Baustein-orientiert statt phasen-zentriert:

Neue Entwicklungsmethoden erfordern neuartige Vorgehensmodelle

Wolfgang Hesse, Philipps-Universität Marburg

1 Traditionelle und OO-Entwicklungsmethoden und ihre Vorgehensmodelle

2 Was heißt "Baustein-orientiert"?

3 EOS: ein Baustein-orientiertes Vorgehensmodell und seine Konsequenzen für das Management

4 Fazit und Ausblick

1 Traditionelle und OO-Entwicklungsmethoden und ihre Vorgehensmodelle

Gegenwärtiger Status:

- Objektorientierung hat zu einem methodischen Umbruch in der Praxis der Software-Entwicklung geführt.

- Kennzeichen gegenüber traditionellen Entwicklungsmethoden:

  - Ziel der meisten Methoden ist eine durchgängige Klassen- und Objektstruktur (möglichst keine Strukturbrüche)

  - *System-Architektur* spielt eine größere, Phasengrenzen überschreitende, oft den gesamten Entwicklungsprozeß umspannende Rolle.

  - Neue Methoden verlangen neue Vorgehensweisen. Bislang dominieren allerdings die Wasserfall-artigen Modelle (Ausnahme: G. Booch)

  - Iteratives Vorgehen wird fast immer empfohlen, allerdings oft wenig spezifiziert und nur innerhalb des Phasenschemas (vgl. z.B. *Rational Objectory Process*).

  - Weiterentwicklung und Wiederverwendung von *System-komponenten* soll gefördert und unterstützt werden. "Komponenten" werden sehr unterschiedlich definiert und spielen für die Prozeß-Definition noch kaum eine Rolle.

- Projekt-Management erfolgt weitgehend traditionell. Phasenmodelle beherrschen nach wie vor das Feld. Sie sind jedoch für OO-Entwicklungsprojekte nicht mehr adäquat.

Vorgehensmodelle für OO-Entwicklungsprojekte sollten ...

- die Software-Architektur berücksichtigen,

- Iteration auf mehreren Ebenen ermöglichen,

- die selbständige Entwicklung, Wiederverwendung, und Weiterentwicklung von *Komponenten* und anderen *Bausteinen* unterstützen.

Dafür ist ein *Wechsel des Leitbildes* notwendig:

- Im Mittelpunkt der Betrachtung sollten nicht länger die *Phasen*, sondern die *Software-Architektur* stehen.

- Elemente der Architektur - z.B. Komponenten, Klassen und Subsysteme müssen klar gegeneinander abgegrenzt werden

- Bestimmten Elementen der Architektur sollten auch *eigene Entwicklungstätigkeiten und -prozesse* (und ggf. entsprechende Management-Tätigkeiten) zugeordnet sein.

2 Was heißt "Baustein-orientiert"?

Software-Baustein: jedes zusammenhängende, in sich abgeschlossene Stück Software, das einem bestimmten definierten Zweck dient, klar abgegrenzte Schnittstellen hat und - relativ zu den in den Schnittstellen getroffenen Vereinbarungen - unabhängig von anderen Bausteinen oder Systemen entwickelt werden kann.

Konsequenz: Bausteine haben eine eigene Entwicklungsgeschichte und können losgelöst von einem bestimmten Kontext betrachtet werden.

Gesichtspunkte zur Bildung von Bausteinen:

- ***Entwurfs-Bausteine:*** Zusammenfassungen von logisch zusammengehörigen Software-Einheiten zum Zwecke des gemeinsamen Entwurfs - oft auch *Komponenten* oder *Pakete (packages)* genannt;
- ***Darstellungs-Bausteine:*** Zusammenfassungen von mehreren Struktureinheiten eines Diagramms zu einer aggregierten Einheit zum Zwecke der kompakten Darstellung;
- ***Organisatorische Bausteine:*** Einheiten, die einer Person oder einem Team zu gemeinsamen Bearbeitung übergeben werden,
- ***Integrations-und Test-Bausteine:*** Zusammenfassung mehrerer Module oder Klassen, um diese gemeinsam zum Ablauf zu bringen und zu testen - oft auch "Subsysteme" genannt;
- ***Verteilungs-Bausteine:*** Zusammenfassungen von Software-Einheiten, die gemeinsam auf einer Plattform zum Ablauf gebracht werden;
- ***Verwaltungs-Bausteine:*** Zusammenfassungen von Software-Einheiten in einer Klassen- oder Projektbibliothek zum Zwecke der Dokumentation, Archivierung und künftigen Verwendung
- ***Wiederverwendbare Bausteine:*** Zusammenfassungen von Software-Einheiten zum Zwecke der Wiederverwendung.

