KobrA und MDA

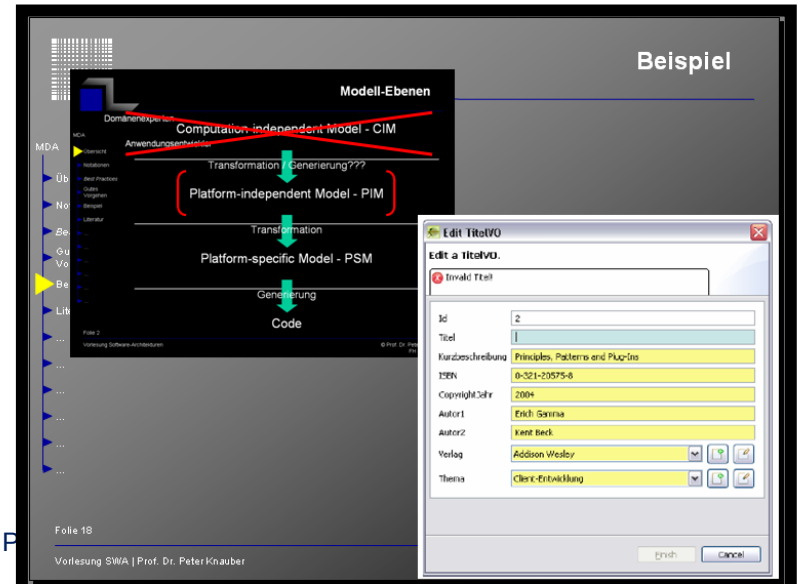
- Motivation / Problem
- KobrA
- Die KobrA-Modelle
- KobrA vs. CIM / PIM
- Code-Generierung
- Literatur

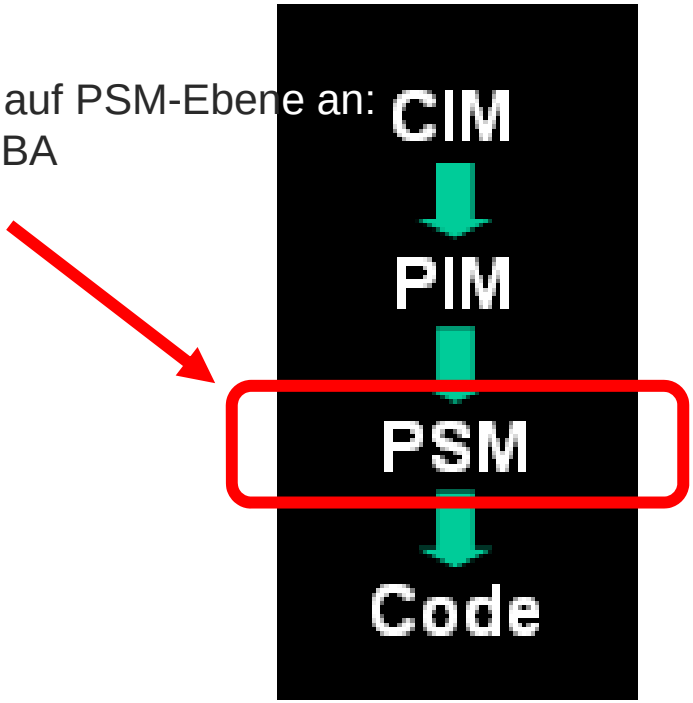
Ausgangspunkt

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

- Modellgetriebener Entwicklungsansatz (MDA)
  - 3 Ebenen
    - → CIM
    - → PIM
    - → PSM
- Komponentenbasierte Entwicklungsansätze
  - *Divide and Conquer*
  - Alle Komponenten*technologien* setzen auf PSM-Ebene an: EJB, OSGI, .NET, Web Services, CORBA





Gesucht ist…

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

ein komponentenbasierter Ansatz, der die „Masse“ an verfügbaren UML-Diagrammen einschränkt und eine klare Verwendung vorgibt

