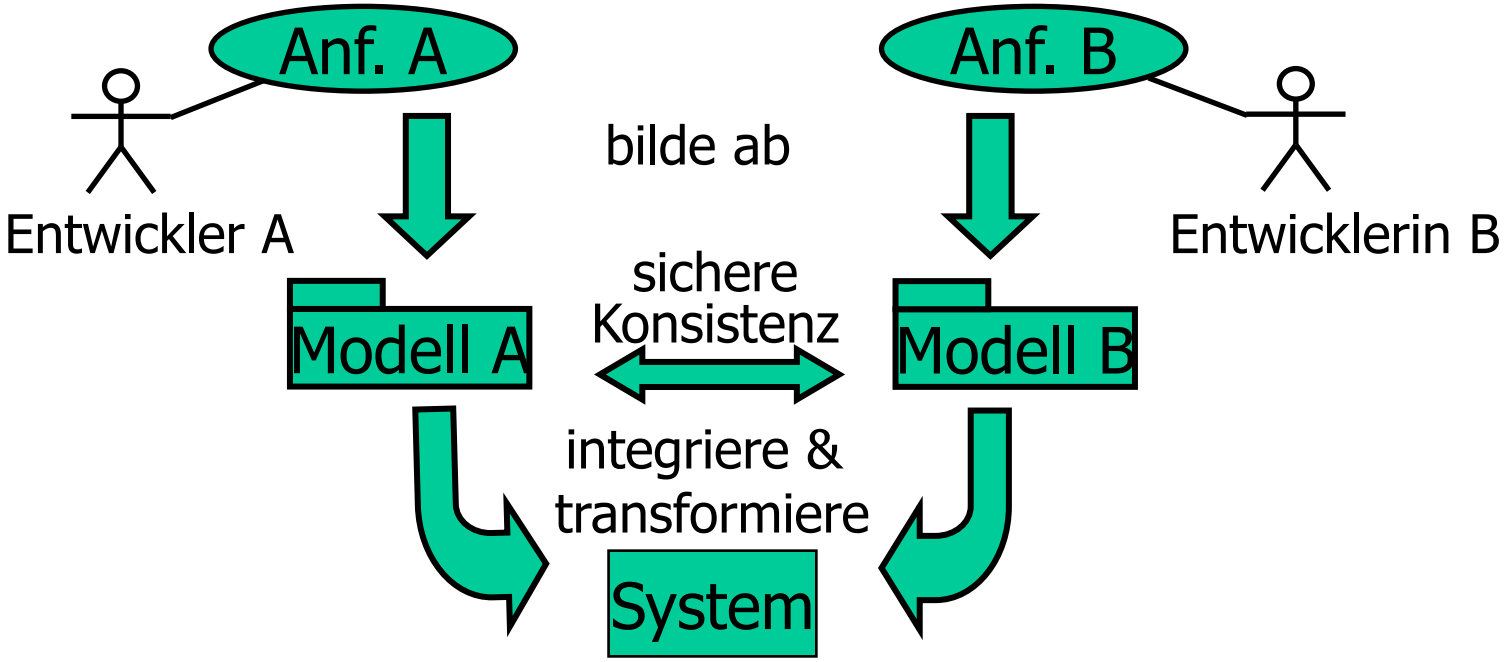
Erkennen von inkonsistenten Anforderungen

Jens Kosiol

Überblick

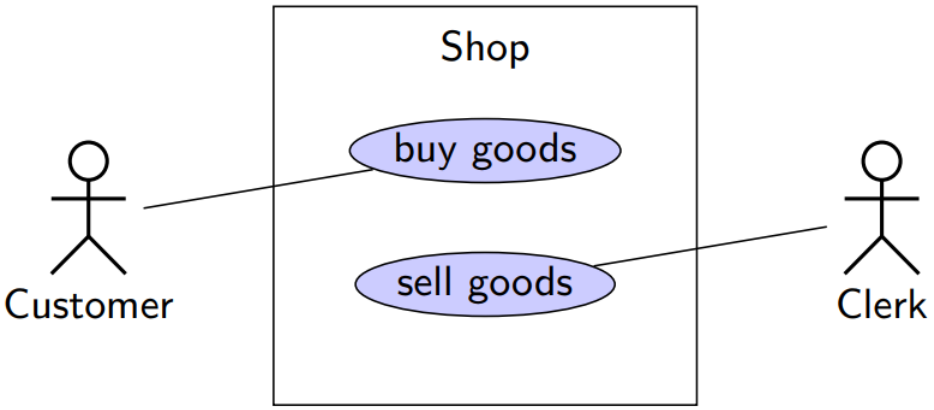
- Anforderungsbeschreibungen für Software enthalten Informationen zu:
  - *Objekten und Beziehungen im System*
  - *Prozessen und Aktivitäten des Systems*
  - *Benutzern und ihren Sichten auf das Systems*
- Wenn Anforderungen im Team spezifiziert werden:
  - *Welche Arten von Inkonsistenzen können zwischen verschiedenen Aspekten von Anforderungsbeschreibungen auftreten?*
- Gibt es ein automatisches Verfahren, um positive und negative Anzeichen für Konsistenz zu finden?

Motivation: Softwareentwicklung als Integration von Sichten



1. Mehrere Entwickler spezifizieren Anforderungen in natürlicher Sprache.
2. Die Anforderungsspezifikationen werden in Modelle übersetzt.
3. Die Modelle werden auf Konsistenz überprüft.

Beispiel: Anforderungen an einen Shop



[HT20]

Buy goods: Kundensicht

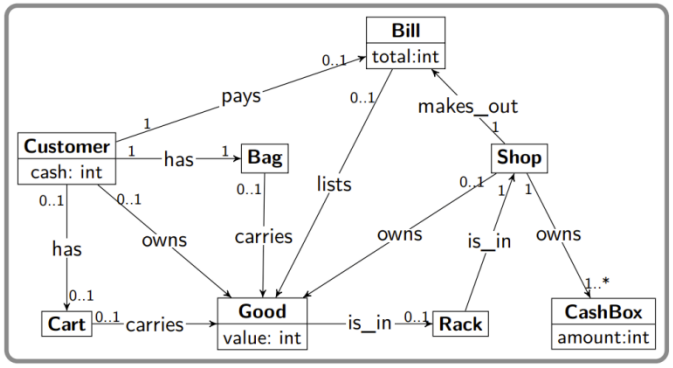
- Ablauf: Der Kunde nimmt einen Wagen und wählt alle Waren aus den Regalen, die er kaufen möchte. Dann wird eine Rechnung erstellt. Der Kunde bezahlt die Waren. Nach dem Kauf gehören die Waren dem Kunden.