UML definiert z.B. zwei Arten von Bausteinen:
Darstellungs-Bausteine *(packages)* und Verteilungs-Bausteine *(components)* [UML 97].

Baustein-Orientierung: Erweiterung des objekt-orientierten Ansatzes auf Bausteine, die *mehrere Klassen oder Objekte* umfassen.

D.h. solche Zusammenfassungen

- werden explizit als Bausteine (etwa der oben angeführten Arten) eingeführt und spezifiziert,

- werden weitgehend unabhängig voneinander entwickelt,

- können über bestimmte Beziehungen (z.B. Vererbung, Aggregation oder Assoziation) miteinander zusammenhängen,

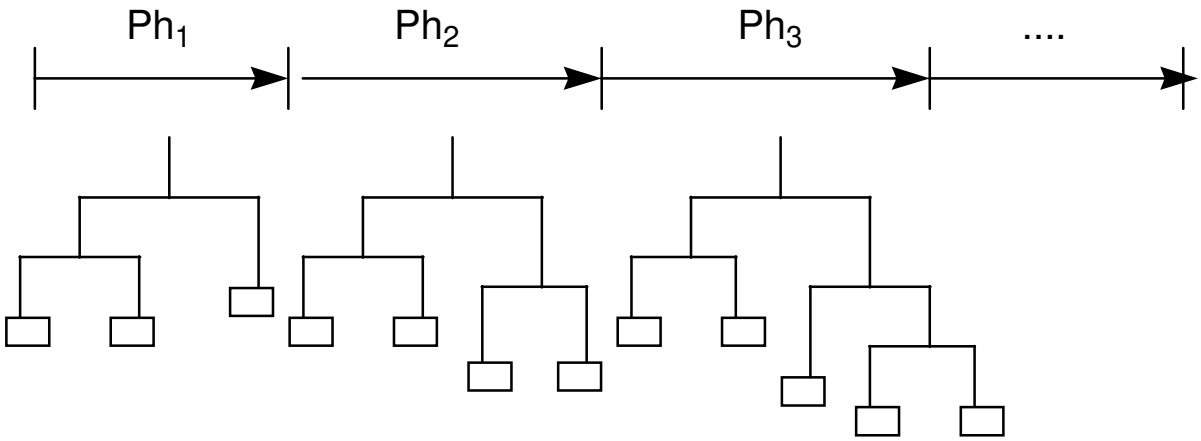
- können in unterschiedlichen Kontexten verwendet werden.

Baustein-orientiertes Vorgehen: ein Vorgehen, das beschreibbar ist in der Form von Prozessen, die sich (in ihrer Mehrzahl) den Bausteinen der System-Architektur zuordnen lassen.

