Software Product Lines

Concepts, Analysis and Implementation

Feature Interaktionen

Dr. Malte Lochau
Malte.Lochau@es.tu-darmstadt.de

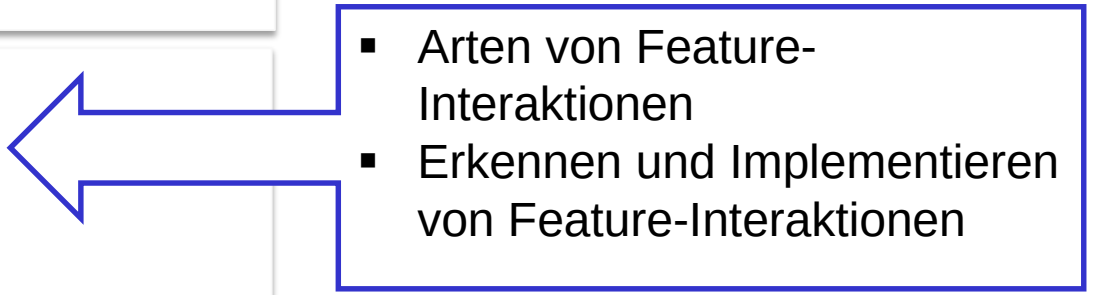
Inhalt

I. Einführung
- Motivation und Grundlagen
- Feature-orientierte Produktlinien

II. Produktlinien-Engineering
- Feature-Modelle und Produktkonfiguration
- Variabilitätsmodellierung im Lösungsraum
- Programmierparadigmen für Produktlinien

III. Produktlinien-Analyse
- Feature-Interaktion
- Testen von Produktlinien
- Verifikation von Produktlinien

IV. Fallbeispiele und aktuelle Forschungsthemen



Fragestellungen

- Was passiert, wenn unabhängig konfigurierbare Features nicht unabhängig im Lösungsraum sind?
- Wie interagieren Features und wie implementiert man diese Interaktionen?
- Wie bewahrt man Modularität der Variabilität trotz dieser Abhängigkeiten?
- Wieviel Variabilität ist sinnvoll?
- Wie erkennt man ungewollte Interaktionen?

Feature Interaktion: Ursprung

- Klassisches Beispiel: Telekommunikationssysteme
  - **Call Forwarding**: forward calls when busy
  - **Call Waiting**: interrupt existing call
  - *FI Problem: Incoming call while other is active: forward or notify with call waiting?*

[P. Zave, 1993]

- Feature Interaktionen beziehen sich auf das **Verhalten von Features** in einem Produkt/System

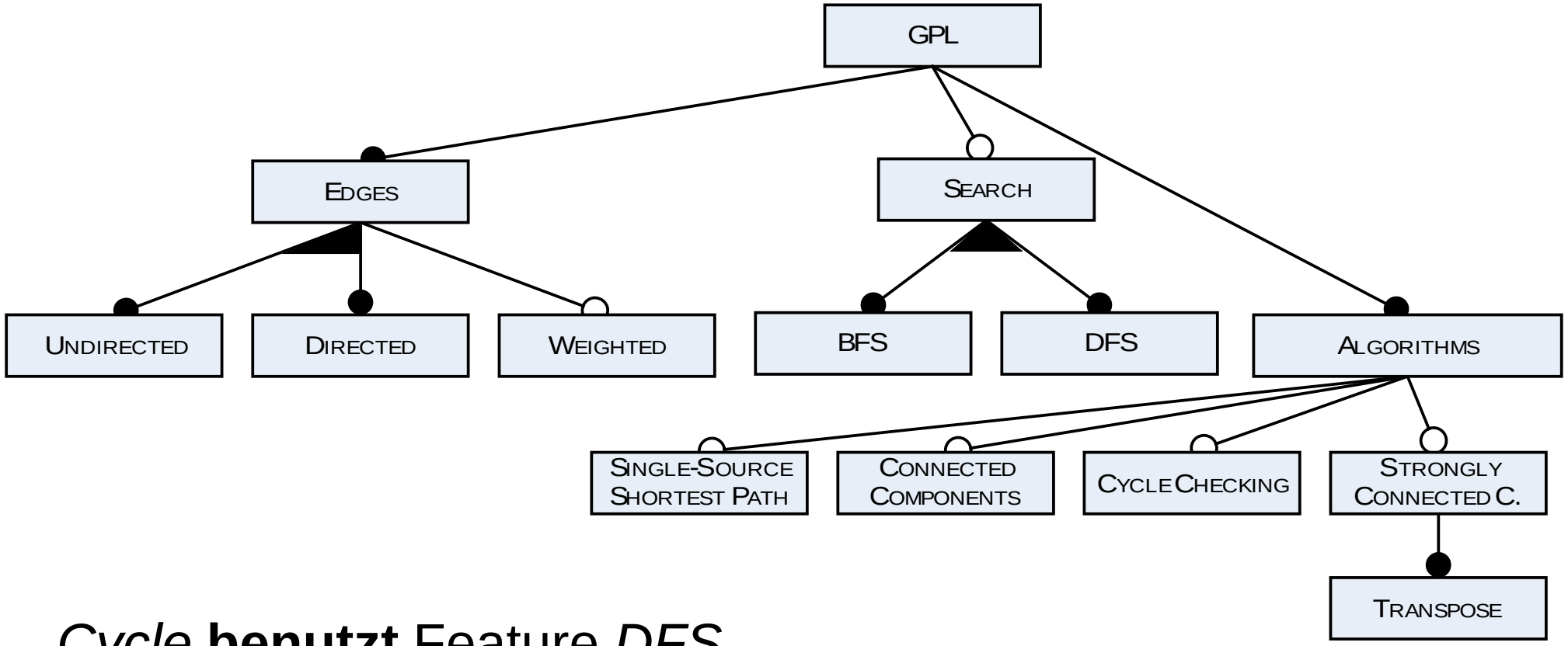
- Wie kann man solche Feature-Abhängigkeiten im Lösungsraum erkennen (Feature Interaction Detection)?

- Problem: Modulare Feature-Spezifikation und Komposition vs. Verhalten, das sich für spezifische Feature-Kombinationen ergibt

Beispiel: Feature Interaktionen

- Flood Control
  - Verhindert Überschwemmung durch Abstellen der Wasserversorgung
- Fire Control
  - Bekämpft Feuer durch Sprinkleranlage

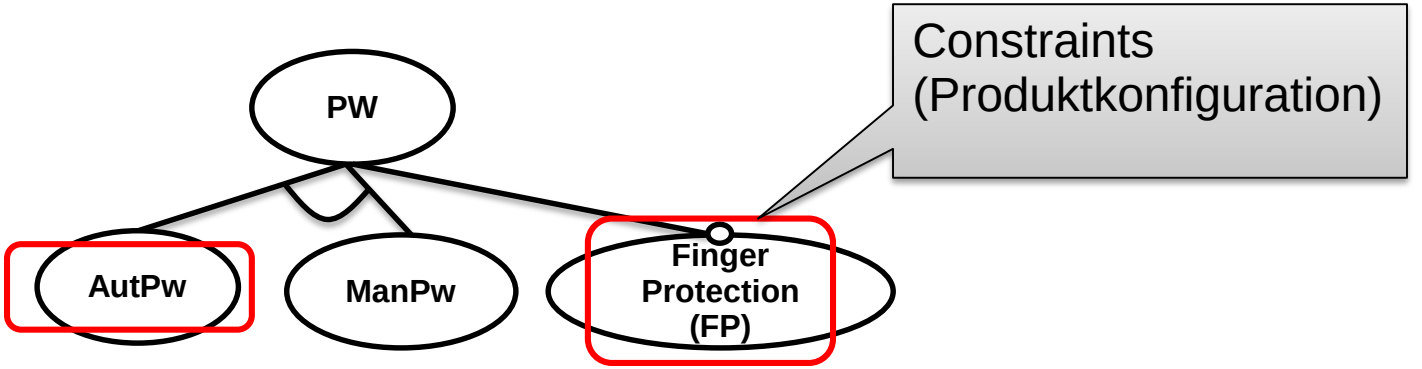
Beispiel: Feature Interaktion



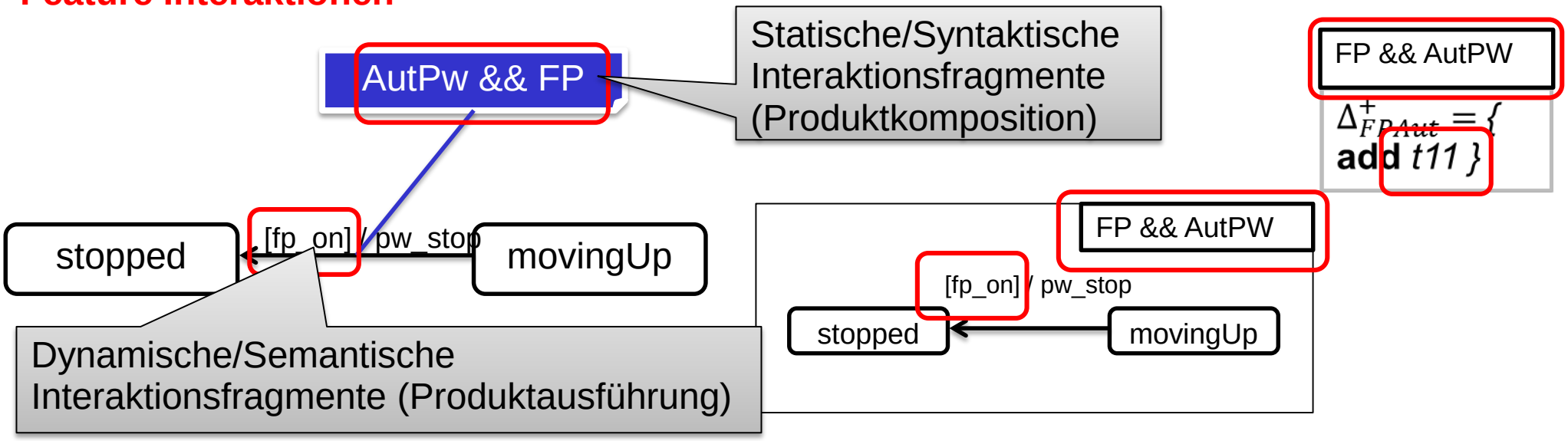
- *Cycle* **benutzt** Feature *DFS*
- *Shortest Path* **benötigt** *gewichtete* Graphen
- *Connected Components* **schließt** *gerichtete* Graphen **aus**
- Strongly Connected **benötigt** gerichteten Graphen **und** DFS

Beispiel: BCS

Problem Space:
Feature Abhängigkeiten



Solution Space:
Feature Interaktionen



FI: Definitionsversuche

[…] A feature interaction occurs in a system whose complete behavior does not satisfy the separate specifications of all its features.