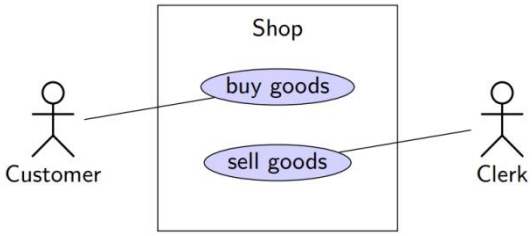
Sell goods: Verkaufssicht

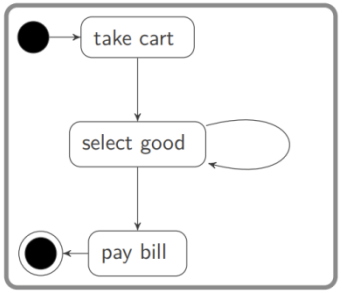
- Die Verkäuferin erstellt eine Rechnung und setzt alle Waren des Kunden auf die Rechnung. Dann schließt sie die Rechnung ab und kassiert. Die Waren gehören nun dem Kunden und nicht mehr dem Shop.

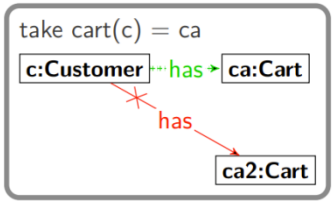
Spezifikation von Anforderungen

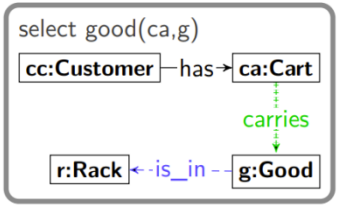
- Statisch:
  - *Klassendiagramme*
  - *Constraints (Graphformeln)*
- Dynamisch:
  - *Anwendungsfalldiagramme*
  - *Verfeinerung von Anwendungs-fällen durch Aktivitätsdiagramme*
- Funktional:
  - *Vor- und Nachbedingungen von Aktionen in Aktivitätsdiagrammen*