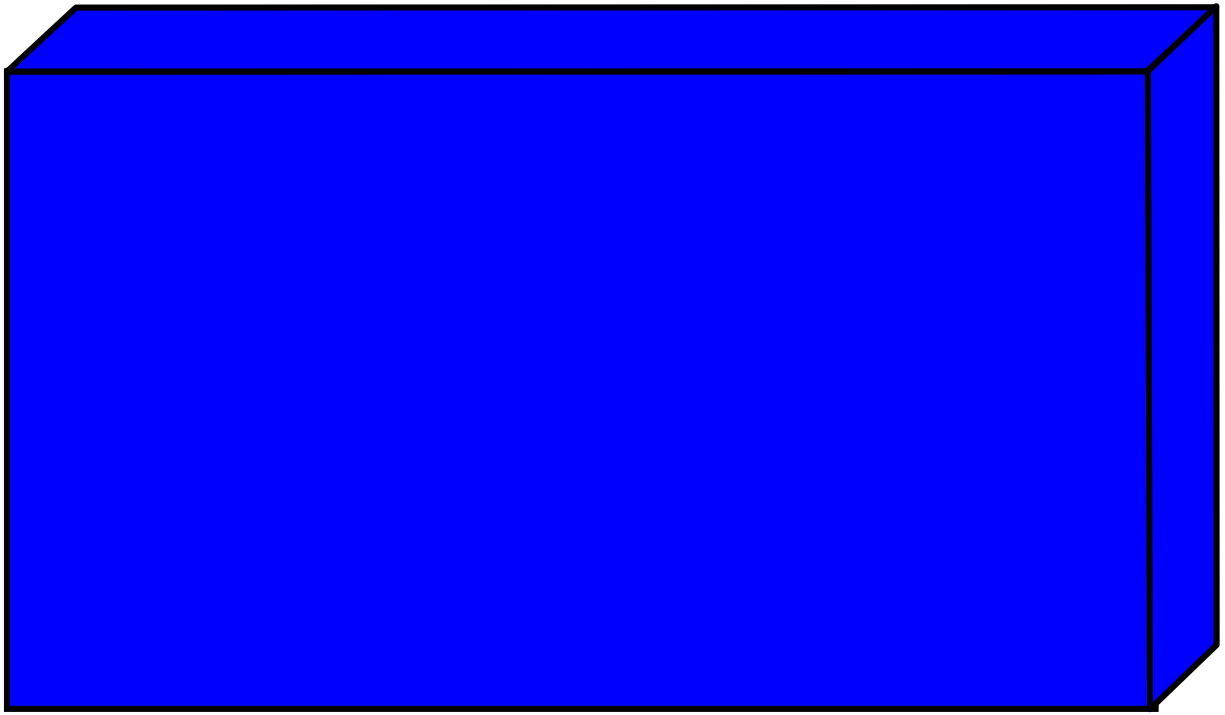
Den gibt es: KobrA

Sicht auf ein Software-System

Ein Software-System ist
aus Sicht des Benutzers
eine (große) Komponente/Klasse

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

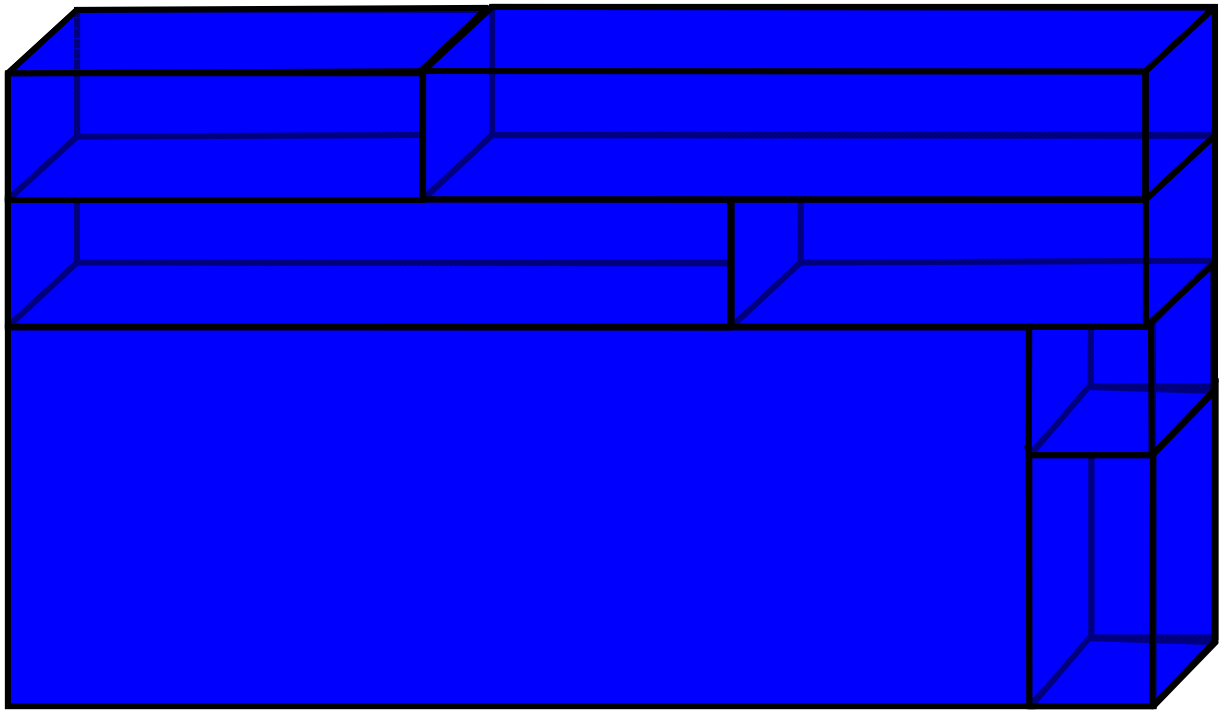


Sicht auf ein Software-System

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

Aus Sicht des Entwicklers
besteht ein Software-System aus
mehreren Komponenten/Klassen

