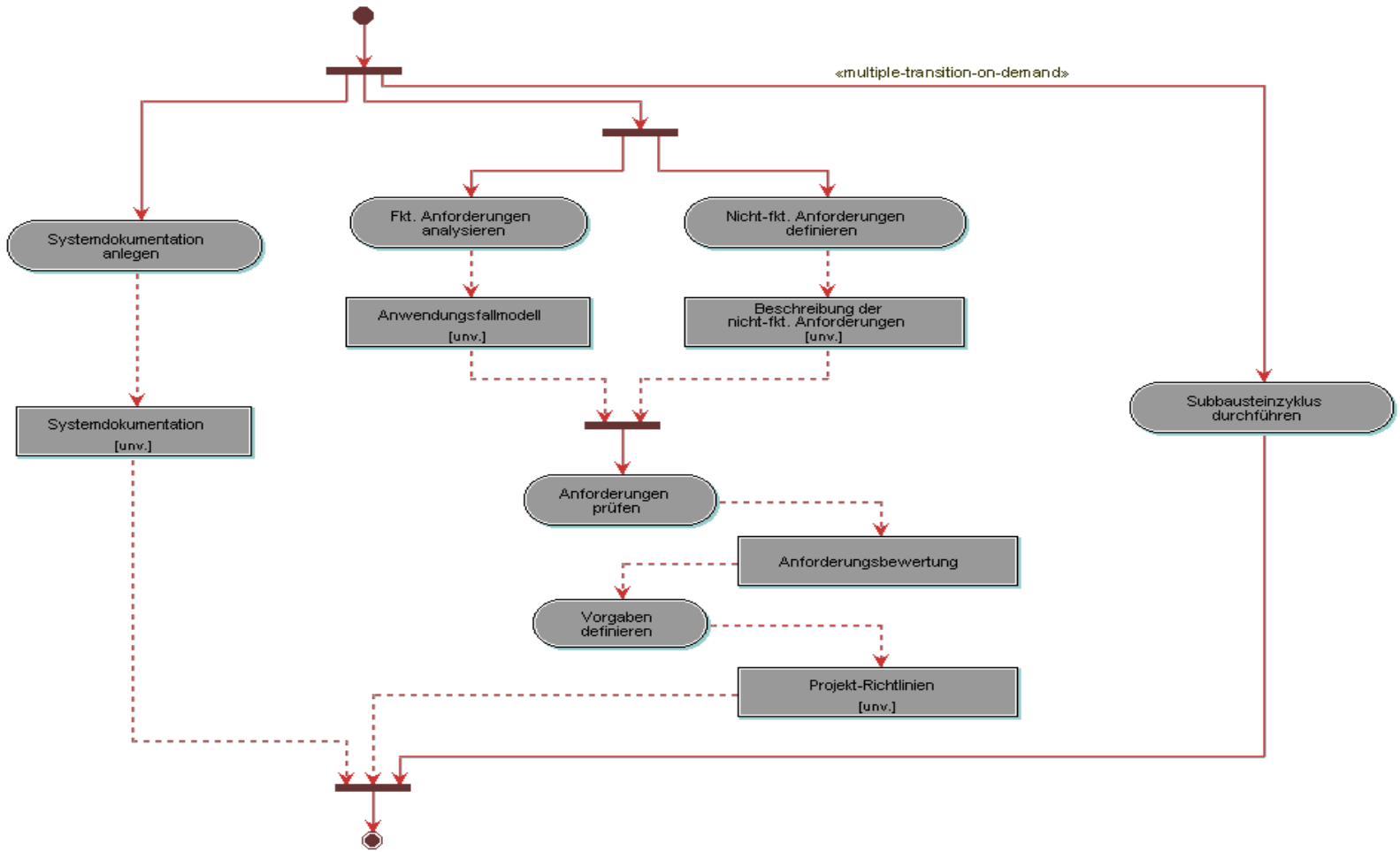
Metamodel for EOS process elements



from: [Beyer, Hesse 2002])

EOS is a Heterogeneous Process

- EOS allows for other process models in each of the four big phases
- Here: UML activity diagram for system analysis (SA) phase