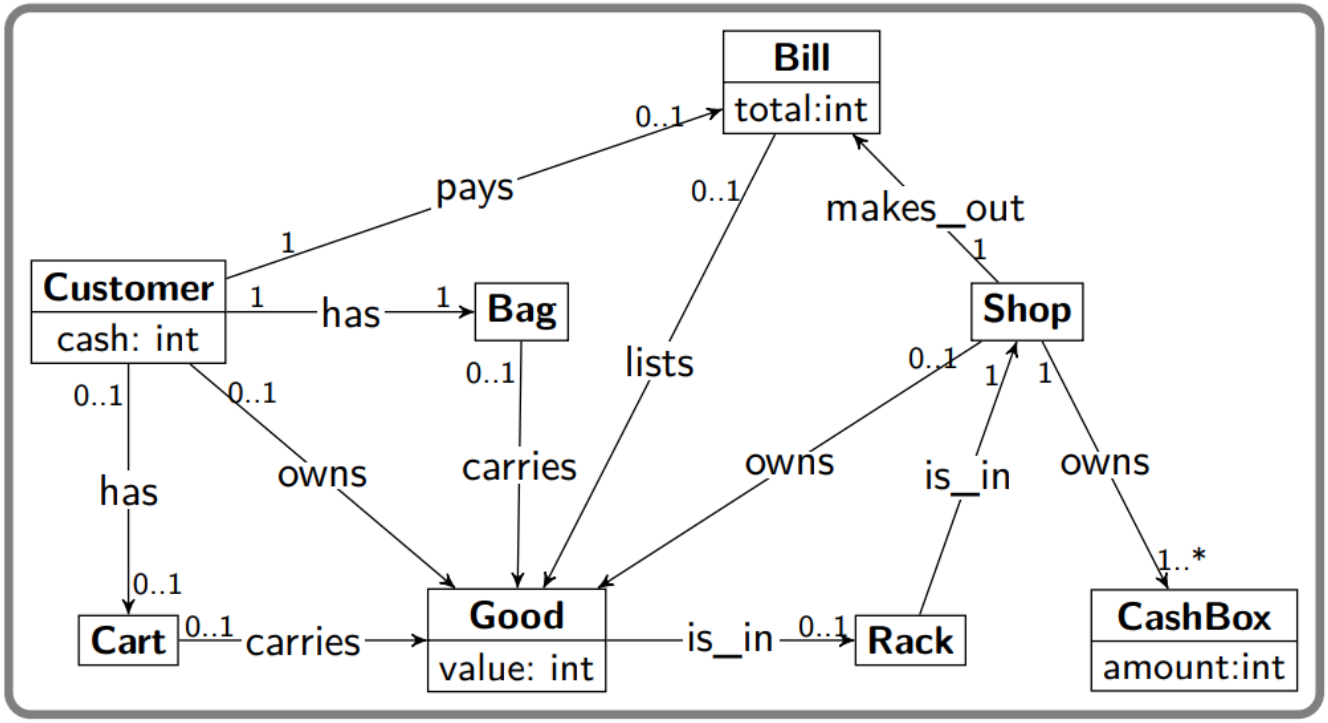






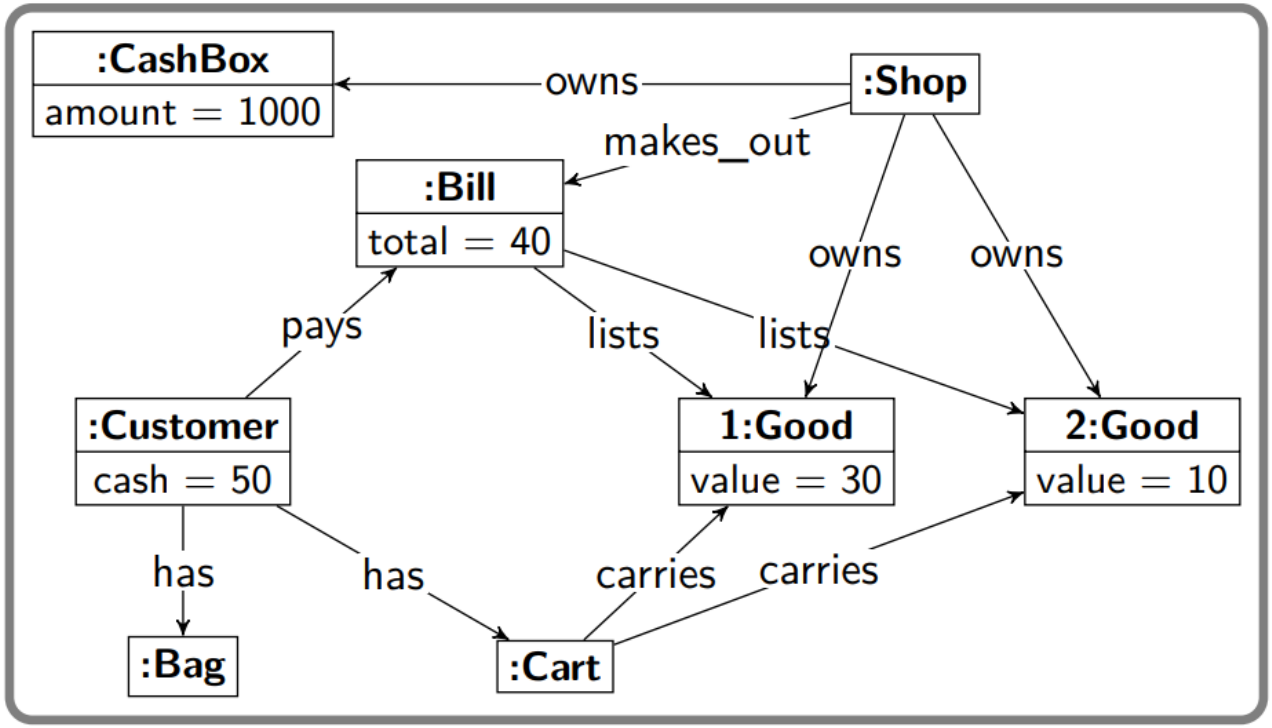
[HT20]

Beispiel: Anforderungen an statische Strukturen in Klassendiagrammen



[HT20]

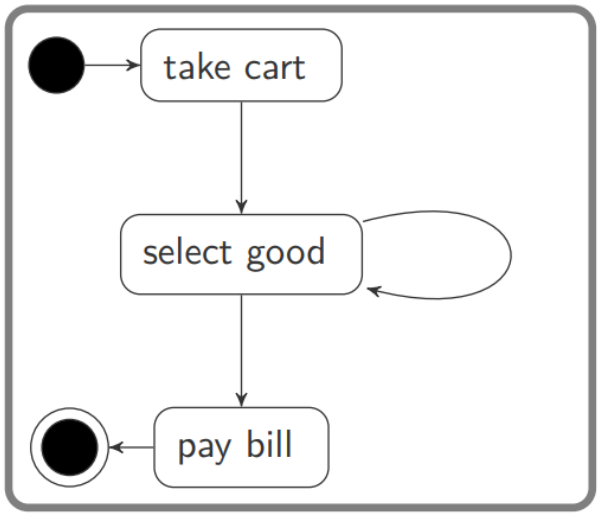
Beispiel: Ein möglicher Zustand des Systems



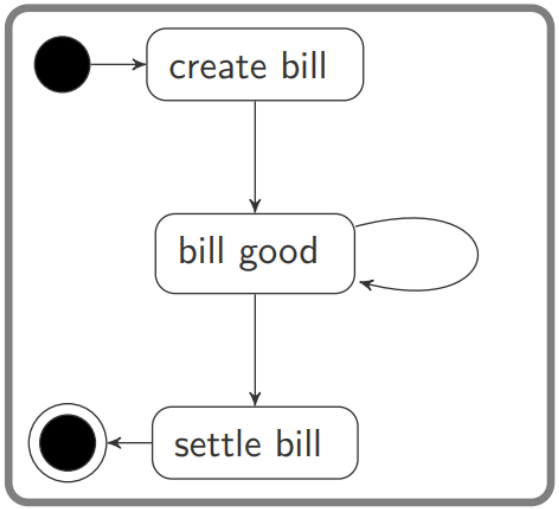
[HT20]

Beispiel: Dynamische Anforderungen an einen Shop

Buy goods:



Sell goods:



[HT20]

- Verfeinerung von Anwendungsfällen durch Aktivitätsdiagramme
- Sind diese Aktivitätsdiagramme konsistent mit dem Klassenmodell?
- Sind die definierten Prozesse konsistent zueinander?

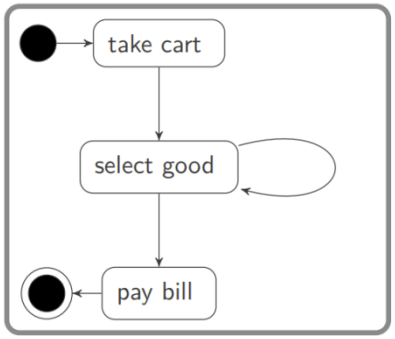
Arten von Inkonsistenzen

- Typinkonsistenz
  - *Dynamische und funktionale Anforderungen enthalten Begriffe, die nicht im Klassendiagramm vorkommen oder dort umbenannt oder umdefiniert wurden.*
  - *Ausführungen von Aktivitätsdiagrammen können unklar sein oder Constraints verletzen.*
- Inkonsistenz zw. dynamischen u. funktionalen Aspekten
  - *Der Kontrollfluss eines Aktivitätsdiagramms passt nicht zu einer oder mehreren Vor- und Nachbedingungen der enthaltenen Aktionen.*
- Inkonsistenz von Sichten:
  - *Verfeinerte Spezifikationen von Anwendungsfällen verschiedener Nutzer behindern sich oder sind voneinander abhängig.*

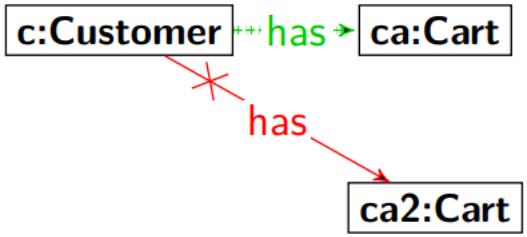
Verschiedene Aspekte in Anforderungsspezifikationen

