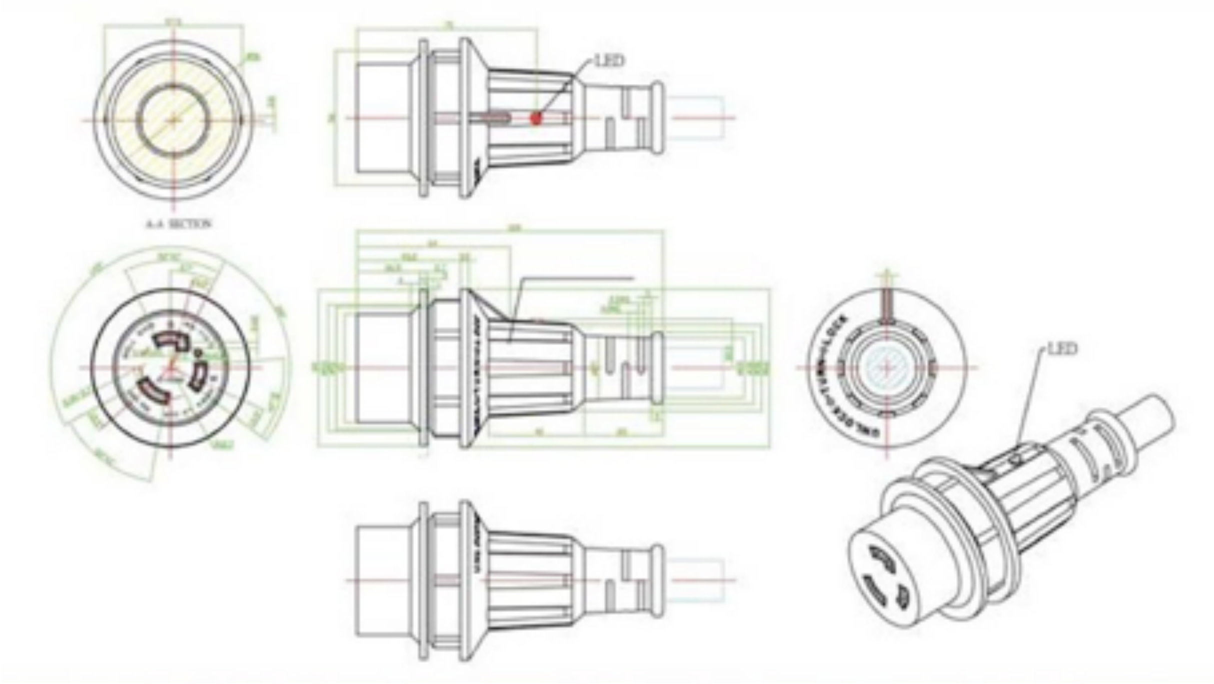
On the Challenges of Composing Multi-View Models

Matthias Schöttle and Jörg Kienzle

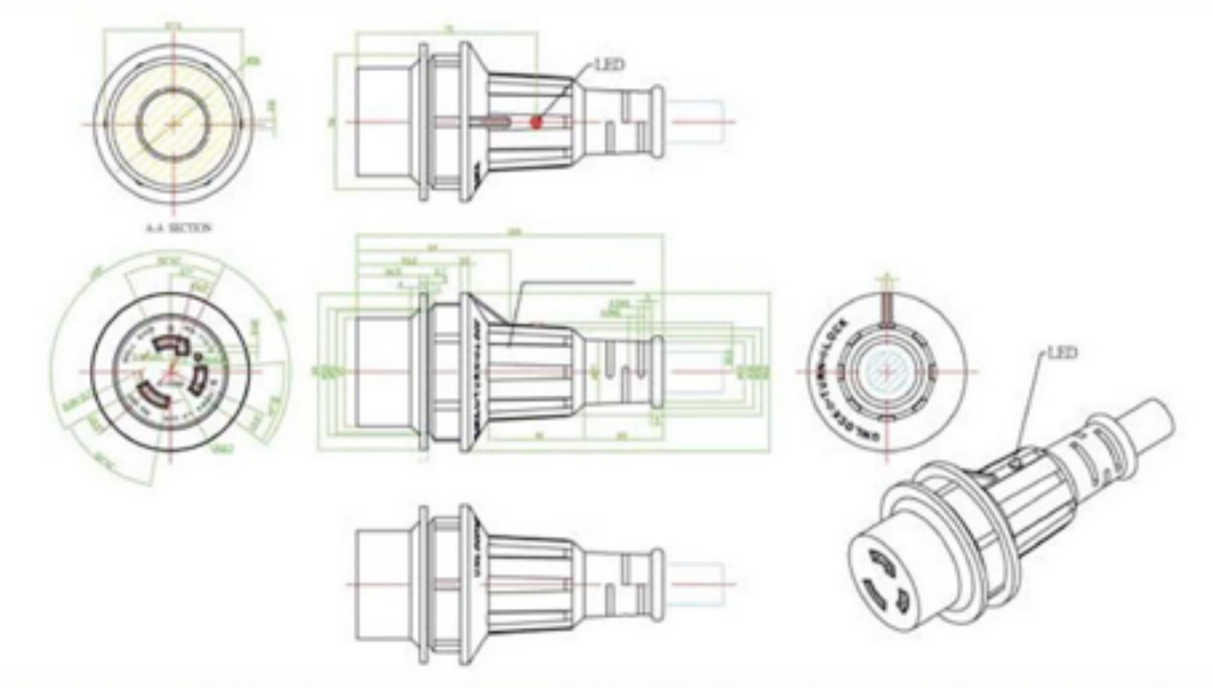
McGill

Multi-View Modelling



[Image credit: http://www.accesscontrolsales.com/mighty-cord-rv-cad.htm]

Multi-View Modelling

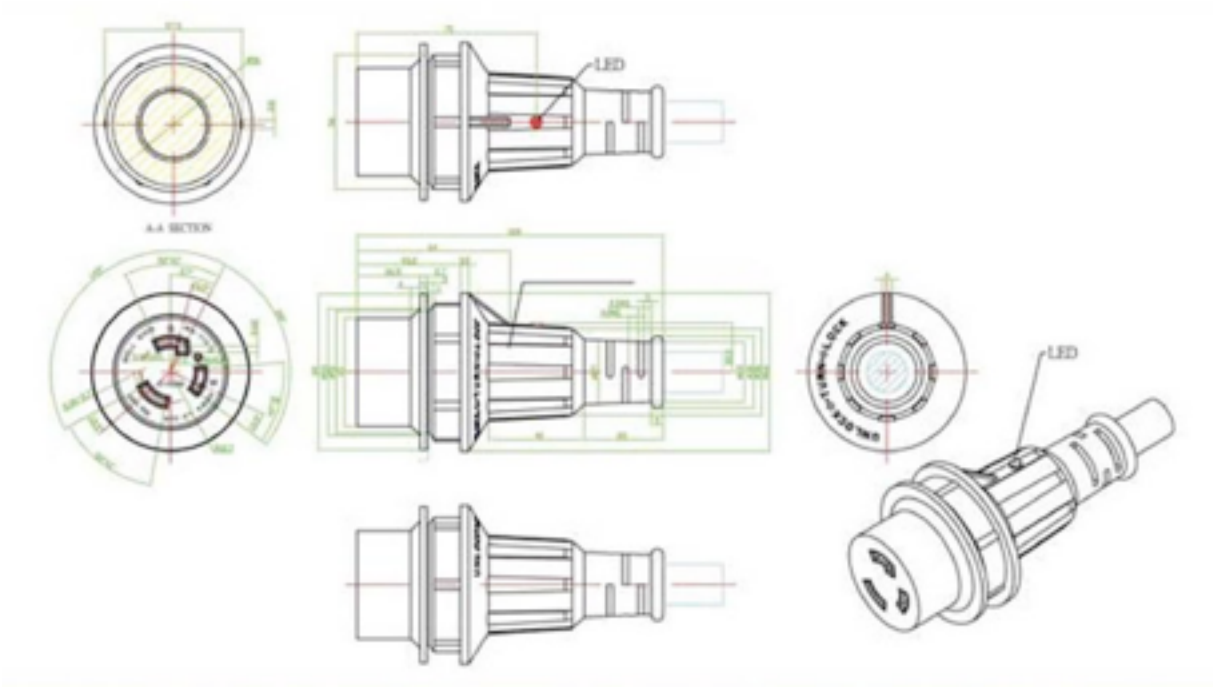


[Image credit: http://www.accesscontrolsales.com/mighty-cord-rv-cad.htm]

Consistent Views

[Image credit: http://www.empowernetwork.com/msycks/blog/how-to-get-consistent-in-your/]

Multi-View Modelling

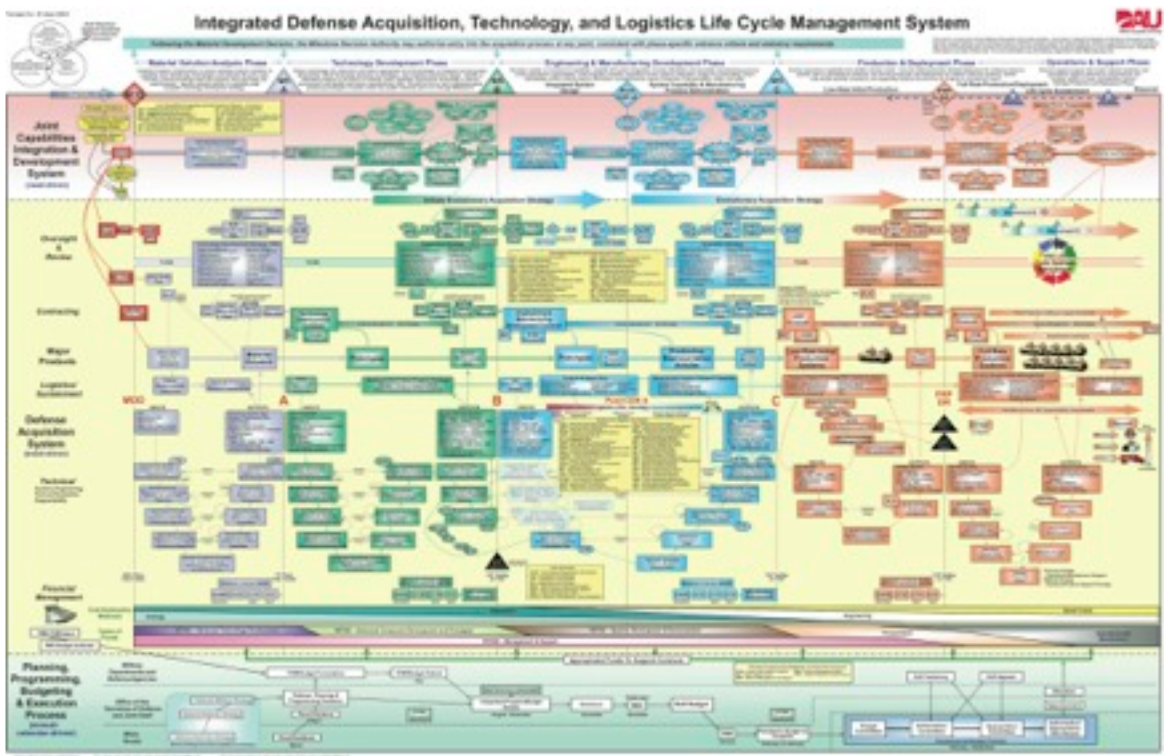


[Image credit: http://www.accesscontrolsales.com/mighty-cord-rv-cad.htm]

Consistent Views

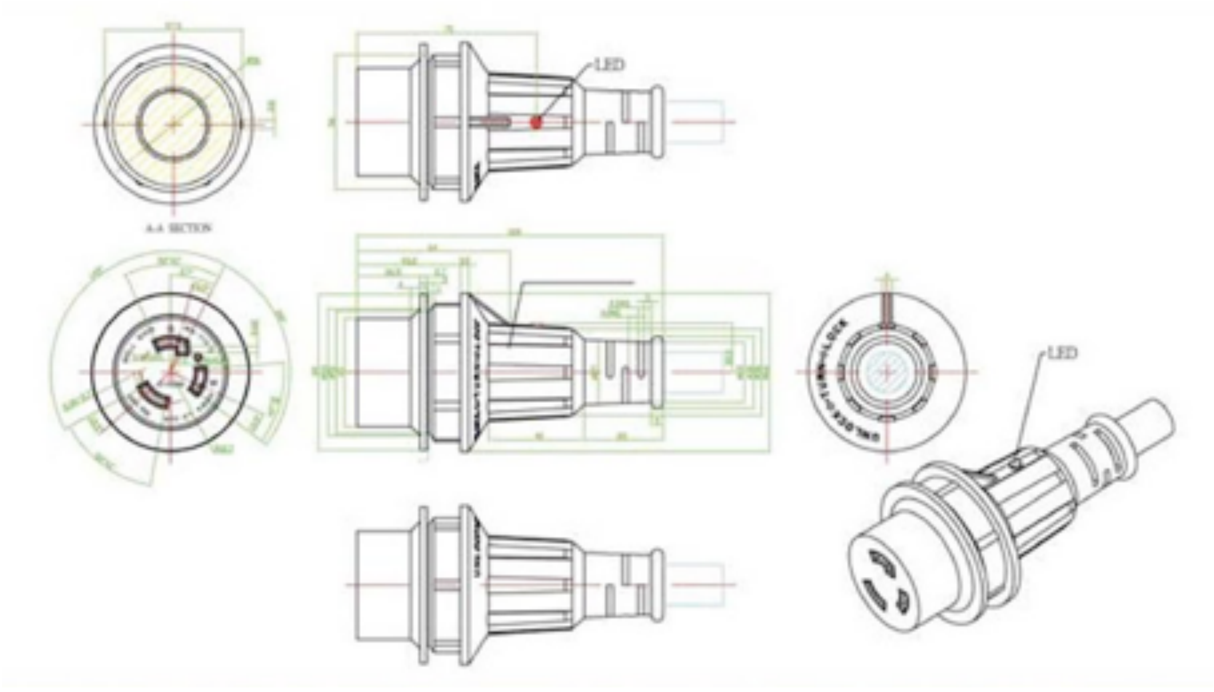
[Image credit: http://www.empowernetwork.com/msycks/blog/how-to-get-consistent-in-your/]

Complex Systems


Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System
DAU

[Image credit: http://www.wired.com/dangerroom/2010/09/revealed-pentagons-craziest-powerpoint-slide-ever/]

Multi-View Modelling

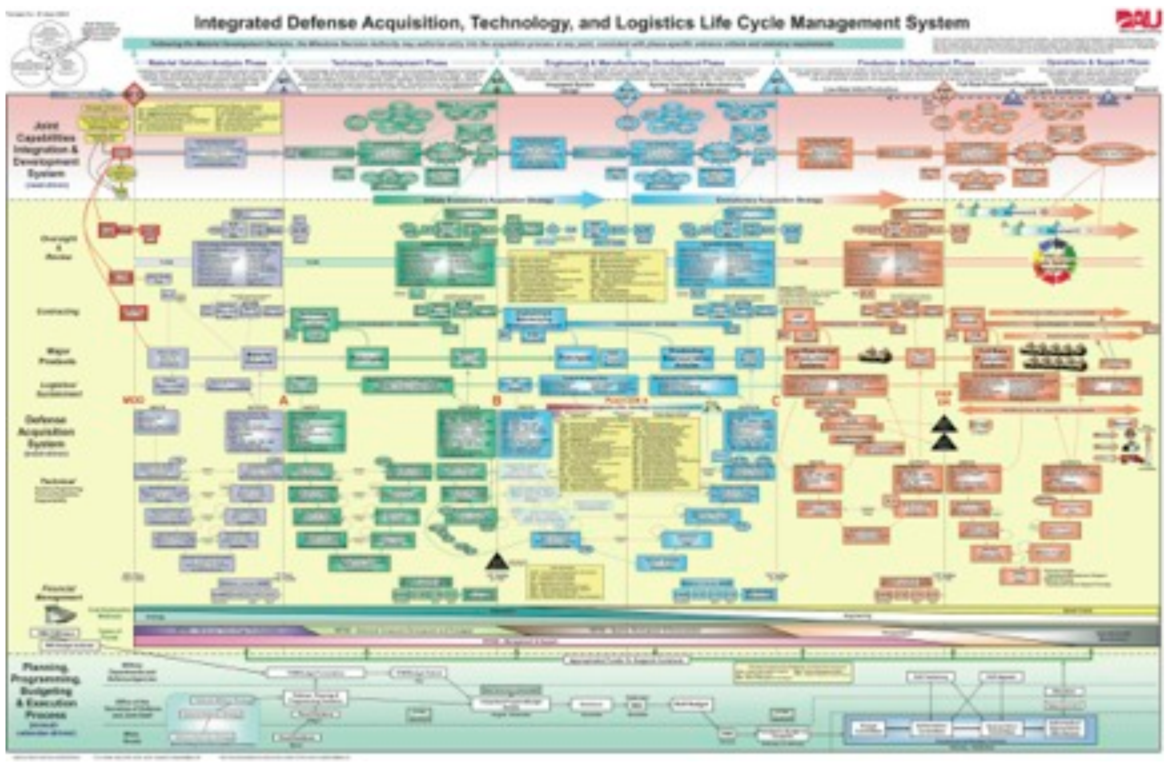


[Image credit: http://www.accesscontrolsales.com/mighty-cord-rv-cad.htm]

Consistent Views

[Image credit: http://www.empowernetwork.com/msycks/blog/how-to-get-consistent-in-your/]

Complex Systems

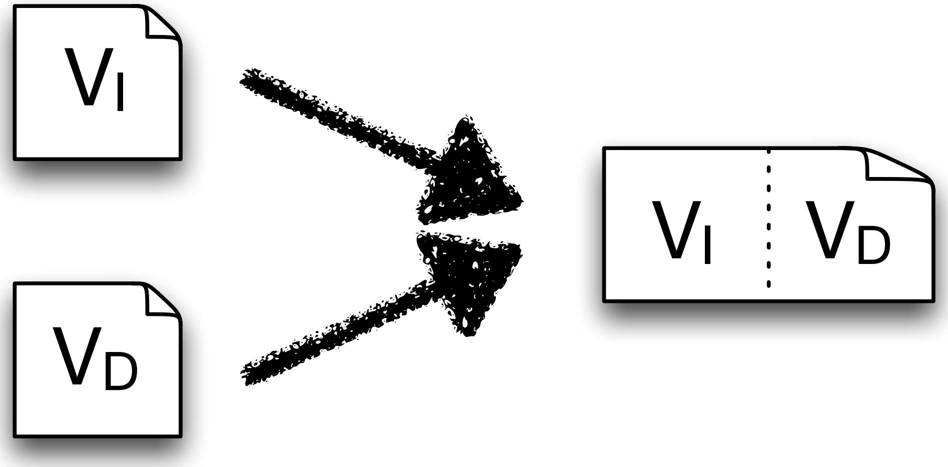

Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System
DAU

[Image credit: http://www.wired.com/dangerroom/2010/09/revealed-pentagons-craziest-powerpoint-slide-ever/]

Composition

[Image credit: http://www.accesscontrolsales.com/mighty-cord-rv-power-cords.htm]