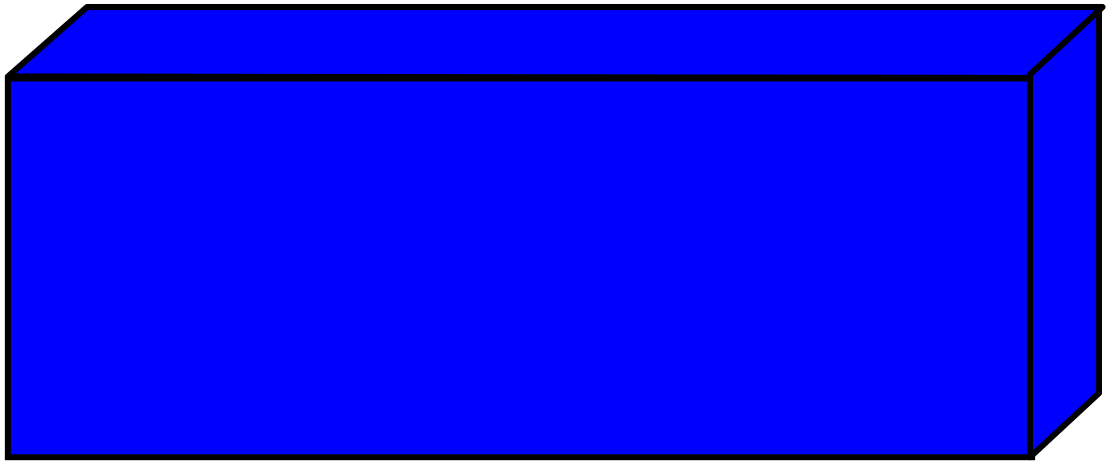


Sicht auf ein Software-System

Große Komponenten/Klassen des Systems kann man weiter zerlegen

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

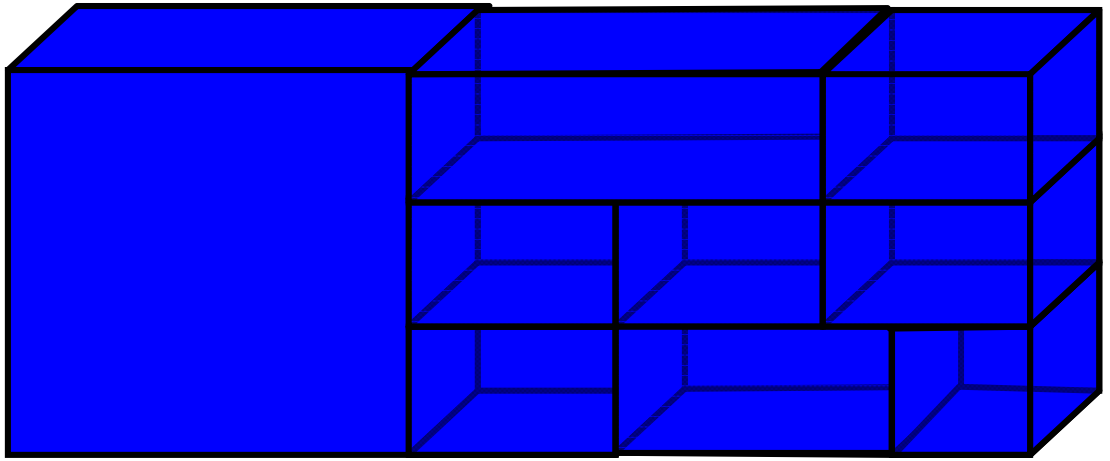


Sicht auf ein Software-System

Große Komponenten/Klassen des Systems kann man weiter zerlegen

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

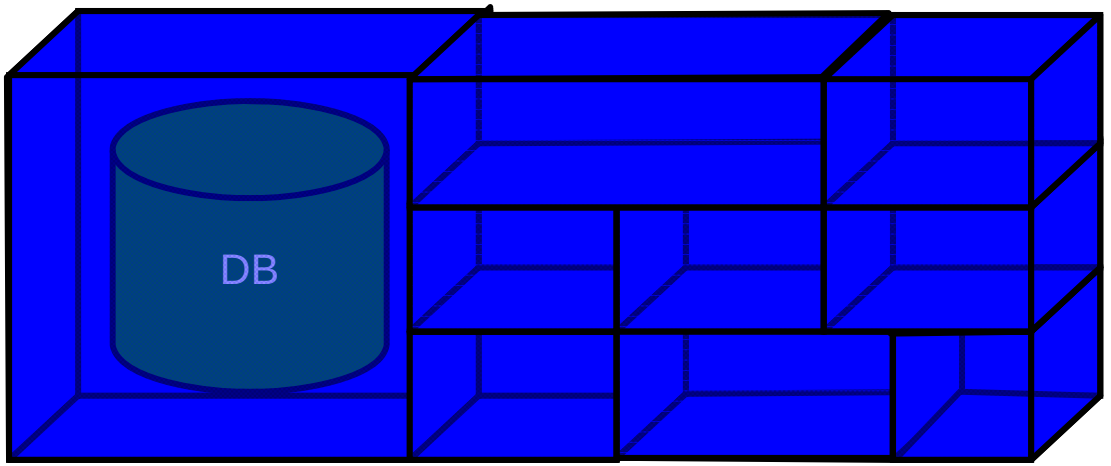


Sicht auf ein Software-System

Je tiefer man einsteigt, desto mehr Details werden sichtbar

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

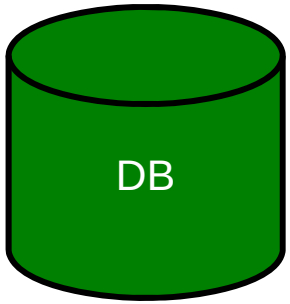


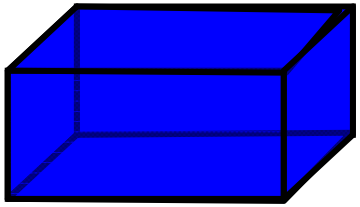
Sicht auf ein Software-System

Am Ende bleiben einzelne,
gut durchschaubare
Komponenten/Klassen übrig

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...



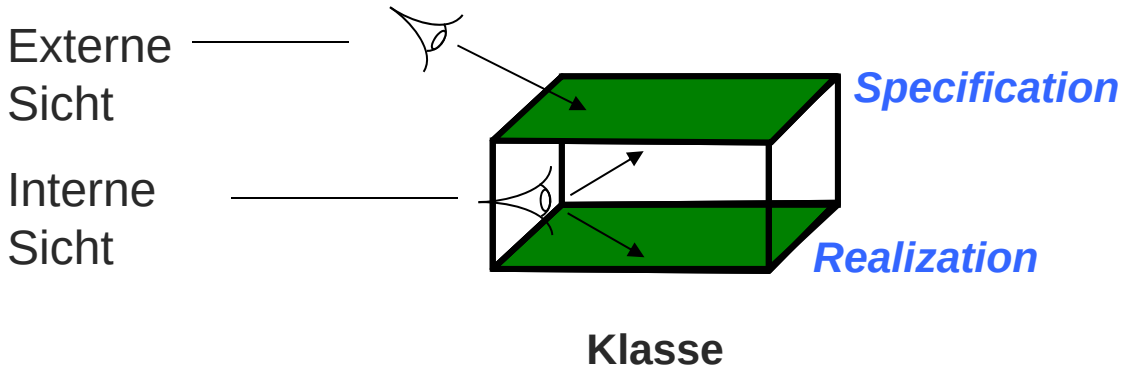


Sichten auf eine Klasse: interne versus externe Sicht

KobrA + MDA
- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

- Die Eigenschaften einer Klasse kann man aus zweierlei Blickwinkel betrachten:
  Externe Sicht
  - Das sind diejenigen Eigenschaften einer Klasse, die nach Außen sichtbar sind (sichtbar sein sollen)
  - In KobrA werden diese durch die **Specification** beschrieben

  Interne Sicht
  - Das sind diejenigen Eigenschaften der Klasse, die innerhalb sichtbar sind
  - Das schließt die von Außen sichtbaren ein!
  - In KobrA werden diese durch die **Realization** beschrieben

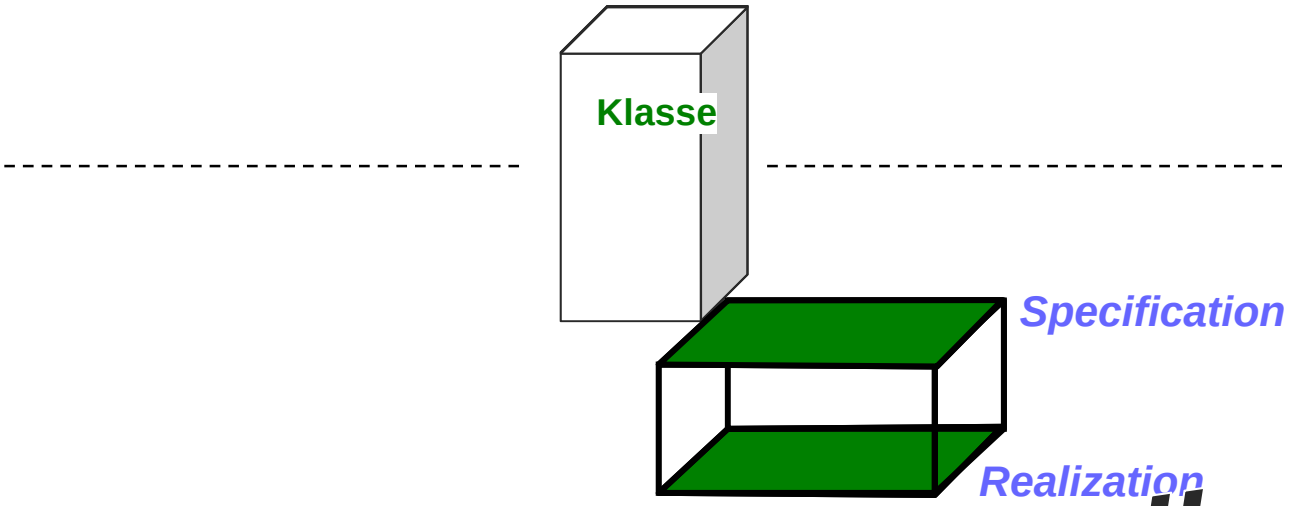


Sichten auf eine Klasse: interne versus externe Sicht

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

Spezifikationsmodelle



Realisierungsmodelle

KobrA-Modelle für die beiden Sichten

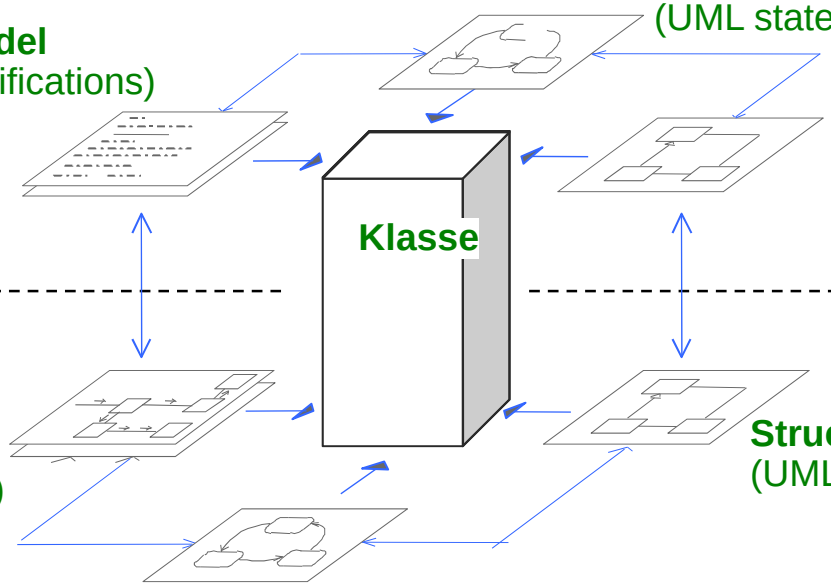
KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

