

15. Evolutionary Object-Oriented Software Development (EOS)

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



1

Prof. Dr. rer. nat. Uwe Aßmann
Lehrstuhl Softwaretechnologie
Fakultät Informatik
Technische Universität Dresden
Version 15-1.1, 20.04.12

1 The EOS process model
2 Managing EOS projects

courtesy Prof. Wolfgang Hesse, University of Marburg

Obligatory Literature

- ▶ [Sarferaz] 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, Wetz 94] W. Hesse, F. Wetz: 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

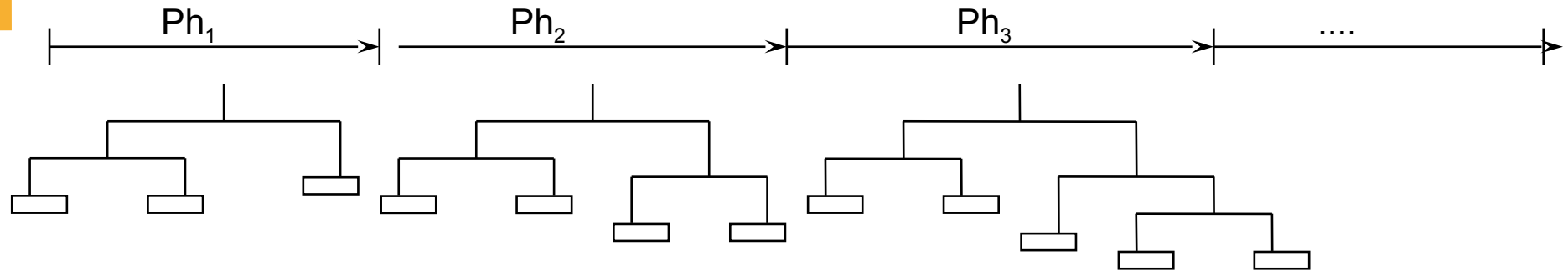
- ▶ [Beyer, Hesse 2002] Use of UML for software process modelling. Internal report, Univ. Marburg 2002
- ▶ [Bittner, Hesse, Schnath 95] U. Bittner, W. Hesse, J. Schnath: Praxis der Software-Entwicklung, Methoden, Werkzeuge, Projektmanagement - Eine Bestandsaufnahme, Oldenbourg 1995
- ▶ [Frese, Hesse 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 Technology, 5th Europ. Workshop EWSPT 96; Springer LNCS 1149, pp. 241-256 (1996)

15.1 The EOS Process Model

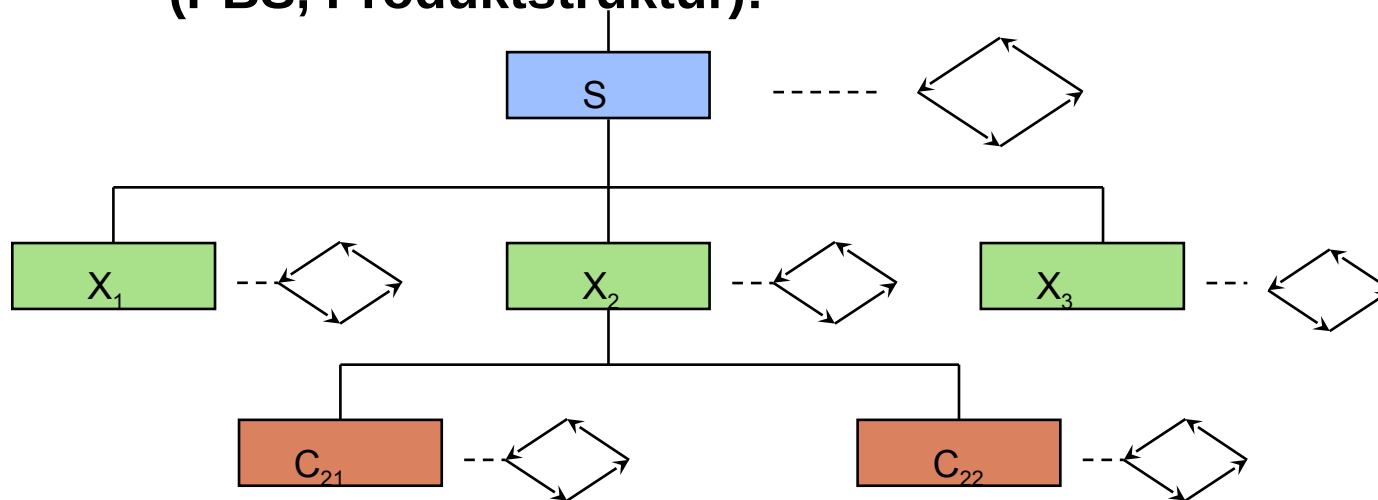
- ▶ Heavy-weight process models are often too bureaucratic and not scalable
 - The aspect of software *evolution* is hardly reflected
 - Planning relies on assumptions and may go wrong
 - *Unforeseen* discoveries, such as change of customer requirements, change the planning
- ▶ Component-oriented, distributed and web-based SW development requires flexible and well-adaptable processes
- ▶ EOS is a *system-structure-based process*, i.e., works if the architecture of the system is clear (standard architecture, standard architectural style, clear component hierarchy)
 - Well-known domain, low innovation project
 - It treats unforeseen dependencies between the components
 - Different availabilities of resources
 - Parallel work possible