- Statische Anforderungen: Klassendiagramme
- Dynamische Anforderungen:
  - *Anwendungsfalldiagramme und*
  - *Ihre Verfeinerung durch Aktivitätsdiagramme*
- Wie hängen statische und dynamische Anforderungen zusammen?
  - *Funktionale Aspekte:*
    - Verfeinerung einzelner Aktivitäten durch Vor- und Nachbedingungen über Systemzuständen
    - Spezifikation einer Verfeinerung durch eine Regel

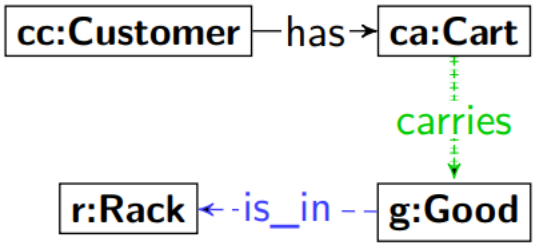
Beispiel: Funktionale Anforderungen



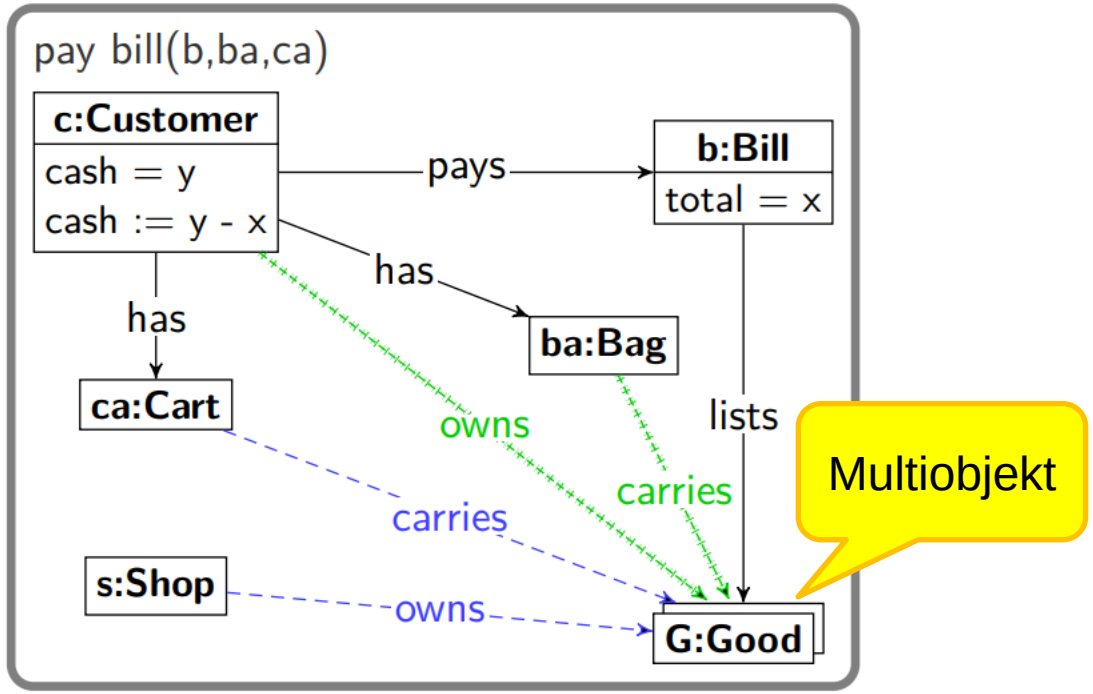
take cart(c) = ca



select good(ca,g)

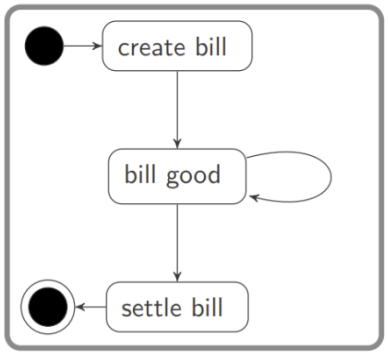


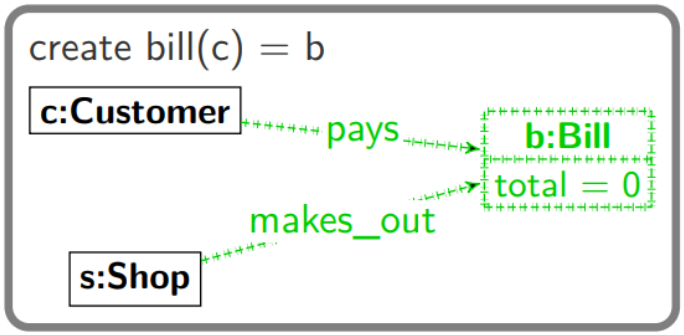
pay bill(b,ba,ca)

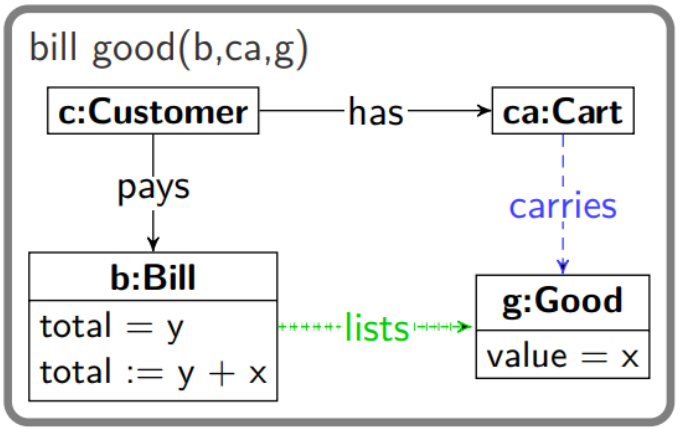


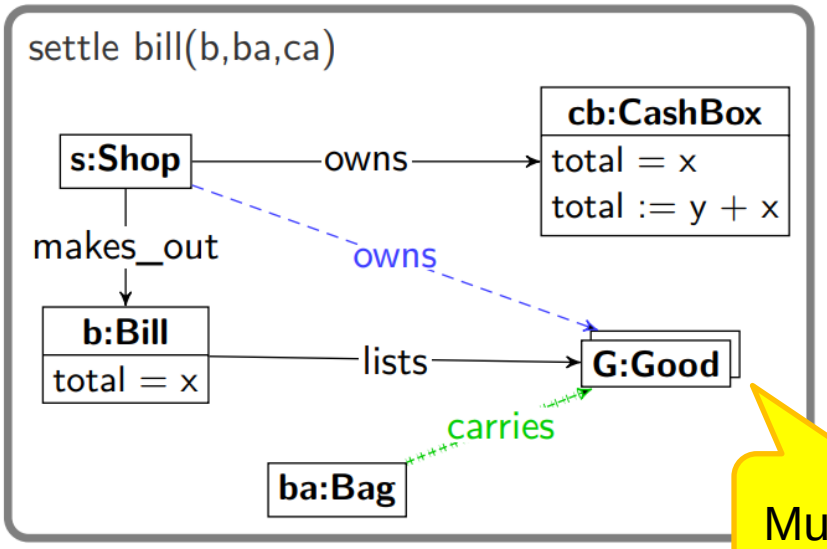
[HT20]