Spezifikationsmodelle

Behavioral Model
(UML statechart diagram)

Functional Model
(operation specifications)



Structural Model

Interaction Model
(UML collaboration diagrams)

Structural Model
(UML class/object diagrams)

Activity Model
(UML activity, use case diagrams)

Realisierungsmodelle

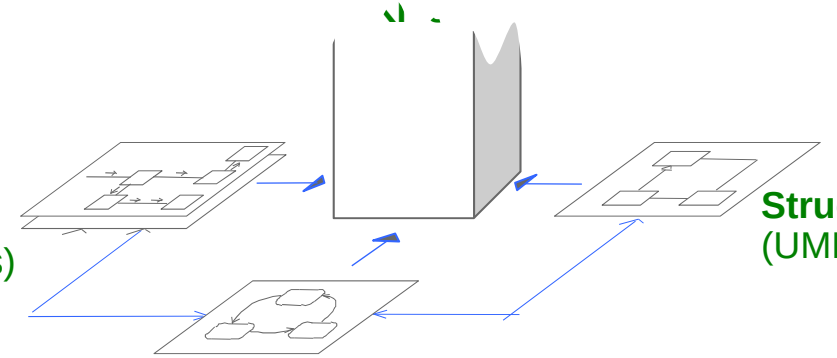
KobrA - Context Realization

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

MDA – konforme CIM-Verwendung:

- Verstehen des Geschäftsprozesses und seiner Umgebung
- als Black-Box
- als Interaktion kooperierender Aktoren



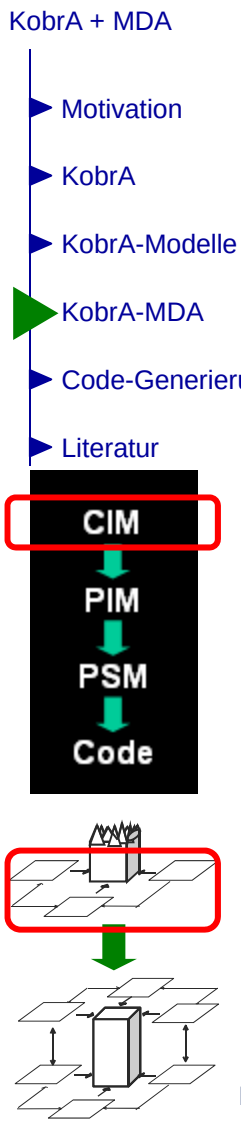
Interaction Model
(UML collaboration diagrams)

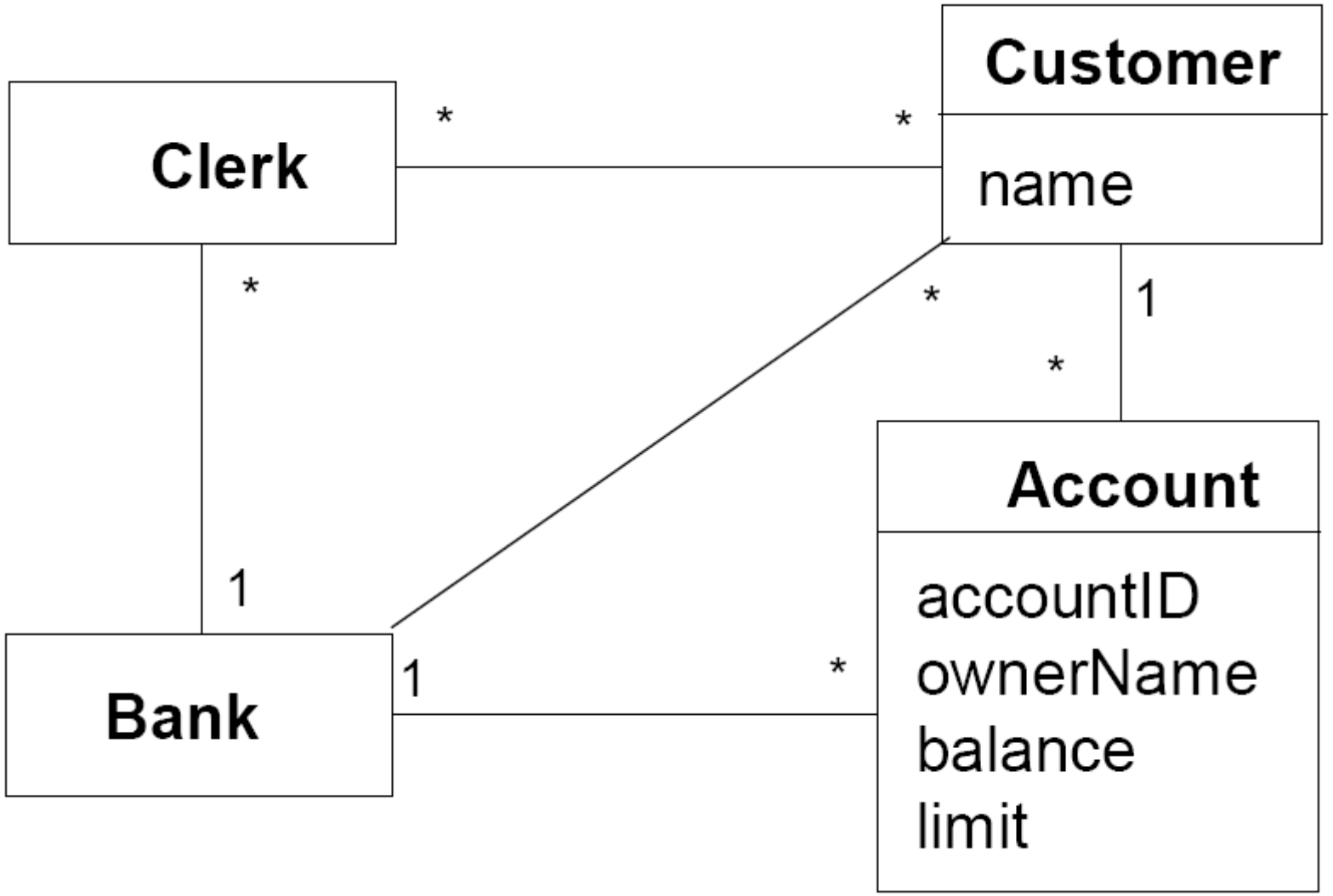
Structural Model
(UML class/object diagrams)

Activity Model
(UML activity, use case diagrams)