Phase-oriented vs. component-oriented process

- ▶ Process in *linear* phases (Phasenmodell):



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



- ▶ S System level
- ▶ X Subsystem level
- ▶ C Class level

Legend:



Building block



Phase or activity

Objects and features of the software process

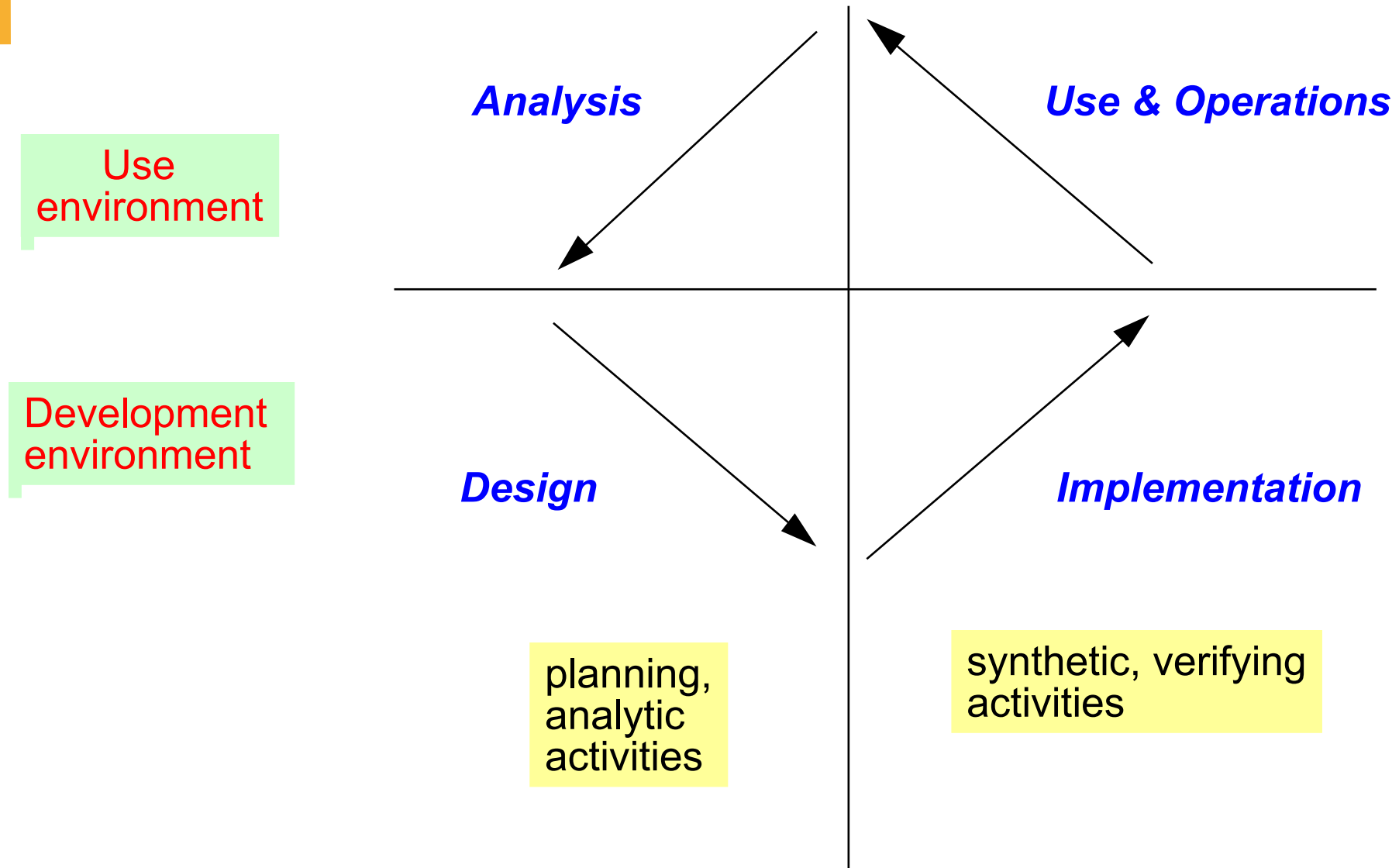
- ▶ The **product breakdown structure (PBS, component hierarchy, Produktstruktur)** is a decomposition of the software product into components
- ▶ In EOS, it is assumed that the PBS is organised in a hierarchy with three level system development structure with three forms of components:
 - S **System** 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 Cycle of Components