[Metzger et al., 2005]

[…] A feature *f* interacts with a feature *f‘* if the behavior of *f* depends on wether *f‘* is present or absent in a product.

[Ferber et al. 2002]

[…] The integration of a set of features requires the modification of features.

[Zave, 1993]

Feature Interaktionen – Kategorien

Feature *Zentralverriegelung* benutzt Feature *Automatischer Fensterheber*, um beim Abschließen des Fahrzeugs automatisch das Fenster zu schließen.

Feature *Einklemmschutz* hindert das Feature *Automatischer Fensterheber* am Hochfahren des Fensters, falls ein Hindernis erkannt wurde.

	Positive FI	Negative FI
Gewollte FI	Feature Kooperation	Feature Vetoing
Ungewollte FI	- Zusätzliche Features - Überlagernde Features	- Fehlende Features - Fehlerhafte Features

Feature *Einklemmschutz* hindert das Feature *Automatischer Fensterheber* am Hochfahren des Fensters, obwohl das Feature Einklemmschutz gar nicht ausgewählt wurde

Das automatische Schließen des Fensters durch die *Zentralverriegelung* erfolgt ohne Koordination mit dem *Einklemmschutz*

Feature *Einklemmschutz* hindert das Feature *Automatischer Fensterheber*, obwohl das Fenster herunterfährt

Das Feature *Einklemmschutz* wird trotz Hindernis nicht wirksam, obwohl es ausgewählt wurde

Nichtfunktionale Feature-Interaktionen

[…] Sendet Feature *Einklemmschutz* ein Signal zum Blockieren des *Fensterhebers* während die *Zentralverriegelung* ein Signal zum Schließen der Fenster sendet, erreicht das Signal des *Einklemmschutzes* den *Fensterheber* nicht rechtzeitig innerhalb der geforderten 10 ms.

- Wieviel **Prozessorzeit** verbraucht ein Feature?
- Wieviel **Speicher** verbraucht ein Feature?
- …

Multi-Interaktionen

Multi-Feature-Interkationen sind gewollte/ungewollte Interaktionen, die in bestimmten Kombinationen von T Features auftreten ($1 \leq T \leq |F|$)

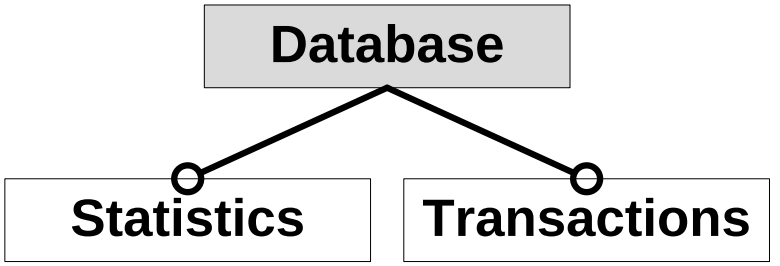
Beispiel BCS:

- 3-wise Feature Interaktionen:
  AutPW – CLS – RCK

  …
- 8-wise Feature Interaktionen:

  AL – RCK – SF – CAP – CLS – PW – FP – LED

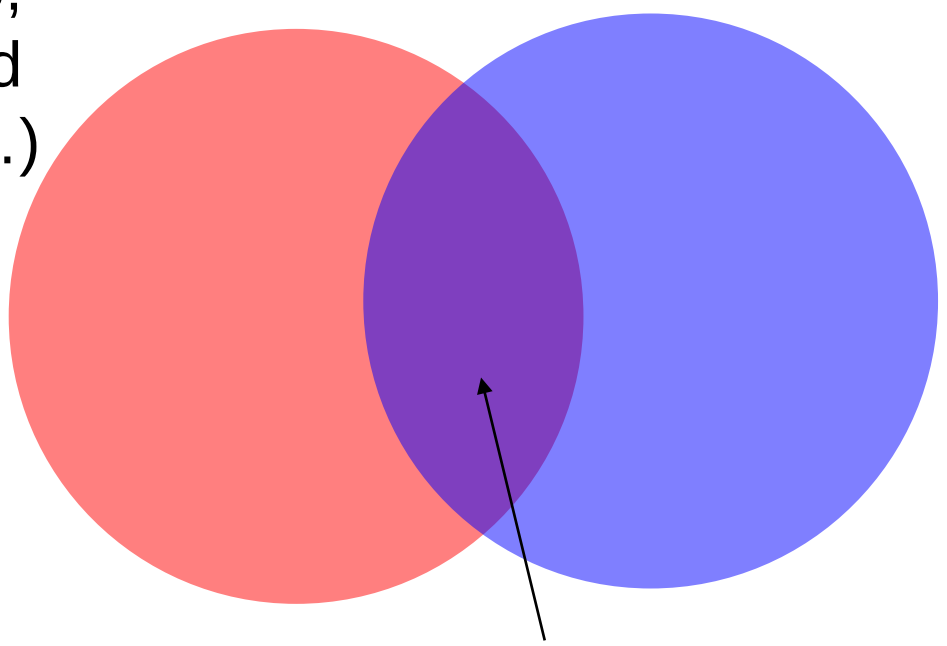
Beispiel: Feature Interaktion



- Datenbank-Produktlinie mit zwei Features
  - Statistik: sammelt Statistiken über Buffer Hit Ratio, Tabellen-Größe, Transaktionen
  - Transaktionen: Sicherstellung der ACID-Eigenschaften
- Beide Features sollen optional sein
  - Aber: Statistik sammelt Informationen über Transaktionen, Transaktionen nutzen evtl. statistische Informationen...
- Modulare Implementierung beider Features?

FI in Datenbanken: Transaktionen und Statistiken

Statistiken (buffer hit ratio, table size and cardinality, …)

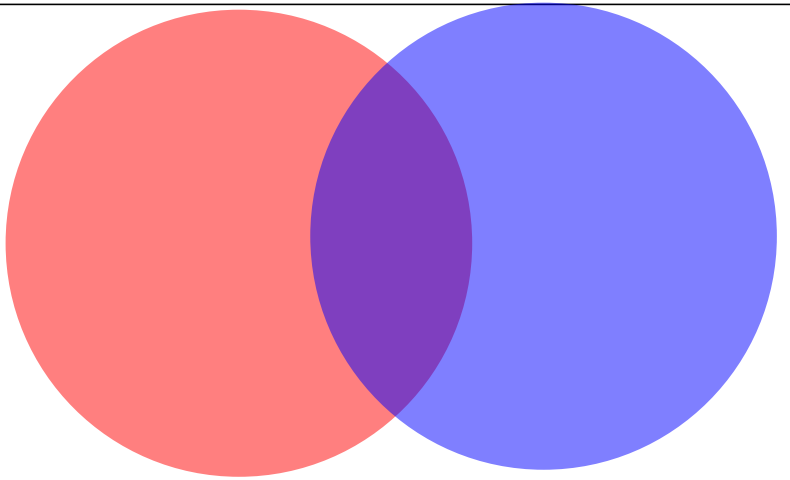


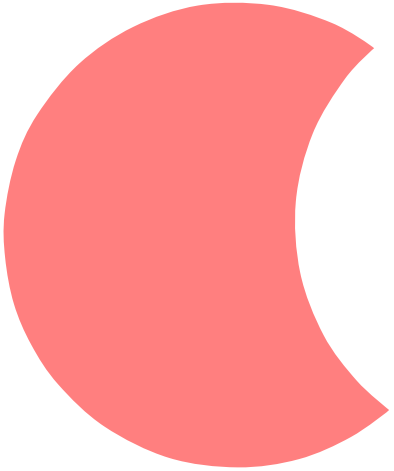
Transaktionen (locks, commit, rollback, …)

Durchsatz-Messung (“Transaktionen pro Sekunde”)

Feature-Kombinationen und gewollte Feature Interaktionen

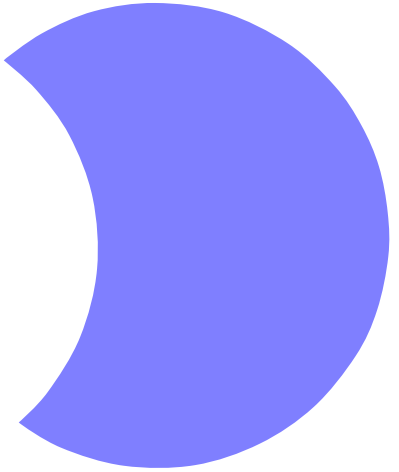
Datenbank mit Statistiken
und mit Transaktionen





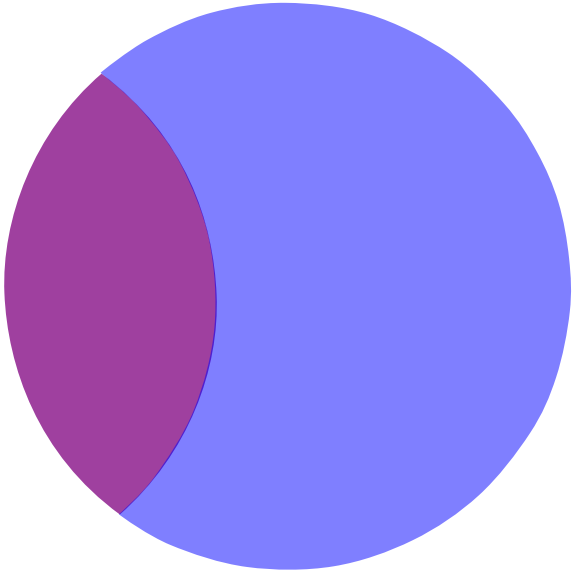
Datenbank mit Statistiken, aber ohne Transakt.