Overview

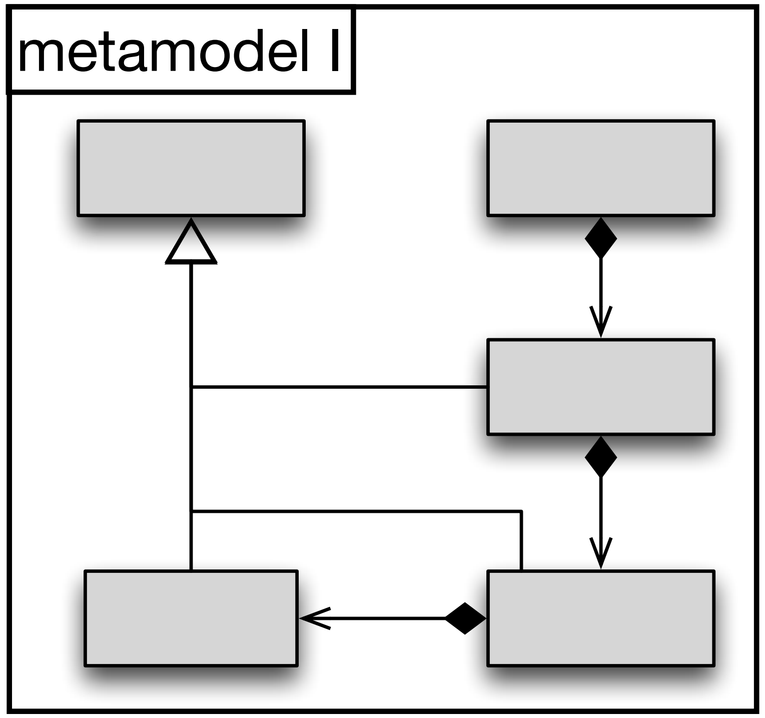


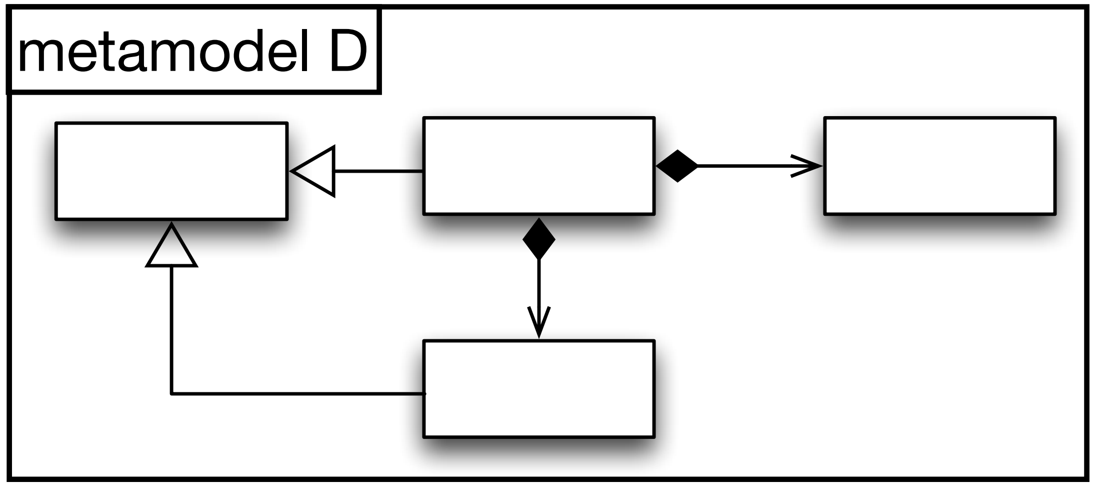
- Two independent modelling notations (I and D)
    - Metamodels and composers exist
- Integration to multi-view modelling notation
    - Ensure consistency between views
    - Ensure compatibility of composers
- Focus on technical side: Reuse

Contents

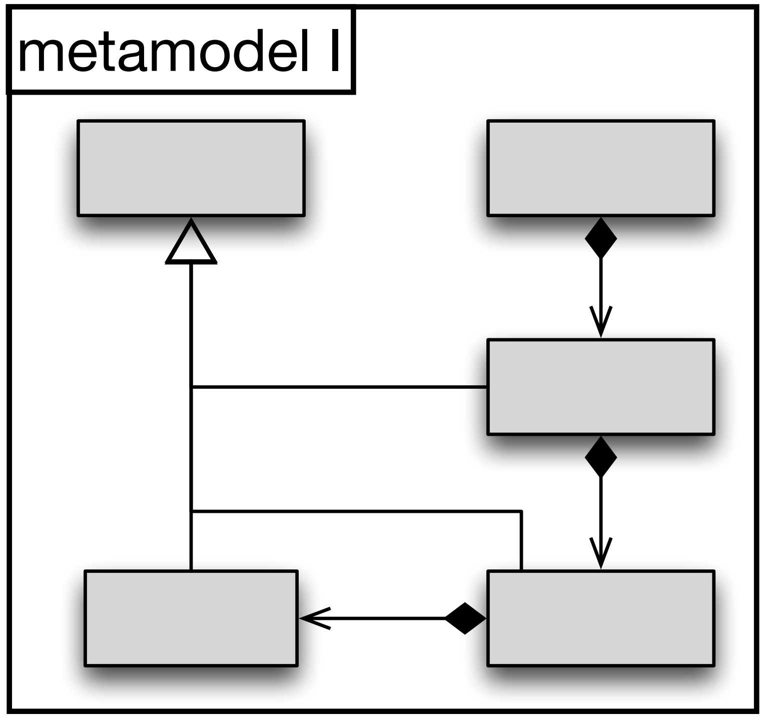
- General strategy for integrating two notations
  - Integration of metamodels
  - Composition of multi-view models
- Practical application of strategy to RAM
- Challenges faced
- Demo