Beispiel: Funktionale Anforderungen





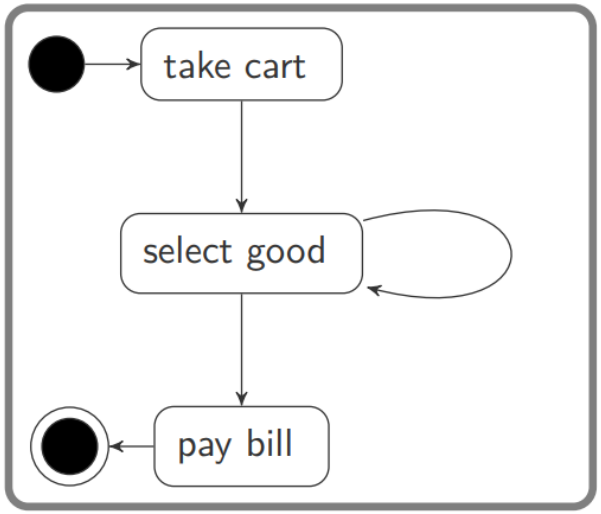




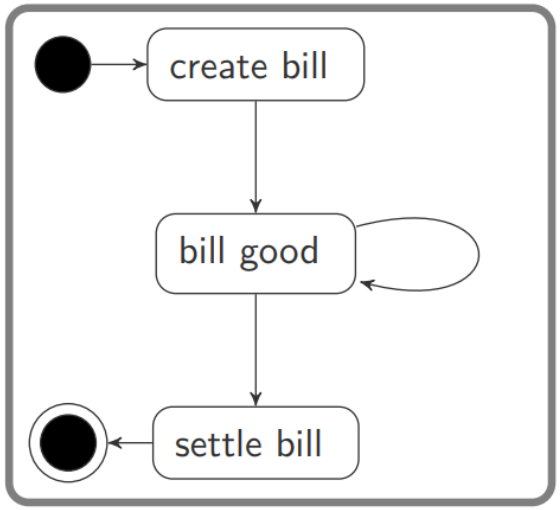
[HT20]

Sind diese beiden Sichten konsistent zueinander?

Buy goods:



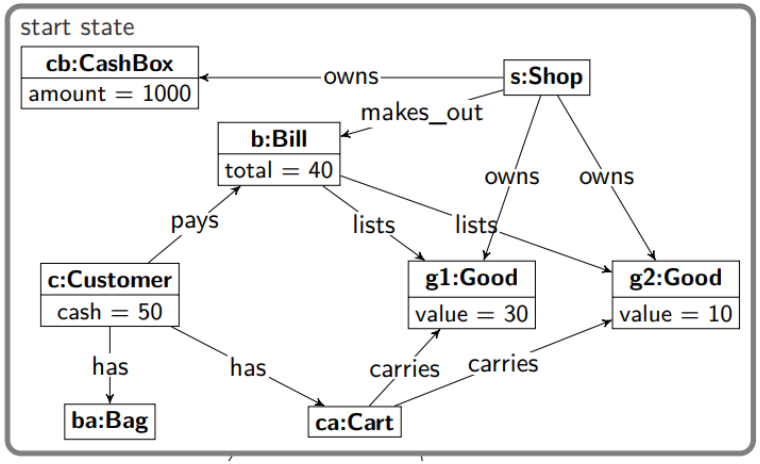
Sell goods:



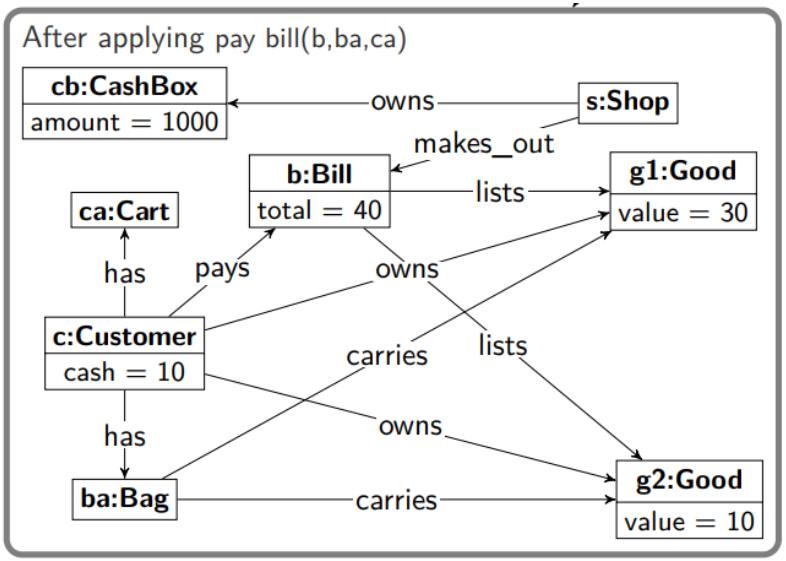
[HT20]

- Welche Konflikte und Abhängigkeiten können zwischen diesen Aktivitäten auftreten?

