Vorlesung Software Engineering

Foliensatz Nr. 2 (20.10.11)

Arbeitsgruppe Software Engineering
Prof. Elke Pulvermüller

Universität Osnabrück
Institut für Informatik, Fachbereich Mathematik / Informatik
Raum 31/318, Albrechtstr. 28, D-49069 Osnabrück

elke.pulvermueller@informatik.uni-osnabrueck.de

http://www.inf.uos.de/se

Sprechstunde: mittwochs 14 – 15 und n.V.

Inhalt

Inhalt

2.1 Phasen der Software-Entwicklung

Die Softwareerstellung gliedert sich in folgende Phasen:

1. Anforderungsphase:
- **Analyse der Kundenwünsche**
- **Exploration des Anwendungsgebietes (Domäne)**
- **Erarbeitung der Anforderungen**

2. Spezifikationsphase
- **Präzise Festlegung der Leistungen**
- **Spezifikationsdokument (→ Kontrakt)**

3. Planungsphase
- **Entwurf der Projektdurchführung**
- **Projektmanagementplan**
- **Zeitdauer, Aufwandsabschätzung**

4. Entwurfsphase

- **Hardware-/Softwareentwurf**
- **Systemarchitektur (Konzeption im Großen, Zerlegung in Module)**
- **Detailentwurf (Implementierung der Module)**

5. Implementierungsphase

- **Implementierung der Programmkomponenten (arbeitsteilig)**
- **Testen der Komponenten (Modultest)**

6. Integrationsphase

- **Integration der Komponenten**
- **Test des Gesamtsystems (Integrations- und Systemtest)**
- **Auslieferung an den Kunden**

2.1 Phasen der Software-Entwicklung

7. Wartungsphase

- **Betrieb beim Kunden**
- **Fehlerbeseitigung**
- **Verbesserung (Effizienz)**
- **Erweiterung**
- **Anpassung**

8. Rückzugsphase

- **Einstellung der Nutzung**
- **Datensicherung/-übertragung**

2.1 Phasen der Software-Entwicklung: Kosten

Kostenaufteilung (Schätzung)

- **Anforderungsphase: 3%**
- **Spezifikationsphase: 4%**
- **Planungsphase: 2%**
- **Entwurfsphase: 6%**
- **Implementierungsphase: 12% (5% Codierung, 7% Test)**
- **Integrationsphase: 6%**
- **Wartungsphase: 67%**
- **Rückzugsphase ?**

Die Kostenaufteilung hängt vom Bereich ab (z.B. Betriebssystem oder Informationssystem)

2.1 Phasen der Software-Entwicklung: Randbedingungen

Randbedingungen

Kunde: Kontinuierlich wechselnde/wachsende Anforderungen durch

- Änderungen von Betriebsabläufen, Organisationsformen und Verfahren
- Änderung gesetzlicher Bestimmungen
- steigende Integration der EDV in Betriebsabläufe

Umgebung: Kontinuierlicher Wandel der Software-/Hardwareumgebung

- Änderung der Software (Betriebssysteme, Anwendungsprogramme)
- Änderung/Erweiterung der Hardware (Speicherausbau, schnellere Prozessoren)
- Änderung der Rechnernetze (Neuanschlüsse, Ortswechsel, Restrukturierung)

2.2 Grundlegende Prinzipien

Grundlegende Prinzipien des Software Engineering im Umgang mit der komplexen Realität:

- **Abstraktion (Abstraction)** (z.B. durch Modellbildung, Datenabstraktion mit ADT oder im Architekturentwurf)
  und Konkretisierung (Verbalisierung, Implementierung)

- **Zerlegung (Partitioning), Dekomposition, Divide et Impera**
  → **Strukturierung, Separation of Concerns, Hierarchisierung, Modularisierung, Lokalität, Geheimnisprinzip**

- **Perspektivenbildung (Projection), Sichten**
  (z.B. durch eine Vielzahl unterschiedlicher Modellarten)

- **Wiederholung und Wiederverwendung (z.B. durch Patterns, Paradigmen)**

- **Automatisierung** (z.B. mit automatischen Tests, Generatoren)

- **Iteration und Evolution / Restrukturierung**

2.3 Prinzip Modularisierung

Prinzip Modularisierung:

- Eine bekannte Technik zur Beherrschung von Komplexität ist der antike Grundsatz: ***Devide et Impera* (Teile und herrsche)** [Dij79]

Zerlegung großer Systeme
in kleinere und übersichtlichere Komponenten bzw. Teile

Das Vorgehen heißt auch: **Dekomposition** [Boo94]

"Software engineers should take a look at the hardware industry and establish a software component subindustry.
The produced software components should be tailored to specific needs but be reusable in many software systems." [McI76]

[Dij79] E. Dijkstra. Programming Considered as a Human Activity, *Classics in Software Engineering*, Yourdon Press, New York, NY, 1979

[Boo94] G. Booch. *Object-Oriented Analysis and Design*, snd. Edition, Benjamin/Cummings, Redwood City, CA, 1994

[McI76] M.D. McIlroy. Mass-produced software components. In J.M. Buxton, P. Maur, und B. Randell, Editoren, *Software Engineering Concepts and Techniques*, Proceedings of 1968 North Atlantic Treaty Organisation (NATO) Conference on Software Engineering Garmisch-Partenkirchen, Seiten 88 -- 98. New York, 1976

2.3 Prinzip Modularisierung: Geheimnisprinzip

Module und das Geheimnisprinzip:

- Grundlegende Gedanken hinter Modularisierung: **Geheimnisprinzip (*Information Hiding Principle*)** [Par72]**:**

  Ein Modul kommuniziert mit seiner Umwelt nur über eine klar definierte Schnittstelle.