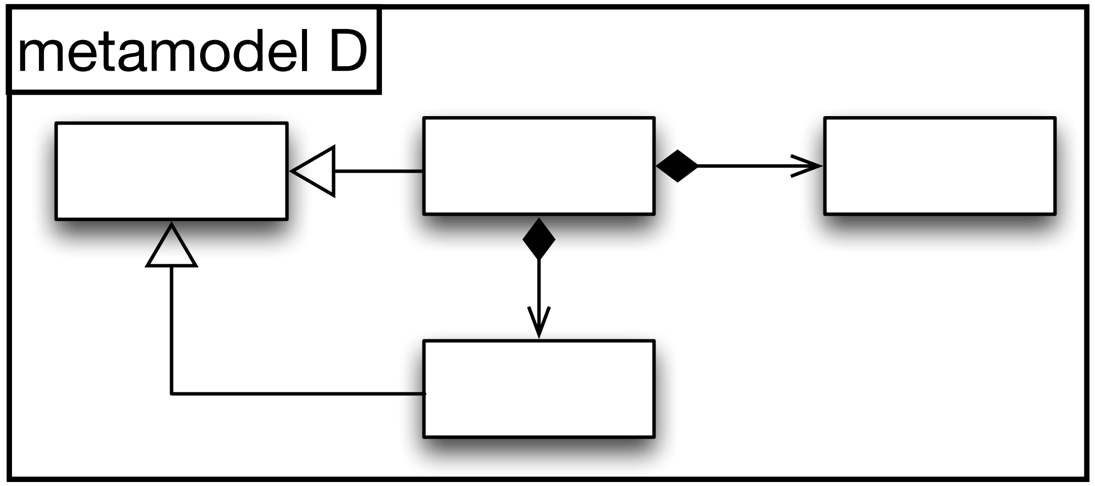
Integration Strategy: Metamodels

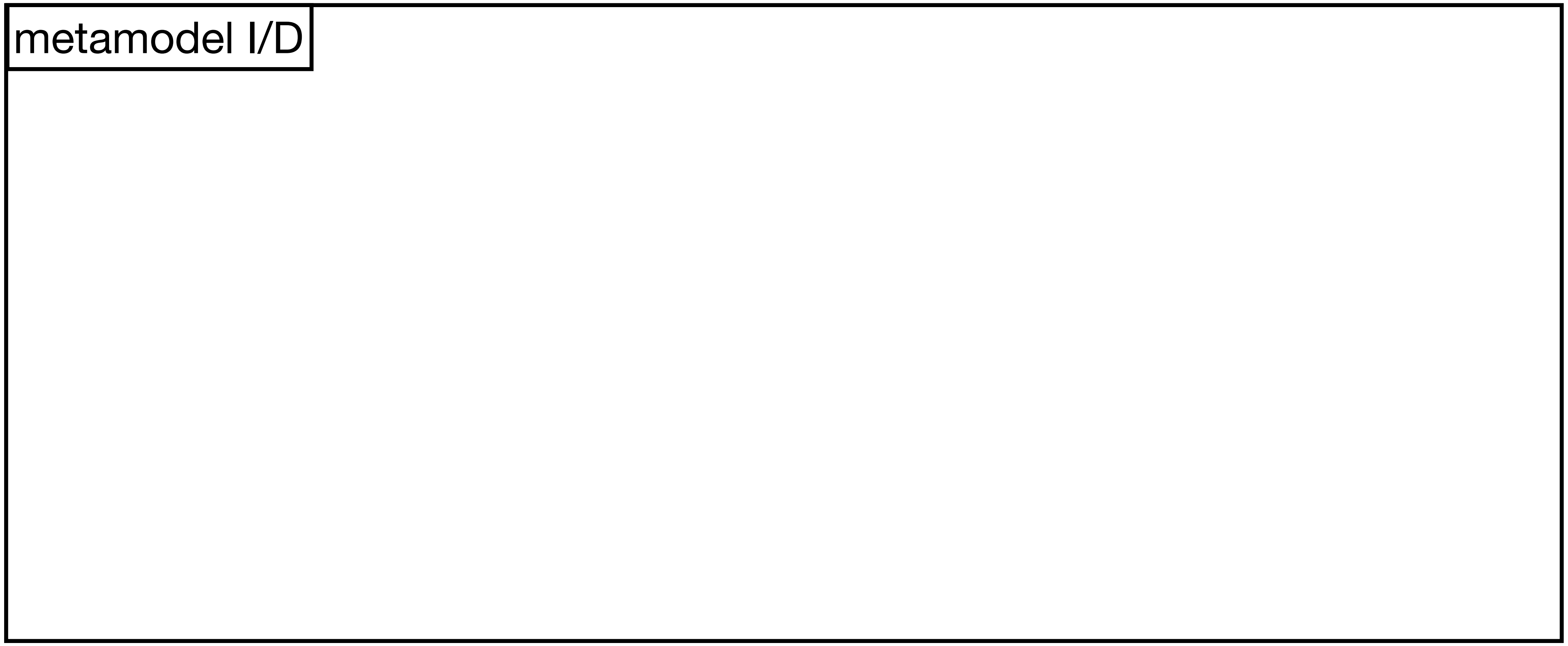




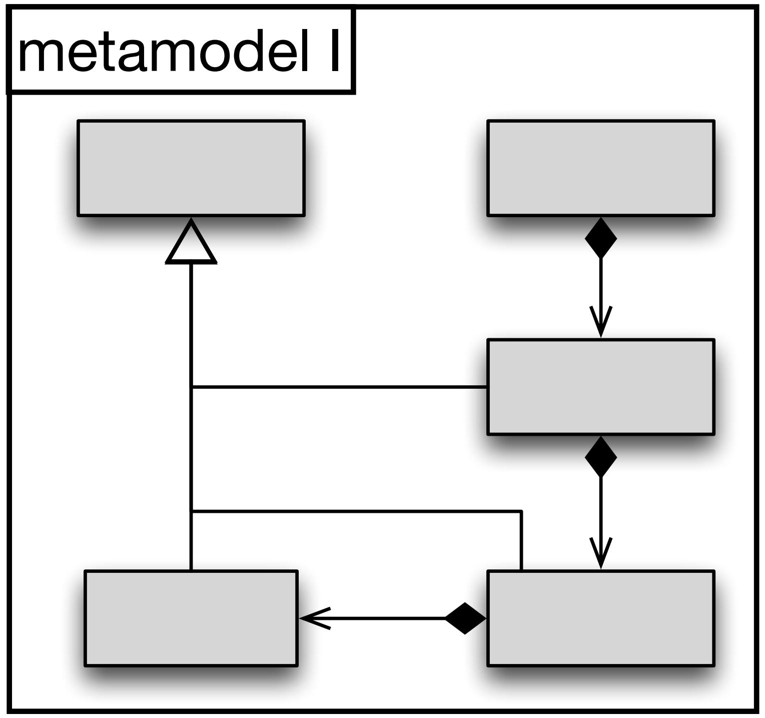
Integration Strategy: Metamodels

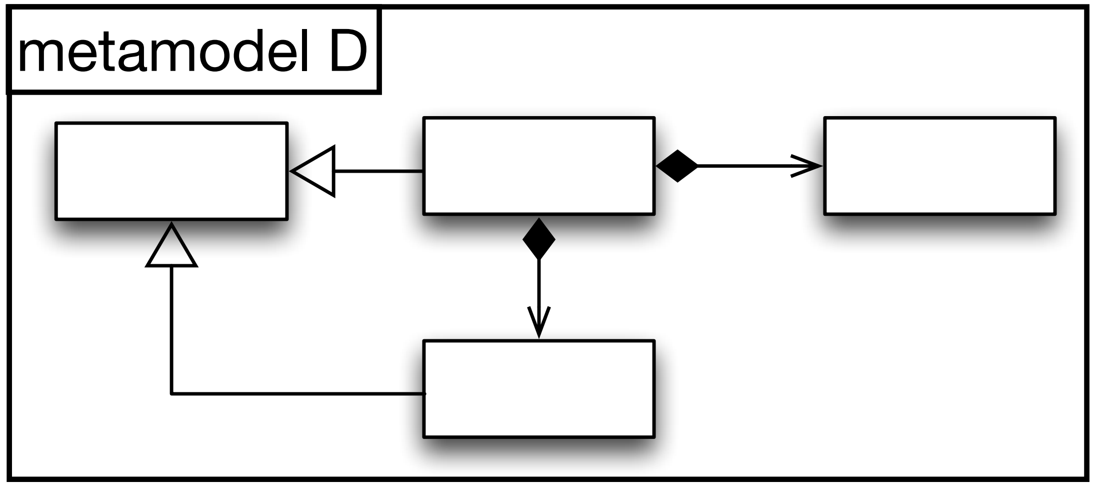


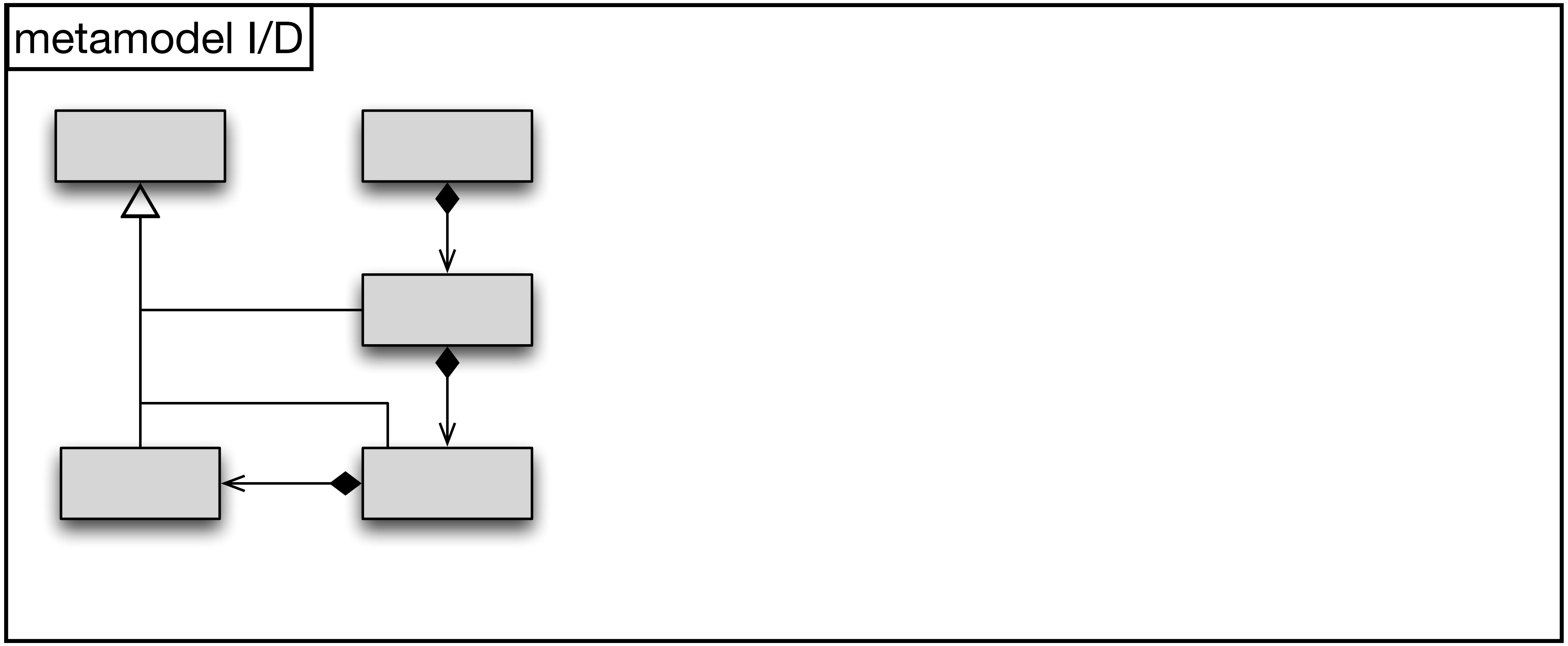




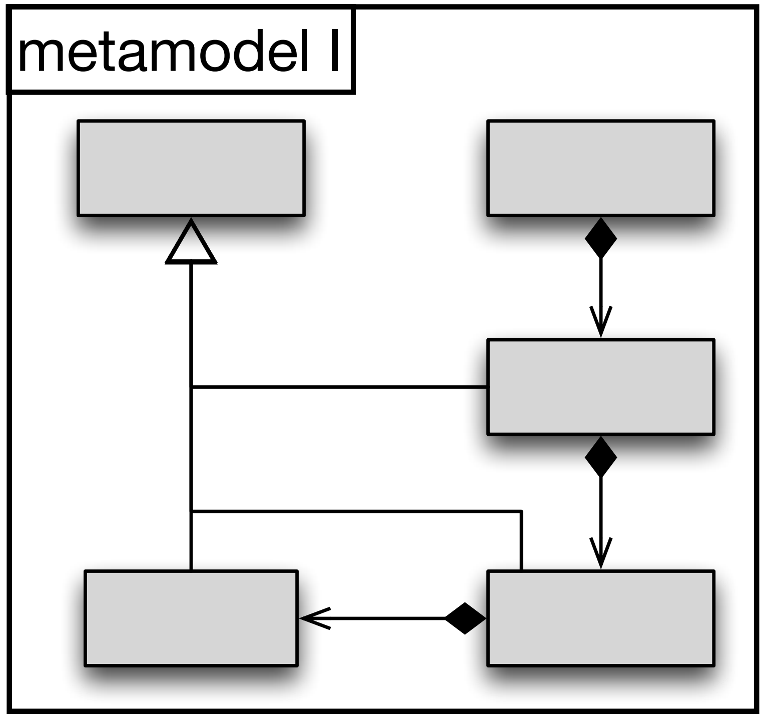
Integration Strategy: Metamodels

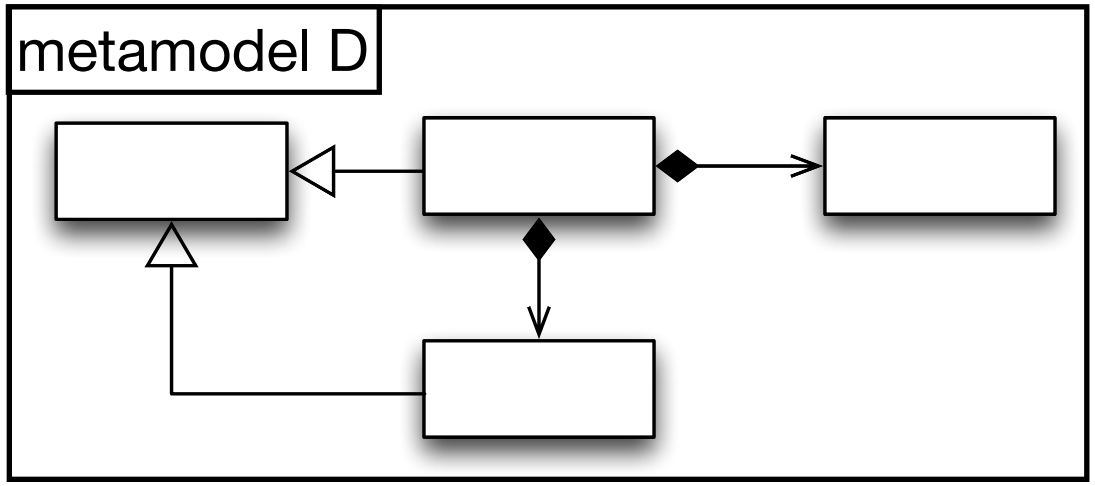


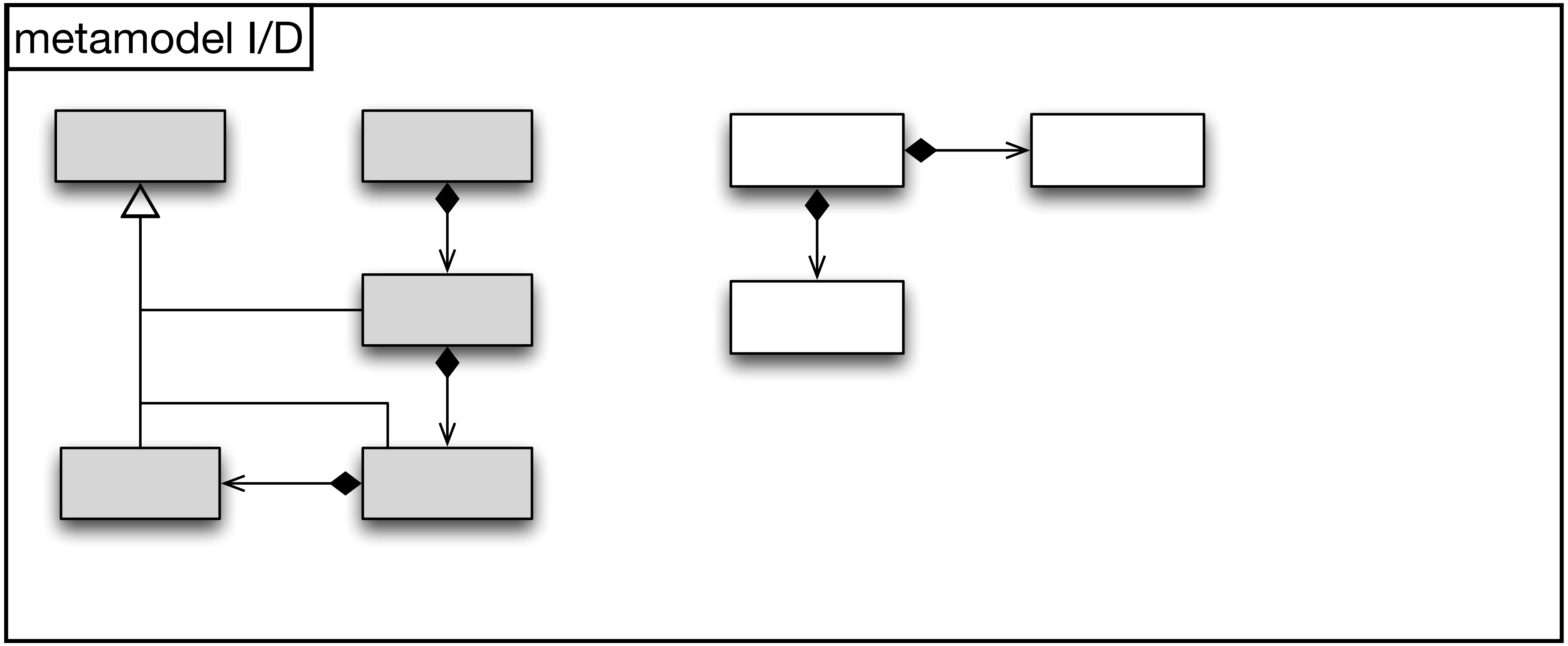




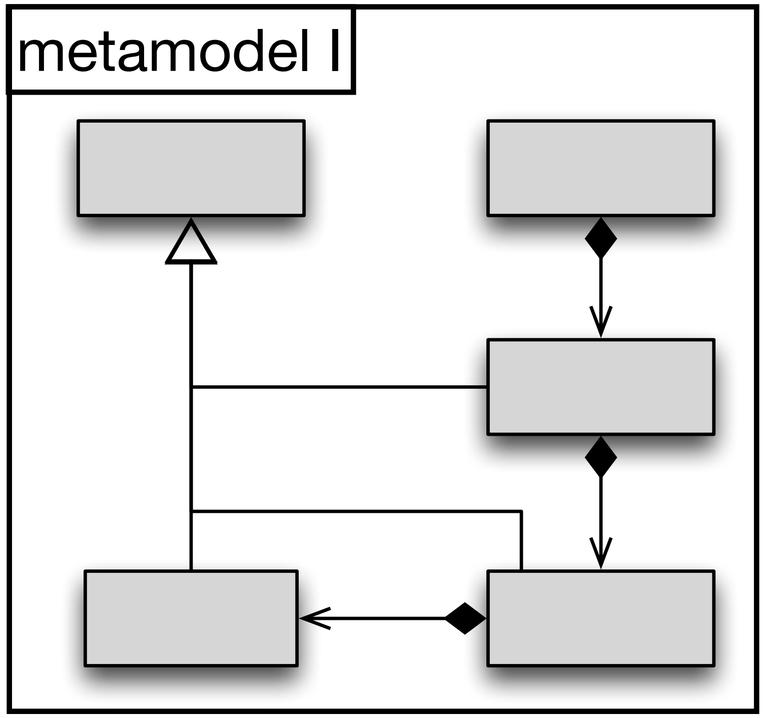
Integration Strategy: Metamodels

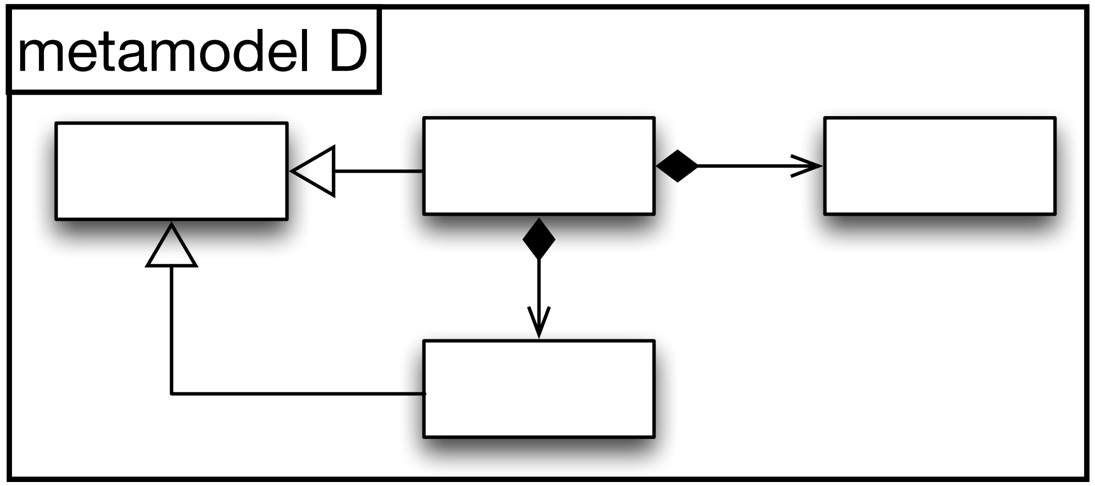


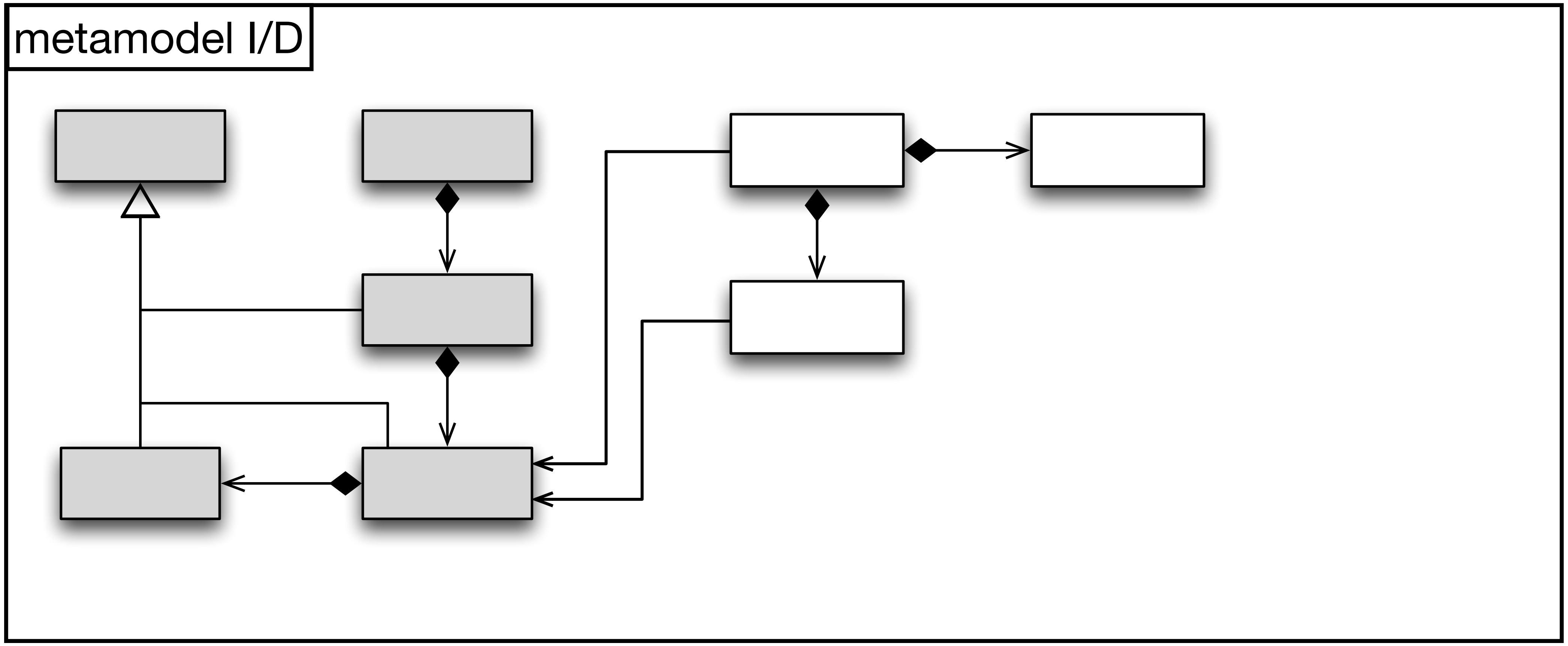




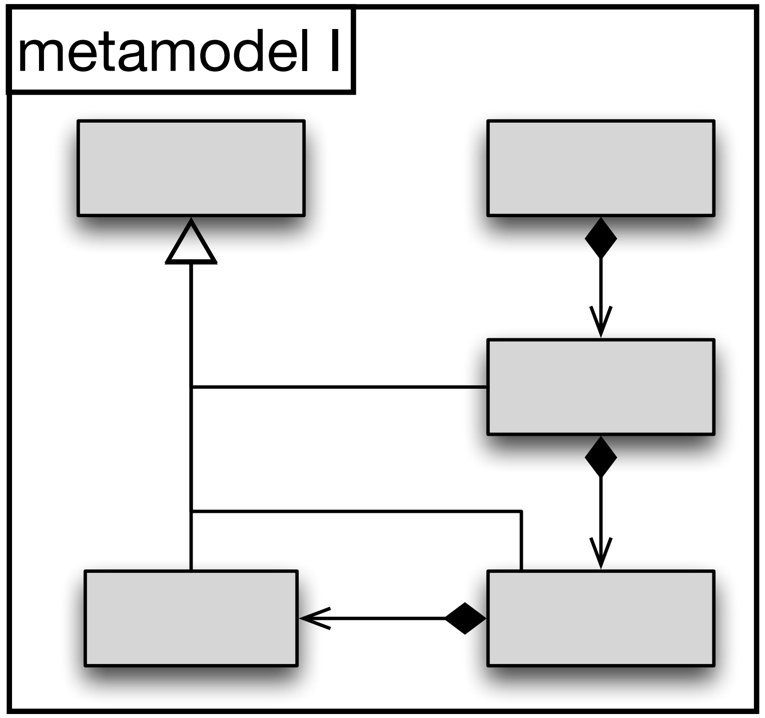
Integration Strategy: Metamodels