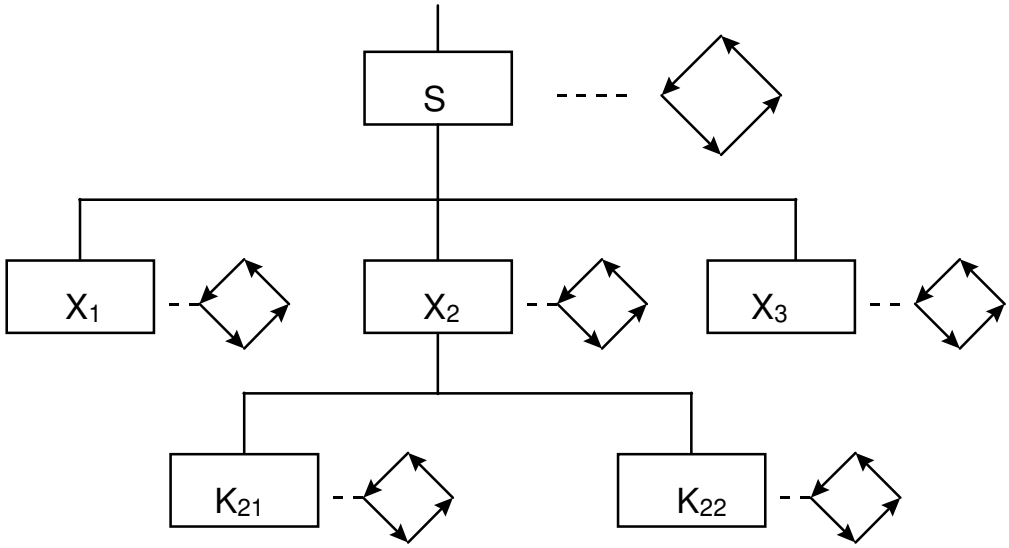
Konsequenz für ein Baustein-orientiertes Vorgehensmodell:

Oberstes Gliederungsprinzip ist *nicht* die Einteilung in *Phasen*, sondern die *Zuordnung von Prozessen zu den Bausteinen* der Entwicklung.

Folgerung für das Management: Projekte müssen baustein-orientiert (satt phasen-orientiert) geplant, organisiert, gesteuert und koordiniert werden.



Phasen-orientiertes und ...



... Baustein-orientiertes Vorgehen

Legende: Baustein → Phase bzw. Tätigkeit

3 EOS: ein Baustein-orientiertes Vorgehensmodell und seine Konsequenzen für das Management

Allgemeines Prozeßmodell:

Software-Entwicklung vollzieht sich im Zusammenspiel verschiedener, i.a. nebenläufiger Prozesse, die sich einzelnen Gruppen von Beteiligten *(Rollen)* zuordnen lassen:

- Entwickler-Prozesse,
- Management-Prozesse,
- Qualitätssicherungs- Prozesse,
- Unterstützungs-Prozesse (z.B. durch Konfigurations-Management und Bibliotheksverwaltung),
- Benutzungs-Prozesse.

Struktur der Entwicklungsprozesse:

Baustein-orientiert , d.h. Entwicklungs-Tätigkeiten erfolgen in *Zyklen*, die grundsätzlich Bausteinen zugeordnet werden.

EOS-Zyklen (entspr. den vorgesehenen Arten von Bausteinen):