Realisierungsmodelle

KobrA → MDA – Beispiel: Simple International Bank; CIM: Domain Model der SIB

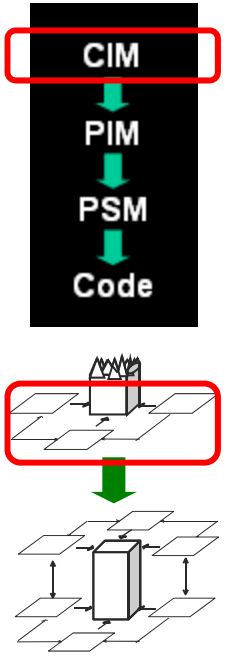


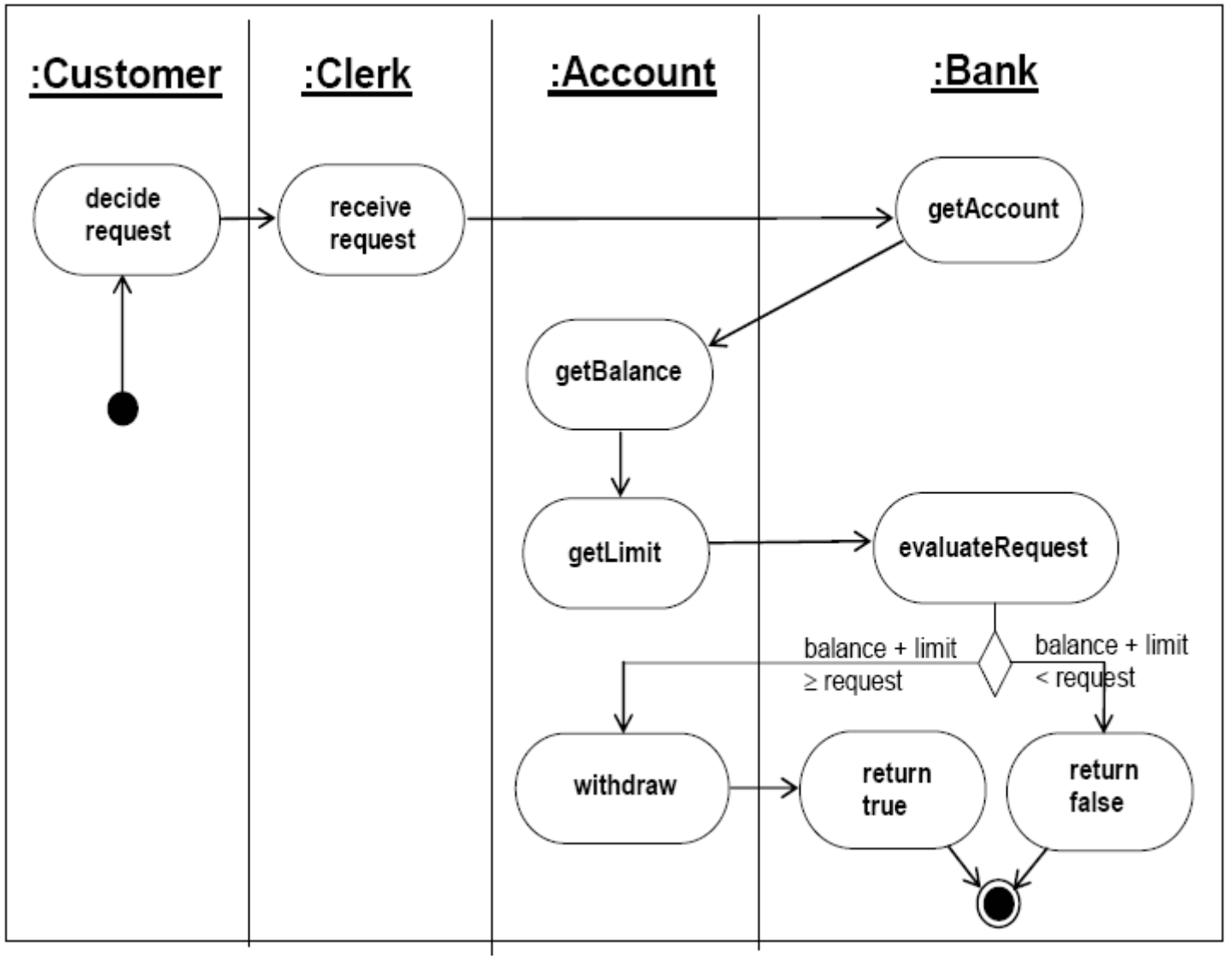


CIM: Geschäftsprozess; Aktivitätsdiagramm der withdraw-Operation

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur

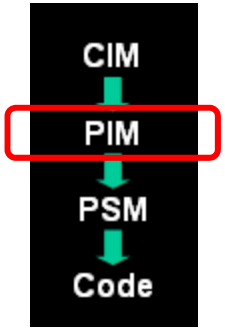


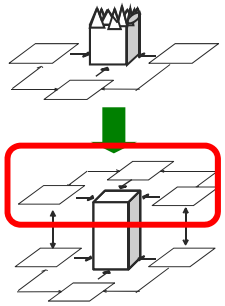


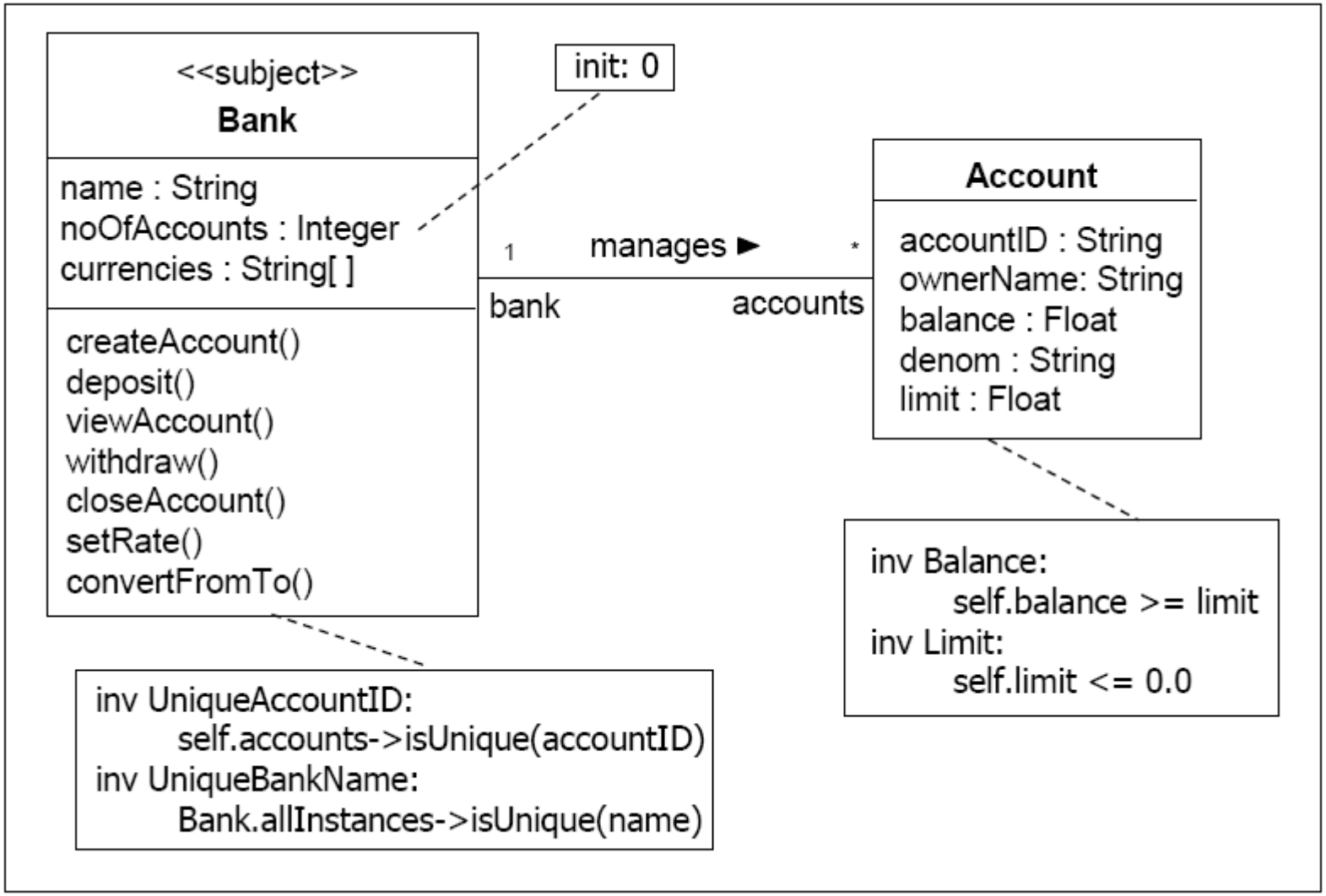
PIM: Spezifikation der SIB; alle von außen sichtbaren Elemente + benötigte Komponenten