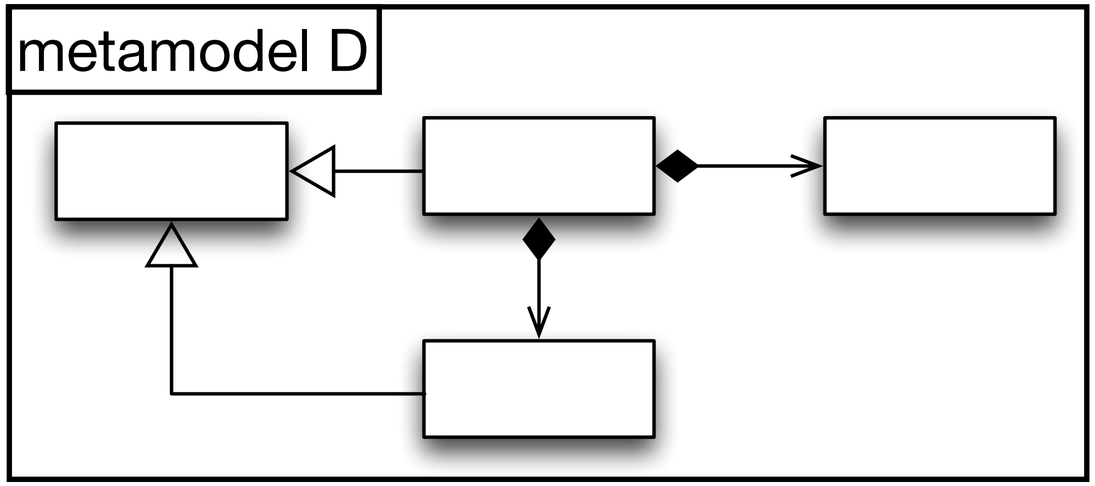


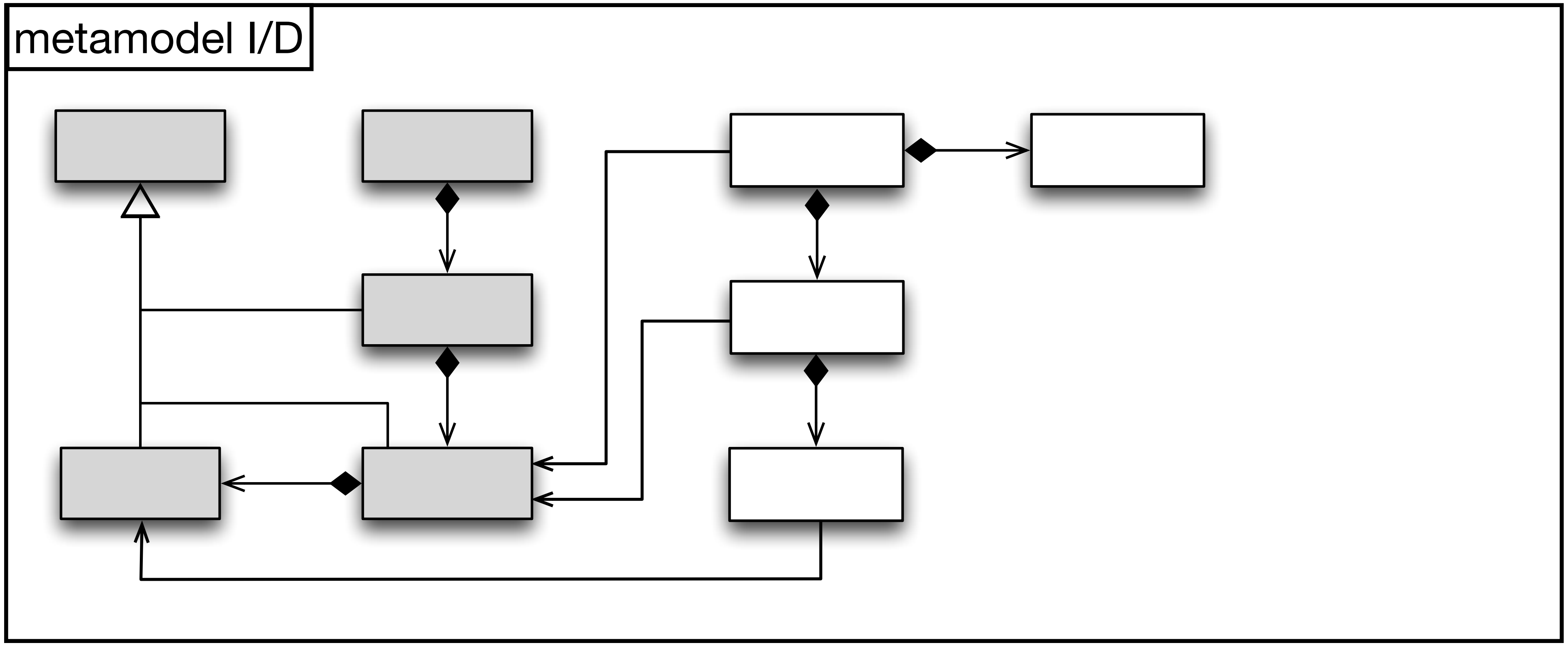




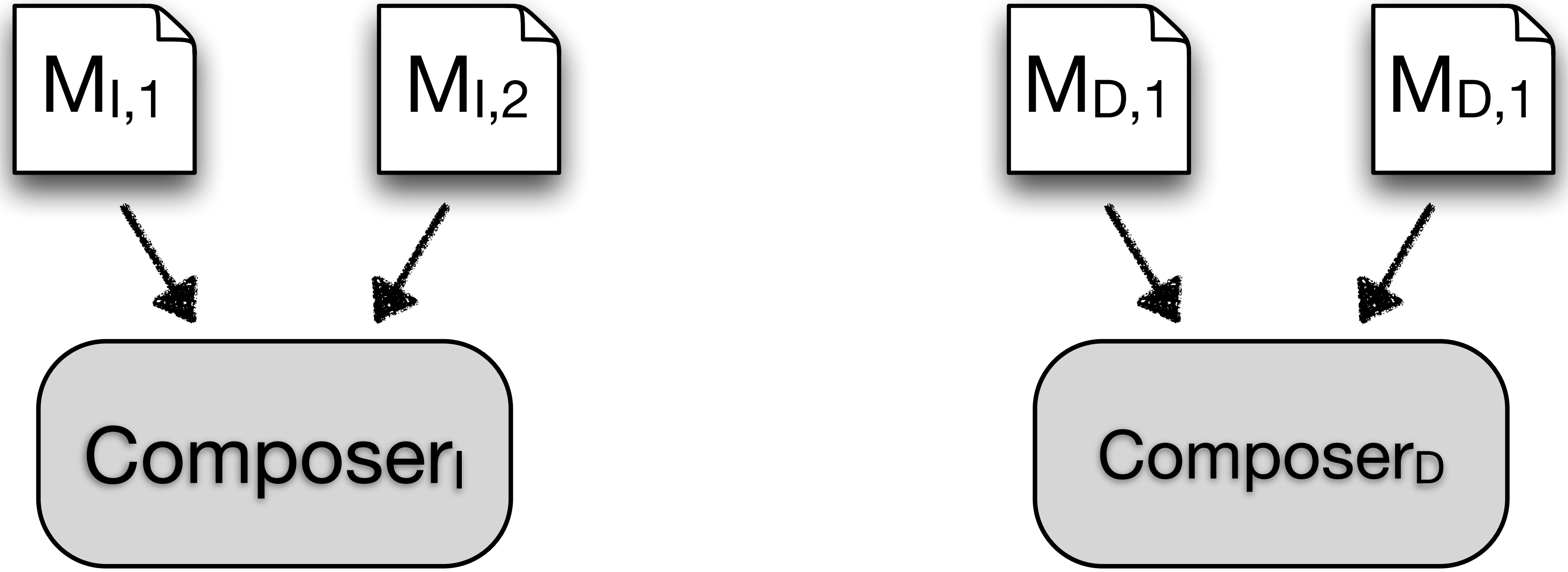
Integration Strategy: Metamodels



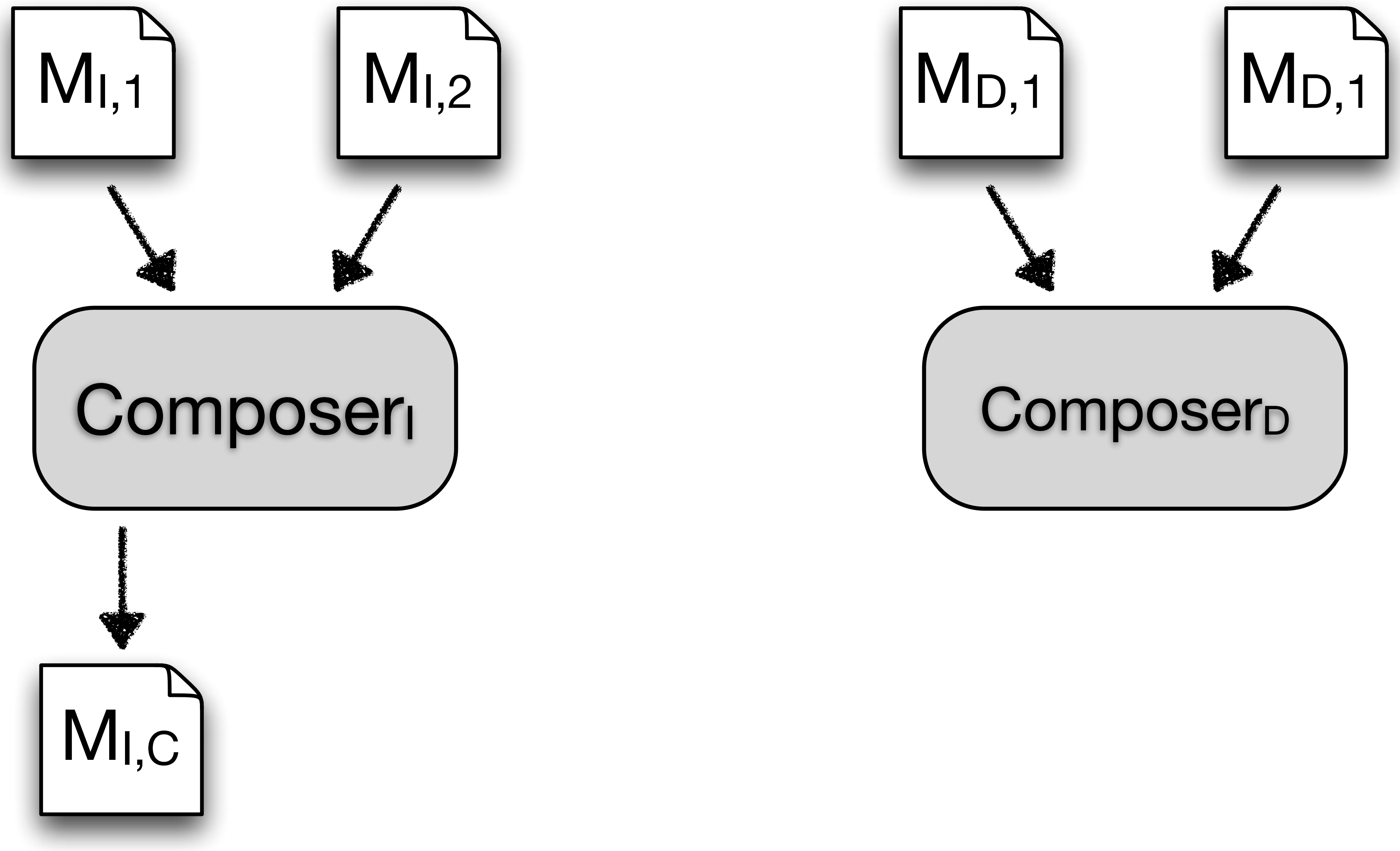




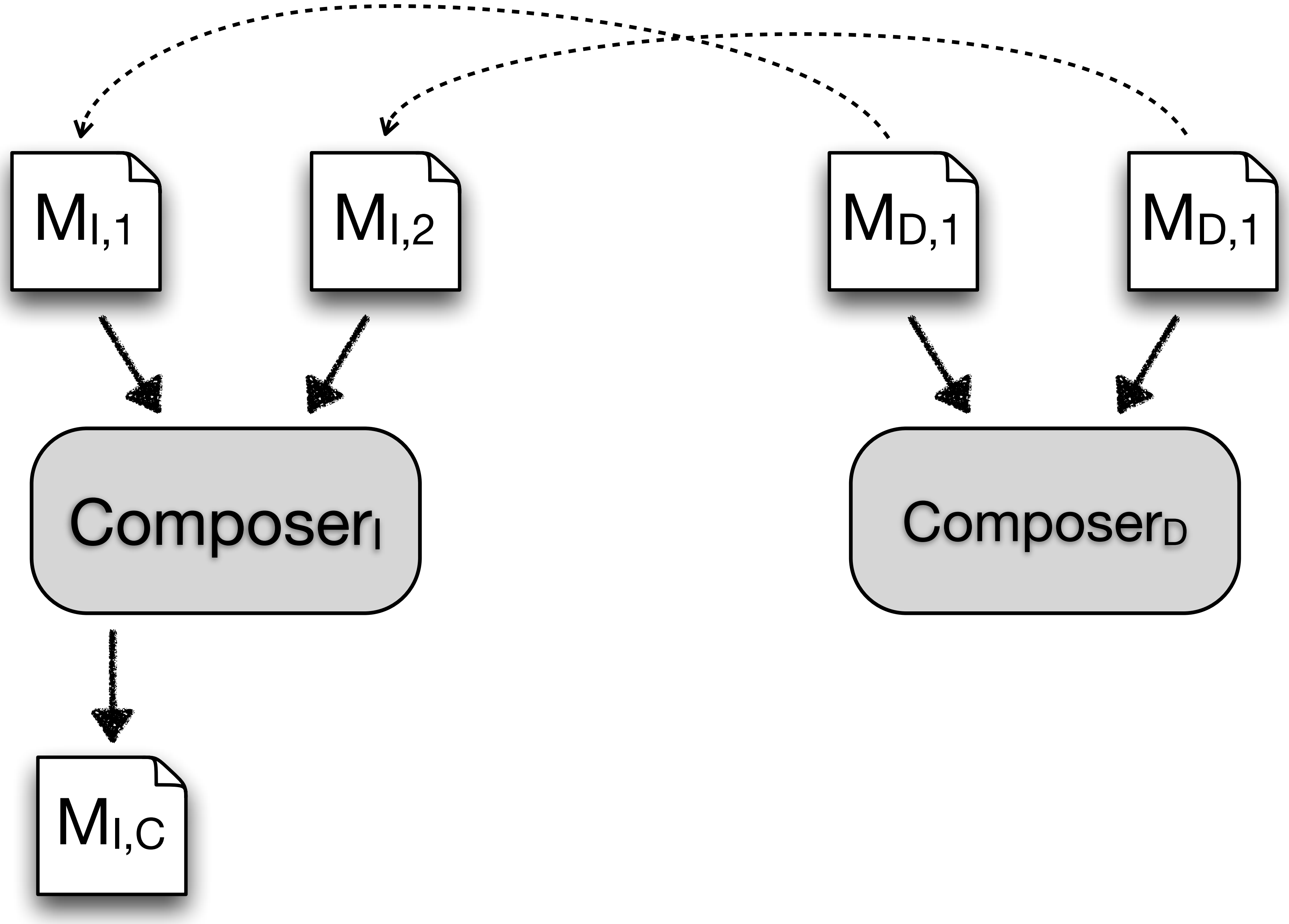
Integration Strategy: Composing



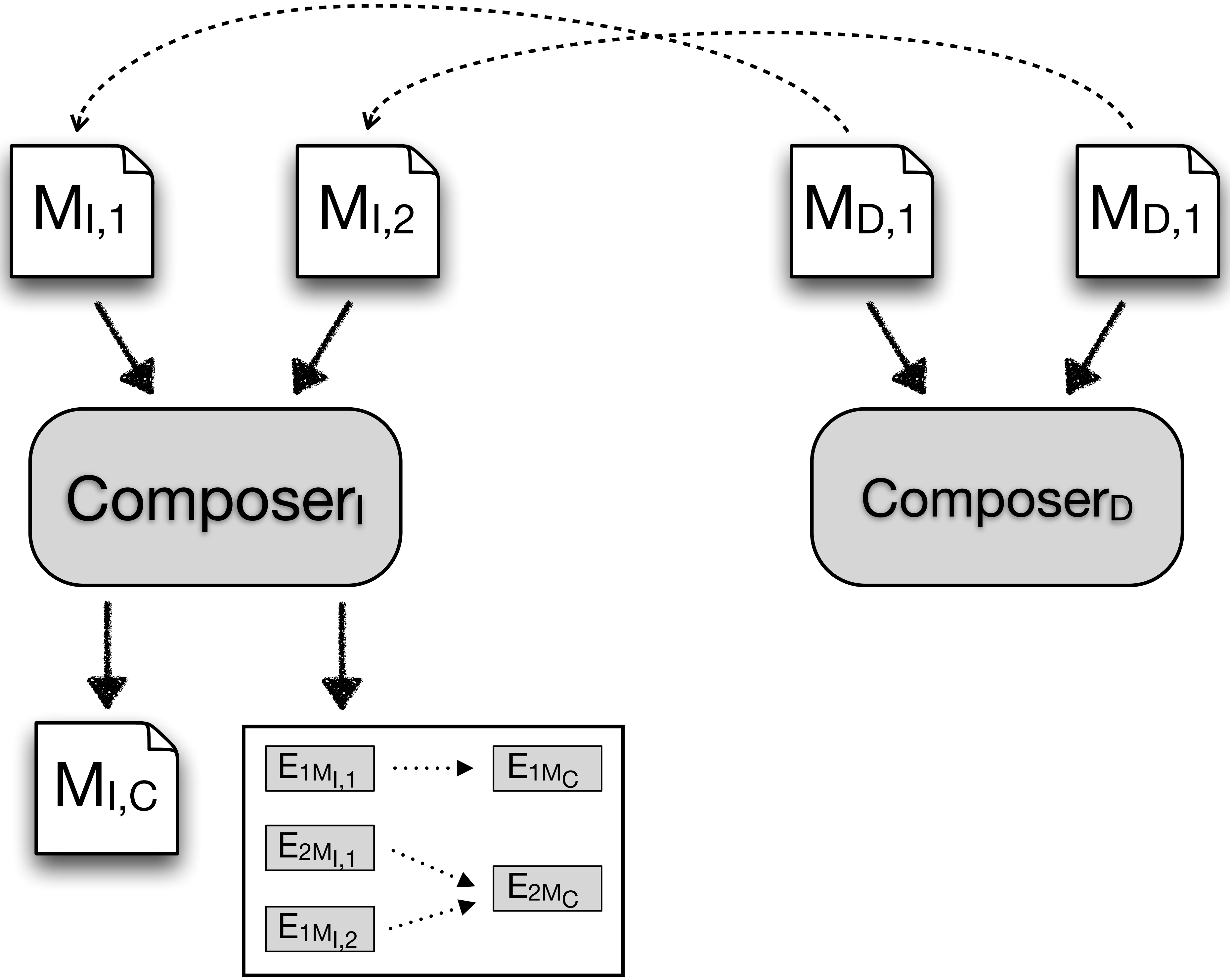
Integration Strategy: Composing



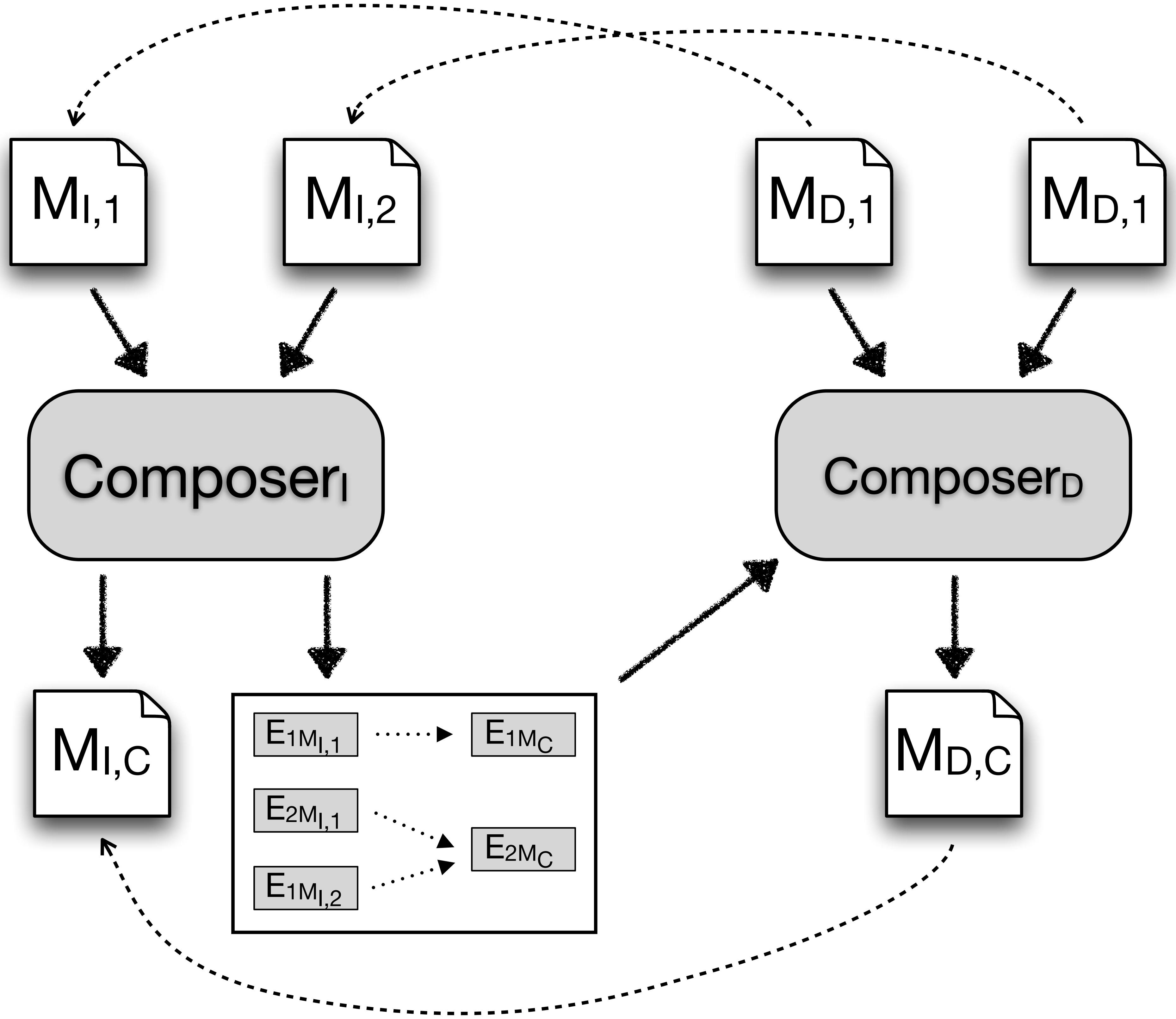
Integration Strategy: Composing



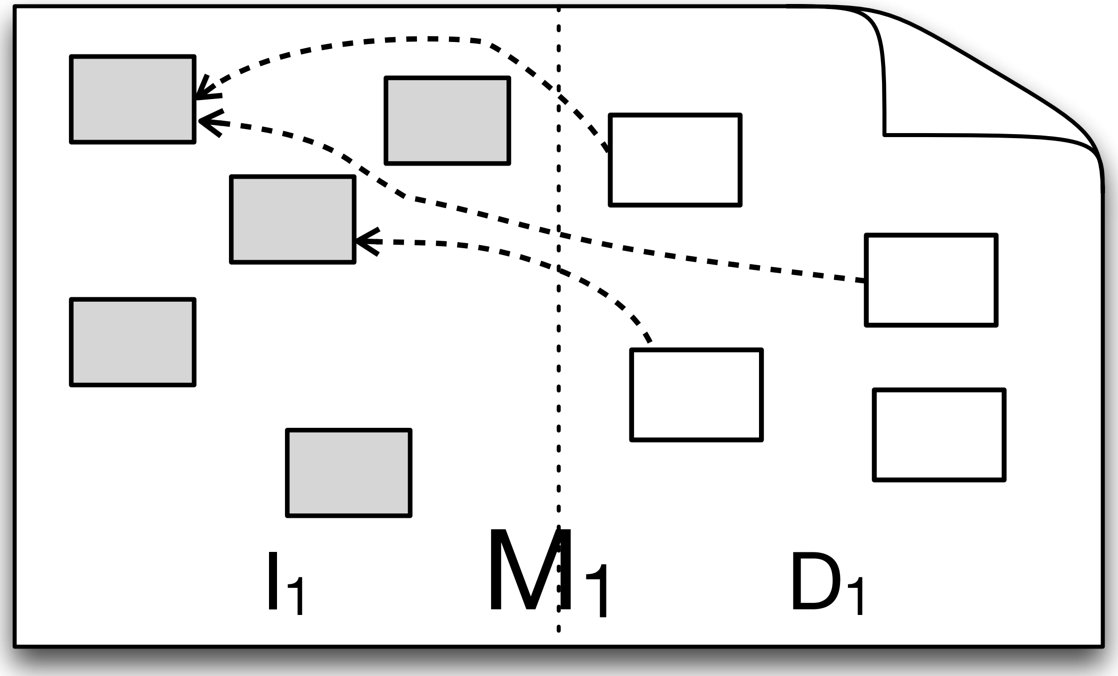
Integration Strategy: Composing

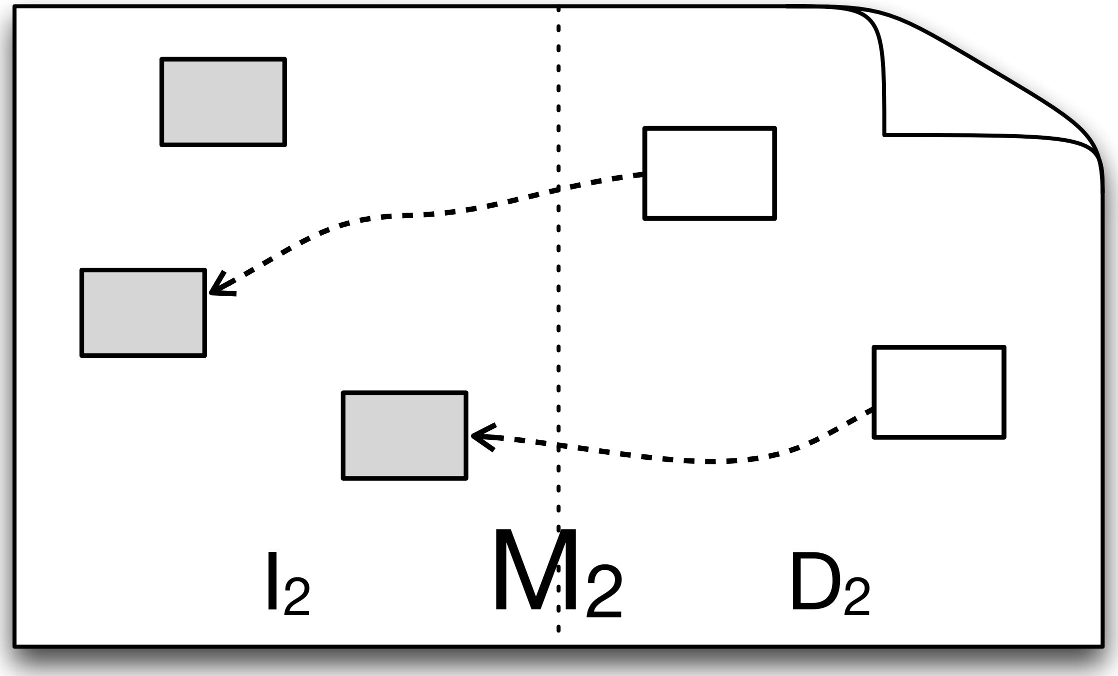


Integration Strategy: Composing

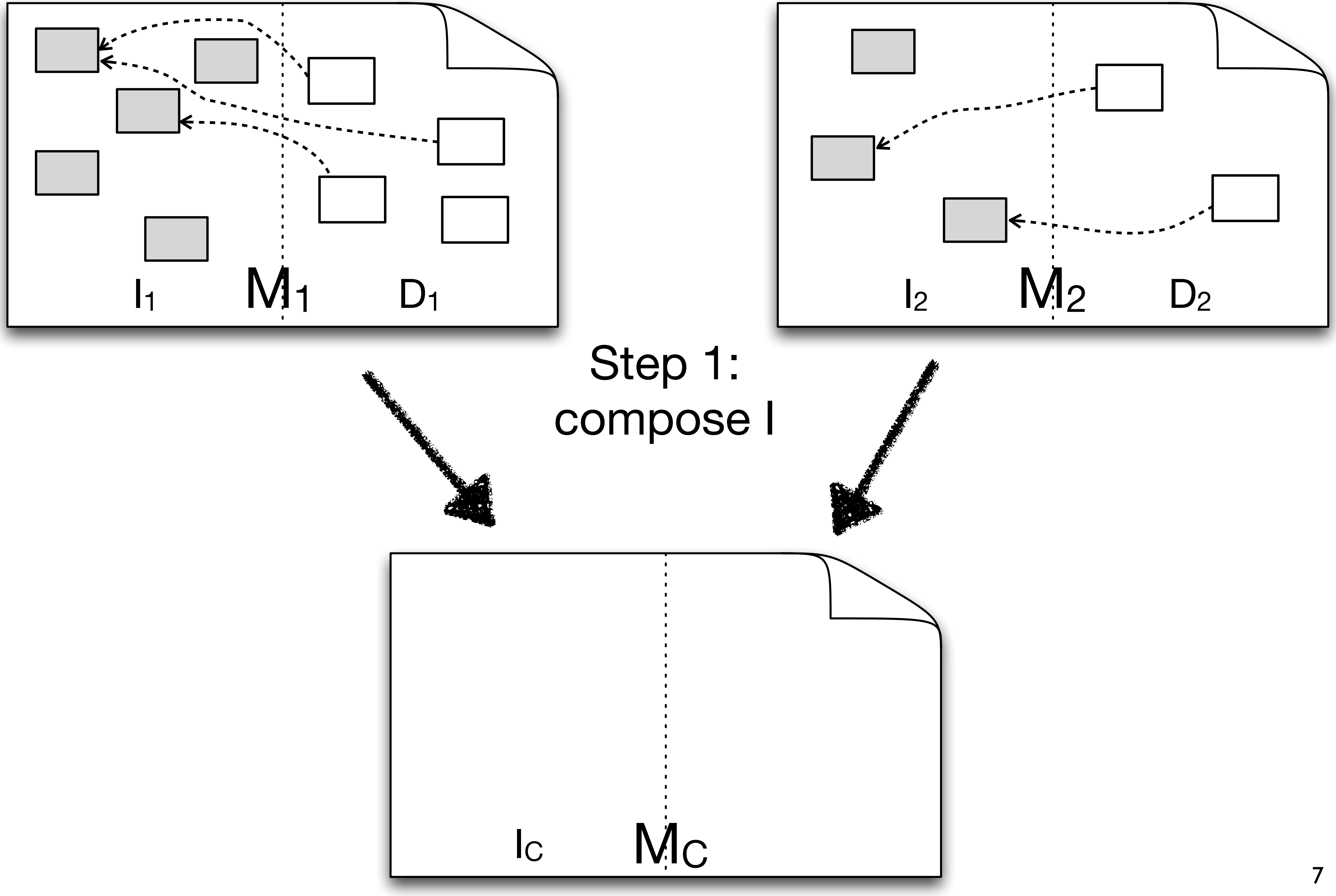


Integration Strategy: Process