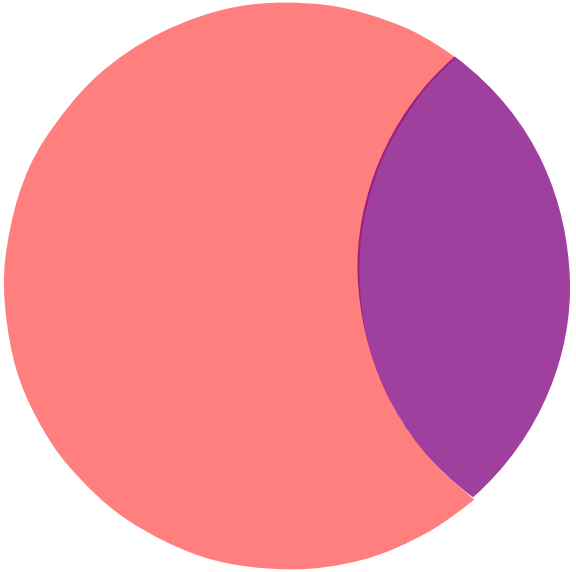
Datenbank mit Transakt., aber ohne Statistiken



Ungewollte Feature Interaktionen

Datenbank mit Transaktionen und ohne Statistiken, die aber denoch den Durchsatz misst
(Größer und langsamer als nötig)





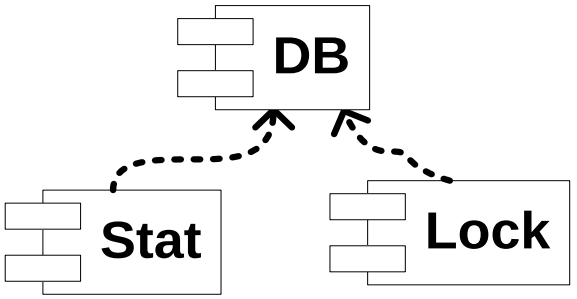
Datenbank mit Statistiken und ohne Transaktionen, die dennoch den Durchsatz misst (?)

Feature Interaktionen: Implementierung

```
class Database {
  List locks;
  void lock() { /*...*/  }
  void unlock() { /*...*/ }
  void put(Object key,Object data){
    lock();
    /*...*/
    unlock();
  }
  Object get(Object key) {
    lock();
    /*...*/
    unlock();
  }
  int getOpenLocks() {
    return locks.size();
  }
  int getDbSize() {
    return calculateDbSize();
  }
  static int calculateDbSize() {
    lock();
    /*...*/
    unlock();
  }
}
```

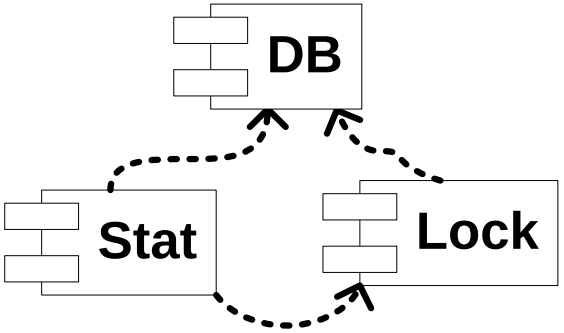
- Sperren (blau)
- Statistiken (rot)
- Die Features überlappen sich an 2 Stellen (violett)
  - Statistiken über Locks
  - Synchronisierung der Statistikmethode

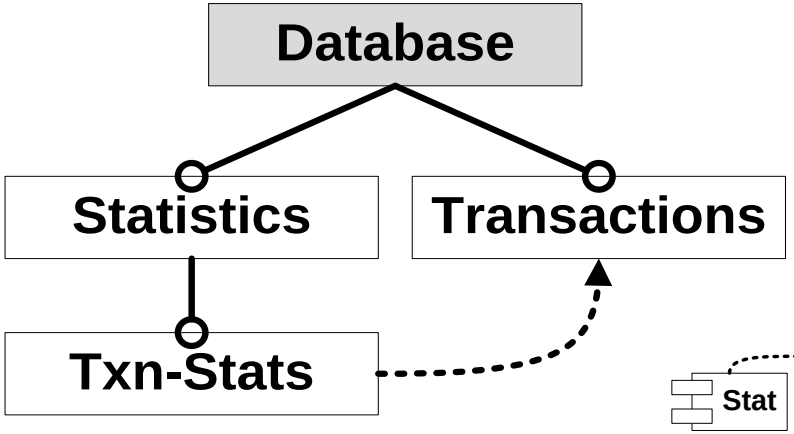
Modularisierung von Feature Interaktionen?

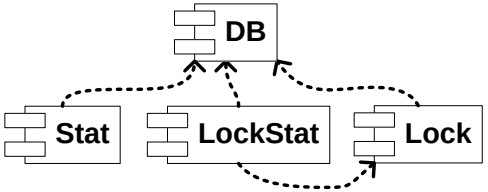


Wo implementiert man die Durchsatz-Messung?

Wie **komponiert** man ein Produkt mit Statistiken aber ohne Transaktionen?







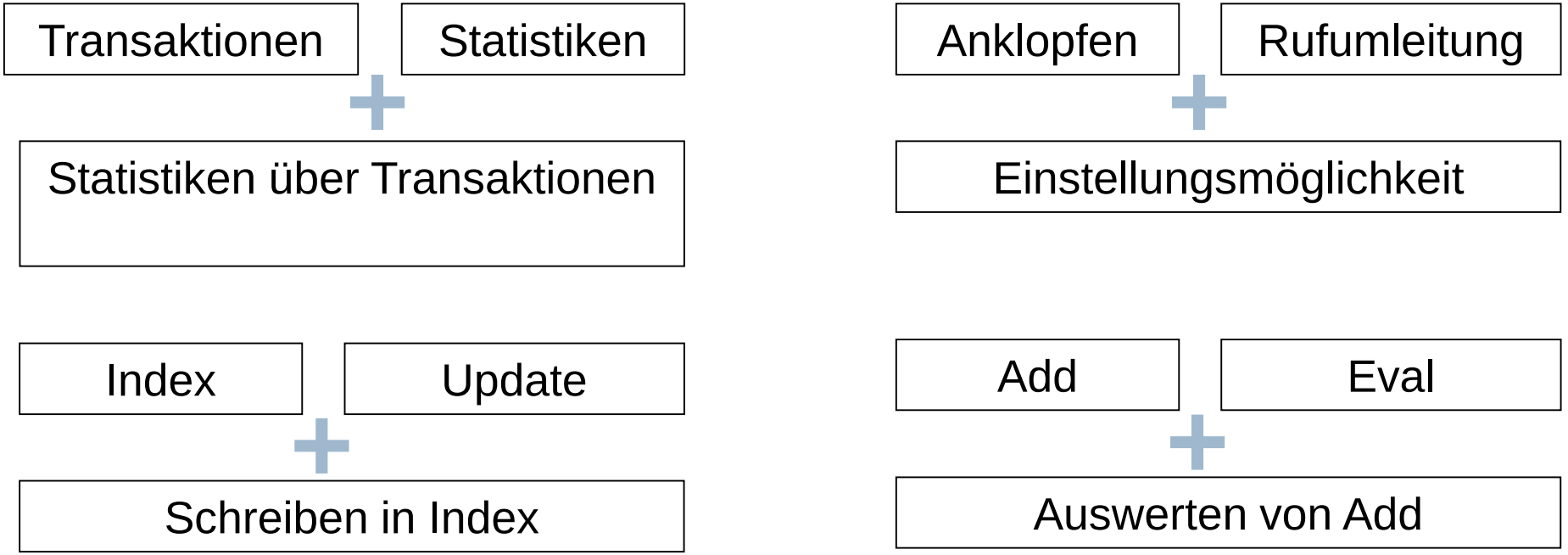
Ist Durchsatz-Messung wirklich ein **Feature**?