  Das Zusammensetzen von Modulen verlangt keine Kenntnis ihrer inneren Arbeitsweise.

  Die Korrektheit eines Moduls kann man ohne Kenntnis seiner Einbettung in das Gesamtsystem nachprüfen. Man kann spezifische Testumgebungen zum Prüfen einsetzen.

  Ein Modul verbirgt wichtige Entwurfsentscheidung und bietet nur seine dokumentierte Funktionalität und eine Schnittstelle für den Aufruf der Funktionalität an (Signatur).

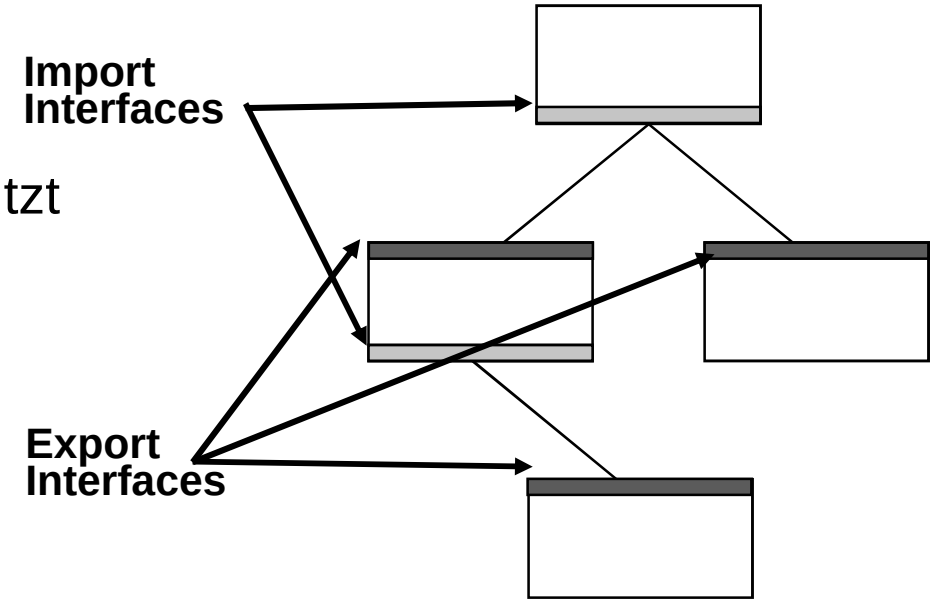
[Par72] D.L. Parnas. *On the Criteria to be Used in Decomposing Systems into Modules*. Communications of the ACM, 15:1053-1058, 1972.

2.3 Prinzip Modularisierung: Geheimnisprinzip

- **Modulschnittstellen**

Exportschnittstelle:
- auf welche Konstrukte (Funktionen oder Variablen) kann von außen zugegriffen werden (auch außen sichtbare Informationen über Typen)

Importschnittstelle:
- welche Konstrukte werden von außen benutzt
- in Sprachen teils nur implizit unterstützt



System mit Modulen

2.3 Prinzip Modularisierung: Geheimnisprinzip

- **Erweiterung des Geheimnisprinzips auf das Konzept der Verträge (*Contracts*)**

  Design by Contract [Mey92]**:**
  durch Vertrag zwischen Anbieter (*Server*) und Abnehmer (*Client*)

  Der Server bietet im Vertrag die Erfüllung einer definierten Leistung an (*Post Condition*).

  Der Client sichert die Eingabe definierter Daten zu (*Pre Condition*).

  Beispiel:
  Server → Führe Division aus; Ergebnis reale oder komplexe Zahl
  Client → Eingabe einer realen Zahl, aber niemals 0

[Mey92] B. Meyer. *Applying Design by Contract*. IEEE Computer, 25(10):40 – 51, 1992.

2.3 Prinzip Modularisierung: Modulkriterien

Kriterien für Module:

Empfehlungen für Module

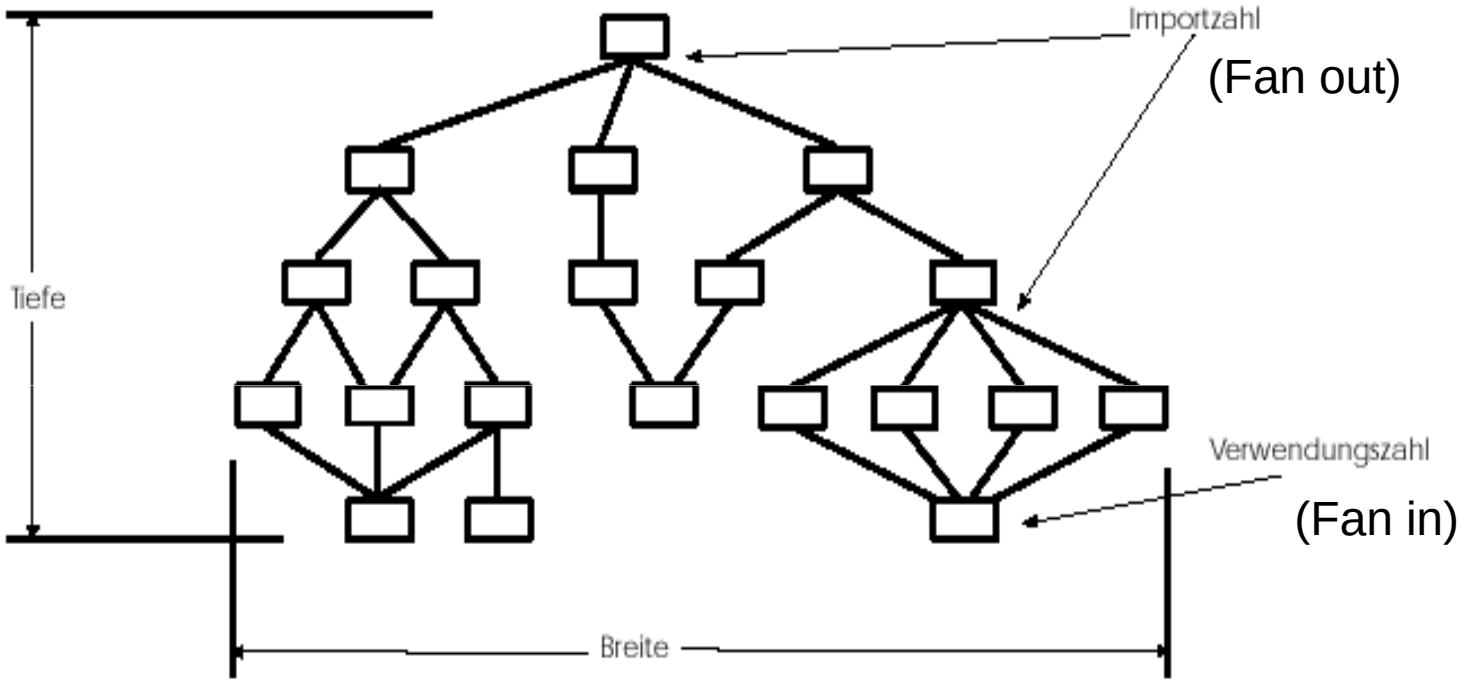
- Every module should communicate with as few others as possible.
- If any two modules communicate at all, they should exchange as little information as possible.
- Whenever two modules and communicate, this must be obvious from the text of one or both.
- All information about a module should be private to the module unless it is specifically declared public.