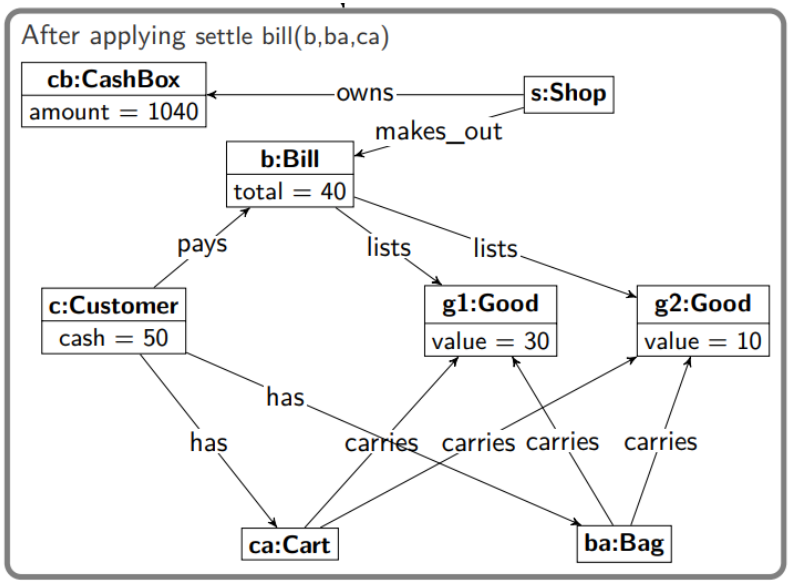
Beispiel: Ein Konflikt



pay bill



settle bill

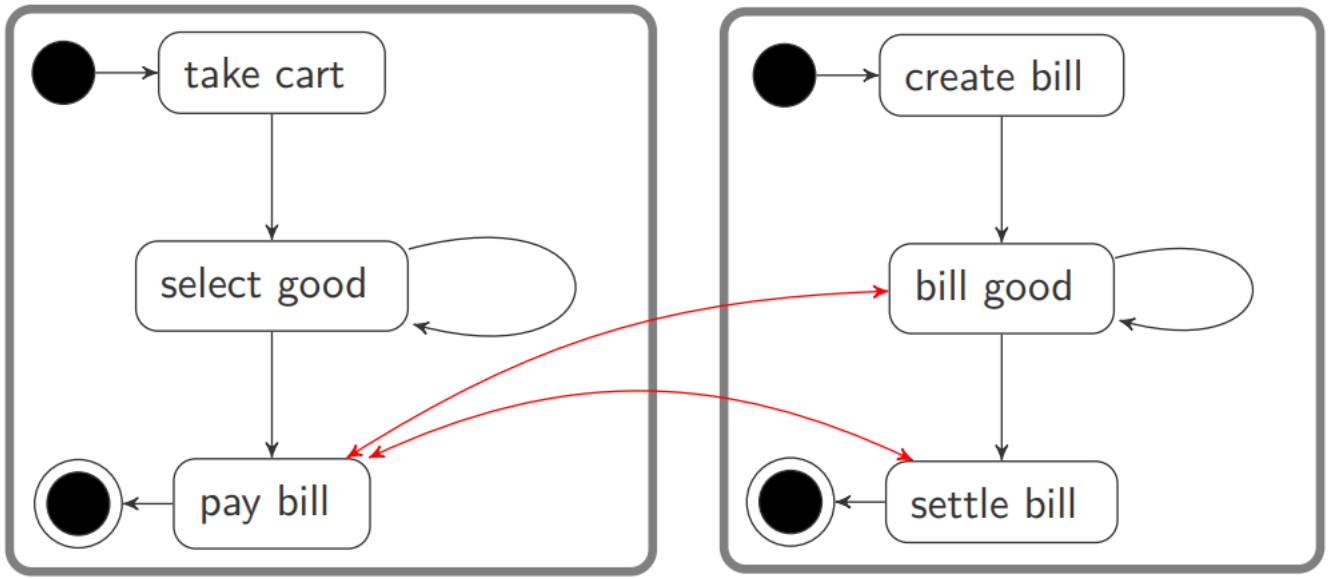


[HT20]

Konsistenz zwischen dynamischen und funktionalen Aspekten

- Beispiele:
  - *Regel B soll nach Regel A angewandt werden, Regel A kann aber mit Regel B in Konflikt stehen.* → *Ein kritisches Zeichen*
  - *Regel B soll nach Regel A angewandt werden und B kann von A abhängig sein.* → *Ein gutes Zeichen*
- Kann die Konflikt- und Abhängigkeitsanalyse Inkonsistenzen aufdecken?
  - *Da sie potentielle Konflikte und Abhängigkeiten findet, kann sie Anzeichen für (In)konsistenzen berechnen.*
  - *Falls Aktivitätsdiagramme mit Objektfluss genutzt werden, kann sie auch echte Inkonsistenzen finden.*

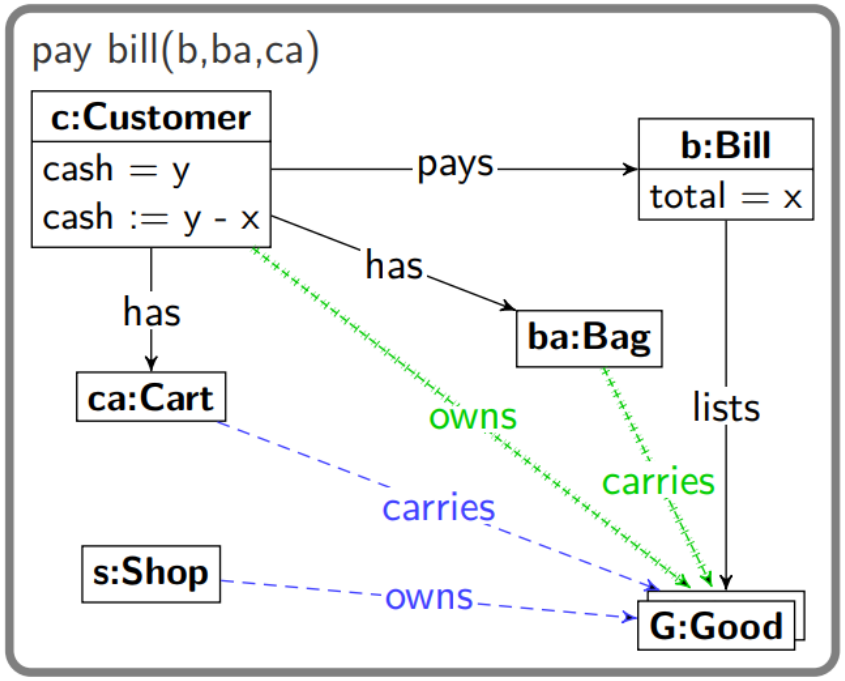
Beispiel: Aktivitätsdiagramme mit potentiellen Konflikten