Feature Interaktion in FOP

- Features **benutzen** Methoden anderer Features
  - *Cycle* nutzt die Suchfunktion Graph.search, die in *DFS* eingeführt wurde
  - *Shorted Path* erwartet, dass die Methode Edge.getWeight vorhanden ist
- Features **erweitern** andere Features
  - Feature *Weighted* implementiert Gewichte, indem es die Methode addEdge aus *Base* überschreibt
- Features **verlassen sich auf** ein bestimmtes Verhalten, welches in einem Feature festgelegt wird
  - *Connected* erwartet das Kanten immer in beide Richtungen zeigen

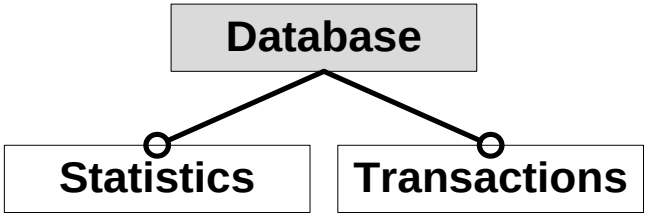
FOP: Optional Feature Problem

- Optionales Feature verhält sich isoliert korrekt
- Problem in Kombination mit anderen Features
- Zusätzlicher Quelltext koordiniert richtiges Verhalten

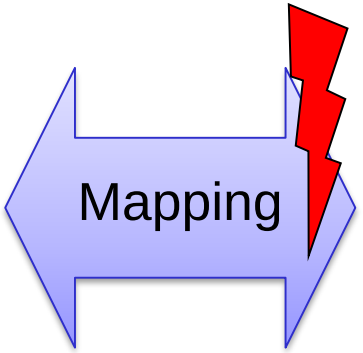


Interaktionen schränken Variabilität ein

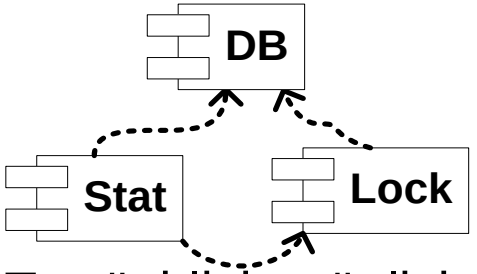
Feature Modell



Gewollt:
4 Produkte

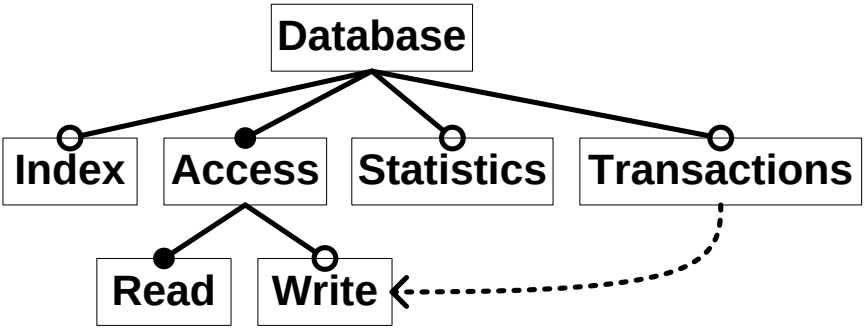


Implementierung

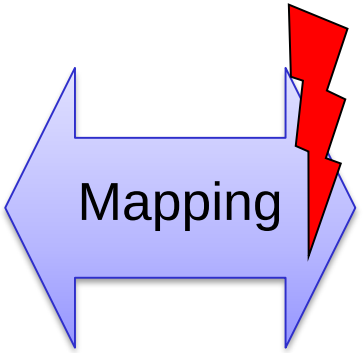


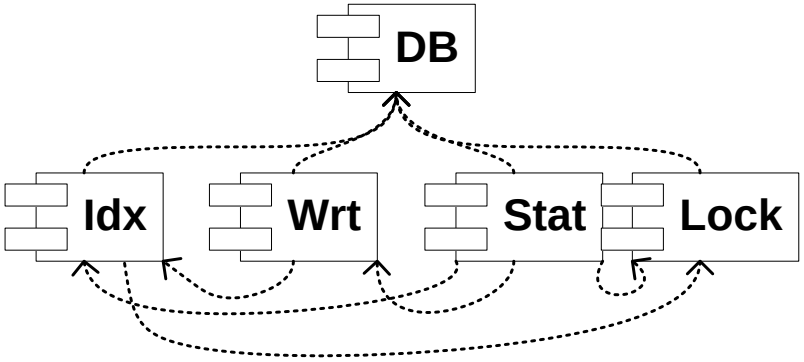
Tatsächlich möglich:
3 Produkte

Eingeschränkte Variabilität



Gewollt:
12 Produkte





Tatsächlich möglich:
5 Produkte

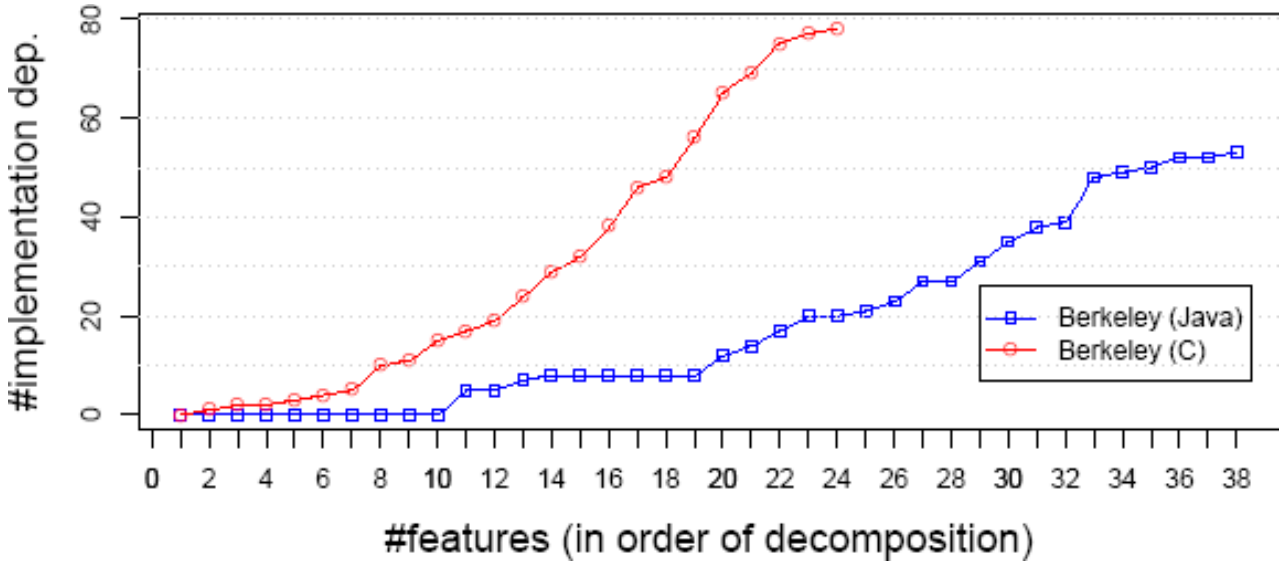
Beispiel: Berkeley DB

Java-Version

- Feature-Modell
  - 38 Features
  - 16 Domänenabh.
  - 3.6 Milliarden Varianten
- Implementierung
  - 53 Implementierungsabh. (eingeschr. Variabilität!)

C-Version

- Feature-Modell
  - 24 Features
  - 8 Domänenabh.
  - ~1 Millionen Varianten
- Implementierung
  - 78 Implementierungsabh. (eingeschr. Variabilität!)



Transitivität von Interaktionen

“A interagiert mit B“ und „B interagiert mit C“ => „A interagiert mit B und C“

- Folge: Einzelne Features können die Auswahl/das Verhalten vieler weiterer Features erzwingen und die Variabilität der SPL deutlich einschränken

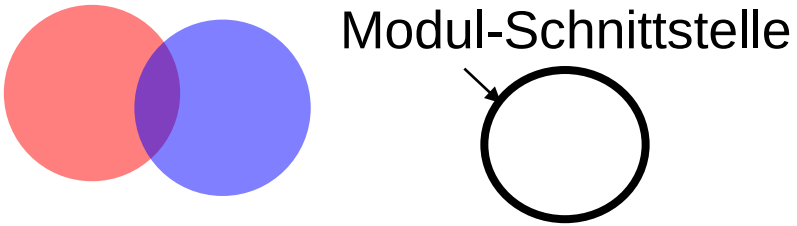
- Beispiel: Berkeley DB
  - Das Statistik-Feature sammelt Statistiken über verschiedene Bereiche des Programms, z. B. Speicherverbrauch, Transaktionen, Schreibzugriffe, Buffer Hit Ratio usw.
  - Die Auswahl des Statistik-Features erzwingt die Auswahl von 14 (von 37) weiteren Features, u. a. Transaktionen, Caches

Implementierungsabhängigkeiten

- Implementierungsabhängigkeiten erschweren einen sauberen Entwurf
- Reduzierte Variabilität, obwohl Varianten in Domäne möglich wären
- Beispiel: Transaktionen vs. Statistiken (s.o.)
  - Lösung 1: im Feature-Modell benötigt Statistik Transaktionen
    ➔ Reduzierte Variabilität
  - Lösung 2: Keine Statistik über Transaktionen
    ➔ Reduzierte Implementierung
- Implementierungsabhängigkeiten auflösen?