[Mey88] B. Meyer, *Object-Oriented Software Construction*, Prentice Hall, 1988.

2.3 Prinzip Modularisierung: Modulkriterien

Tiefe / Breite der Modulstruktur

- Tiefe und Breite der Modulstrukturen sollen in vernünftiger Relation zueinander stehen.

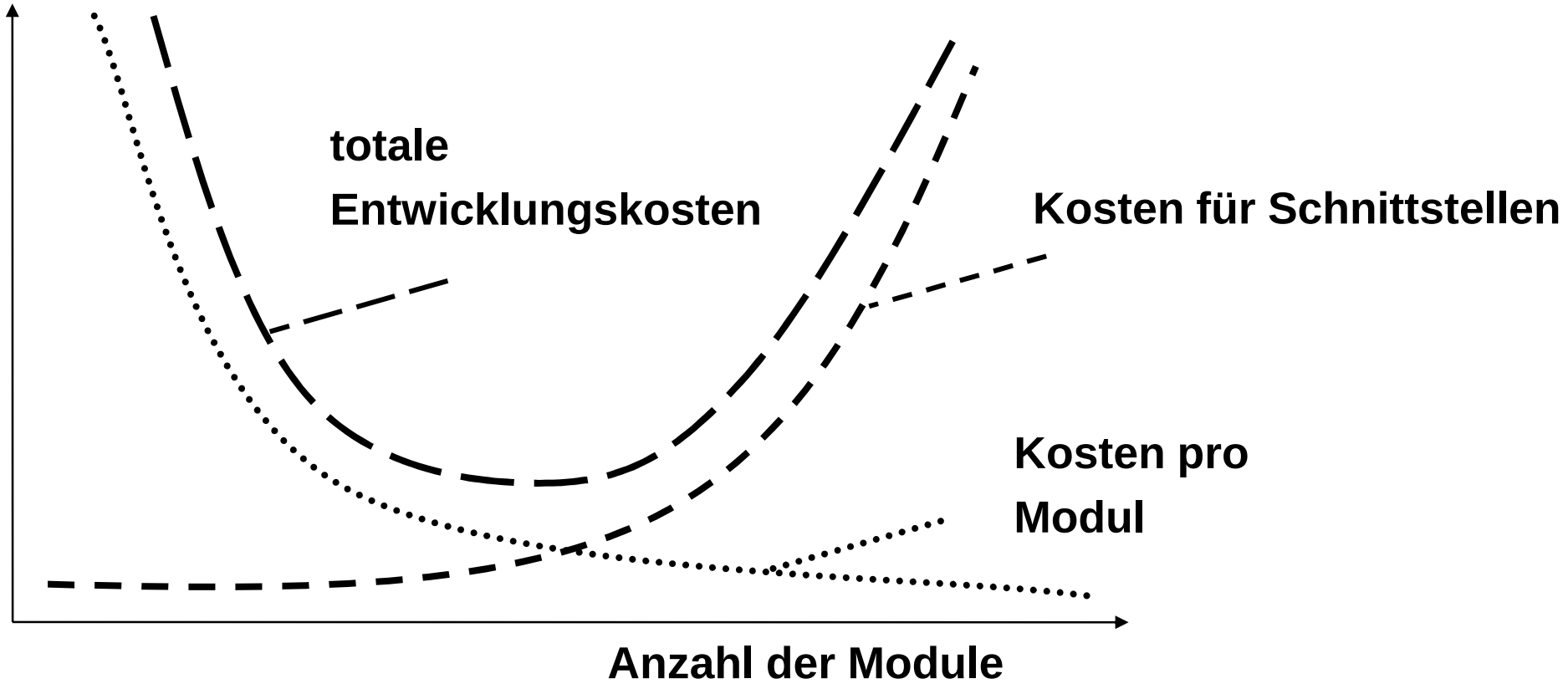


[Kla2001] H. Klaeren. Skript Software Technik, Universität Tübingen, 2001

2.3 Prinzip Modularisierung: Modulkriterien

Modulgröße / Modulzahl

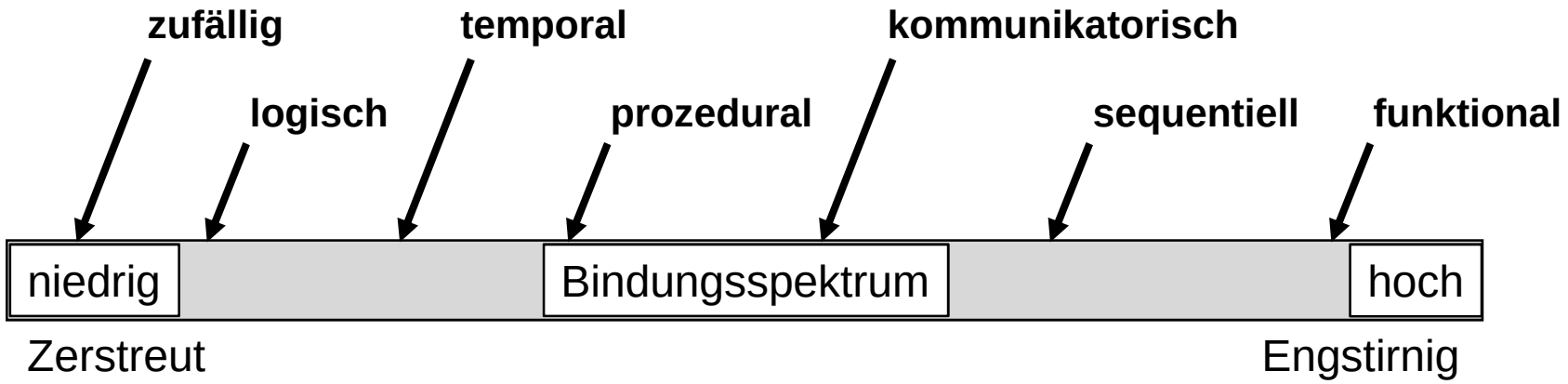
- Modulgröße und Modulzahl sind zueinander gegenläufige Größen.



2.3 Prinzip Modularisierung: Modulkriterien

Bindung von Modulen

- Modulbindung *(Cohesion)*: sollte möglicht eng sein
  ist ein Maß dafür, wie stark der Zusammenhang zwischen den einzelnen Prozeduren (und Daten) eines Moduls wirklich ist.
- Die Modulbindung ist eine Eigenschaft jedes einzelnen Moduls.
- Bindungstypen:



2.3 Prinzip Modularisierung: Modulkriterien

Bindung von Modulen

- Zufällige Bindung (coincidental) zufällige Zusammenstellung, ohne klare Absicht (für Dritte beinahe unmöglich zu verstehen)