15.2 Managing EOS Projects

Principles of Managing EOS Projects

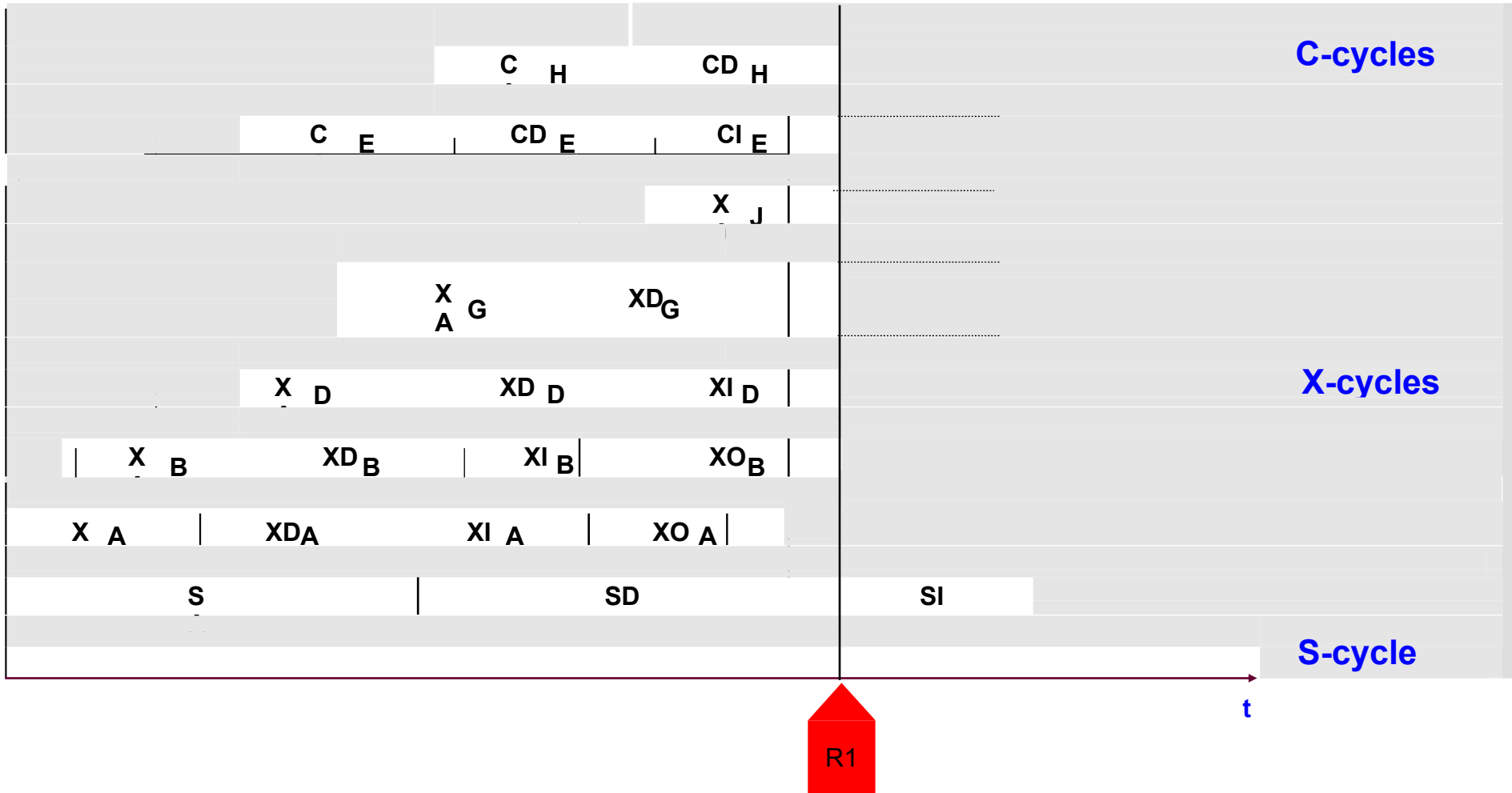
- **Management structure follows system structure (PBS)**
  - Starting point: the EOS hierarchy levels
  - S-cycle: Global planning (project-wide)
  - X-cycles: Detailed steps (e.g. team work packages)
  - C-Cycles: Activities of single developers
- Differenciated units of planning and control (on each level)
  - 1st planning stage: development cycle as a whole
  - 2nd planning stage: phases within cycle
- Dynamic, situative planning (agile)
  - Rather informal planning, "stand by"-management
  - Situation-driven adjustment of plans (backlogs)
  - Frequent plan revisions

Management principles (cont'd)

- "Object oriented" resp. "component-oriented" workpackages
  - Developers are primarily responsible for "objects" and "comnponents" - not for activities
    - Planning refers to objects rather than to activities:
    - on S- and X-level: by development (&support) teams (with users participating whereever necessary)
    - on C-level: by single developers or users
- Transparent planning, reliable plan control
  - Continuous information of teams on the project status
  - Plan revisions at defined points of time (→ revision points)
- Dynamic and adaptable cost and effort estimation
  - based on the EOS process structure, experience data and statistical regression methods [Sarferaz, Hesse 2000]
- EOS is *not* time-boxed, but clearly structured along the PBS
  - If the PBS is stable, but it remains unclear, how long it takes to realize the activities, EOS is a very amenable process

Revision points

- A revision point is a special *milestone,* more differentiated and flexible, because lying between small or large activities
- Revision points allow for replanning and reestimation



Summary and Outlook

- EOS combines the ideas of *evolutionary, agile, component-oriented,* and *object-oriented* software development
- The development process is structured along the PBS
  - by three hierarchy levels (system, component/subsystem, class)
  - by four phases (analyse, design, implement, operate)
- Cycles and phases are linked in a *systematic* and *orthogonal* manner
- Wavefront algorithm for parallel development
- Development cycles are planned and executed *on demand* and in a *dynamic* way
- Project managers can plan and survey the project on every level of *detail* by means of *revision points*