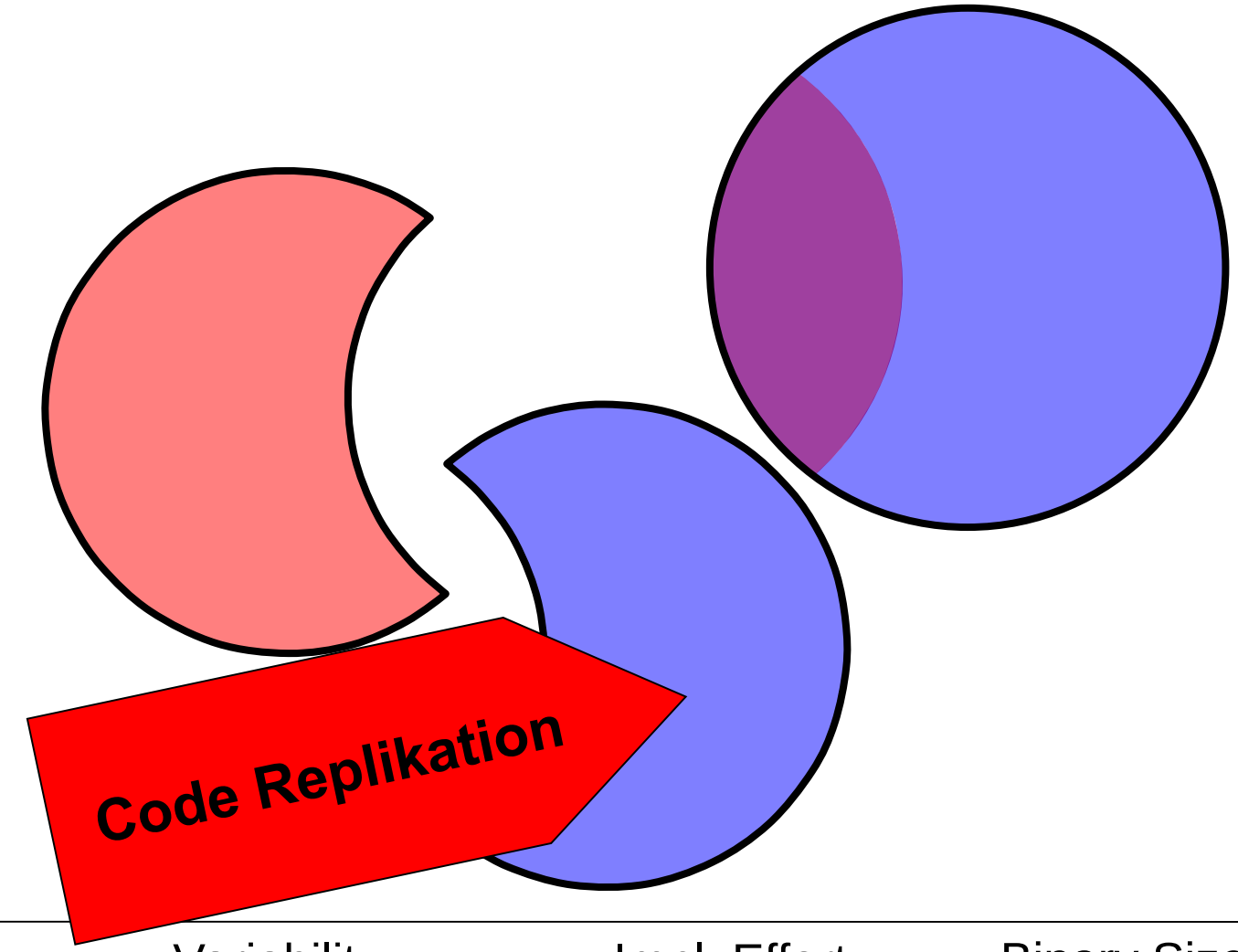
Optionale Features implementieren

- Wie modularisiert man zwei interagierende Features?

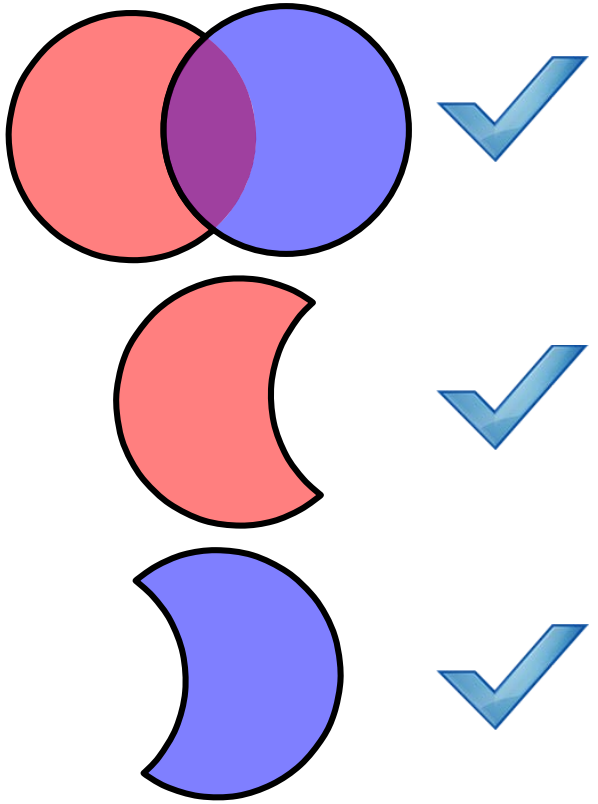


- Ziele:
  - Variabilität wie im Feature-Modell vorgesehen
  - Geringer Implementierungsaufwand
  - Effiziente Implementierung (Code-Größe, Performance)
  - Code-Qualität (Trennung v. Belangen, Modularität)

Lösung 1: Multiple Implementierungen

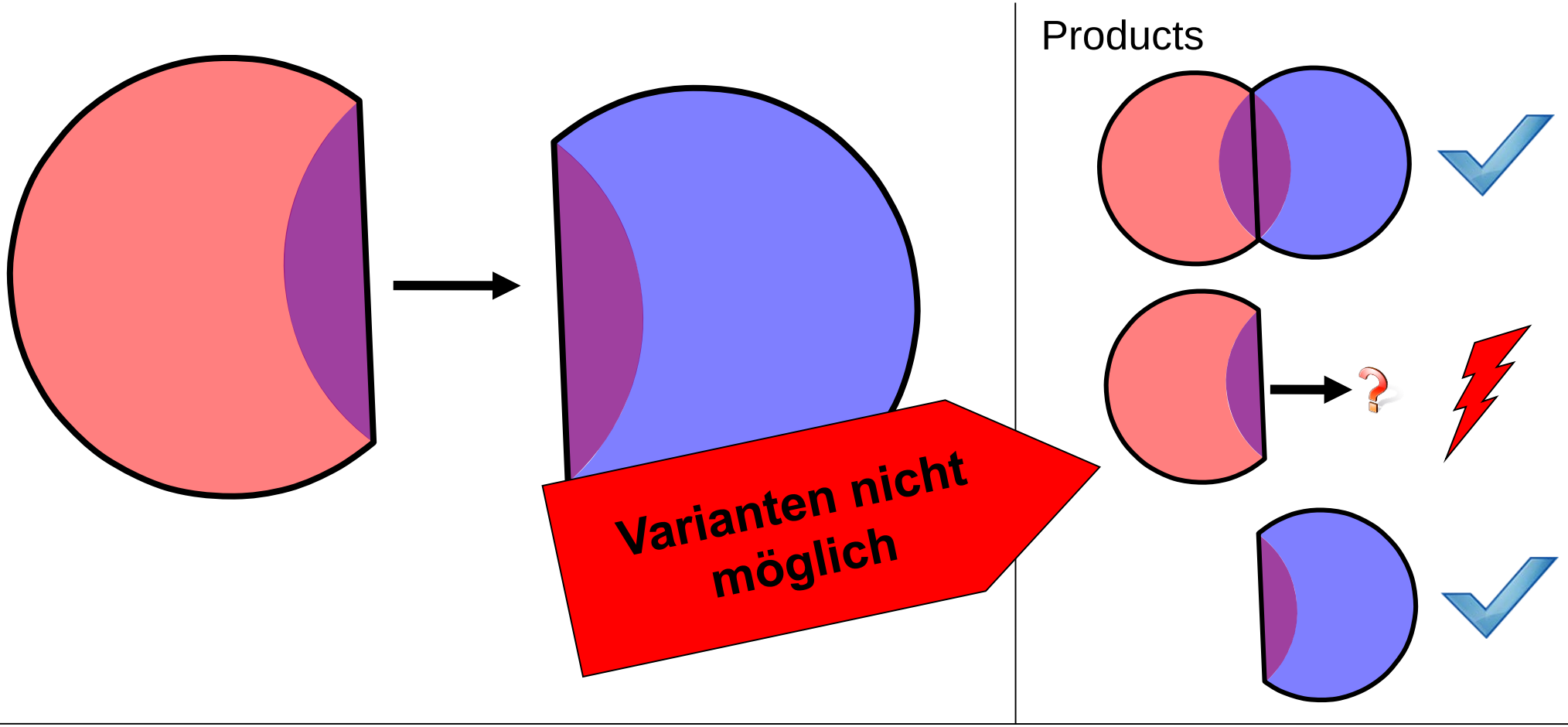


Produkte



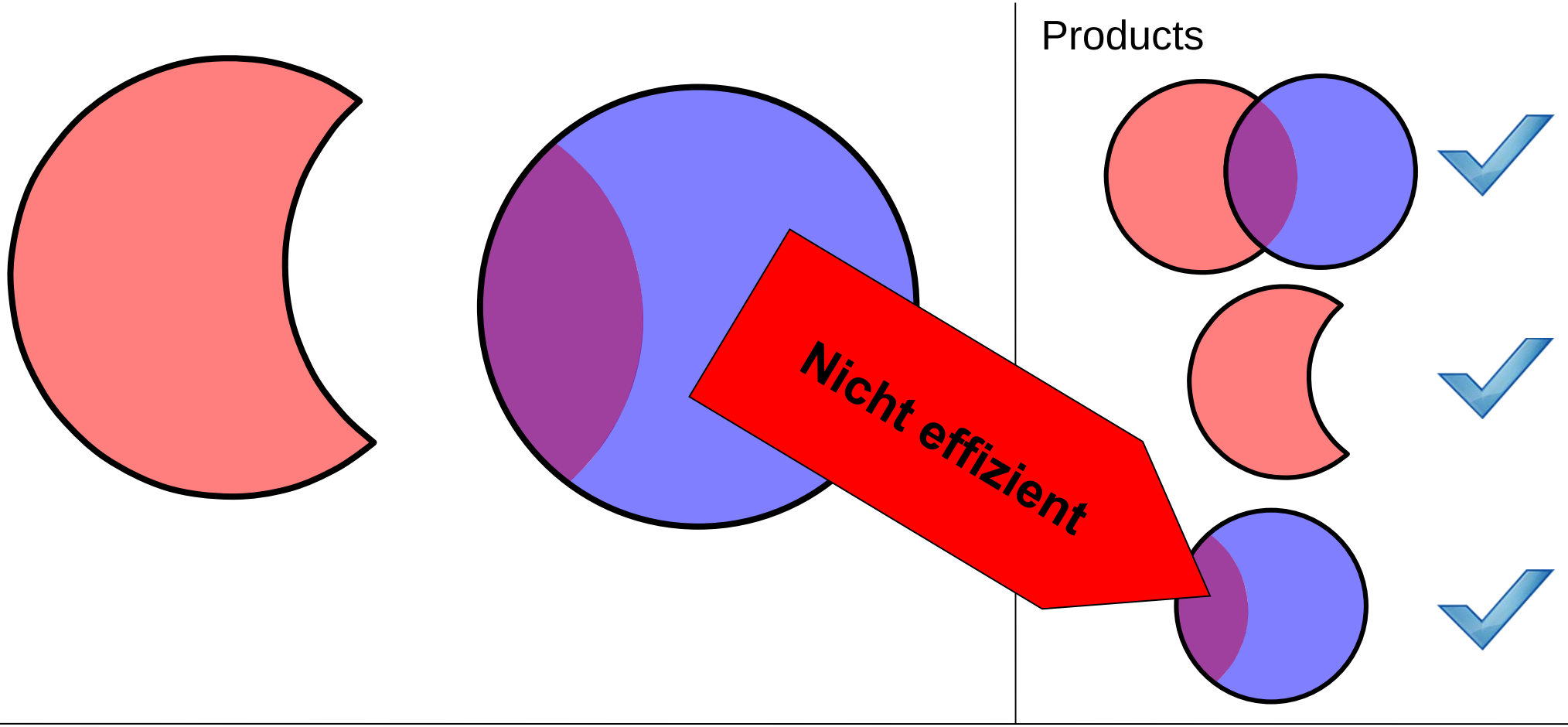
Variability	Impl. Effort	Binary Size & Perf.	Code Quality
✓	↯	✓	↯