- Logische Bindung: logische Zusammenfassung von ähnlichen Funktionalitäten
- Temporale Bindung: Prozeduren werden zeitlich zusammenhängend aufgerufen, d.h. im gleichen Zeitabschnitt
- Prozedurale Bindung: die Elemente eines Moduls sind voneinander (z.B. im Sinne einer bestimmten Aufrufreihenfolgen) abhängig
- Kommunikationsbindung: alle Funktionen operieren auf einer gemeinsamen Datenstruktur (abstrakter Datentyp ADT)
- Sequentielle Bindung: alle Prozeduren müssen in nur einer bestimmten sequentiellen Folge ausgeführt werden, die Sequenz muss vollständig ausgeführt werden
- Funktionale Bindung: das Modul exportiert nur eine einzige Prozedur

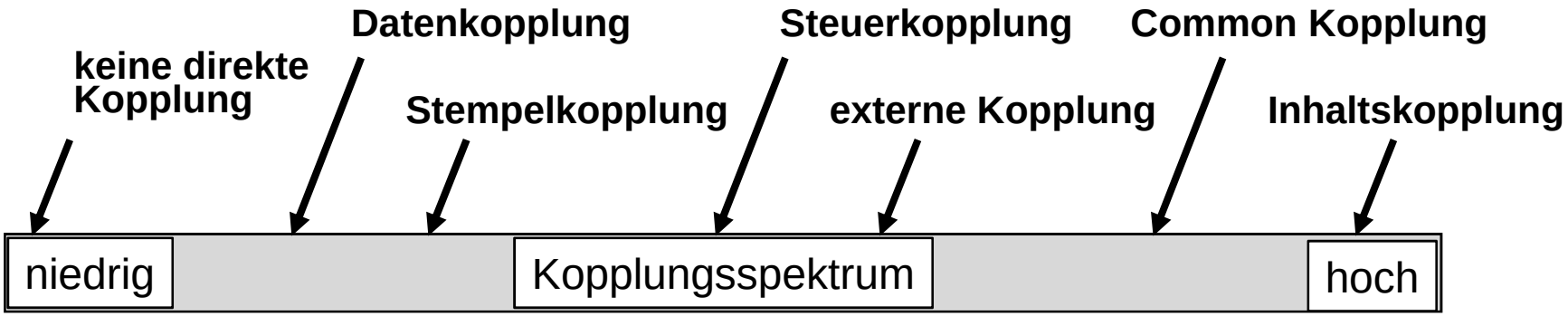
2.3 Prinzip Modularisierung: Modulkriterien

Kopplung von Modulen

- Modulkopplung *(Coupling)*: sollte möglicht lose sein
  ist ein Maß dafür, wie stark die einzelnen Module eines Systems voneinander abhängig sind.

- Kopplung ist eine Eigenschaft zwischen je zwei Modulen eines Systems.