7

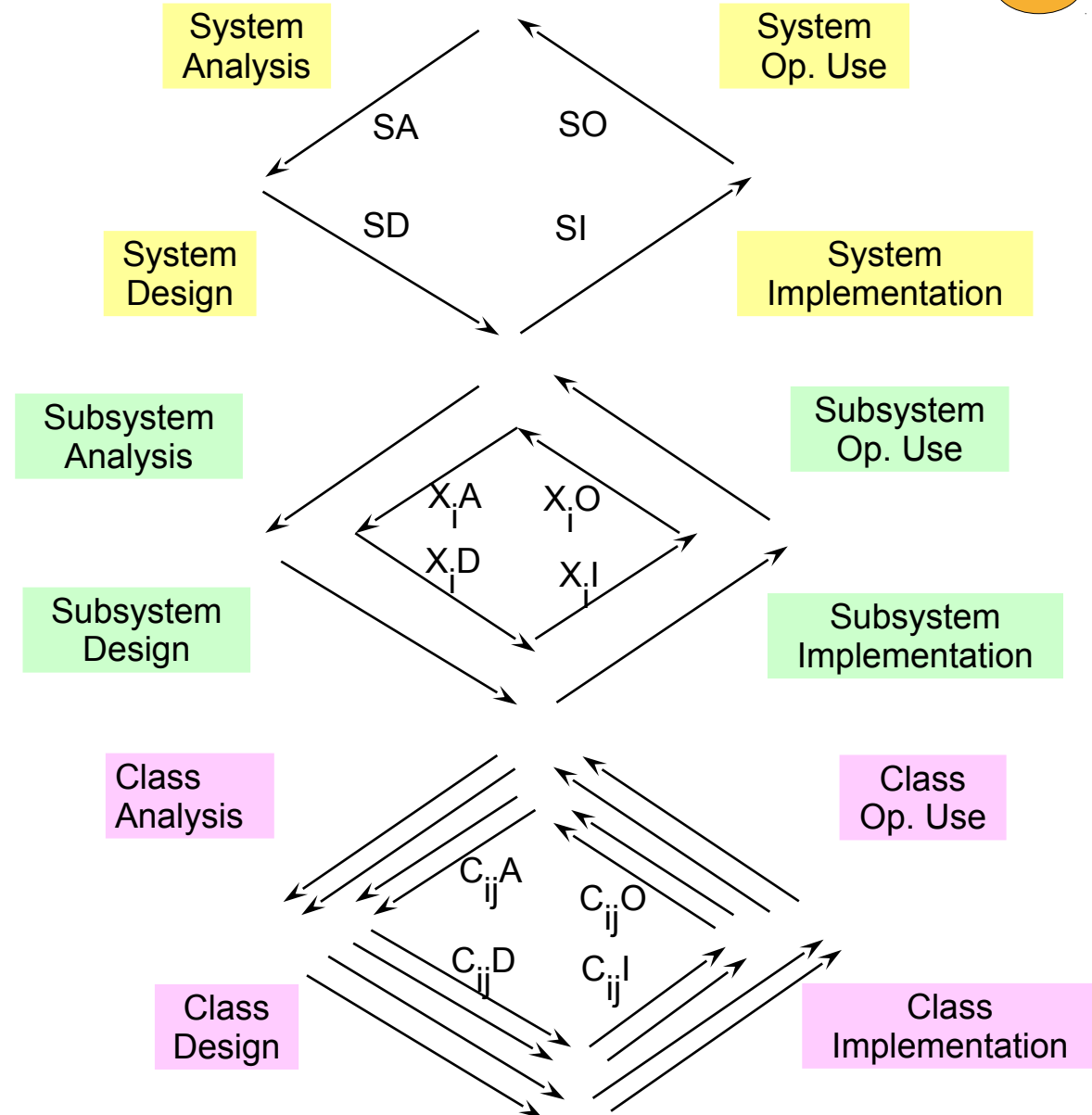
- ▶ Each development cycle, for every component on every level, has the *same structure* and consists of
 - *(.A) Analysis:* Define requirements, build model, consult building block (BB) library
 - *(.D) Design:* Specify and construct BB's
 - *(.I) Implementation:* Transform designed BB's to code, test,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