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur

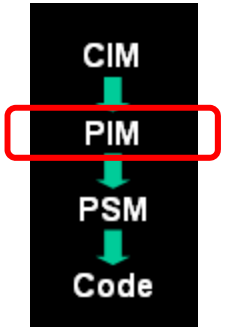


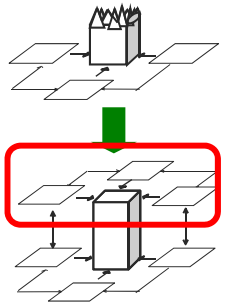




PIM: Spezifikation der SIB; *Operation Specification* von withdraw mit nach Außen sichtbaren Effekten

KobrA + MDA
- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierur
- Literatur



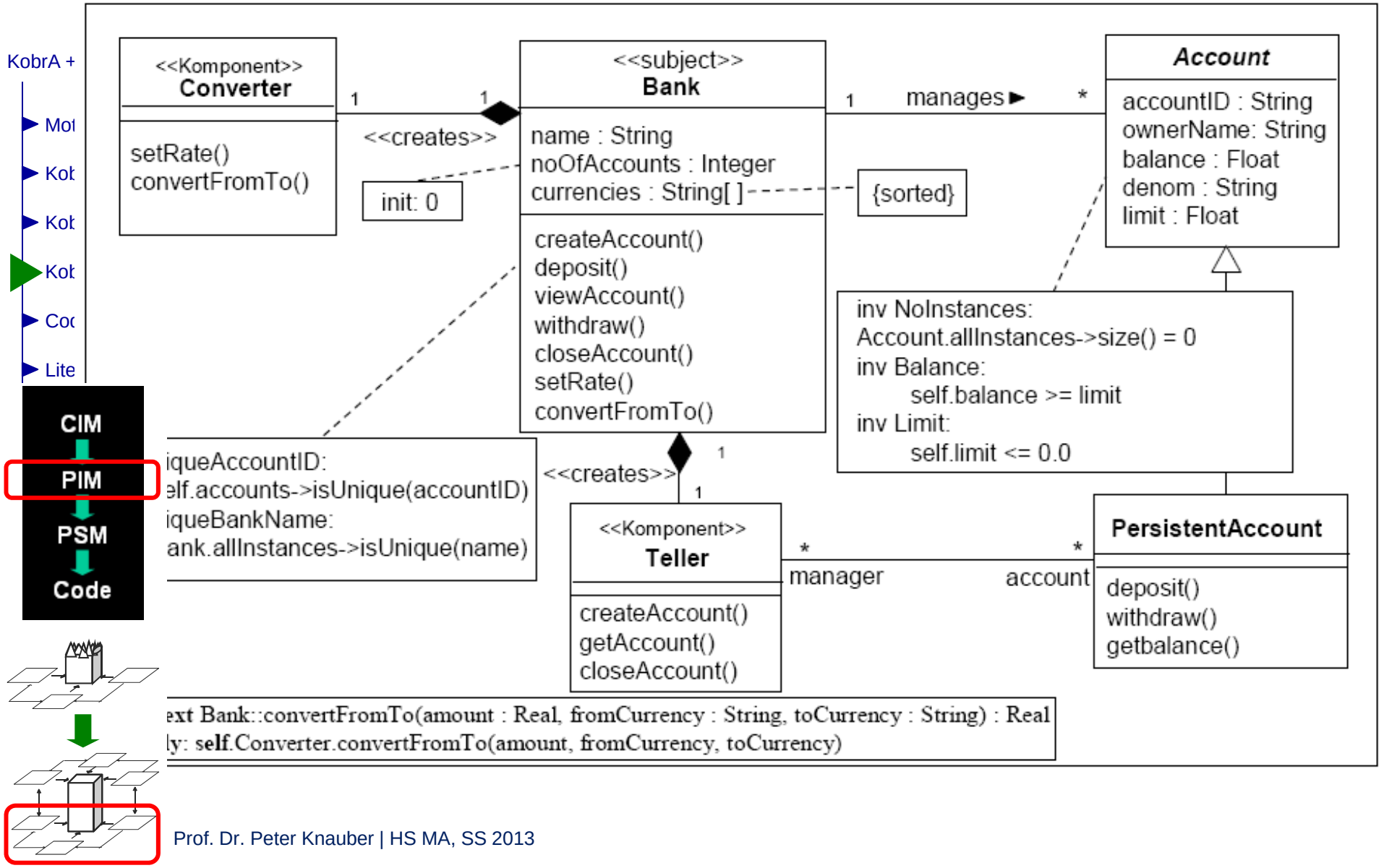


Name	withdraw
Informal Description	An amount of money in a particular currency is withdrawn from an account.
Receives	ID : String, currency : String, amount : Real
Returns	wdpossible(ID : String, currency : String, amount : Real) : Boolean = let acc : Account = self.accounts->select(accountID = ID)->asSequence()->first() in currencies->includes(currency) and amount > 0 and acc.balance - convertFromTo(amount, currency, acc.denom) > acc.limit
Changes	account with accountID = ID