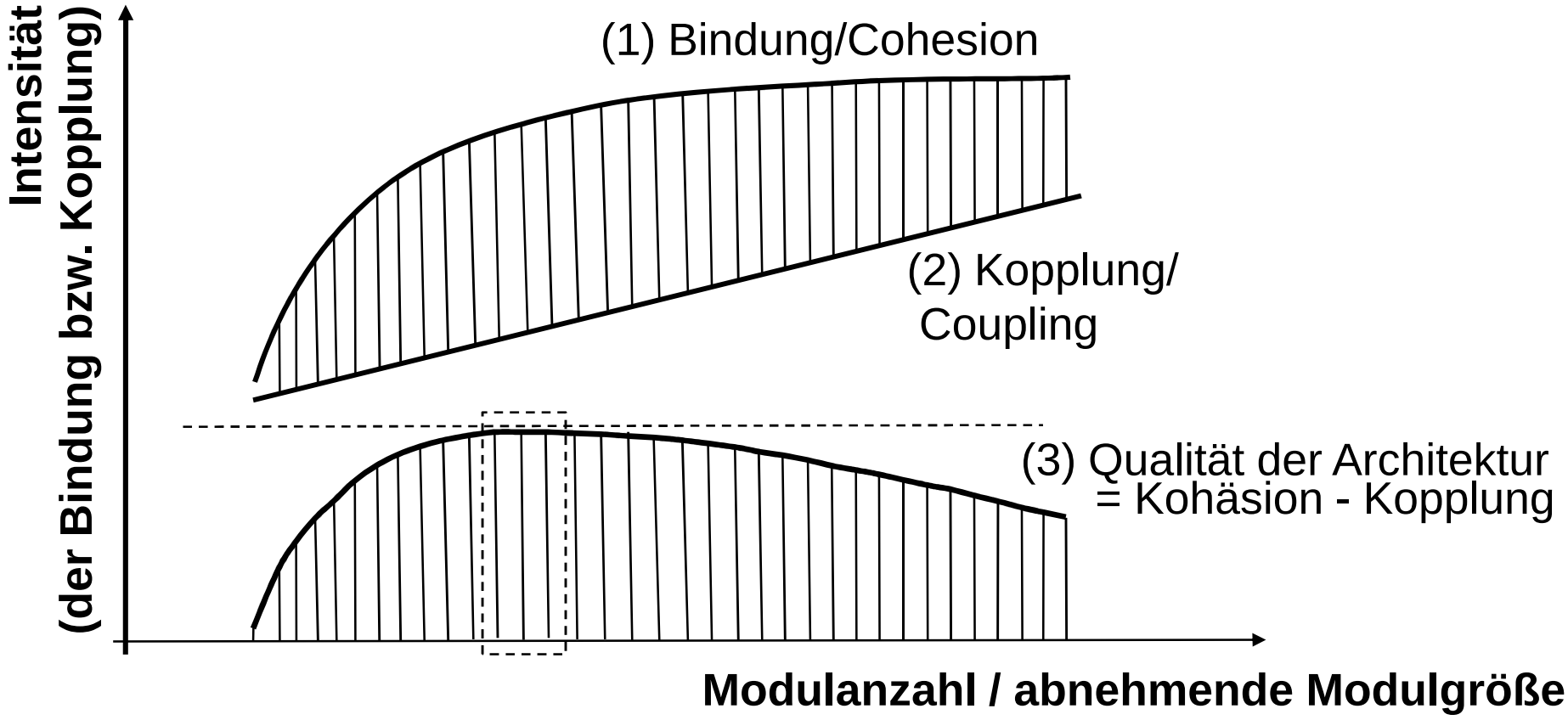
- Kopplungstypen:



Kopplung von Modulen

- Datenkopplung: Zugriff auf Prozeduren des anderen Moduls mittels Argumentlisten, die nur einfache Datenelemente beinhalten
- Stempelkopplung *(Stamp Coupling*): die Argumentlisten beinhalten komplexe Datenstrukturen (das aufrufende Modul muß somit diesen Datentyp ebenfalls kennen)
- Steuerkopplung: bestimmte Argumente eines Aufrufs werden dazu verwendet, um im gerufenen Modul bestimmte Pfade im Steuerfluss auszuwählen (z.B. `OPEN (filename, false)` ergibt Fehlermeldung, wenn `filename` nicht existiert; `OPEN (filename, true)` legt `filename` neu an, falls es noch nicht existiert.)
- Externe Kopplung und Common-Kopplung: Module kommunizieren über globale Datenbereiche außerhalb der Modulstruktur. („the road to hell“)
- Inhaltskopplung: das Modul basiert auf / verwendet Details der Implementierung des aufgerufenen Moduls (z.B. in Modula-2 nicht möglich)