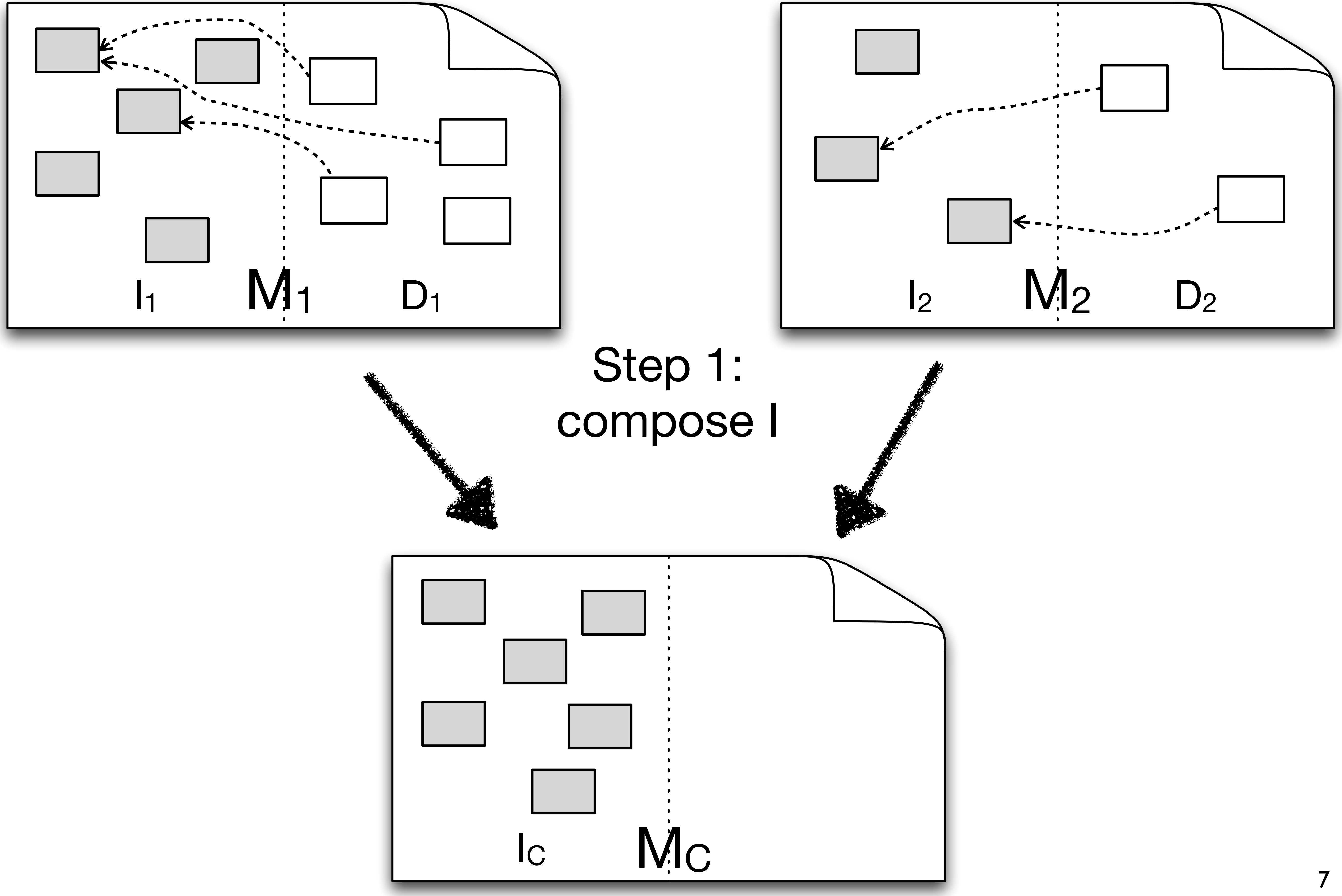


Integration Strategy: Process



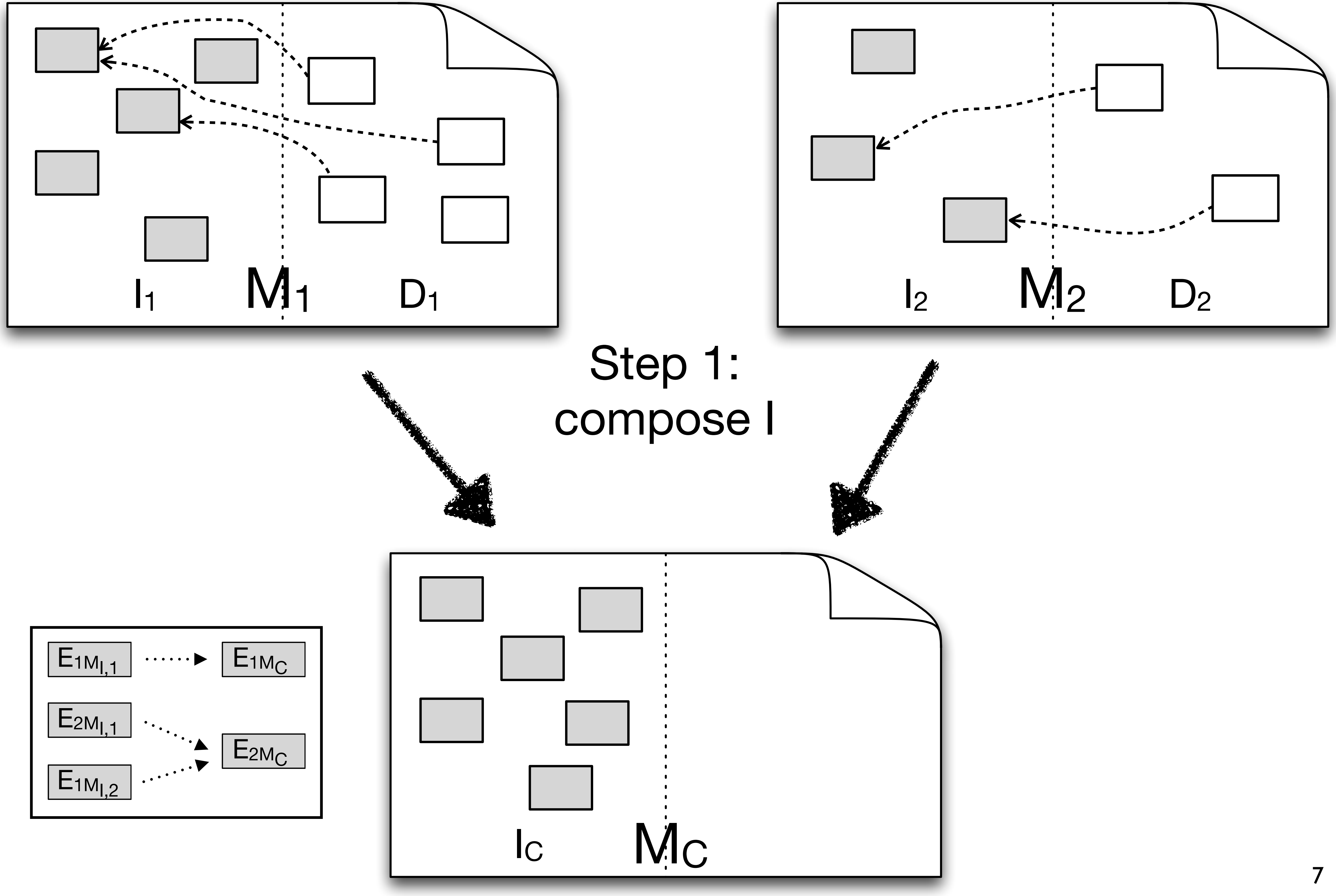
Step 1:
compose I

Integration Strategy: Process



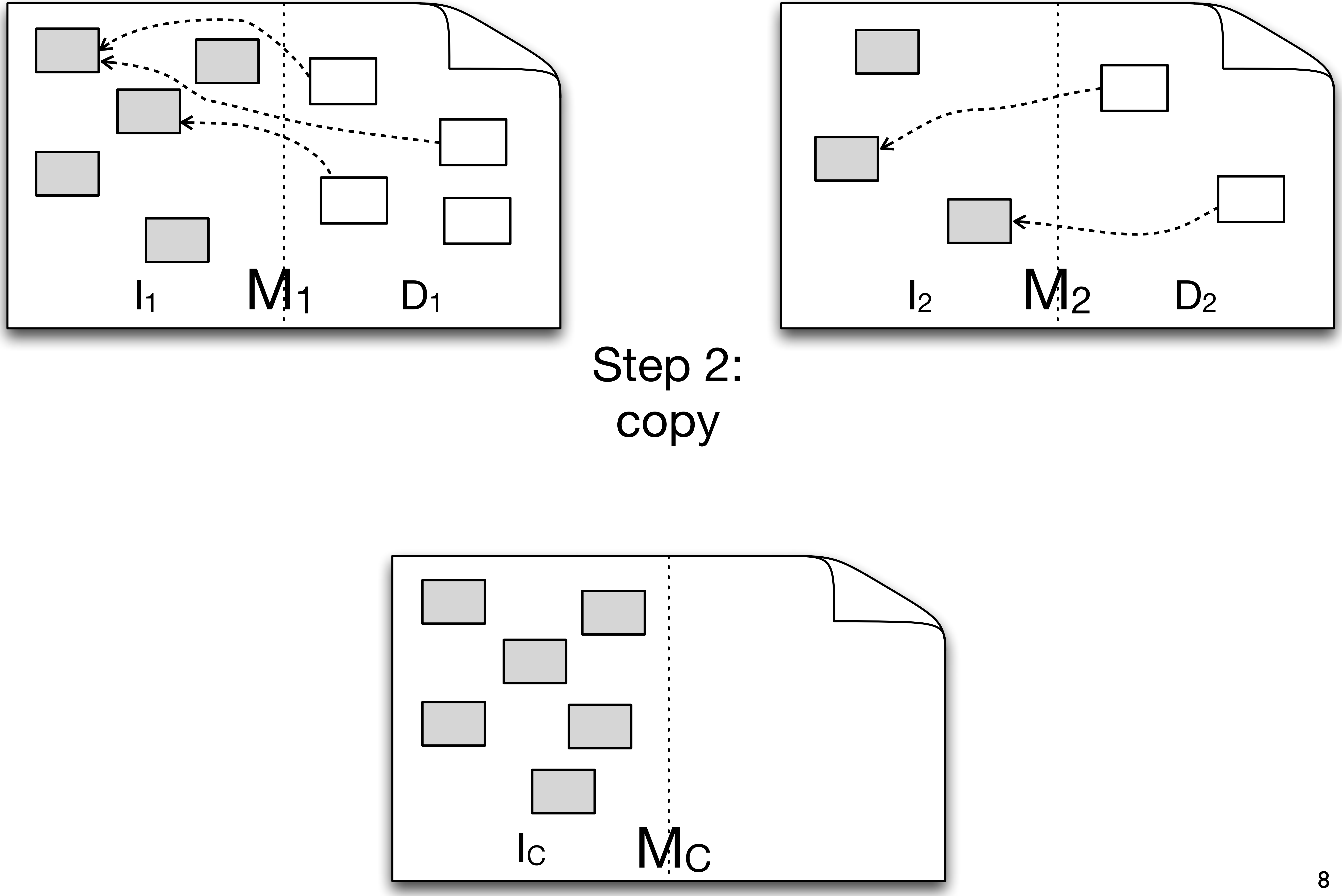
Step 1:
compose I

Integration Strategy: Process



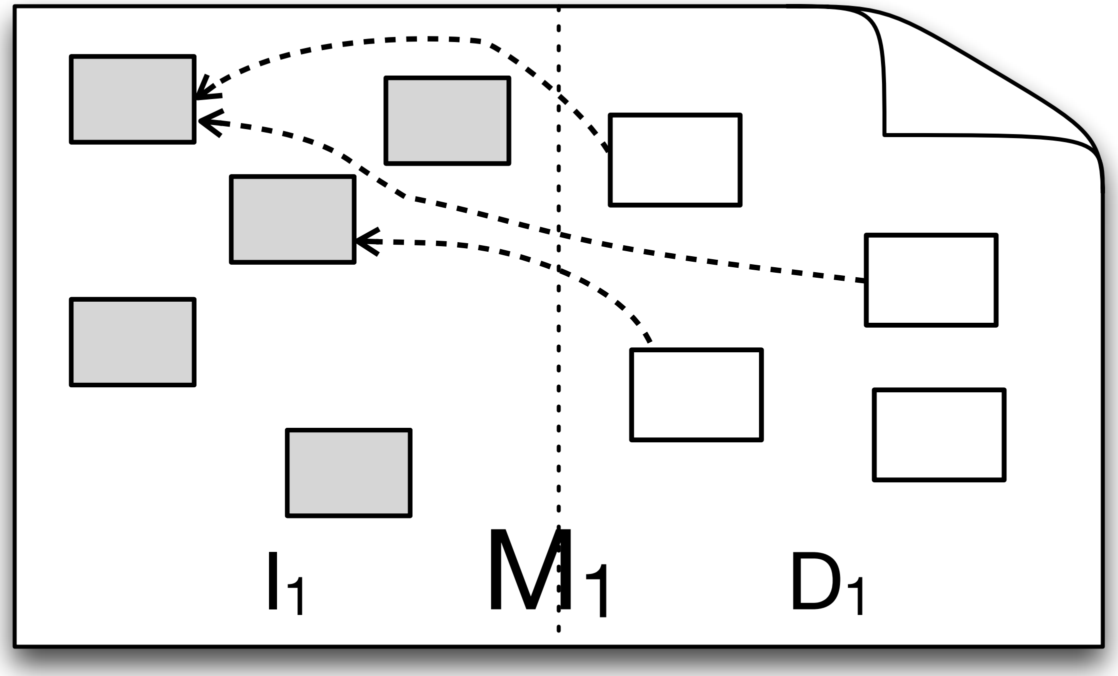
Step 1: compose I

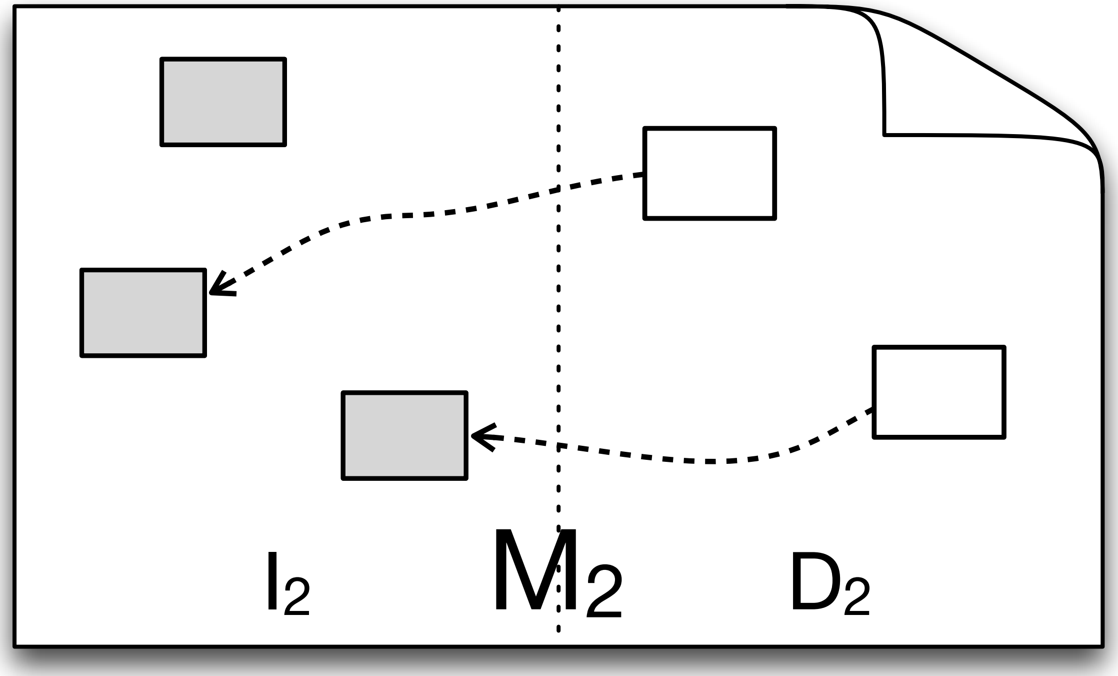
Integration Strategy: Process



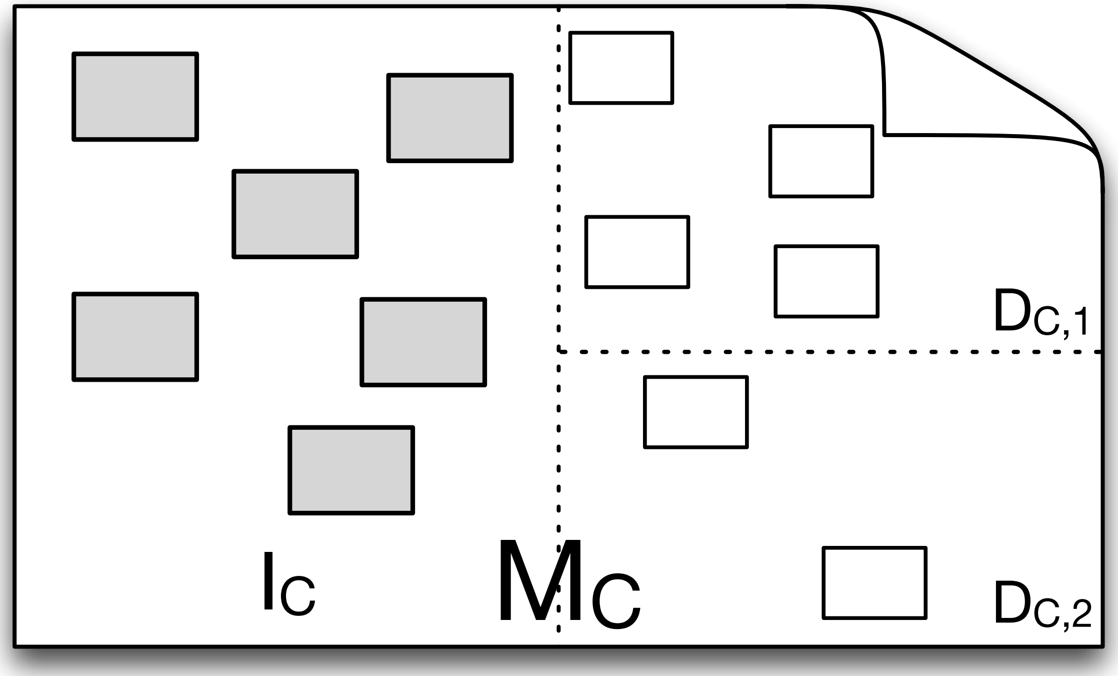
Step 2:
copy

Integration Strategy: Process

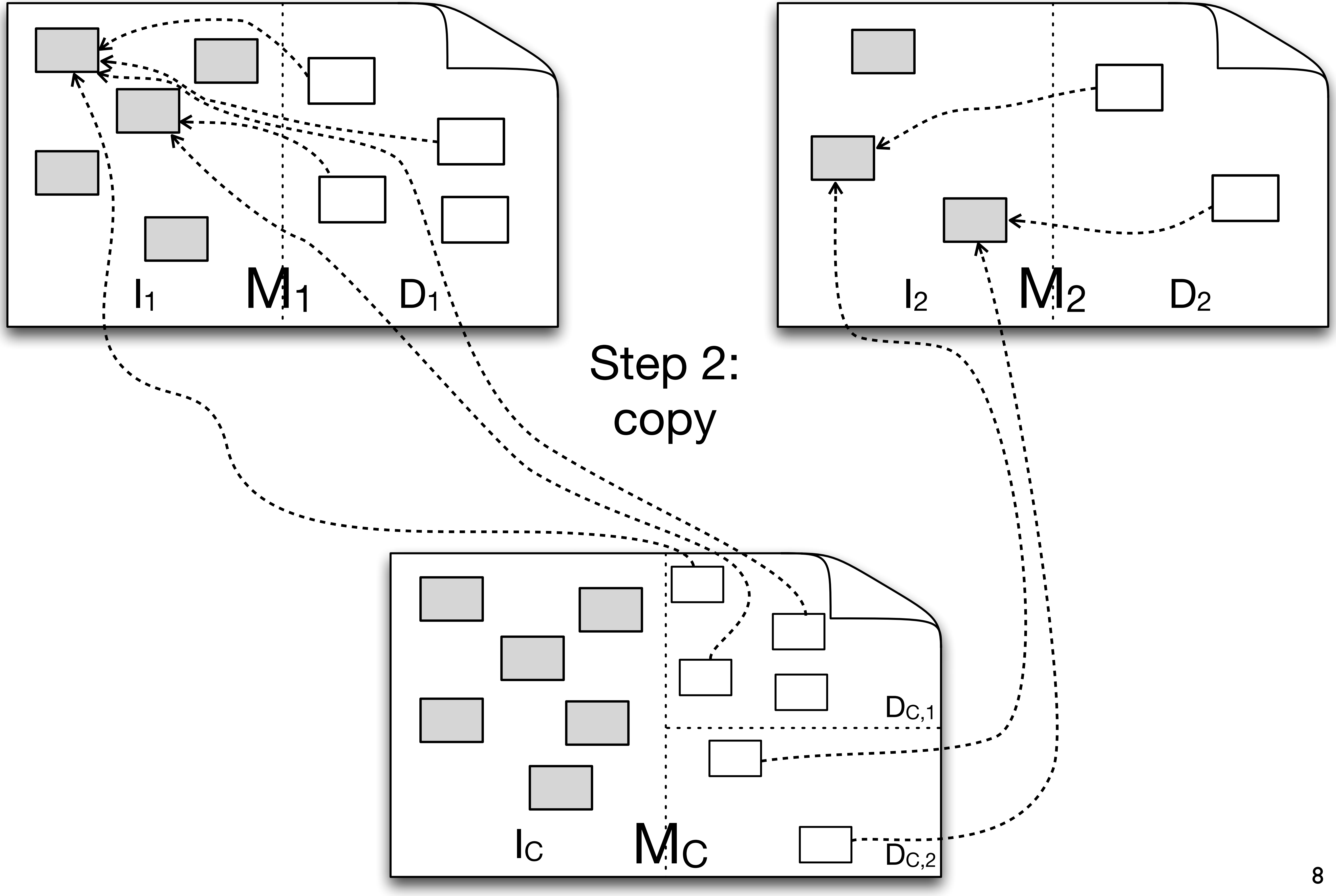




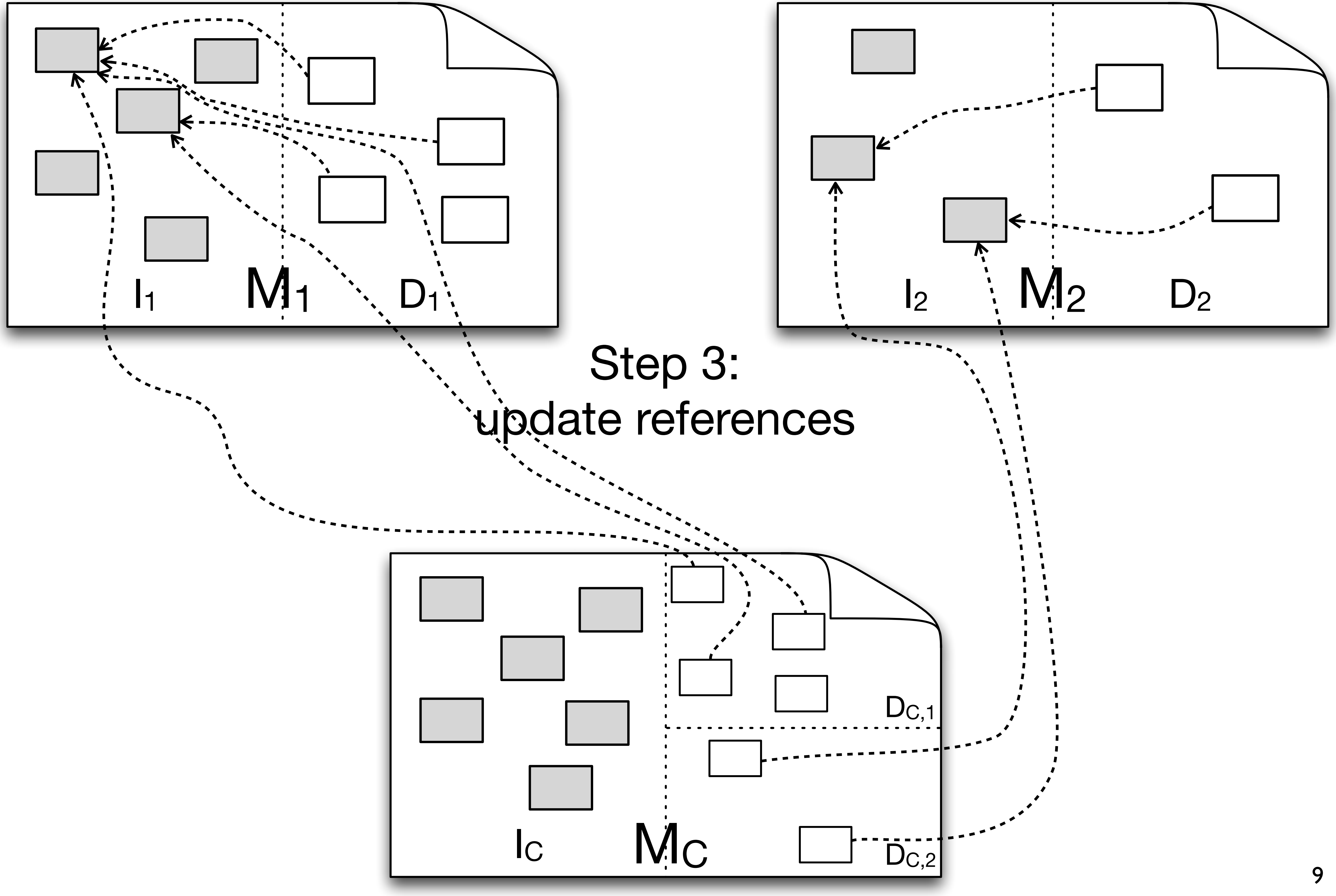
Step 2:
copy



Integration Strategy: Process

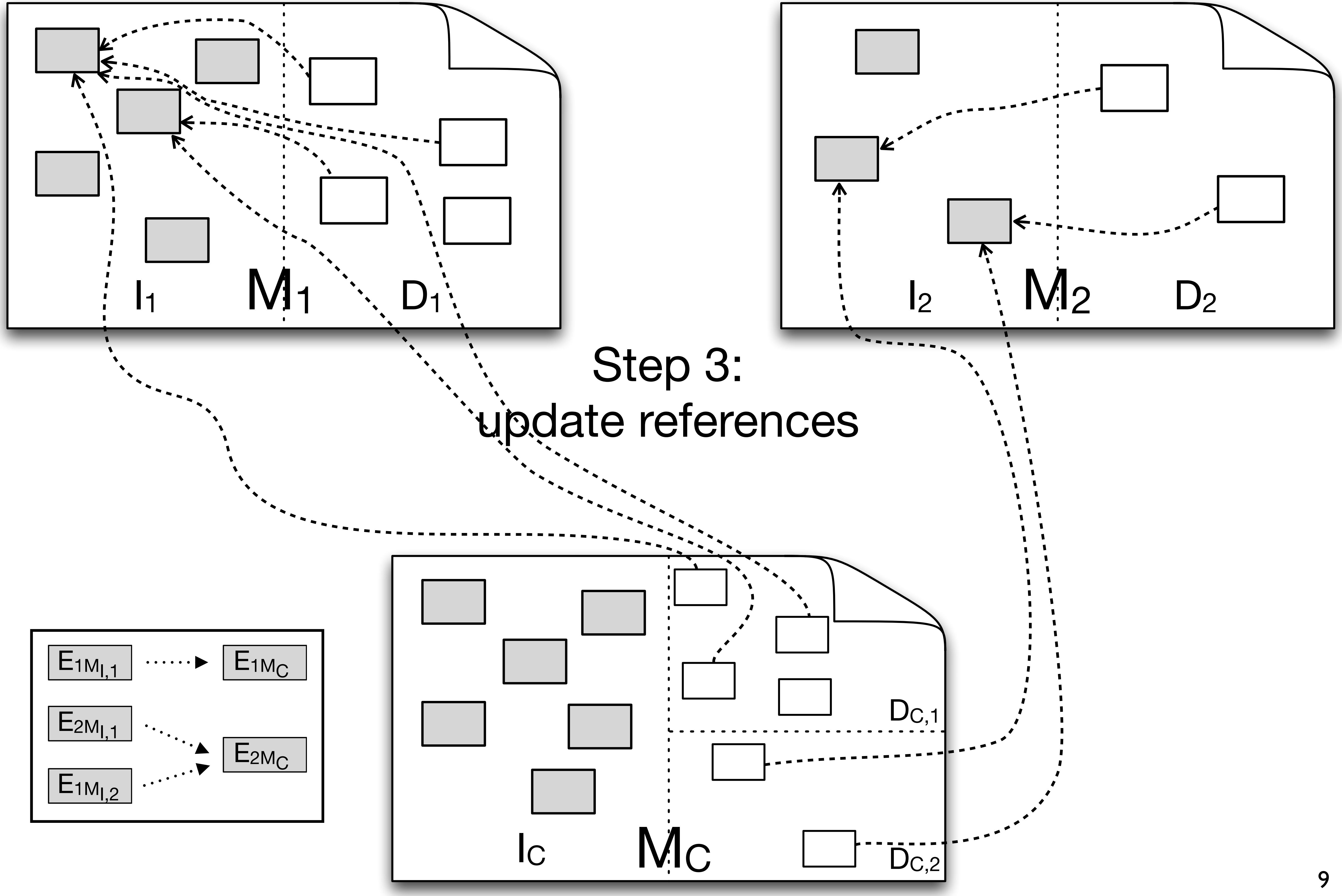


Integration Strategy: Process