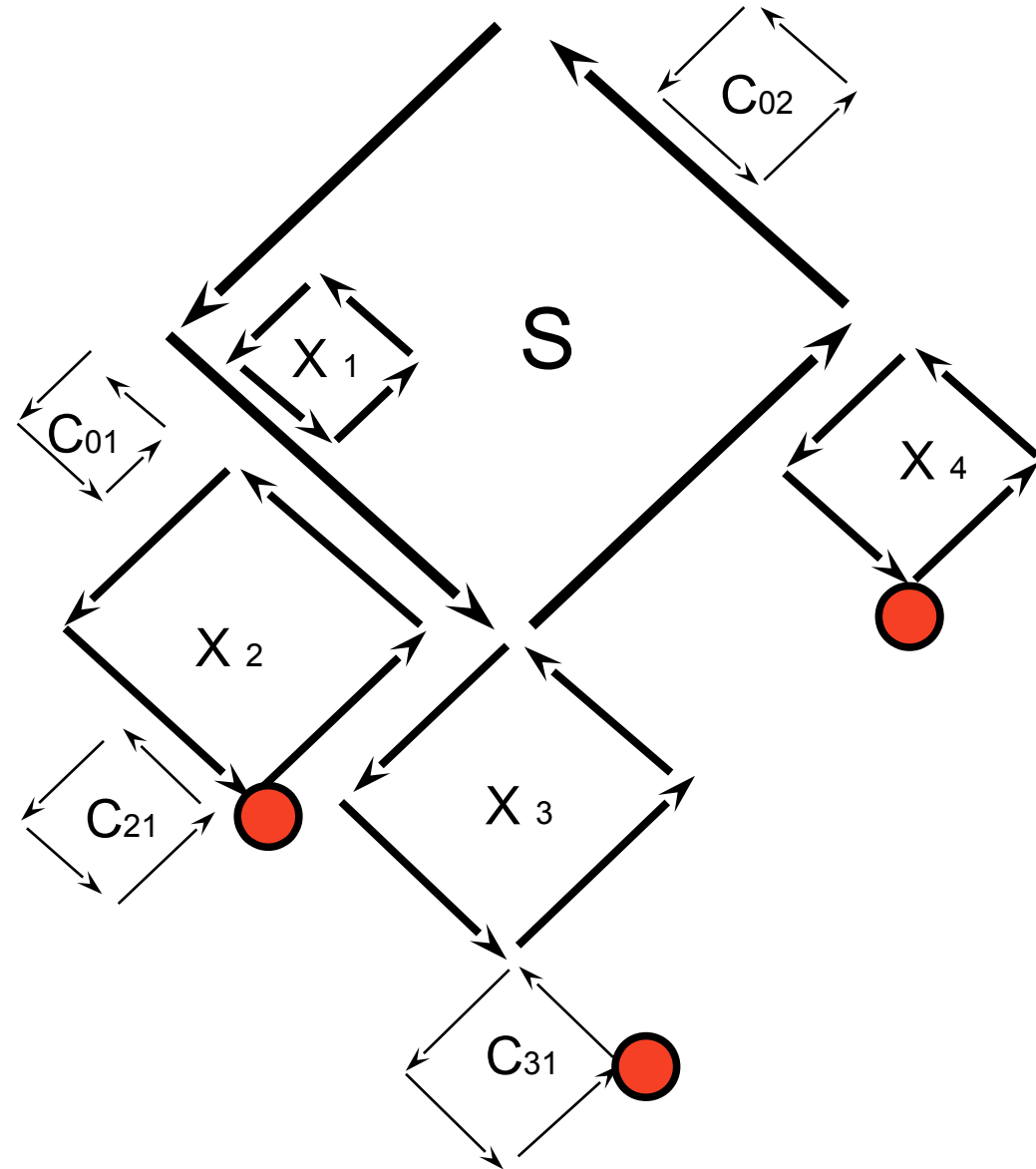
Combining development cycles in a traditional way