2.3 Prinzip Modularisierung: Modulkriterien



2.3 Prinzip Modularisierung: unterstützende Techniken

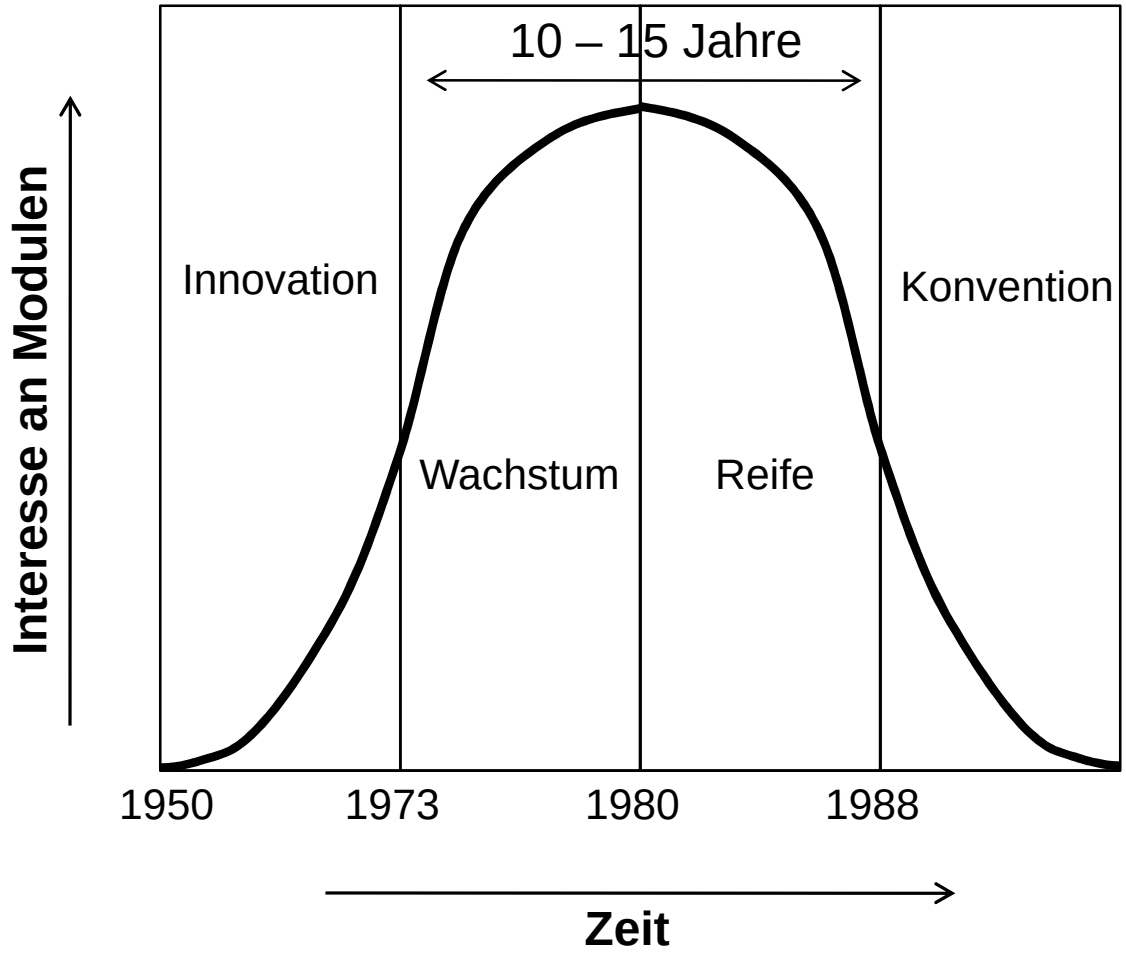
Techniken zur Unterstützung von Modulen:

- **Softwaretechniksprachen zur Unterstützung der Modularisierung (z.B. mit expliziten Import-/Exportschnittstellen)**
- **Sichtbarkeit von Objekten (Scopes) und Blockstruktur in Programmiersprachen**
- **Versionierungswerkzeuge (z.B. CVS Concurrent Versions System)**
- **`make` als Werkzeug (oder andere Build-Werkzeuge) zur Kontrolle von Stapelprozessen (*Batch Processing*); besonders geeignet bei einer getrennten Compilation von Modulen**
- **integrierende Software-Entwicklungsumgebungen mit Projektunterstützung (z.B. Eclipse, diverse Visual Studios)**

Das Konzept der Module findet sich in den verschiedenartigsten Programmierparadigmen (z.B. prozedural, objektorientiert), Programmiersprachen oder Werkzeugen wieder.

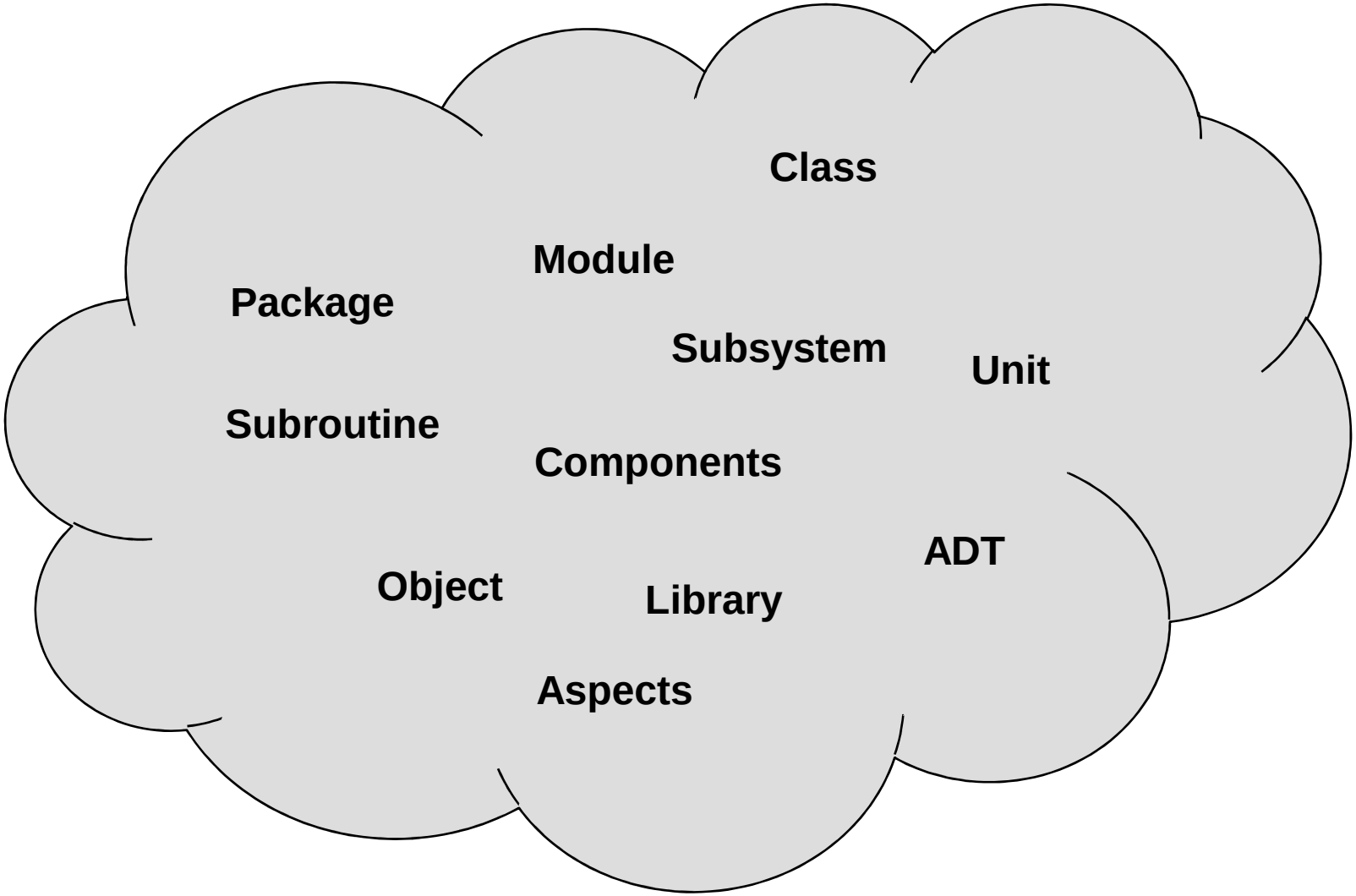
2.3 Prinzip Modularisierung: Module heute