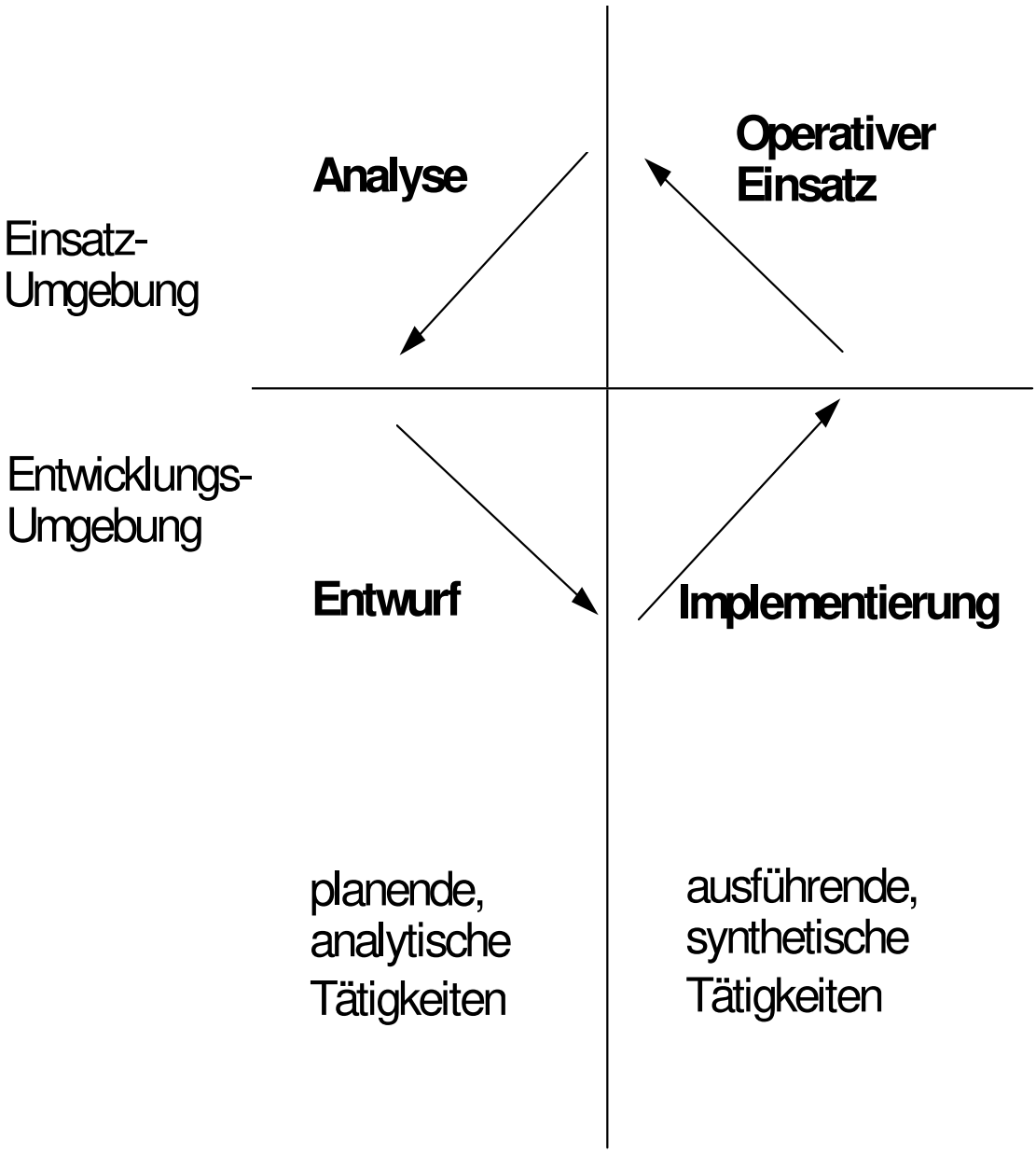
- ***S-Zyklus:*** zur Entwicklung gesamter, i.a aus vielen Bausteinen bestehender *Systeme*

- ***X-Zyklus:*** zur Entwicklung von Entwurfs-Bausteinen, bei EOS *Komponenten* genannt,

- ***K-Zyklus:*** zur Entwicklung von *Klassen* im Sinne der OO-Methoden,

- ***SS-Zyklus:*** zur Entwicklung von Integrations-Bausteinen, bei EOS *Subsysteme* genannt.

Tätigkeiten eines Entwicklungszyklus



Feinstruktur der Entwicklungszyklen (für alle Zyklen analog):

Vier Grundtätigkeiten:

- Analyse
- Entwurf
- Implementierung
- Operationaler Einsatz

Konsequenzen für das Vorgehen und für ein *Baustein-orientiertes Projekt-Management:*

- Jede Baustein-Entwicklung ist ein eigenständiger (Teil-) Prozeß.

  Er läuft in definierten Tätigkeitsschritten, möglicherweise in Iterationen ab.

  Ihm sind Personal (Bearbeiter und dafür Verantwortliche), Ressourcen, Pläne, Termine, Qualitäts-Maßnahmen etc. zugeordnet.

- Jeder Baustein-Zyklus enthält die oben aufgeführten Grund-Entwicklungstätigkeiten. D.h. im besonderen:

  Der Einsatz des Produkts (in einer Test- oder Zielumgebung) gehört mit zum Prozeß. Damit ist auch die Benutzung in das Prozeßmodell integriert.