Lösung 2: Abhängigkeiten beibehalten



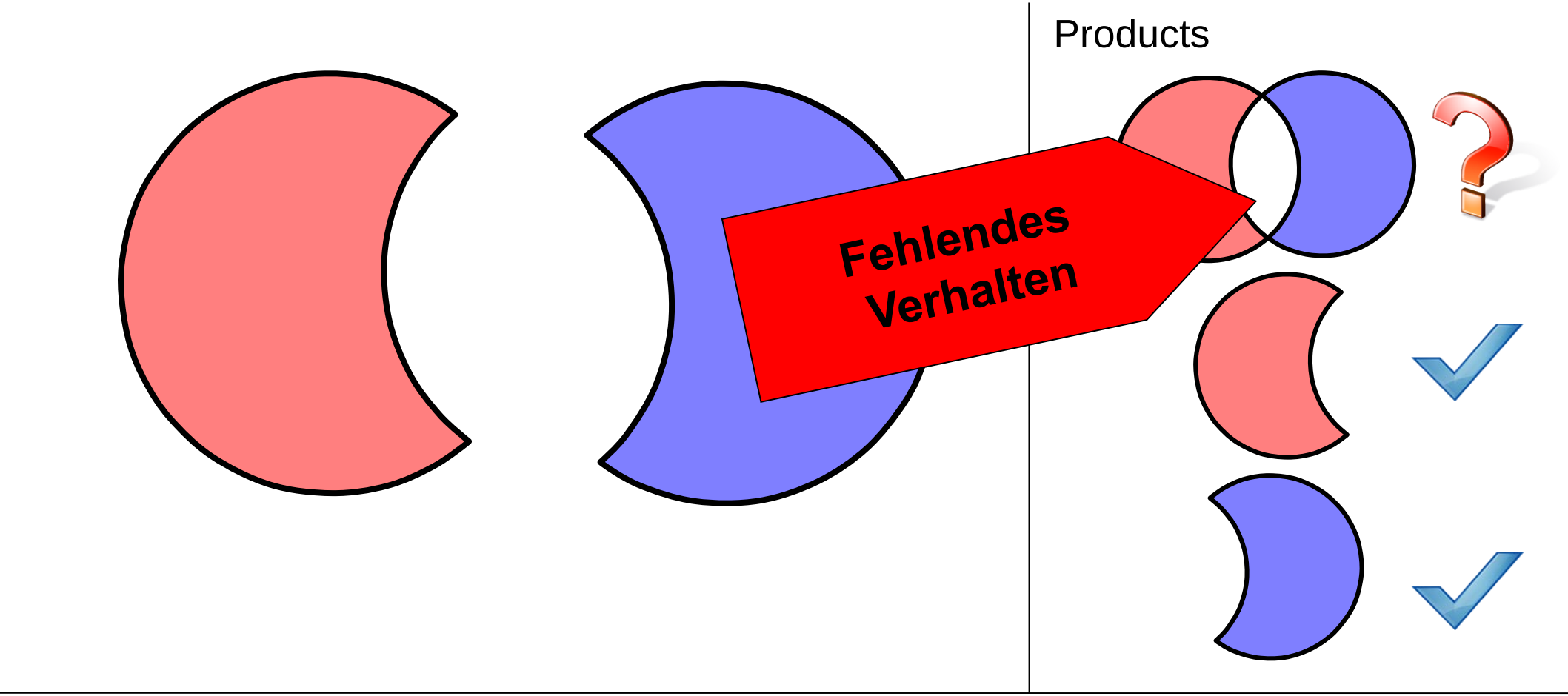
Variability	Impl. Effort	Binary Size & Perf.	Code Quality

Lösung 3: Code verschieben



Variability | Impl. Effort | Binary Size & Perf. | Code Quality

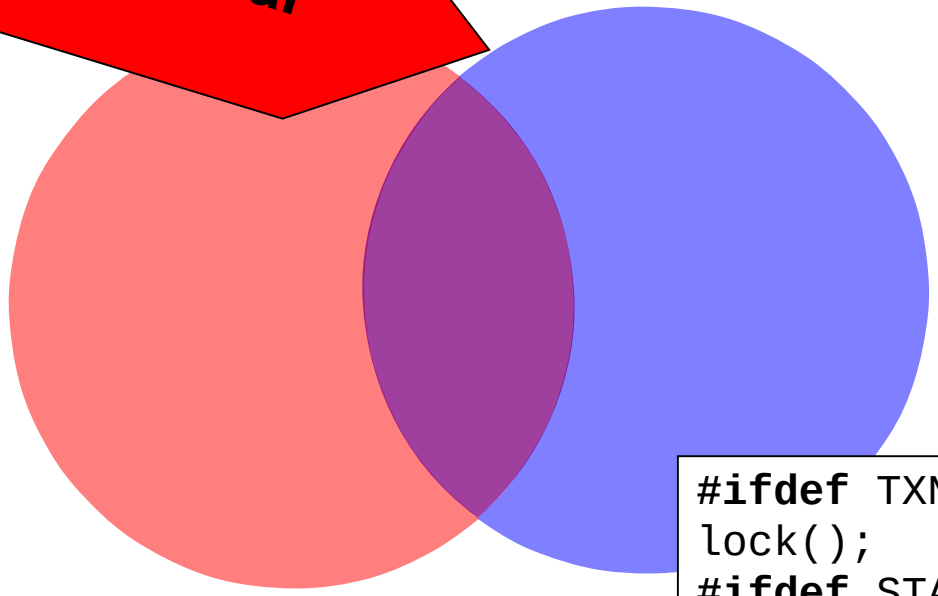
Lösung 4: Verhalten anpassen



Variability | Impl. Effort | Binary Size & Perf. | Code Quality

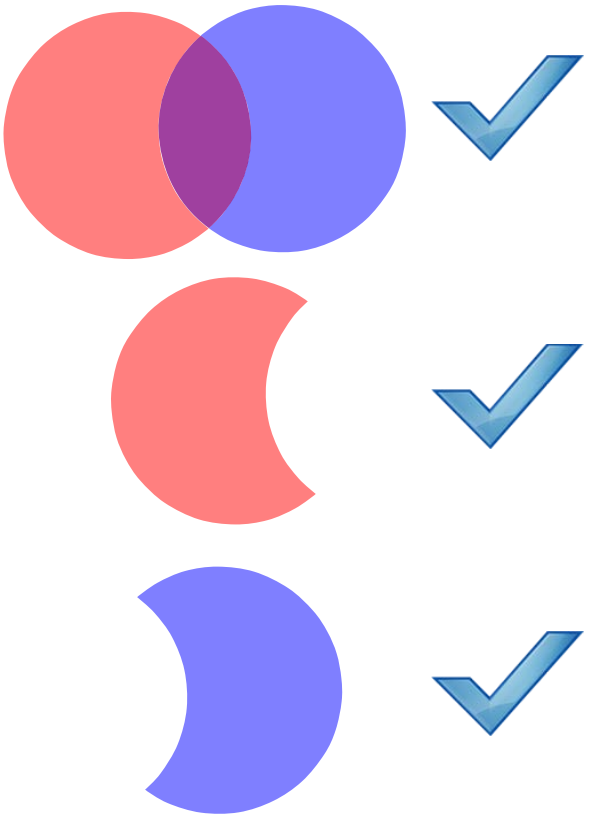
Lösung 5: Präprozessor

nicht modular



```
#ifdef TXN
lock();
#ifdef STAT
lockCount++;
#endif
#endif
```