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



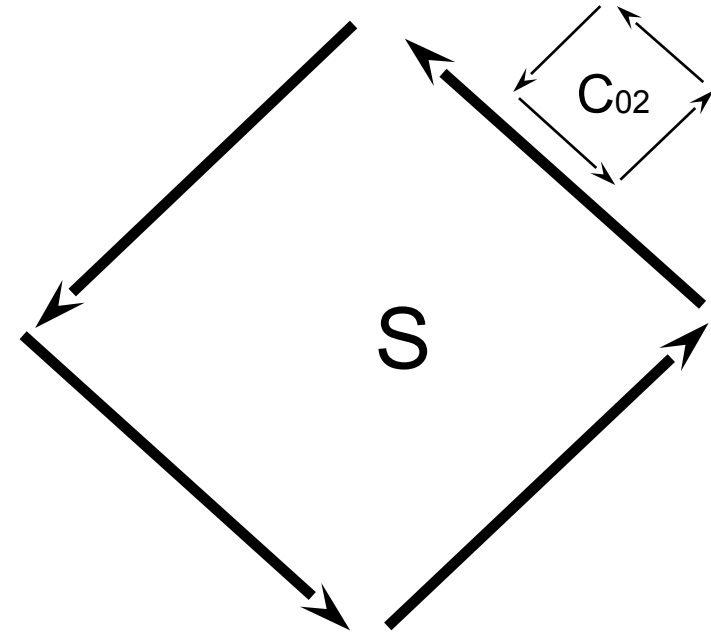
Typical EOS-like Process Structure

- ▶ 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)