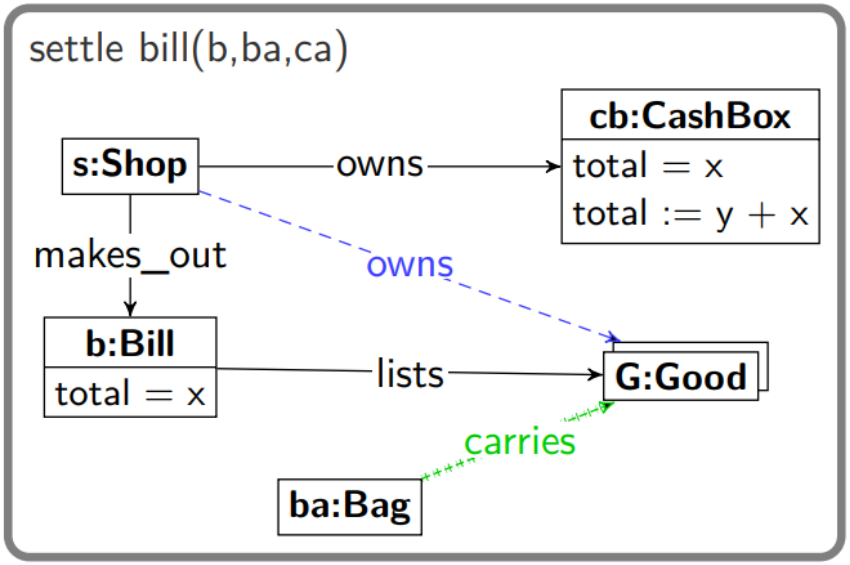


[HT20]

- Welche potentiellen Konflikte können auftreten?
- Können diese durch Änderungen der Aktivitäts-diagramme aufgelöst werden?

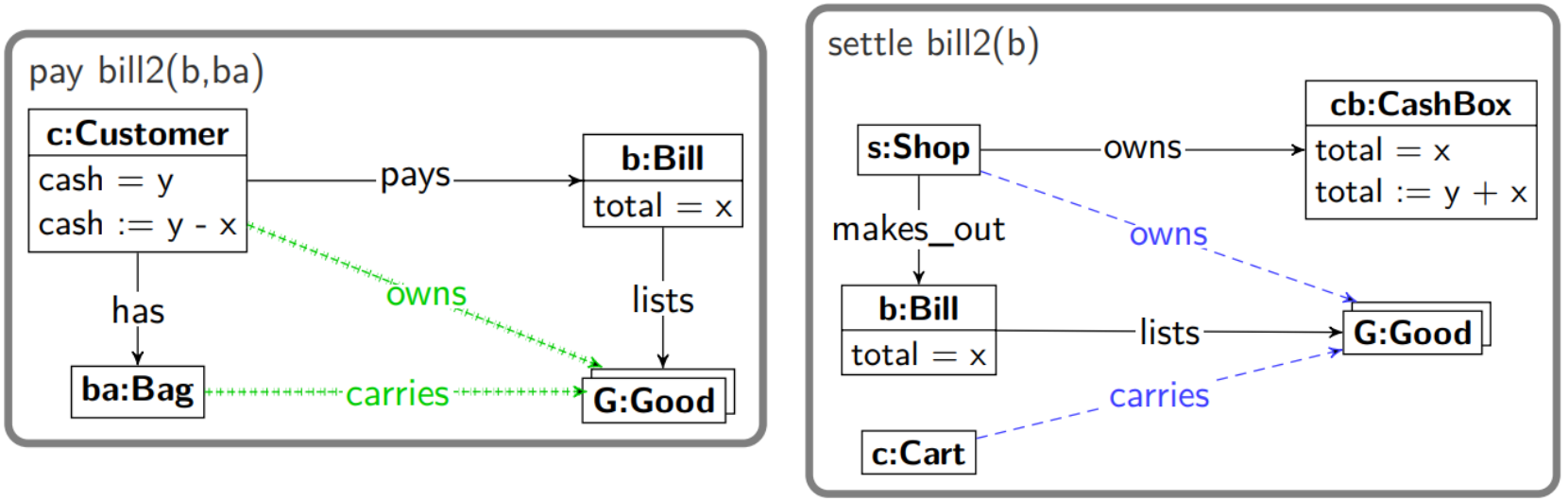
Welche Konflikte können auftreten?





[HT20]

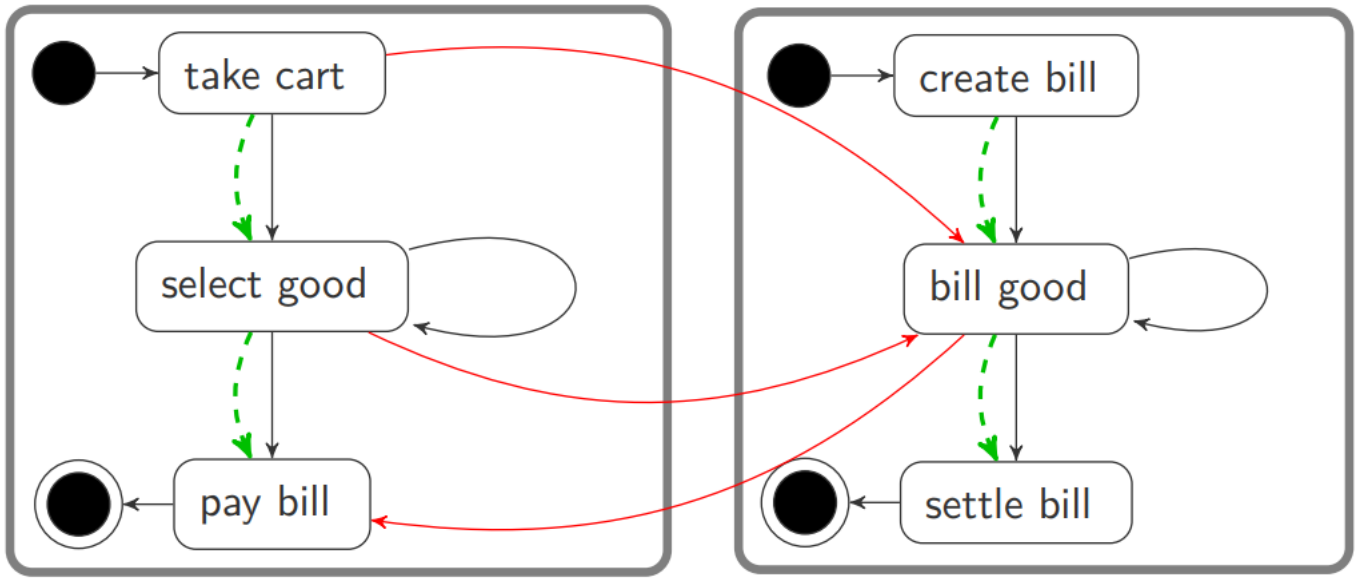
Beispiel: Konfliktlösung



[HT20]

- Die Anwendungsfälle werden an die jeweilige Sicht angepasst.