Step 3:
update references

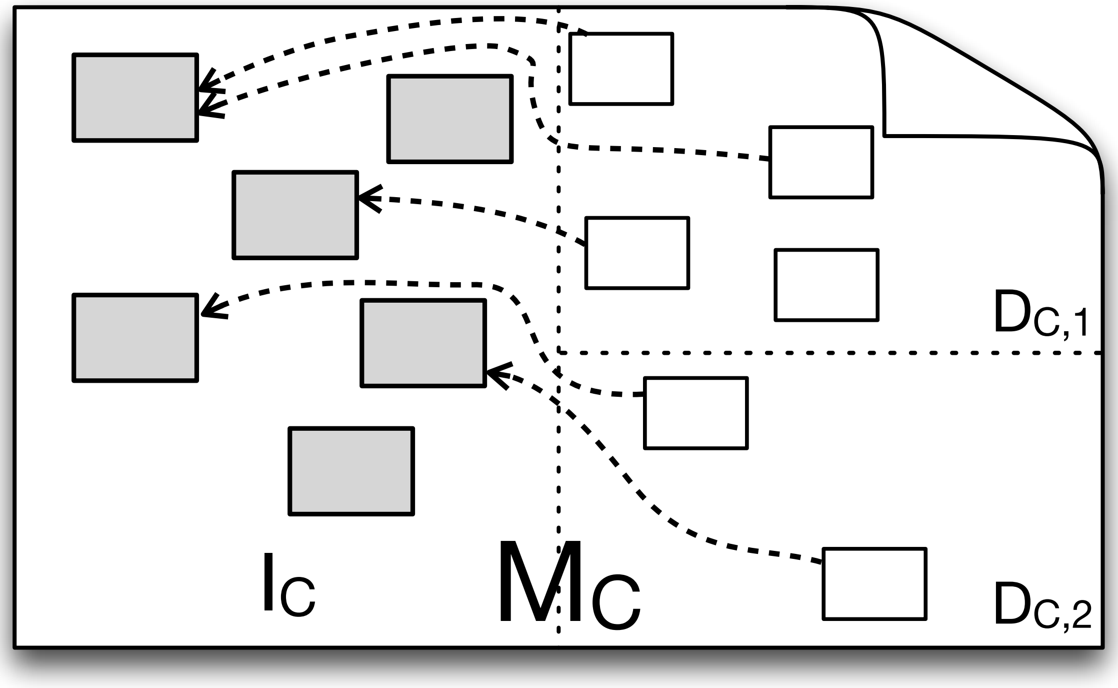
Integration Strategy: Process



Step 3:
update references

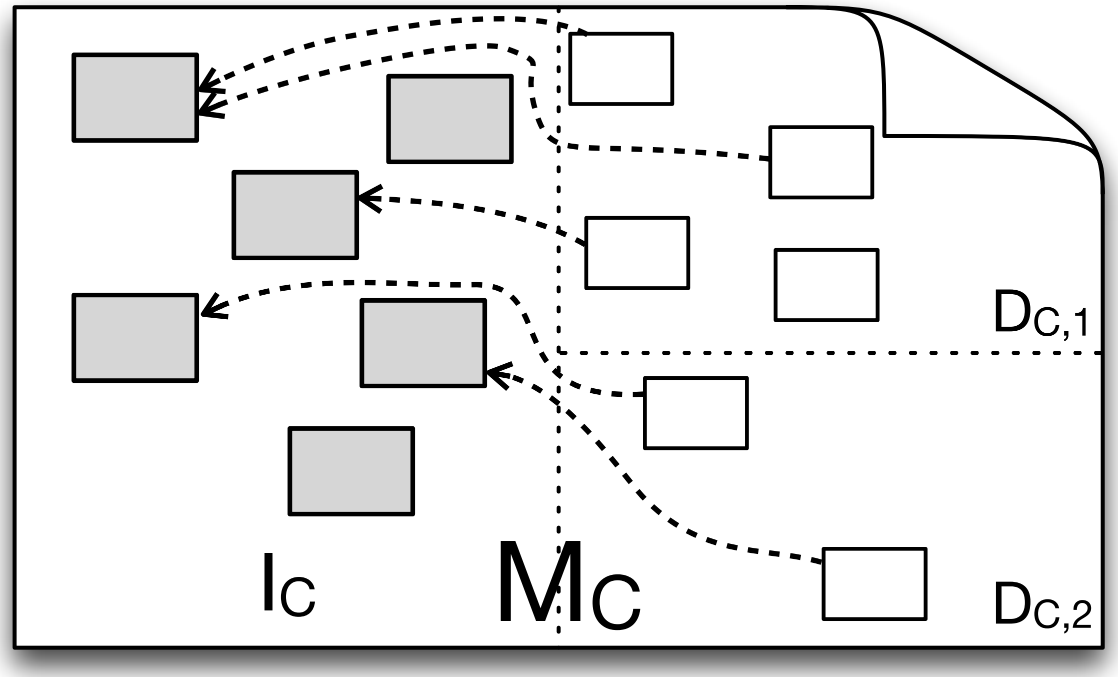
Integration Strategy: Process

Step 3:
update references



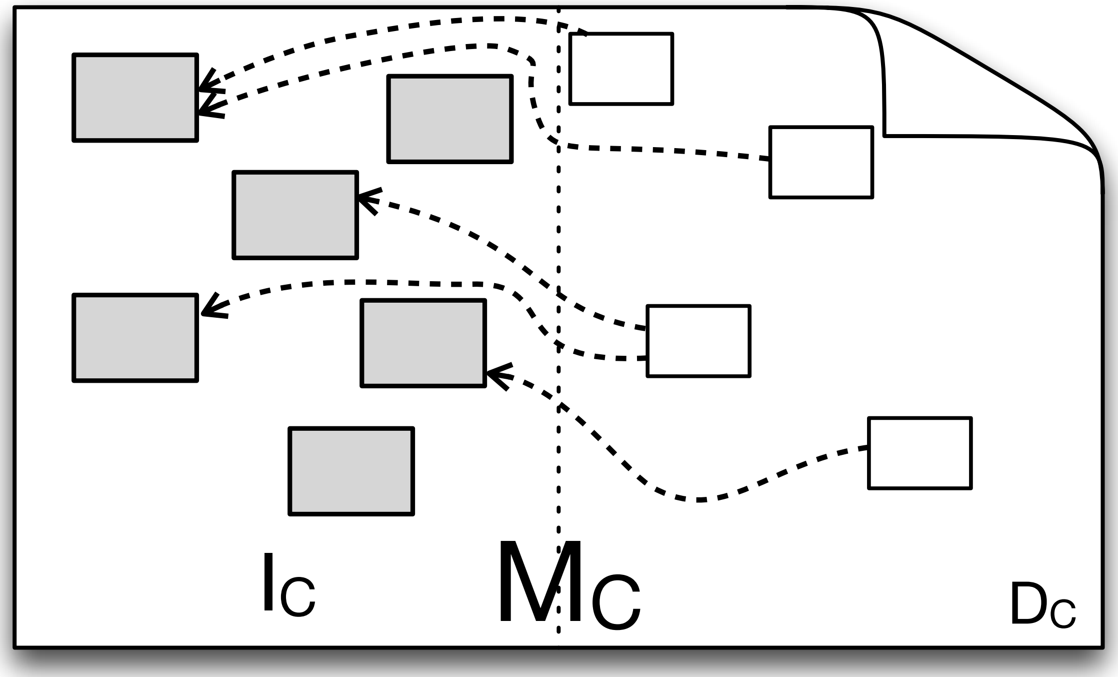
Integration Strategy: Process

Step 4:
compose D_C

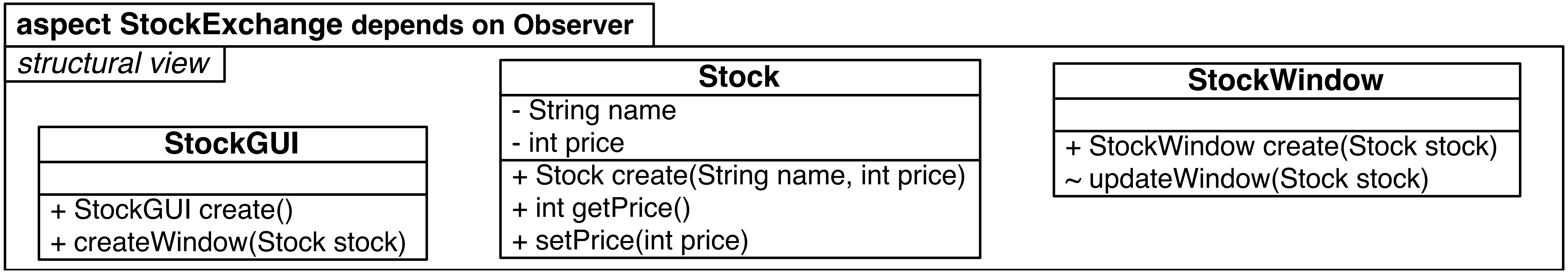


Integration Strategy: Process

Step 4:
compose D_C



Practical Application to Reusable Aspect Models (RAM)



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

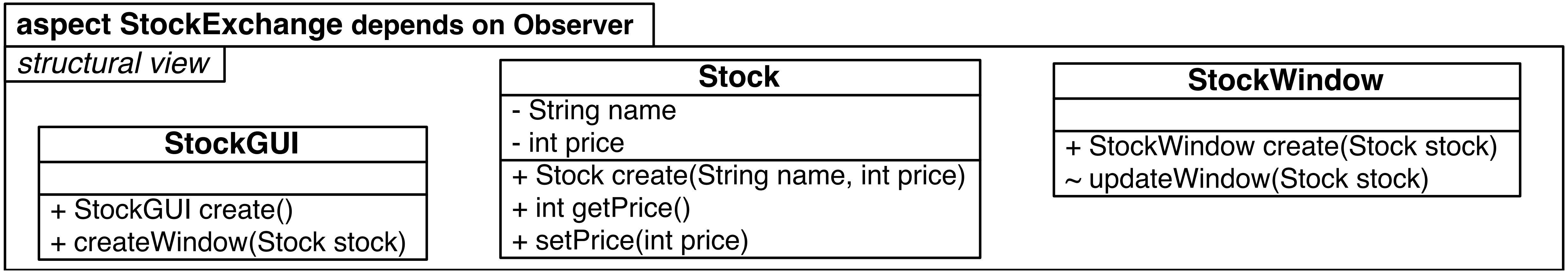
- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

Practical Application to Reusable Aspect Models (RAM)