Feature Extension During Evolution

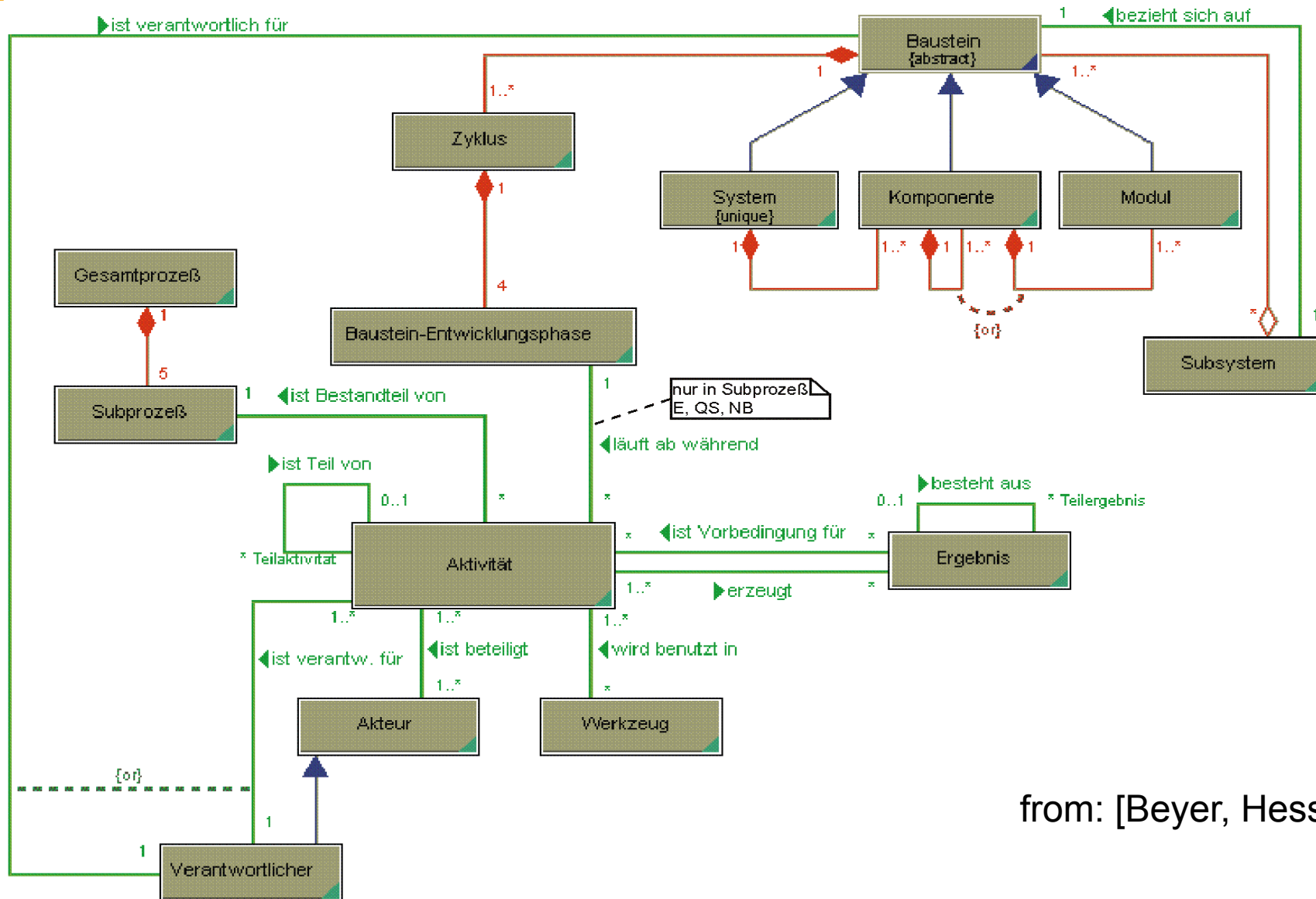
- ▶ With a new class C_{02}



EOS is Agile with Backlogs

- ▶ As in SCRUM, there is a backlog of prioritized next activities
- ▶ k backlogs with $l < k$ parallel development threads
- ▶ 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)
- ▶ Very flexible
- ▶ Customer can be involved, but need not