- Für das Management sind Baustein-Entwicklungen grundsätzlich *selbständige Planungseinheiten* mit eigenen Resourcen, Plänen, Terminen, etc..

  Das gilt bis zu einer für Management-Zwecke sinnvollen Granularität.

  Viele Bausteine entstehen erst *während* der Entwicklung → die entsprechenden Planungs-Aktivitäten können nicht a priori (bei Projektbeginn) durchgeführt werden, sondern müssen *dynamisch im Zuge der fortschreitenden Definition von Bausteinen* erfolgen.

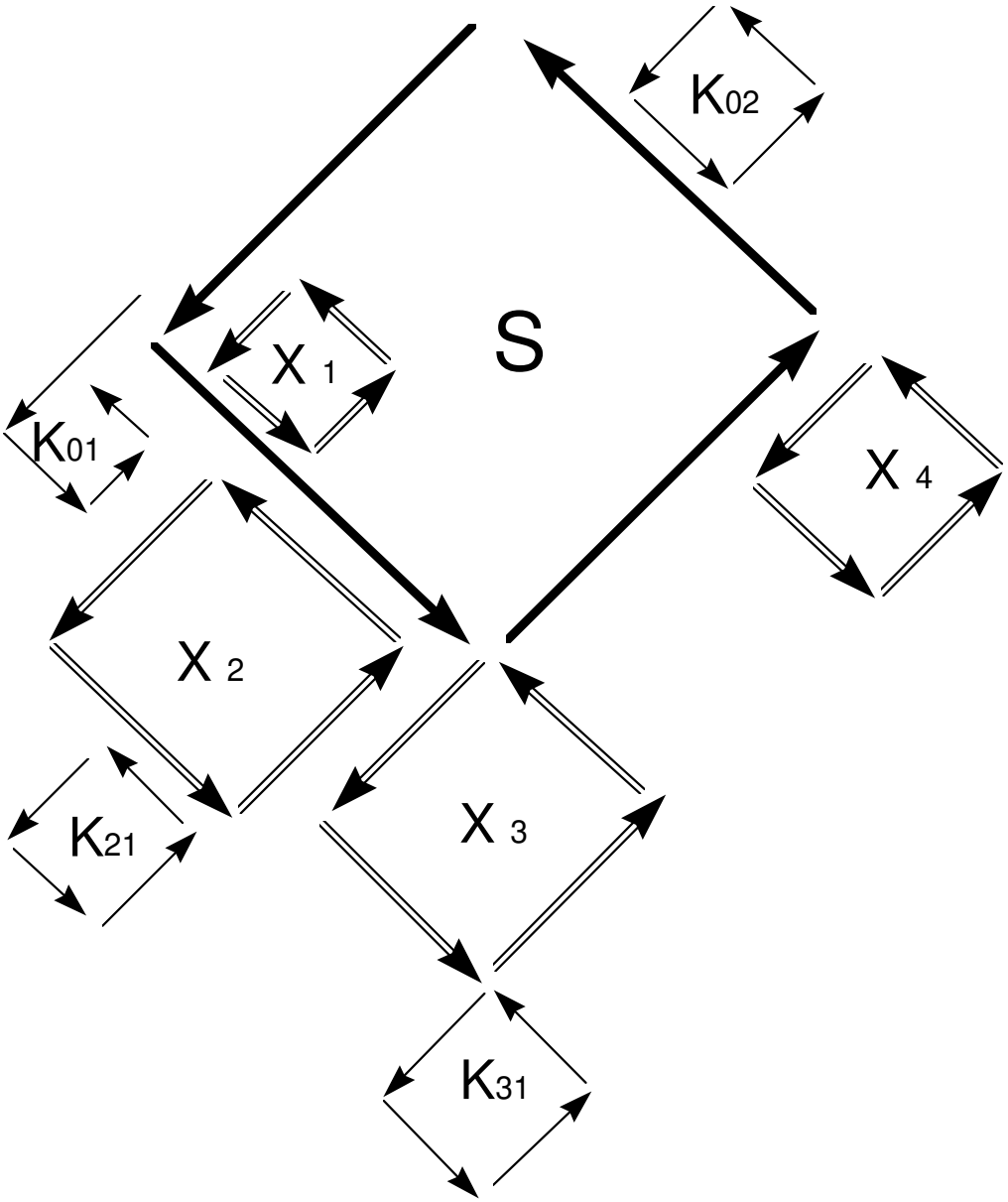
Baustein-orientiertes Vorgehen schließt ein dynamisches Projekt-Management ein.