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

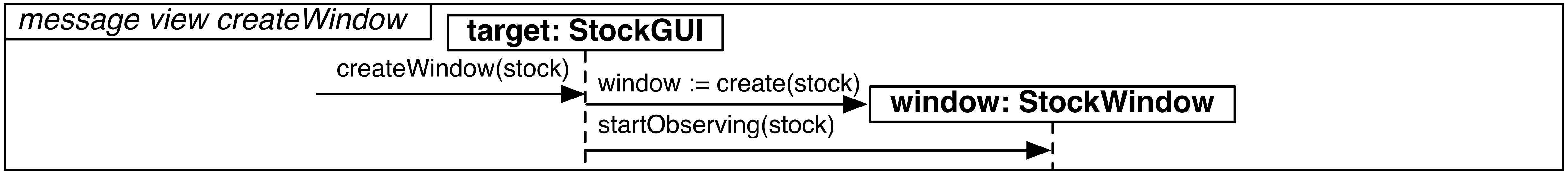
Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)



message view createWindow

target: StockGUI

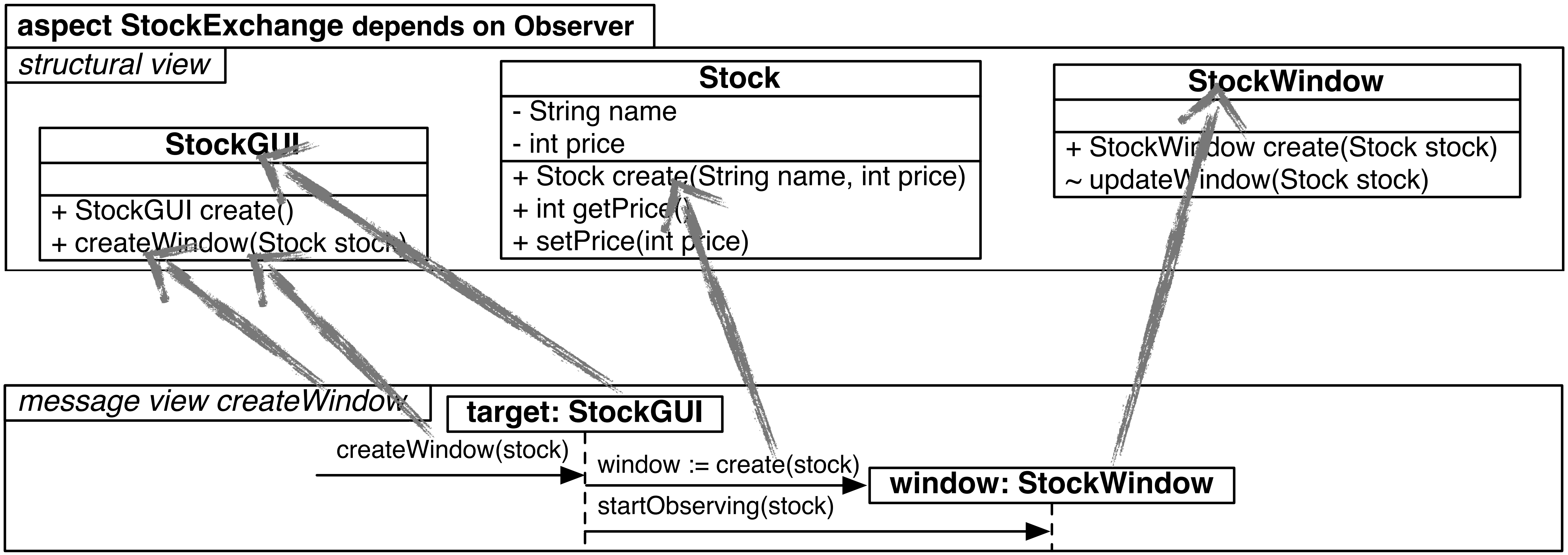
createWindow(stock)

window := create(stock)

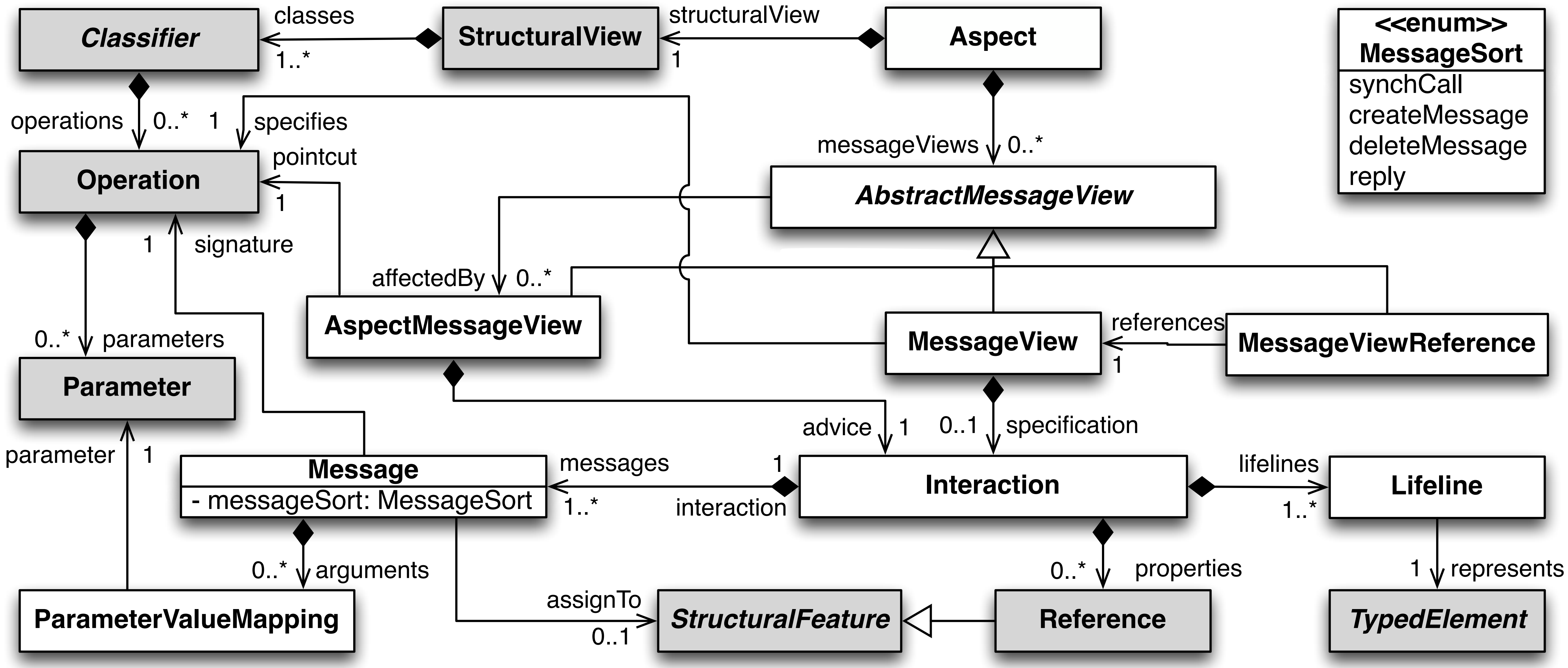
window: StockWindow

startObserving(stock)

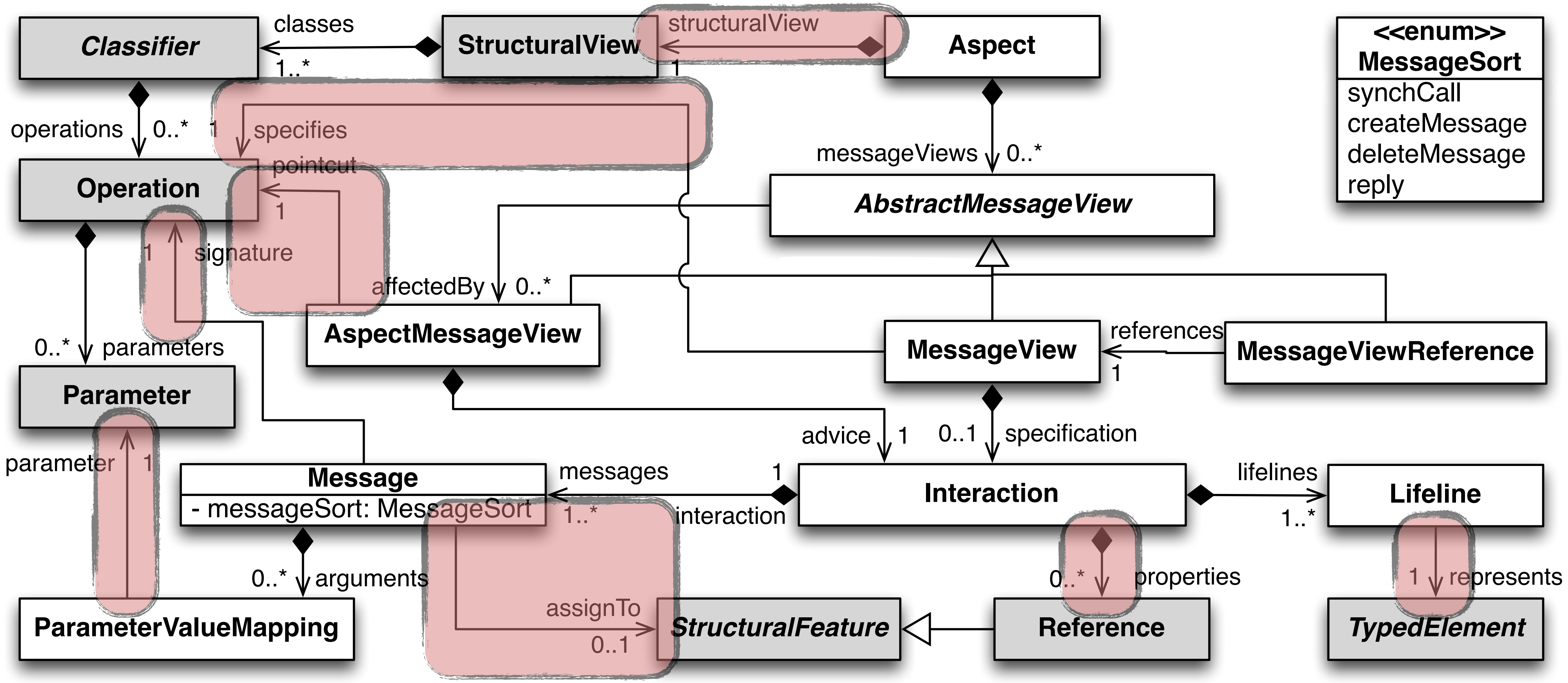
Practical Application to Reusable Aspect Models (RAM)



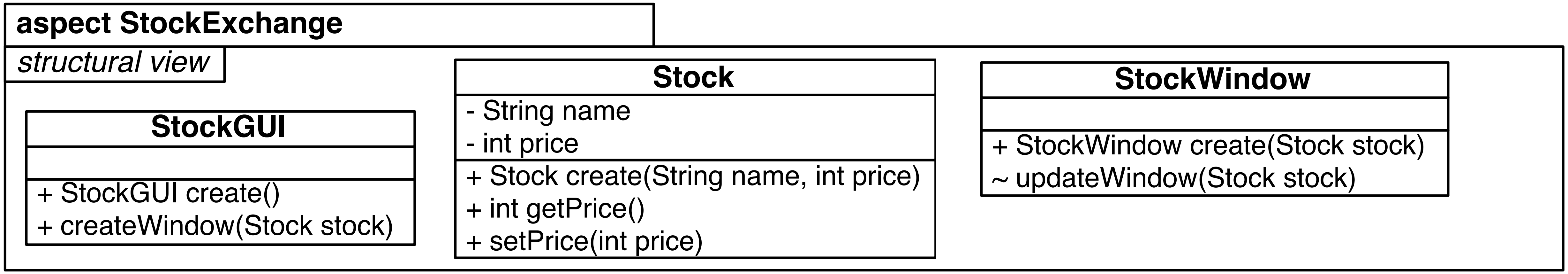
RAM: Metamodel



RAM: Metamodel



RAM: Concern Reuse



aspect StockExchange

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

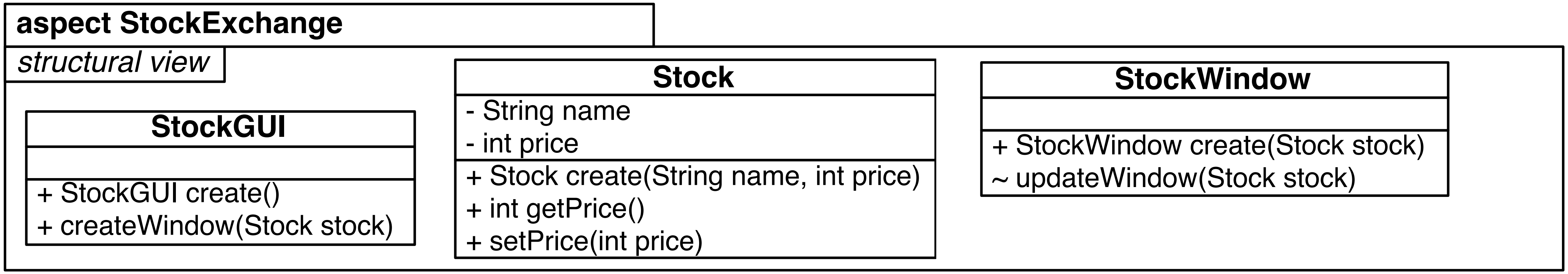
- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

RAM: Concern Reuse



aspect StockExchange

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

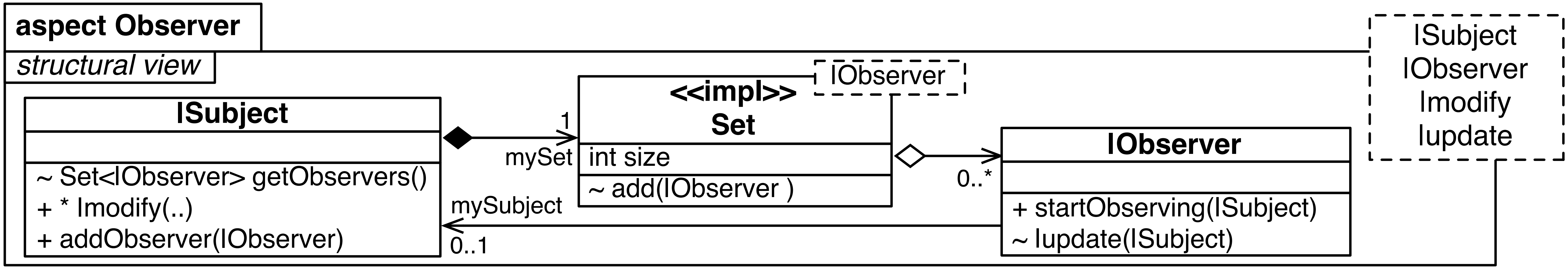
Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)



aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

<<impl>>
Set

IObserver

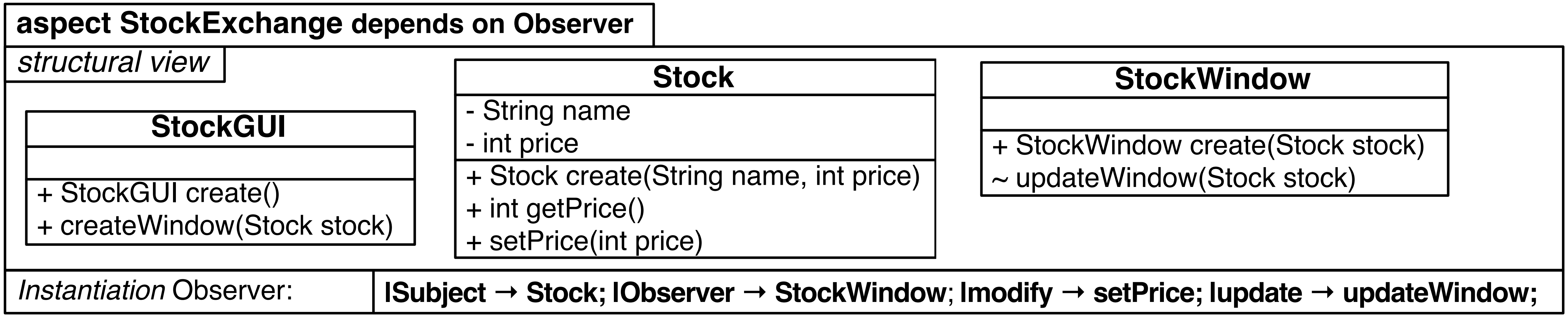
int size

~ add(IObserver)

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

RAM: Concern Reuse



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

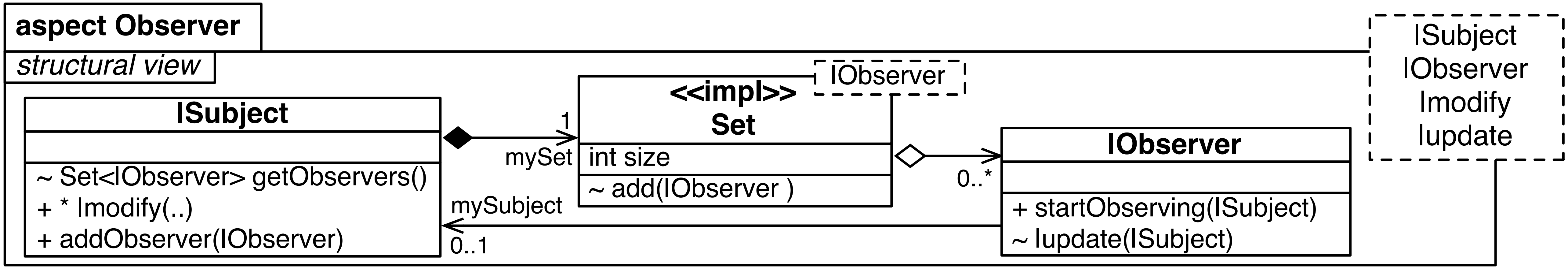
- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

Instantiation Observer: **ISubject → Stock; IObserver → StockWindow; Imodify → setPrice; Iupdate → updateWindow;**



aspect Observer

structural view

ISubject
IObserver
Imodify
Iupdate

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

<<impl>>
Set

IObserver

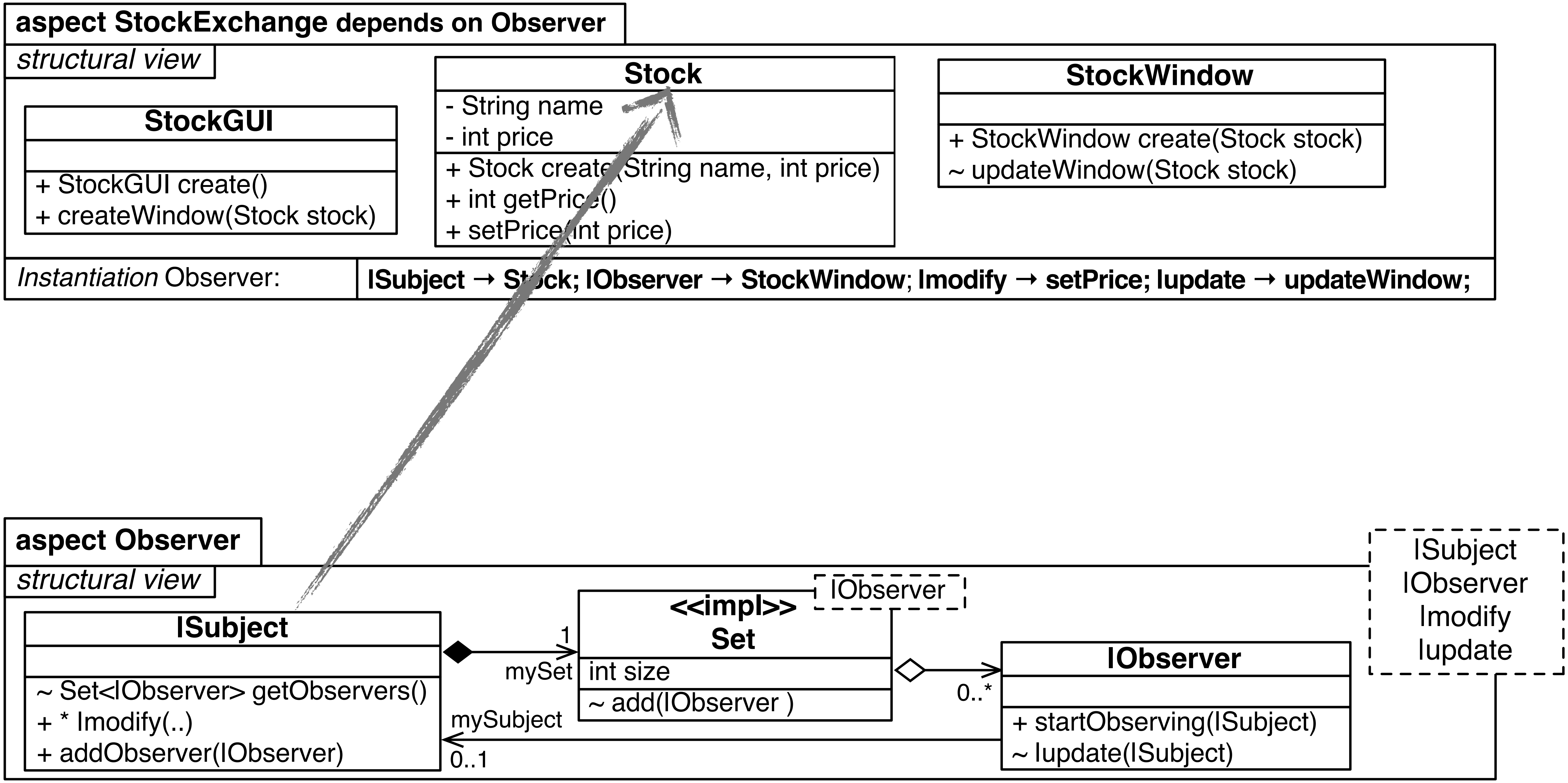
int size

~ add(IObserver)