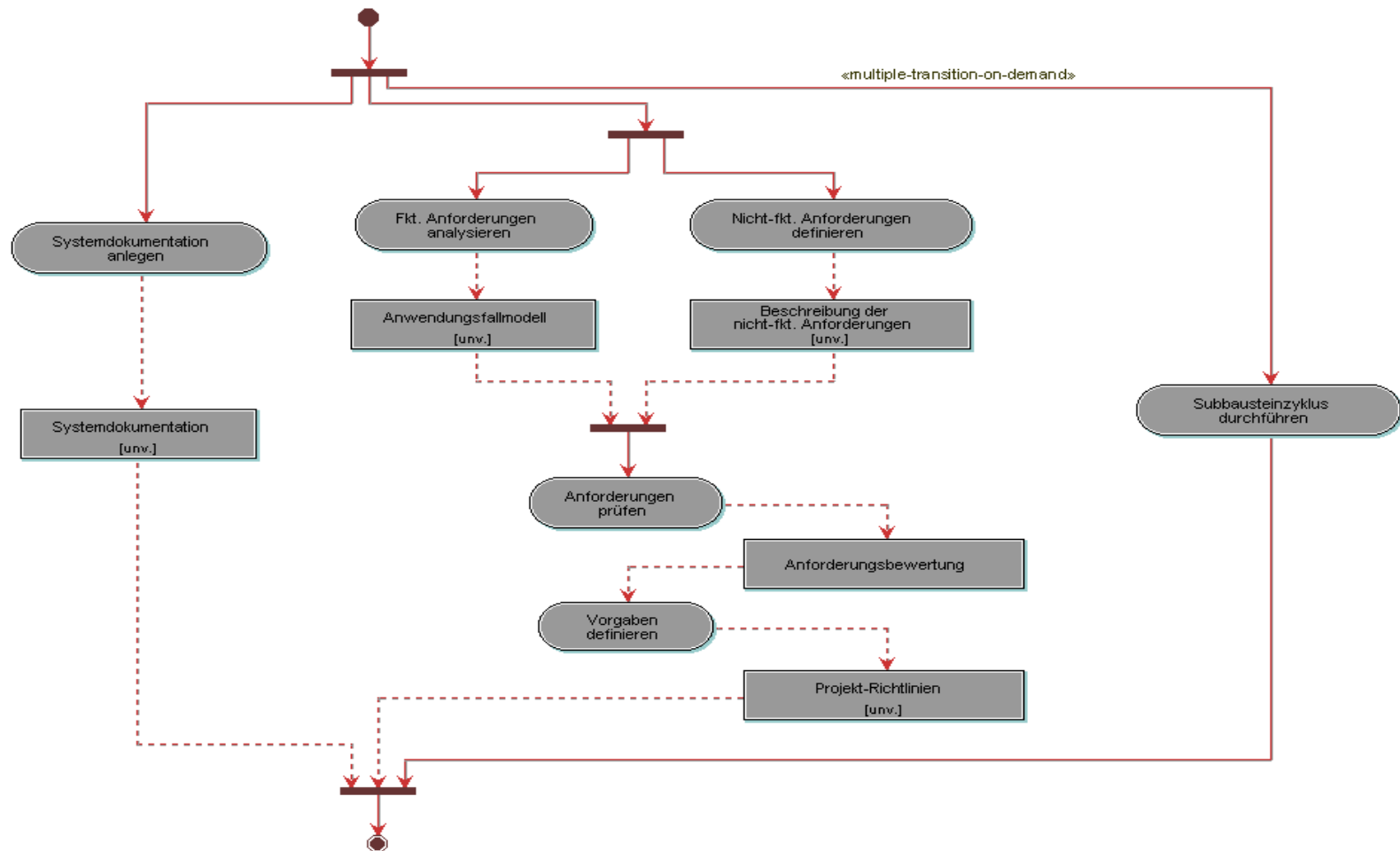
Metamodel for EOS process elements



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



15

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
- ▶ **Differentiated units of planning and control (on each level)**
 - 1st planning stage: plan the 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 “components” - not for activities
 - Planning refers to objects rather than to activities:
 - on S- and X-level: by development (&support) teams (with users participating wherever 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]