- Dynamisches Projekt-Management erfordert hohen Koordinationsaufwand. In der Regel laufen mehrere (i.a. in unterschiedlichen Zuständen befindliche) Baustein-Entwicklungszyklen parallel ab.

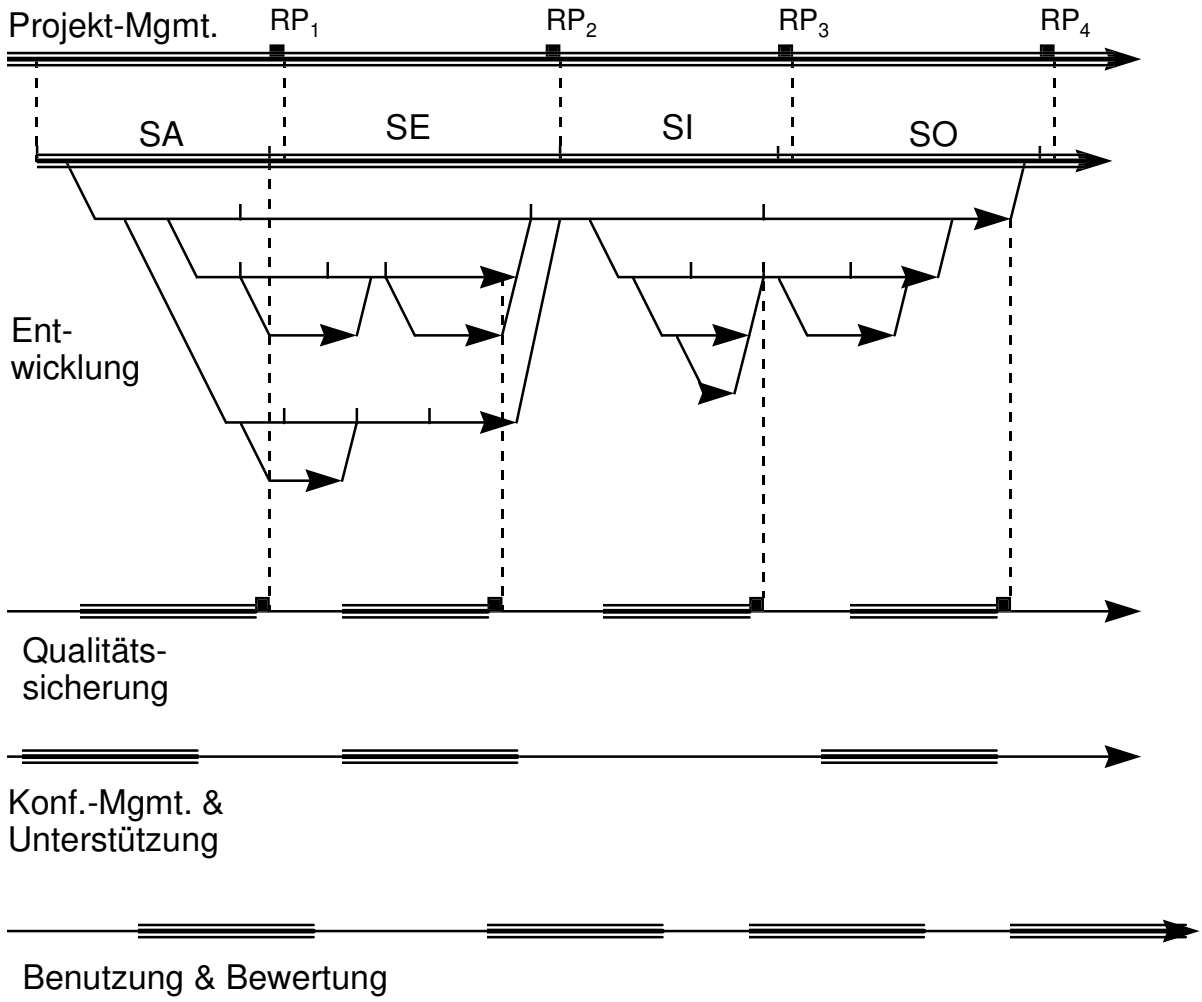
  Der (immer vorhandene) erhebliche Steuerungs- und Koordinationsbedarf wird in den herkömmlichen Wasserfall-Modellen hinter einer realitätsfernen Phasenstrukur versteckt.