Module heute:



[Raccoon, L. B. S: Fifty Years of Progress in Software Engineering. ACM Software Engineering Notes, 22(1):88-104, 1997]

2.3 Prinzip Modularisierung: Module heute



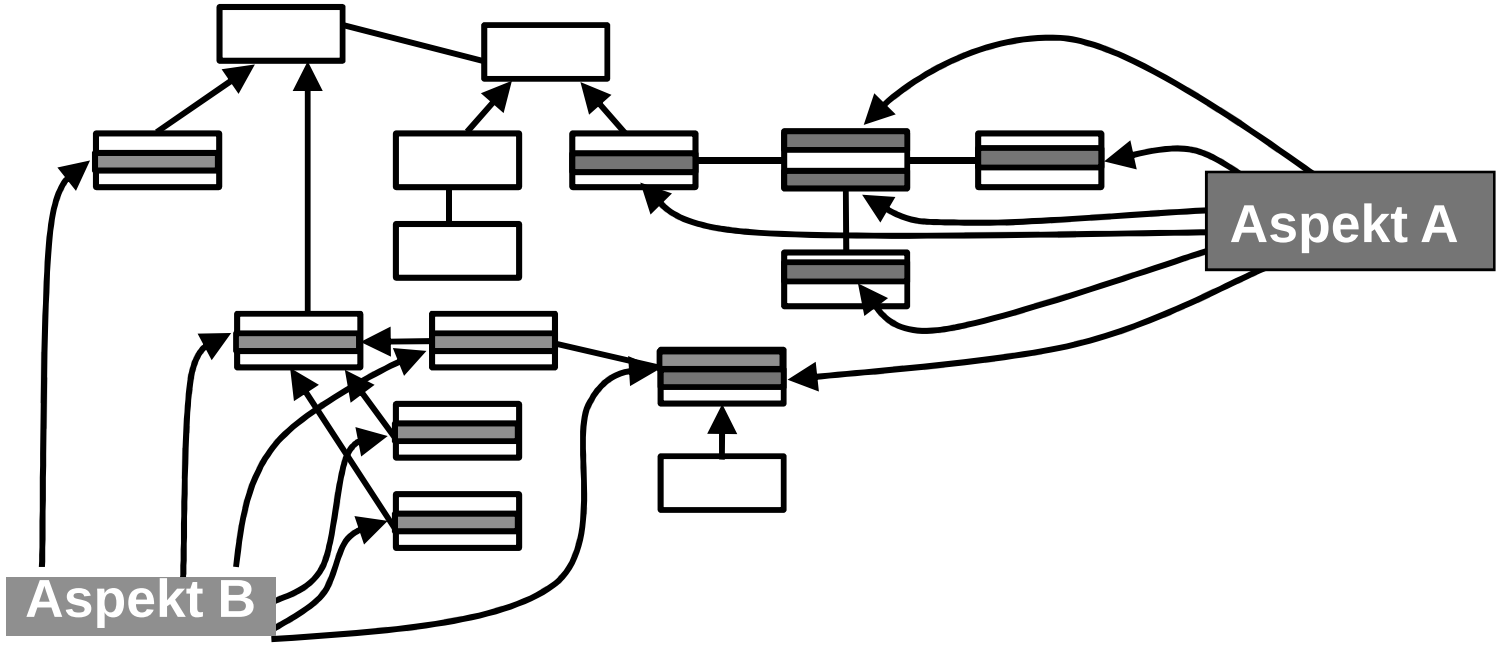
2.3 Prinzip Modularisierung: Module heute