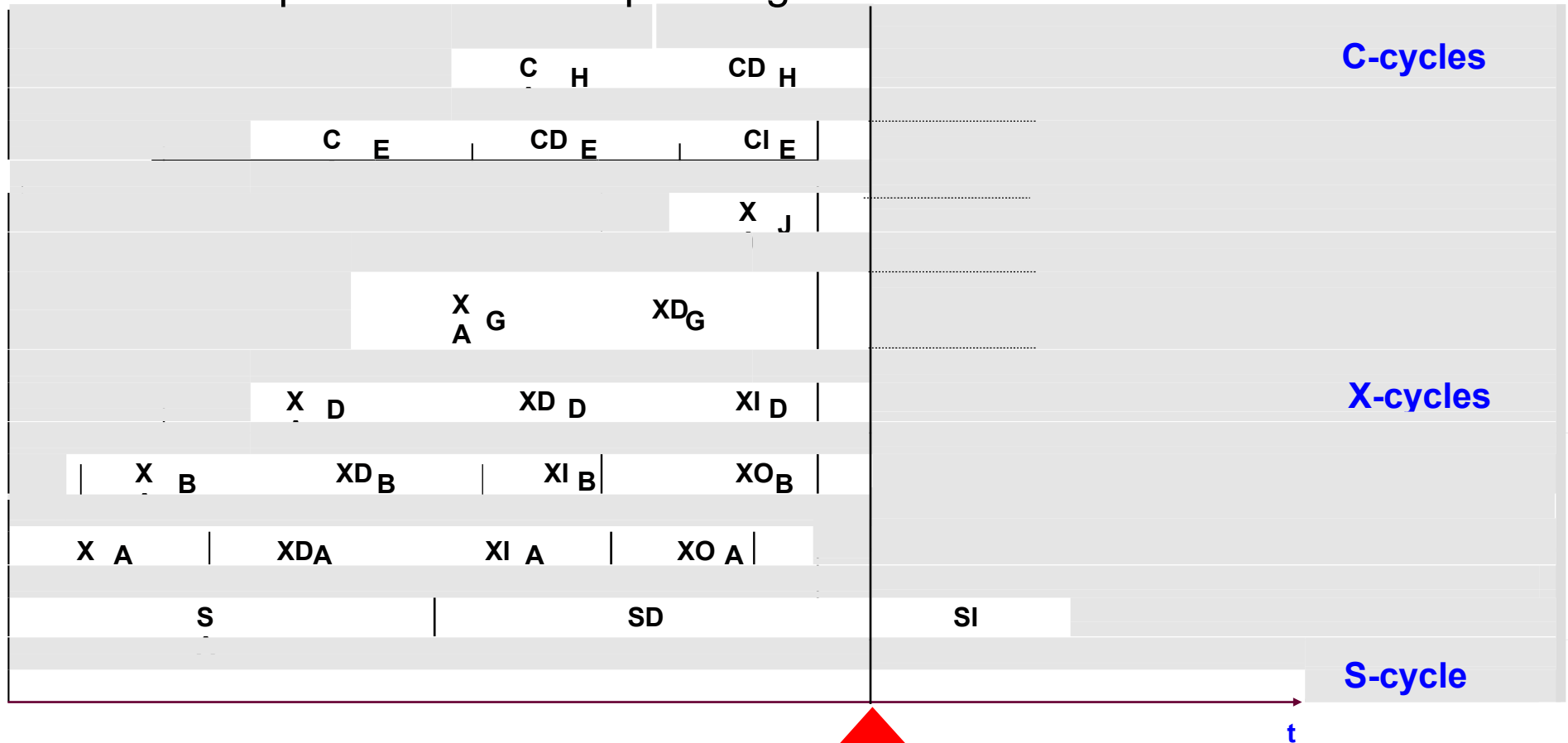
•

EOS and SCRUM

- ▶ The classical 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 amenable
- ▶ **EOS can be combined with time-boxing (tEOS)**
 - k “component backlogs” for all components
 - k “sprint backlogs” for the current sprint in a component

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
- ▶ 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*