  EOS bietet zur Steuerung und Koordination paralleler Baustein-Entwickungszyklen das Konzept der *Revisionspunkte* an.

Zeitlich verzahnte Tätigkeitszyklen



Allgemeines Prozeßmodell



SA: System-Analyse
SE: System-Entwurf
SI: System-implementierung
SO: System-Einsatz

Rp_i: Revisionspunkt i

4 Fazit und Ausblick

Wird es ein Standard-Vorgehensmodell für die OO-Entwicklung (analog zum traditionellen Wasserfall-Modell) geben?

Einschränkung der ursprünglich geplanten *Unified Method* auf eine *Unified Modelling Language* spricht eher für das Gegenteil.

Einflußfaktoren, die dem entgegenstehen:

- Unterschiedliche Zielsetzungen, Anforderungen, Anwendungsgebiete, Typen von Software
- Unterschiedliche Menschen, Organisationen, Traditionen und "kulturelle" Hintergründe.

EOS kann (und will) nur eines von vielen möglichen Vorgehensmodellen sein.

Das Leitbild der *Baustein-Orientierung* könnte jedoch die nächste Generation von Vorgehensmodellen mit bestimmen.

Für das Projekt-Management ergeben sich daraus erhebliche Konsequenzen und die Notwendigkeit zum Umdenken.

Literaturhinweise

[Boo 94] G. Booch: *Object-Oriented Analysis and Design with Applications;* Second Edition, Benjamin/Cummings Publ. Comp. 1994

[C-Y 91] P. Coad, E. Yourdon: *Object-oriented analysis;* 2nd ed., Yourdon Press 1991

[FRS 89] Ch. Floyd, F.-M. Reisin, G. Schmidt: STEPS to software development with users; in: *C. Ghezzi, J. McDermid (eds.): ESEC '89, Second European Software Engineering Conference.* LNCS 387, pp. 48-64; Springer 1989

[Hes 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 European Workshop, EWSPT 96, Springer LNCS 1149,* pp. 241-256 (1996)

[Hes 97a] W. Hesse: Wie evolutionär sind die objekt-orientierten Analysemethoden? Ein kritischer Vergleich; *Informatik-Spektrum 20.1,* S. 21-28 (1997).

[Hes 97b] 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,* St. Petersburg 1997

[Hes 97c] W. Hesse: Improving the software process guided by the EOS model. In: *Proc. SPI '97 European Conference on Software Process Improvement*. Barcelona 1997

[H-W 94] W. Hesse, F. Weltz: Projektmanagement für evolutionäre Software-Entwicklung; *Information Management 3/94***,** pp. 20-33, (1994)

[Jac 93] I. Jacobson: *Object-Oriented Software Engineering - A Use Case Driven Approach;* Revised Printing, Addison- Wesley 1993

[Rum 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen*: Object-Oriented Modelling and Design*; Prentice Hall 1991

[UML 97] G. Booch, J. Rumbaugh, I. Jacobson: *Unified Modelling Language (UML) for Object-Oriented Development* (Documentation Set - Version 1.0), Rational Software Corp. 1997

Adresse des Autors:

Prof. Dr. Wolfgang Hesse, FB Mathematik und Informatik, Universität Marburg, Hans Meerwein-Str., D-35032 Marburg

Tel.: +49-6421-281515, Fax: +49-6421-285419, email: hesse@informatik.uni-marburg.de