ADT:

$$\begin{aligned}
&\text{datatype } \mathtt{Liste}(A);\\
&\\
&\text{sorts} \quad \mathtt{List}(A),\ A;\\
&\\
&\text{constructors}\\
&\qquad \mathtt{empty} : \rightarrow \mathtt{List}(A);\\
&\qquad \mathtt{cons} \ : A \times \mathtt{List}(A) \rightarrow \mathtt{List}(A);\\
&\text{operations}\\
&\qquad \mathtt{map} \ : (A \rightarrow A) \times \mathtt{List}(A) \rightarrow \mathtt{List}(A);\\
&\\
&\text{equations}\\
&\qquad \mathtt{map}(f,\ \mathtt{empty}) = \mathtt{empty}\\
&\qquad \mathtt{map}(f,\ \mathtt{cons}(a,l)) = \mathtt{cons}(f(a),\ \mathtt{map}(f,l))
\end{aligned}$$

2.3 Prinzip Modularisierung: Module heute

Aspekte (z.B. implementiert mit AspectJ von Xerox)

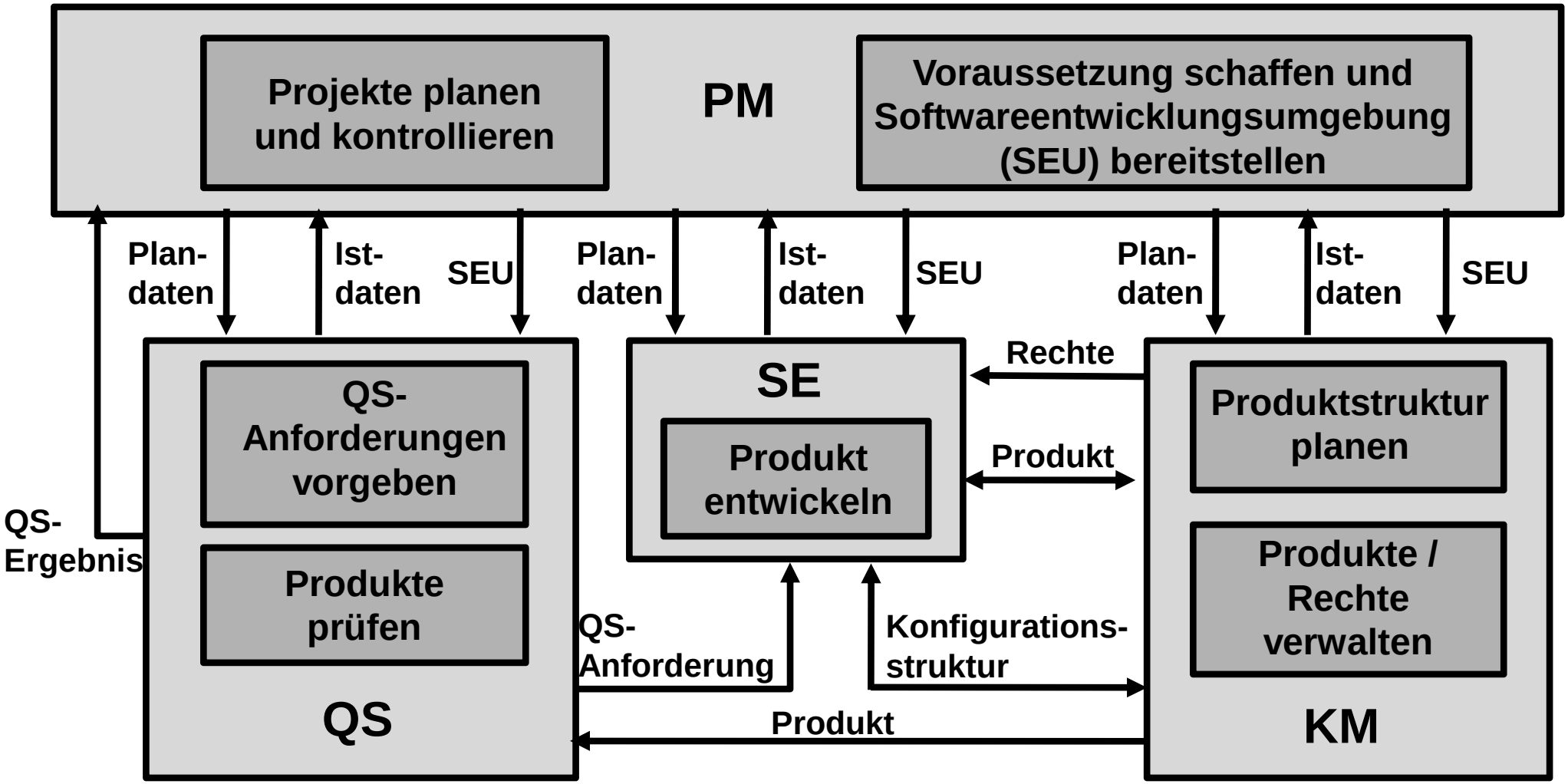


Aspekte durchschneiden Systeme (cross-cutting)

z.B. Logging, Synchronisation

2.4 Bereiche des Software Engineering

Schnittstellen des Projektmanagements



SEU: Software-Entwicklungsumgebung
PM: Projektmanagement
QS: Qualitätssicherung
KM: Software-/Konfigurationsmanagement
SE: Software-Entwicklung (beinhaltet auch Wartung und Evolution)

aus [Zuser, SW Engineering, 2004]

Grundlagen des SW Engineering

Zusammenfassung und Ausblick

1 **Software-Krise und Software Engineering**

2 **Grundlagen des Software Engineering**
- 2.1 Phasen der Software-Entwicklung
- 2.2 Grundlegende Prinzipien
- 2.3 Prinzip Modularisierung
- 2.4 Bereiche des Software Engineering

3 **Projektmanagement**

4 **Konfigurationsmanagement**

5 **Software-Modelle**

6 **Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle**

7 **Qualität**

8 **Fortgeschrittene Techniken**

→ **Wege im Umgang mit der Software-Krise**

systematische Planung, Steuerung und Kontrolle von Projekten: Projektmanagement