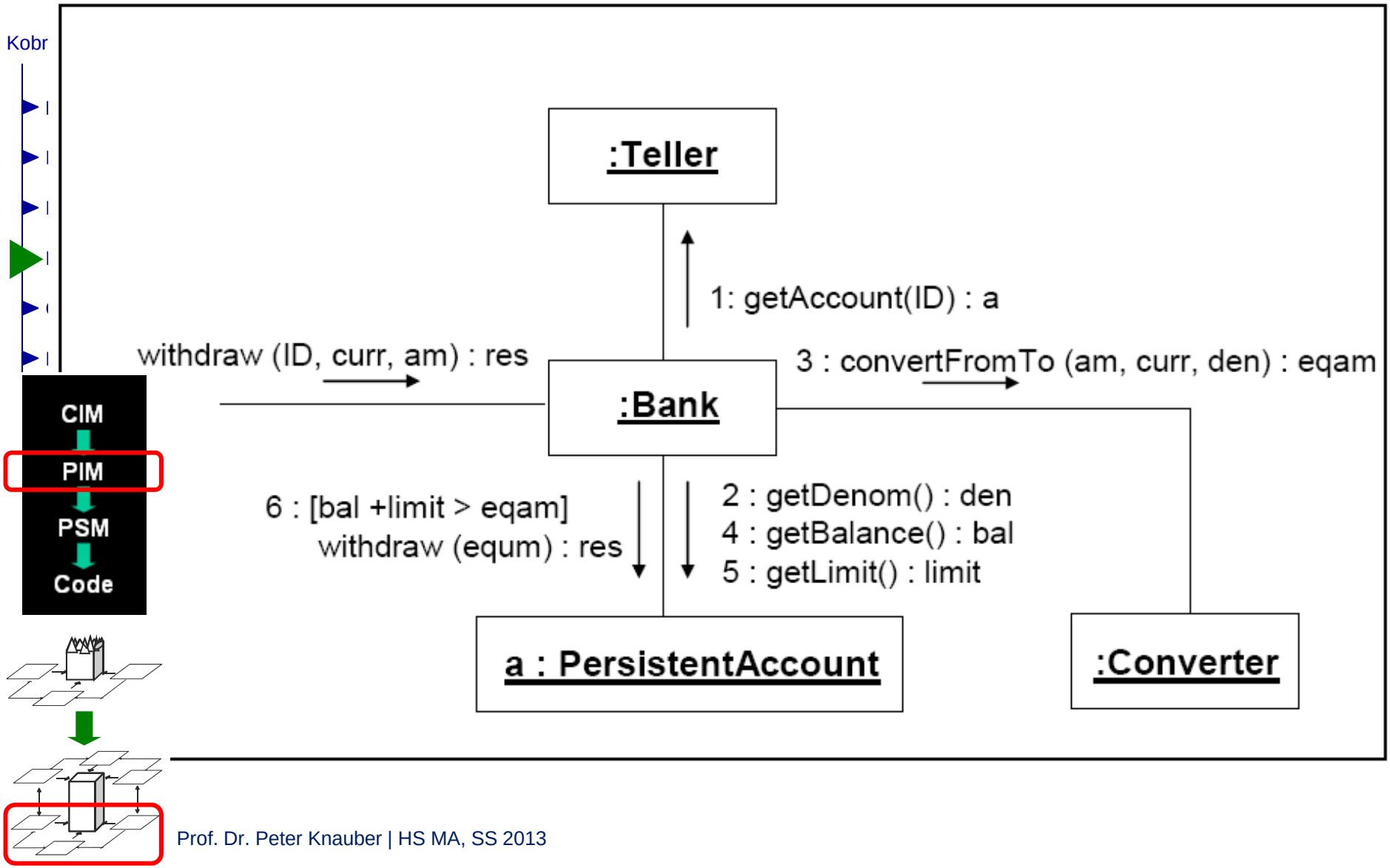
Rules	let acc : Account
	= self.accounts->select(accountID = ID)->asSequence()->first()
	in if currency = acc.denom then
	equivalentAmount = amount
	else
	equivalentAmount = convertFromTo(amount, currency, acc.denom)
	endif

Result	post: if (wdpossible(ID, currency, amount) then
	acc.balance = acc.balance@pre – equivalentAmount
	else
	acc.balance = acc.balance@pre
	endif

PIM: Realisierung der SIB; entspricht Spezifikation + zur Realisierung notwendige Elemente

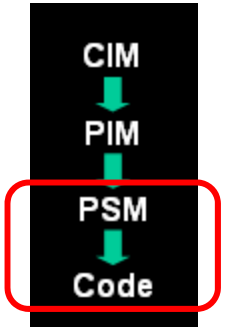


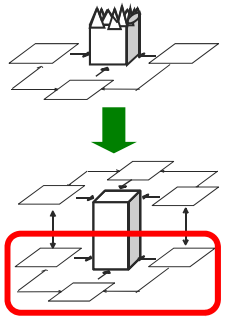
PIM: Realisierung der SIB; Interaktion am Beispiel withdraw



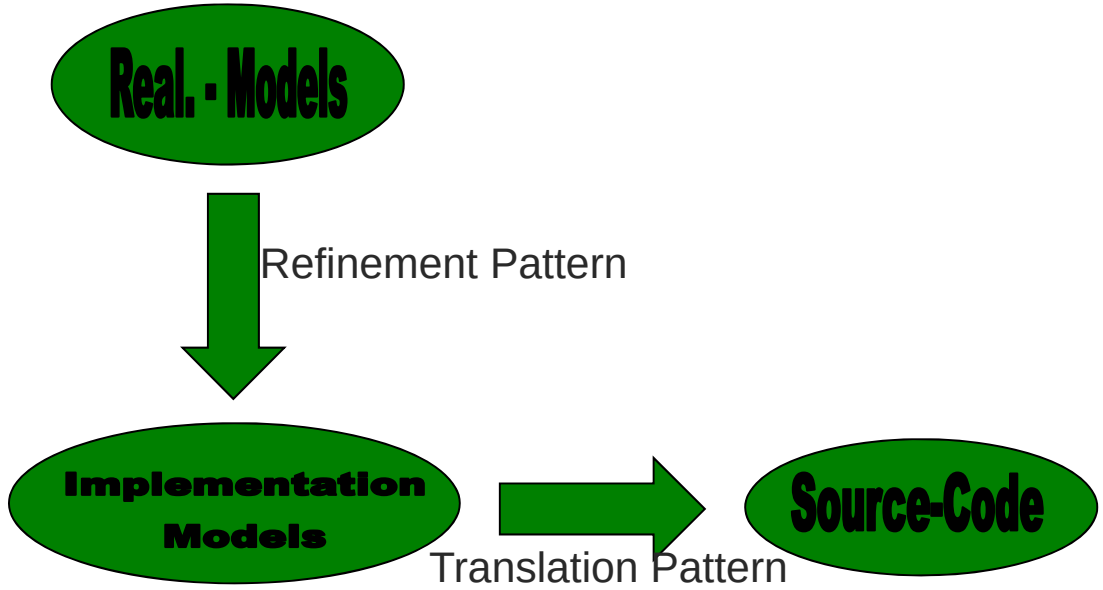
Code-Generierung

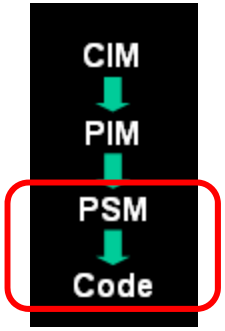
KobrA + MDA
- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur

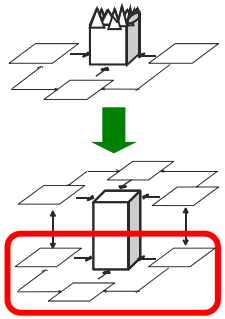




- Transformationsschritte
  - Refinement Pattern
  - Translation Patterm
- Mittels *Implementation Models* (spezielle Form der *Realization Models*) + *Translation Pattern*

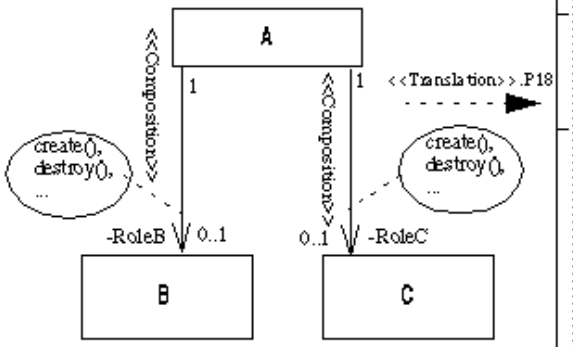






Translation Pattern

18§ Pattern "Clientship <<Composition>> (1)"§ Target: C++§ Category: Structure§




```
class B {(
friend class A;(
...(
};¶
```

```
class C {(
friend class A;(
...(
};¶
```

```
class A {(
public:(
} A();(
} ...(
private:(
} B b;(
} C c;(
} ...(
...(
};¶
```

```
A::A():b(),c(){(
{(
}§
```

Context:¶

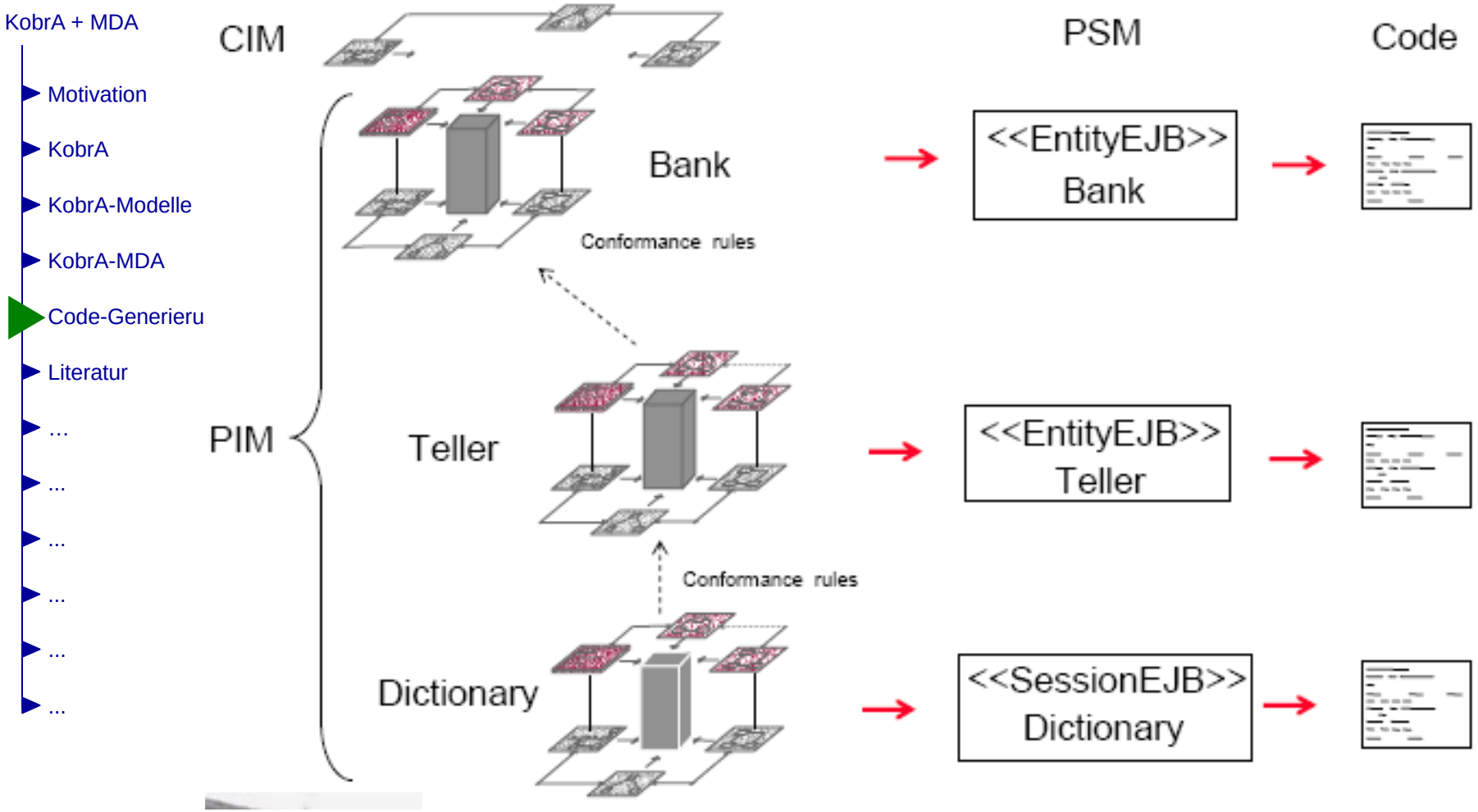
§	Influence (- -, -, o, +, ++)§
Maintainability§	o§
Performance§	+§
Reliability§	o§
Reusability§	o§
Security§	--§
Space§	--§
...§	§

Description:¶

-) Another way of implementing an <<Aggregation>> dientship is to use embedded objects. The composite class A defines a variable which holds zero or one objects of a composite part (i.e., variables b and c). The variables b and c are initialized during the creation method of class A. The lifetime of the composite parts of object A, namely objects of class B and C, is restricted to the lifetime of the composite object due to the nature of composition. Therefore A's creation method instantiates class B and C in an inheritance like manner. If an object of class A ceased to exist the embedded instances of classes B and C are automatically destroyed. In the field of programming this strategy is known as embedded objects.(
Please note that the implementation of constraints (e.g., ordering), or -specifiers is discussed by pattern: ***25-27¶***

Trace:) - Refinement Patterns: 2, 3, 7, 20, 21,-23(
) - Translation Patterns: 1-7, 8, 25-27§

CIMs, PIMs und PSMs in KobrA



Literatur

KobrA + MDA

- Motivation
- KobrA
- KobrA-Modelle
- KobrA-MDA
- Code-Generierung
- Literatur
- …
- ...
- ...
- ...
- ...
- ...

- Colin Atkinson, Matthias Gutheil, Oliver Hummel: Komponentenorientierter Entwurf von PIMs und CIMs mit der KobrA-Methode. Architekturen, Komponenten, Anwendungen - Proc. 1. Verbundtagung AKA 2004, Augsburg, Dezember 2004, S. 93-110
- Atkinson, Bayer, Bunse, Kamsties, Laitenberger, Laqua, Muthig, Paech, Wüst, Zettel: Component-based Product Line Engineering with UML, Addison Wesley (das Buch zur KobrA-Methode)