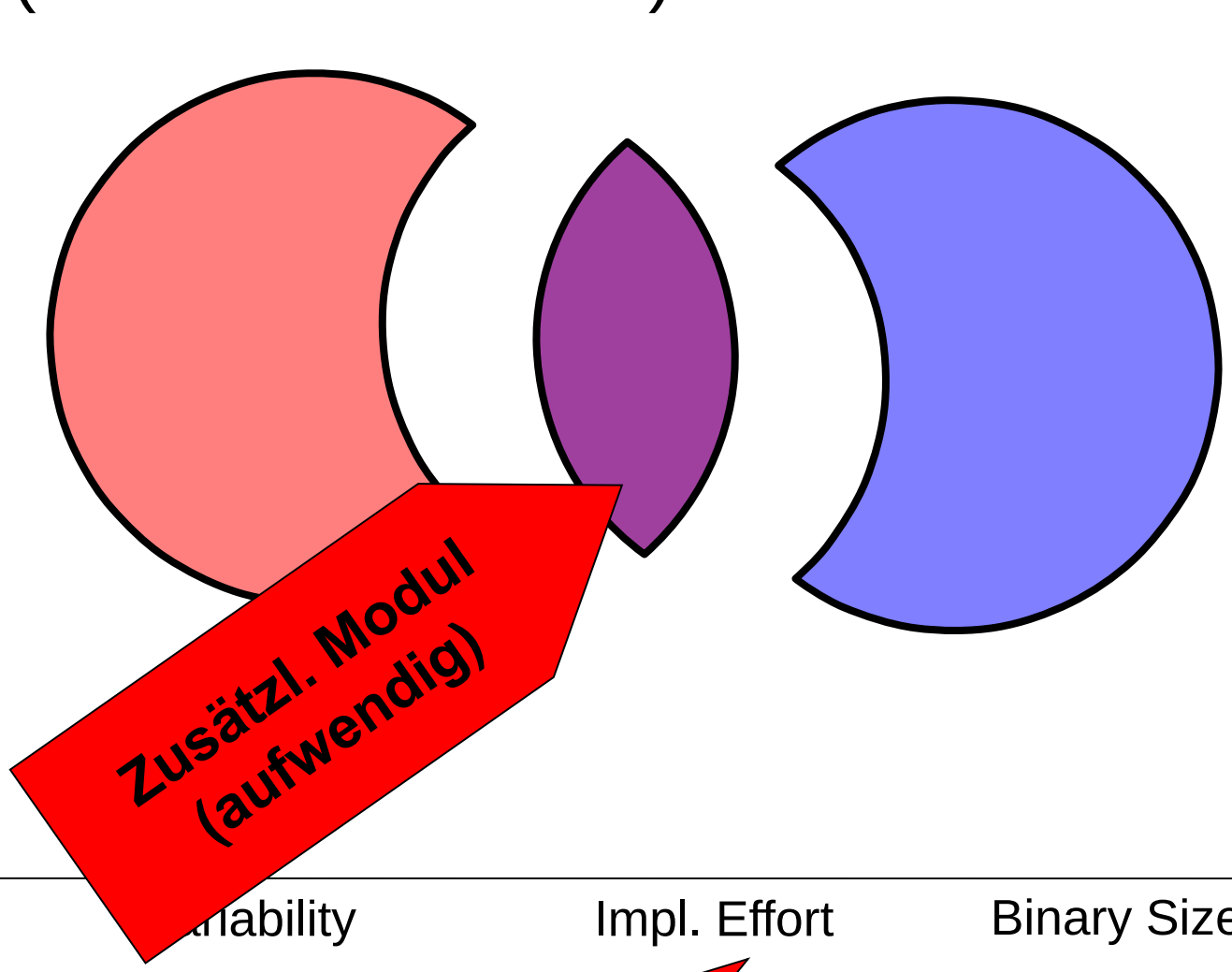
Products



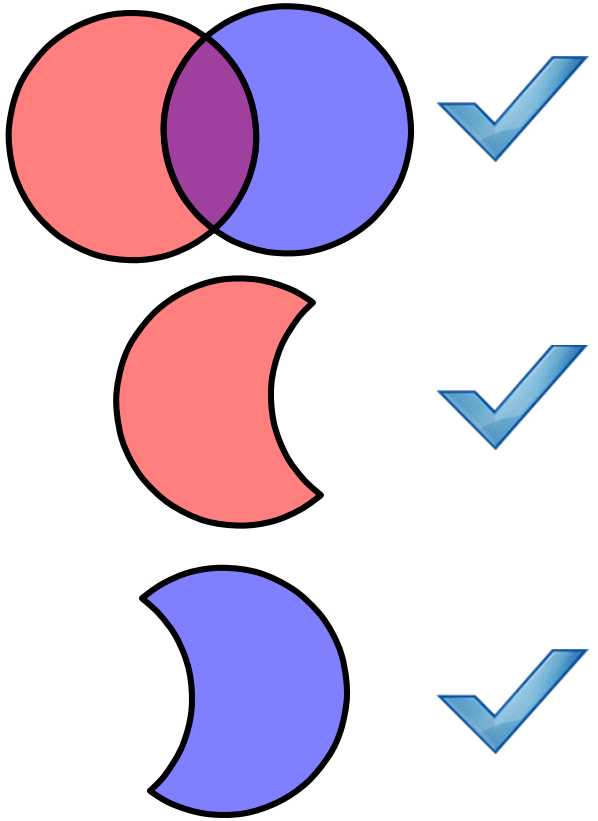
Variability	Impl. Effort	Binary Size & Perf.	Code Quality
✓	✓	✓	⚡

Lösung 6: Interaktion extrahieren

(Glue-Code-Module)



Products

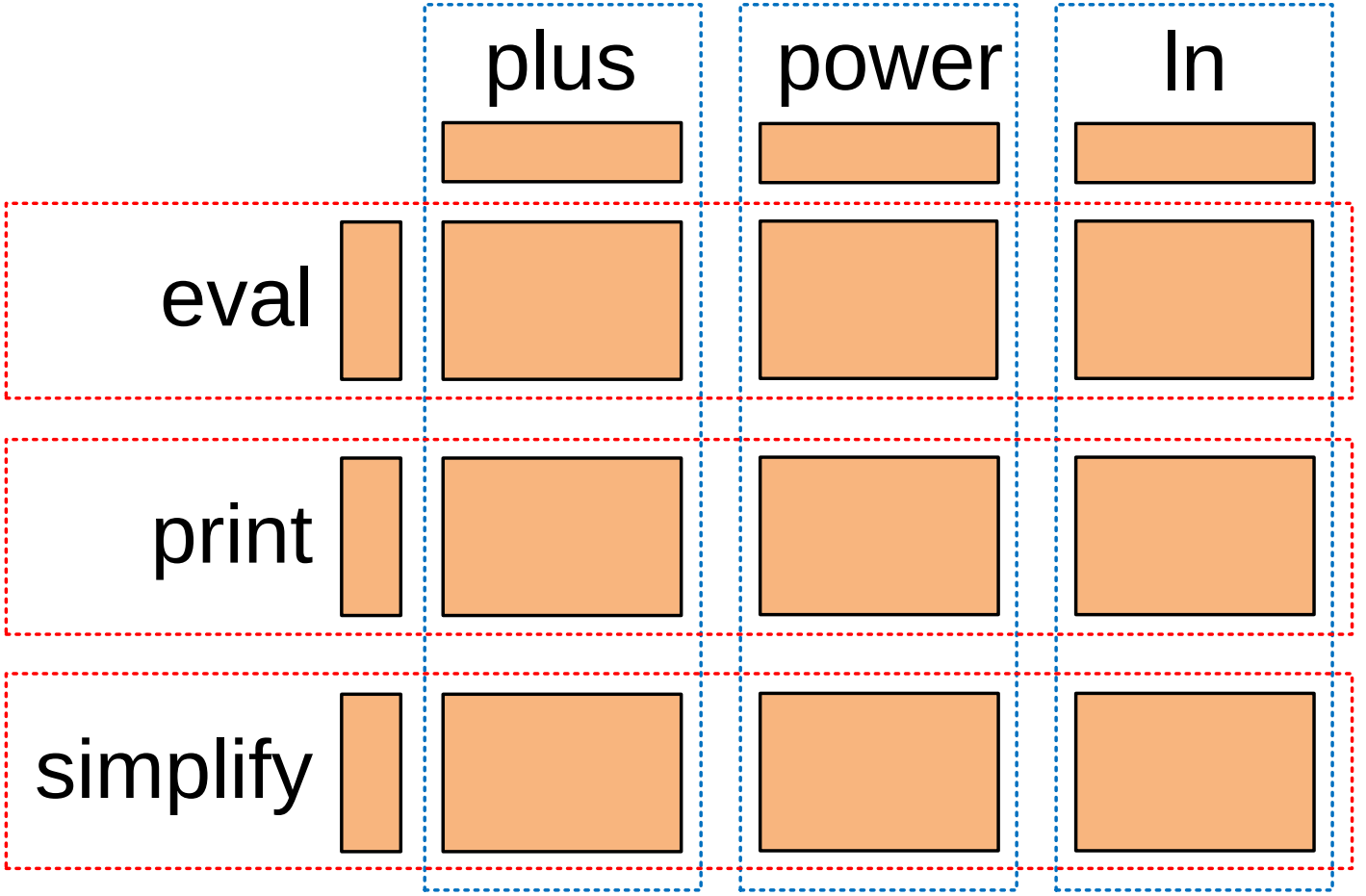


...liability	Impl. Effort	Binary Size & Perf.	Code Quality

Zusammenfassung

Lösung	Variabilität	Aufwand	Größe & Performance	Qualität
Mehrere Implementierungen	✓	⚡	✓	⚡
Abhängigkeiten beibehalten	⚡	✓	✓	✓
Quelltext verschieben	✓	✓	⚡	⚡
Verhalten ändern	?	✓	?	✓
Präprozessor	✓	✓	✓	⚡⚡
Extraktion der Interaktion	✓	⚡	✓	✓