Beispiel: Aktivitätsdiagramme mit potentiellen Abhängigkeiten



[HT20]

- *Gute Anzeichen: pot. Abhängigkeiten entlang des Kontrollflusses*
- *Kritische Anzeichen: potentielle Abhängigkeiten zwischen Anwendungsfällen verschiedener Sichten*

Gute und kritische Anzeichen für Konsistenz: Konflikte

	With control flow	Against control flow	No control flow
Conflict	**Critical sign**	—	**Critical sign**
No conflict	**Favourable sign**	—	**Favourable sign**
Dependency	**Favourable sign**	**Critical sign**	**Critical sign**
No dependency	**Critical sign**	**Favourable sign**	**Favourable sign**

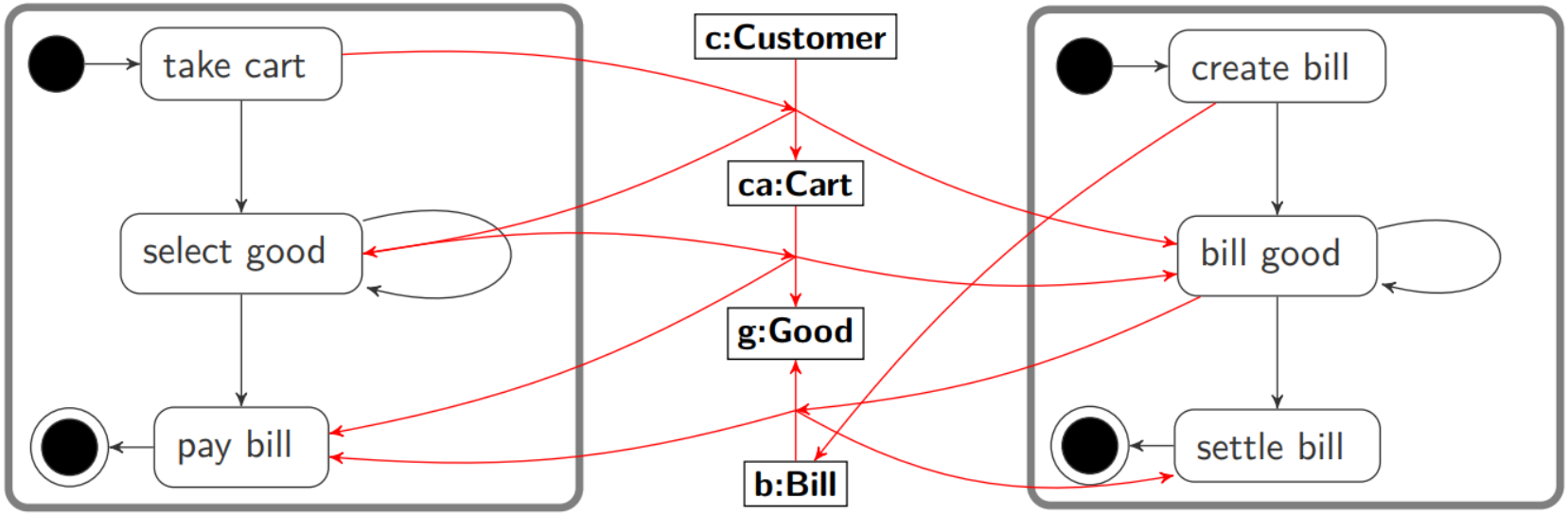
- Konsistenz zwischen Regeln A,B und Kontrollfluss:
  - *A → B, A kann Konflikt mit B verursachen: kritisch*
  - *A → B, A kann keinen Konflikt mit B verursachen: gut*
  - *B → A, A kann Konflikt mit B verursachen: egal*
  - *B → A, A kann keinen Konflikt mit B verursachen: egal*
  - *A und B nicht in Kontrollflussabhängigkeit:*
    - A kann Konflikt mit B verursachen: kritisch
    - A kann keinen Konflikt mit B verursachen: gut

Gute und kritische Anzeichen für Konsistenz: Abhängigkeiten

	With control flow	Against control flow	No control flow
Conflict	**Critical sign**	—	**Critical sign**
No conflict	**Favourable sign**	—	**Favourable sign**
Dependency	**Favourable sign**	**Critical sign**	**Critical sign**
No dependency	**Critical sign**	**Favourable sign**	**Favourable sign**

- Konsistenz zwischen Regeln A,B und Kontrollfluss:
  - *A $\rightarrow$ B, B kann von A abhängig sein: gut*
  - *A $\rightarrow$ B, B kann nicht von A abhängig sein: kritisch*
  - *B $\rightarrow$ A, B kann von A abhängig sein: kritisch*
  - *B $\rightarrow$ A, B kann nicht von A abhängig sein: gut*
  - *A und B nicht in Kontrollflussabhängigkeit:*
    - B kann von A abhängig sein : kritisch
    - B kann nicht von A abhängig sein: gut

Beispiel: Aktivitätsdiagramme mit Gründen für Abhängigkeiten



[HT20]

- *Gründe für Abhängigkeiten zeigen die potentiell in Abhängigkeit stehenden Objekte und Referenzen.*
- *Kann mit Aktivitätsdiagrammen mit Objektfluss verglichen werden.*

Zusammenfassung

- Spezifikation von Anforderungen:
  - *Statische Strukturen: Klassendiagramme*
  - *Dynamisch: Aktivitätsdiagramme (definieren Kontrollflüsse)*
  - *Funktional: Vor- und Nachbedingungen (in Regeln dargestellt)*
- Inkonsistenzen zwischen Anforderungsaspekten:
  - *Statisch – funktional: Typisierungsfehler oder Verletzung von Invarianten*
- Hier im Fokus:
  - *Inkonsistenzen zwischen funktionalen und dynamischen Anforderungen*
  - *Inkonsistenzen zwischen Sichten*
  - *Konflikt- und Abhängigkeitsanalyse hilft Inkonsistenzen zu finden.*