Beispiel: Expression Problem



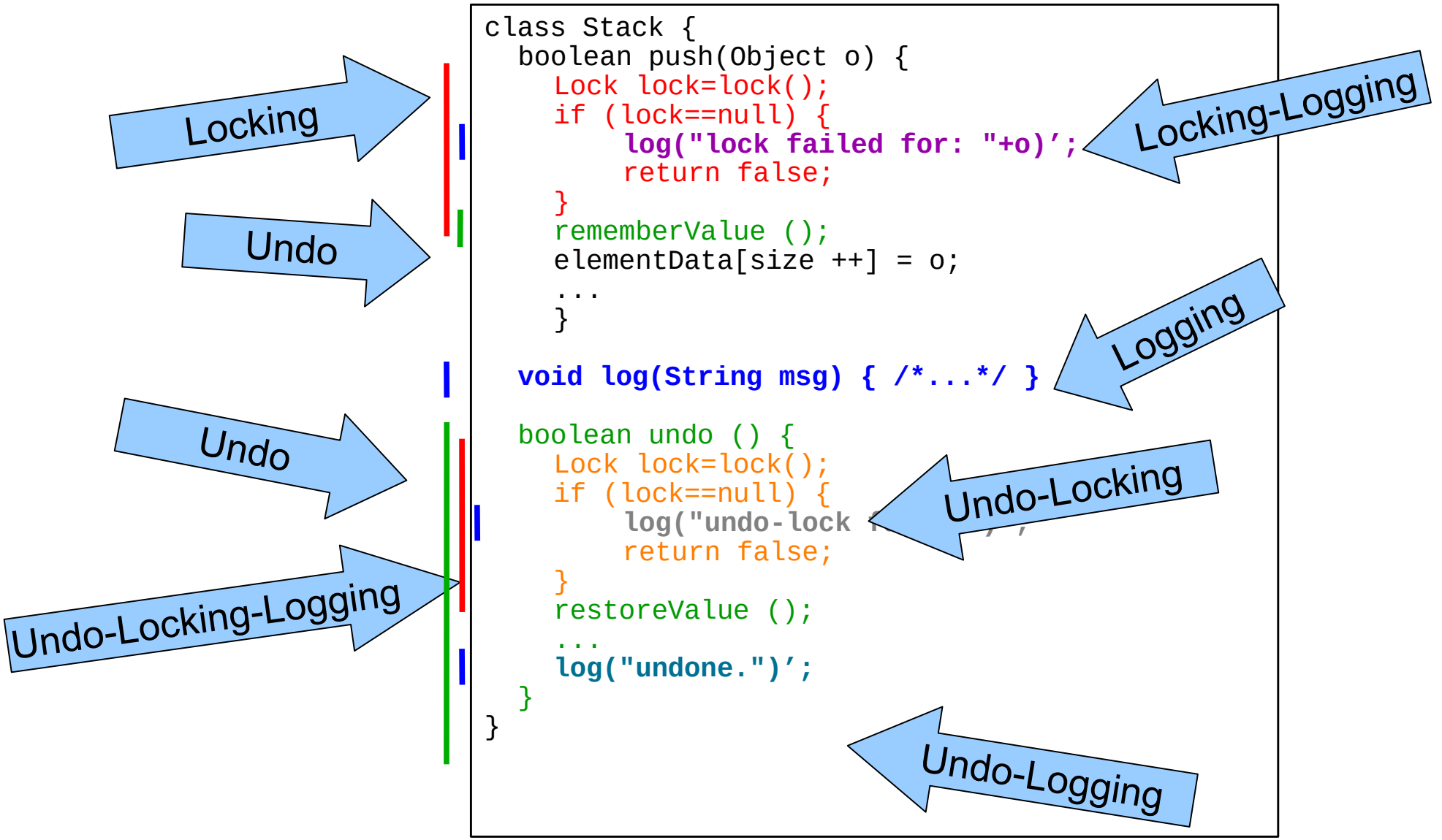
- Fast der gesamte Quelltext würde in neue Module extrahiert
- 6 Features → 15 Module

Paarweise Feature-Kombinationen

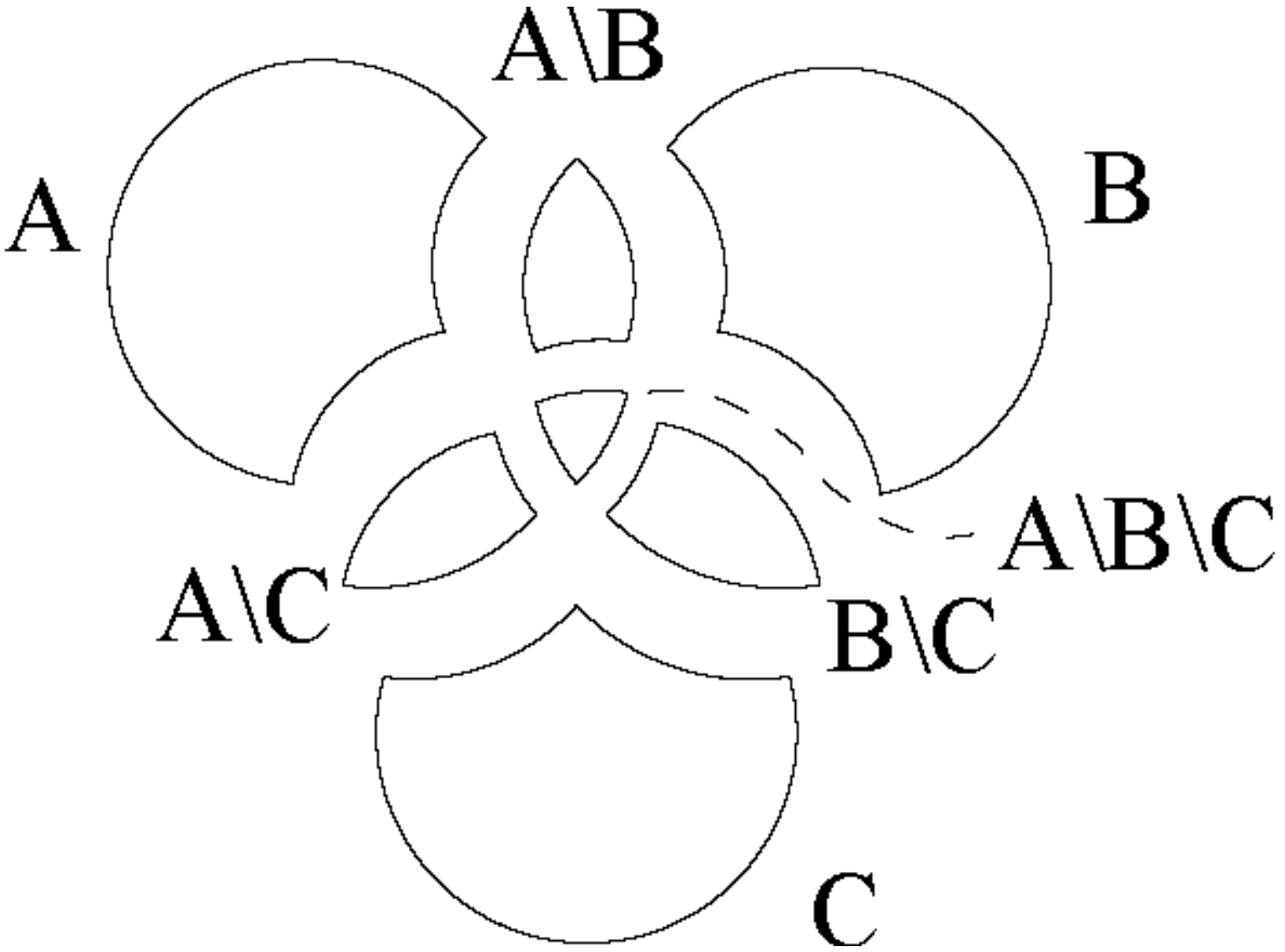
- Manuelles Extrahieren sehr aufwendig
- Zusätzliche Module ➔ höhere Komplexität
- Interaktionen sind oft stark verteilte, heterogene Erweiterungen
- Pro Feature-Paar f, f' vier Kombinationen möglich:
  $(f, f'), (\neg f, f'), (f, \neg f'), (\neg f, \neg f')$
- Dadurch theoretisch sehr viele **Feature-Kombinationen** mit gewollten/ungewollten Interkationen möglich

$$i_{max} = \binom{n}{2} = \frac{n^2 - n}{2}$$

Beispiel: Multi-Interaktionen



Multi-Interaktionen



Multi-Interaktionen

- Theoretisch mögliche Multi-Interaktionen der Ordnung *o*:

$$h_{\text{max}} = \sum_{o=1}^{n-1} \binom{n}{o-1} = 2^n - n - 1$$

- Tatsächliche Anzahl
  - Deutlich geringer
  - Aber trotzdem häufig mehr als Features im System

- Detektion und Analyse?
  - **Gewollte Interkationen richtig implementiert?**
  - **Ungewollte Interkationen treten nicht auf?**

Referenzen

- *J. Liu, D. Batory, and C. Lengauer:* **Feature Oriented Refactoring of Legacy Applications.** In Proc. Int'l Conf. on Software Engineering, 2006.
- *C. Kästner, S. Apel, S. S. ur Rahman, M. Rosenmüller, D. Batory, and G. Saake:* **On the Impact of the Optional Feature Problem: Analysis and Case Studies.** In Proc. Int'l Software Product Line Conference (SPLC), 2009.
- *P. Zave:* **Feature Interactions and Formal Specifications in Telecommunications.** IEEE Computer 26(8): 20-30 (1993)
- *M. Calder, M. Kolberg, E.H. Magill, S. Reiff-Marganiec:* **Feature Interaction: A critical review and considered forecast.** *Computer Networks*, Volume 41, Issue 1, 2003, pp. 115-141
- *M. Lochau, S. Oster, U. Goltz, A. Schürr:* **Model-based Pairwise Testing for Feature Interaction Coverage in Software Product Line Engineering.** In: Software Quality Journal, 20:567-604, 2012
- *A. Metzger:* **Feature Interactions in Embedded Control Systems.** Computer Networks, 45, pp. 625 – 644, 2004