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

RAM: Concern Reuse



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

Instantiation Observer: **ISubject → Stock; IObserver → StockWindow**; **Imodify → setPrice; Iupdate → updateWindow;**

aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

<<impl>> Set

IObserver

int size

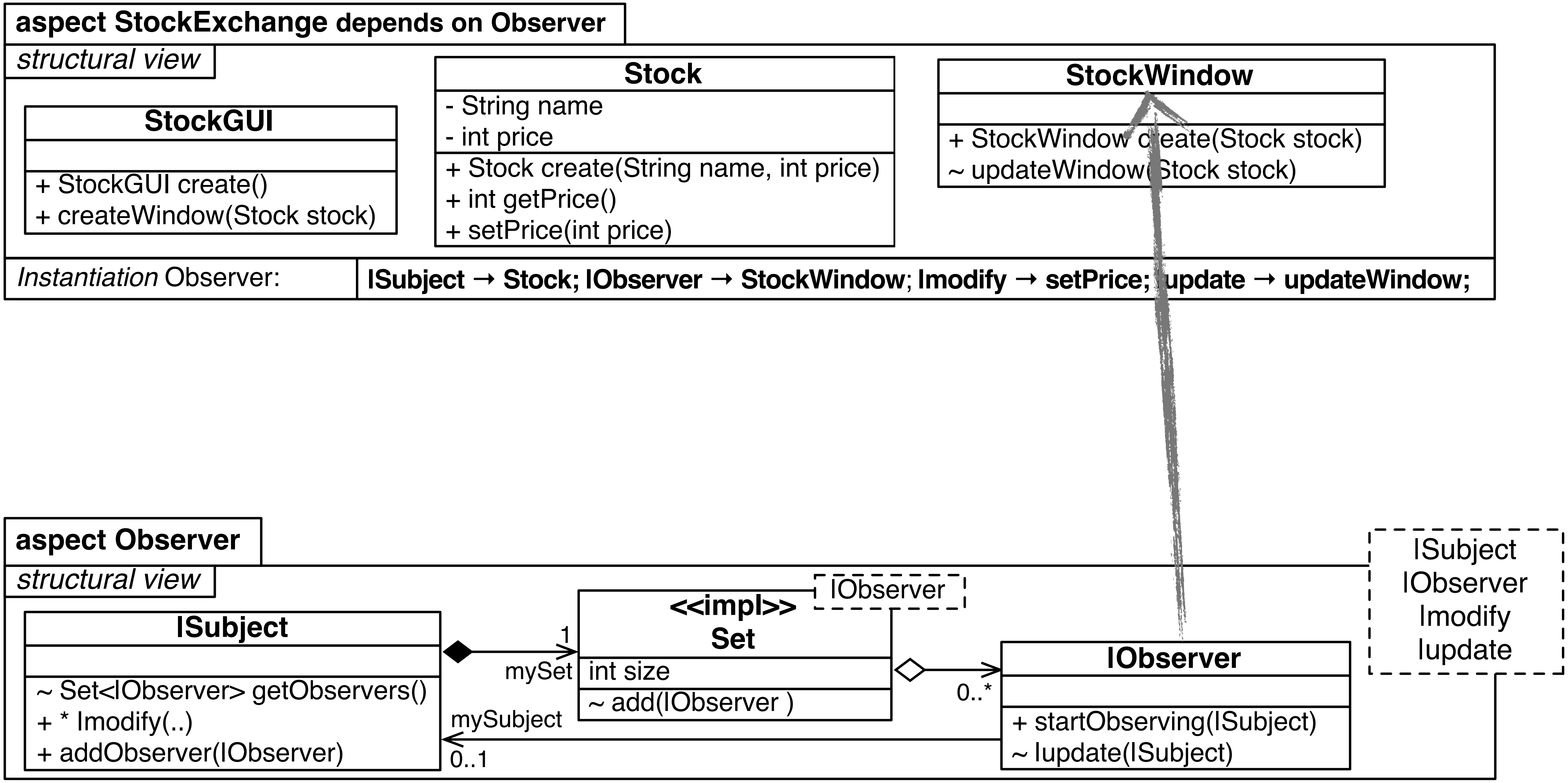
~ add(IObserver)

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

ISubject
IObserver
Imodify
Iupdate

RAM: Concern Reuse



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

Instantiation Observer: **ISubject → Stock; IObserver → StockWindow; Imodify → setPrice; Iupdate → updateWindow;**

aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

1 mySet

<<impl>> Set

IObserver

int size

~ add(IObserver)

0..*

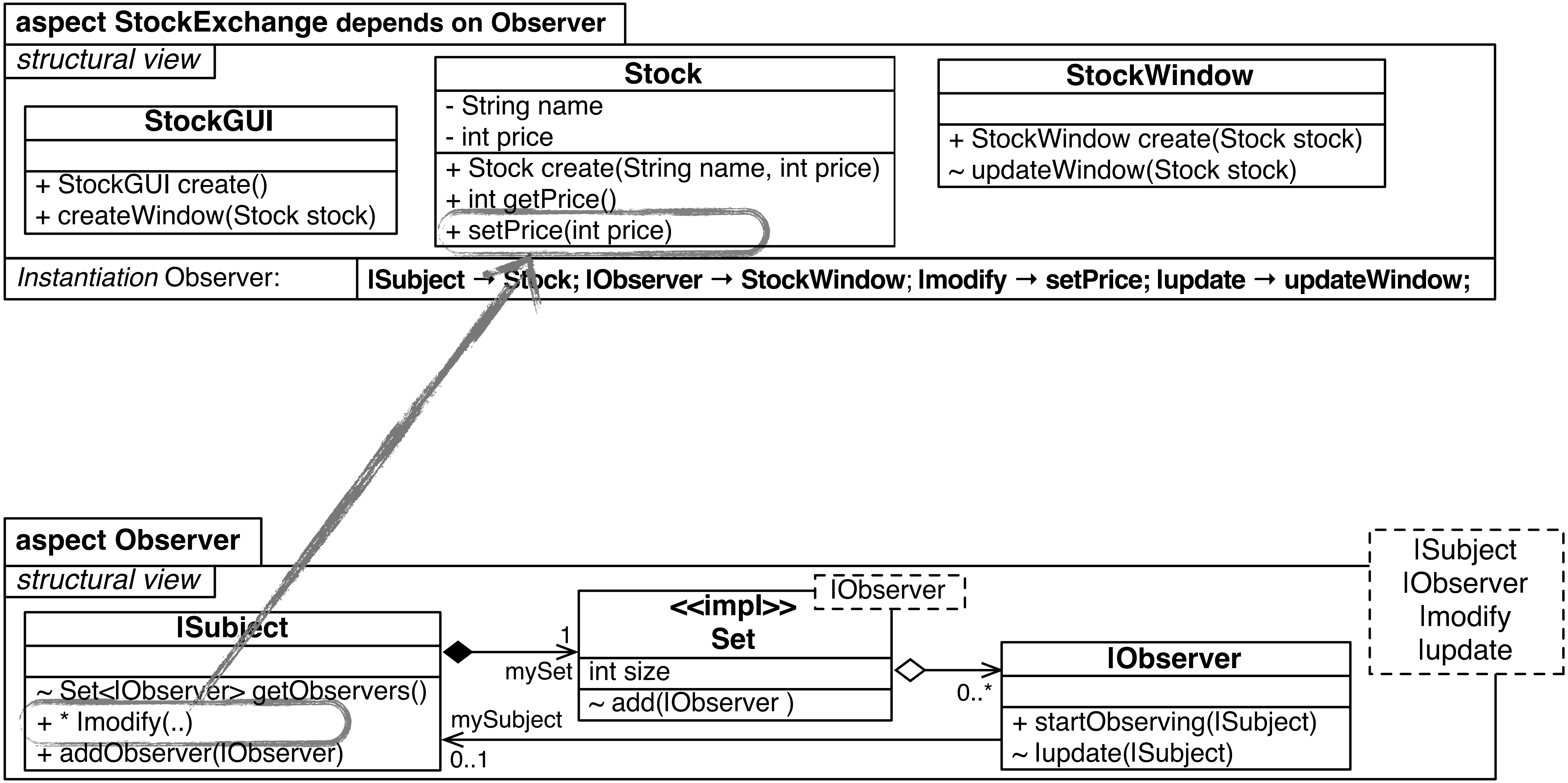
IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

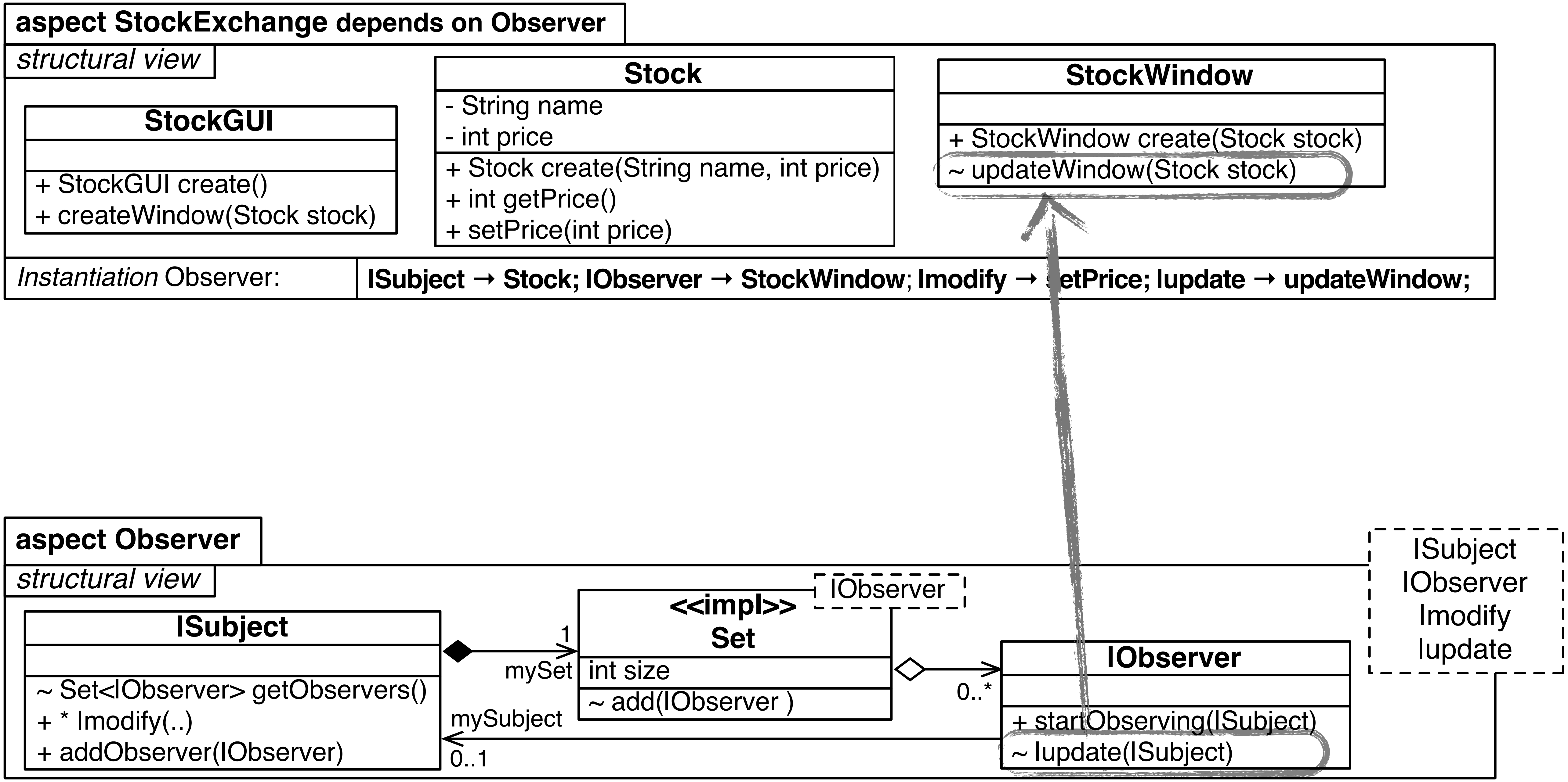
mySubject 0..1

ISubject
IObserver
Imodify
Iupdate

RAM: Concern Reuse



RAM: Concern Reuse



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

Instantiation Observer: **ISubject → Stock; IObserver → StockWindow; Imodify → setPrice; Iupdate → updateWindow;**

aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

1
mySet

<<impl>> Set

IObserver

int size

~ add(IObserver)

0..*

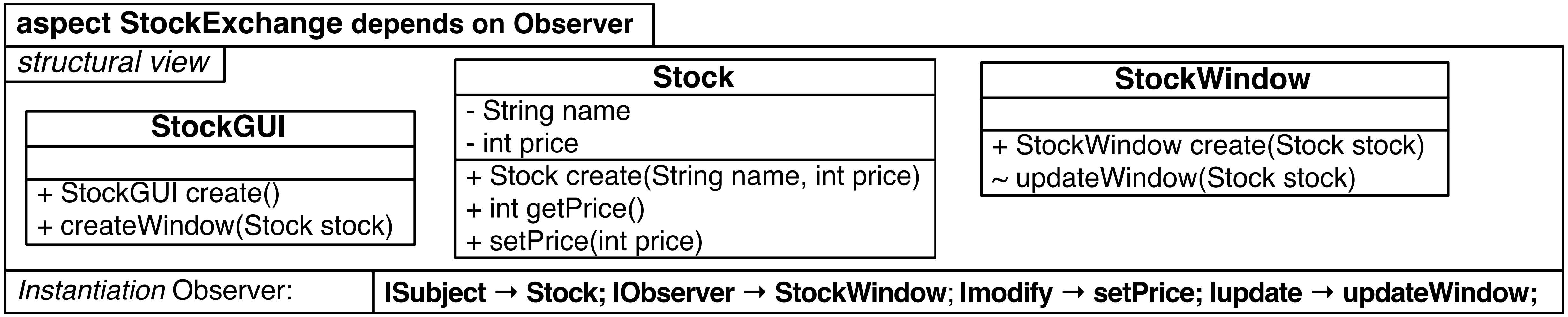
IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

mySubject
0..1

ISubject
IObserver
Imodify
Iupdate

RAM: Weaving



aspect StockExchange depends on Observer

structural view

StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

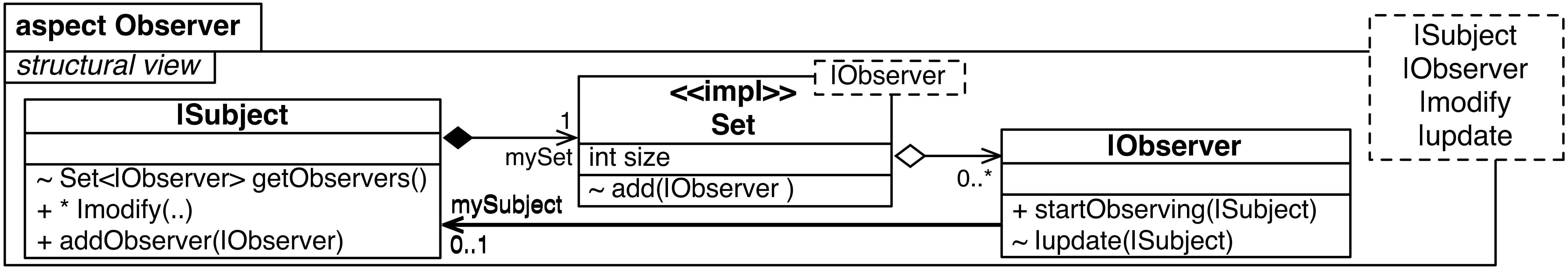
- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)

Instantiation Observer: **ISubject → Stock; IObserver → StockWindow; Imodify → setPrice; Iupdate → updateWindow;**



aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

<<impl>> Set

IObserver

int size

~ add(IObserver)

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

1

mySet

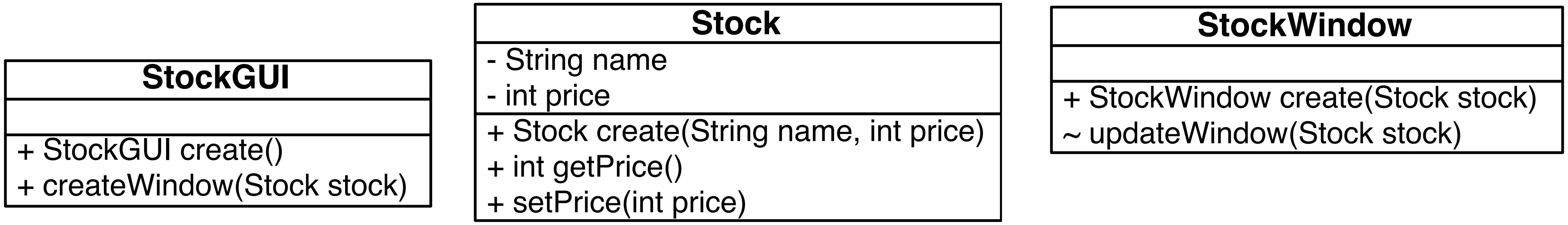
0..*

mySubject

0..1

ISubject
IObserver
Imodify
Iupdate

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

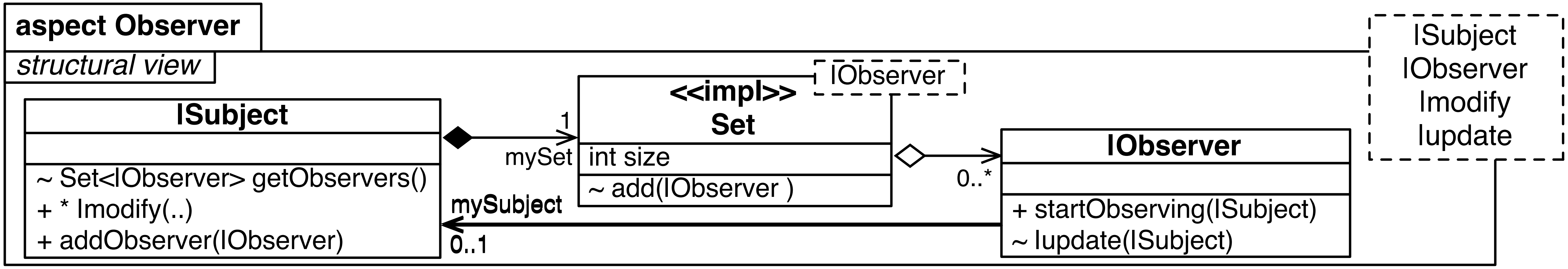
Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)



aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

1
mySet

<<impl>> Set

IObserver

int size

~ add(IObserver)

0..*

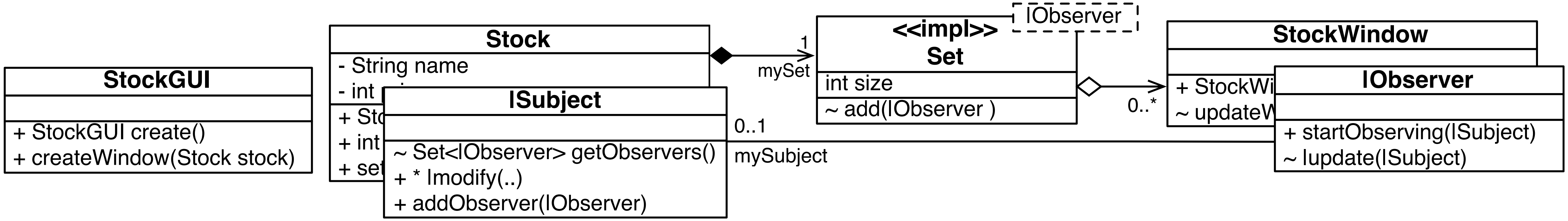
IObserver

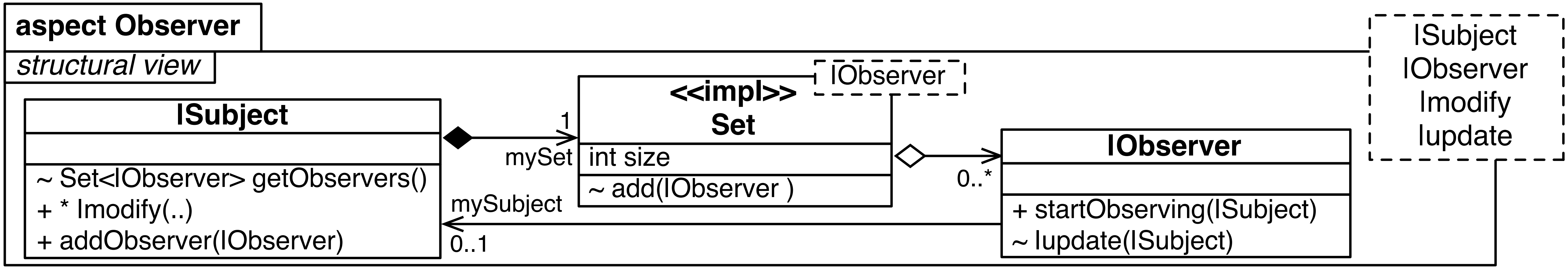
+ startObserving(ISubject)
~ Iupdate(ISubject)

mySubject
0..1

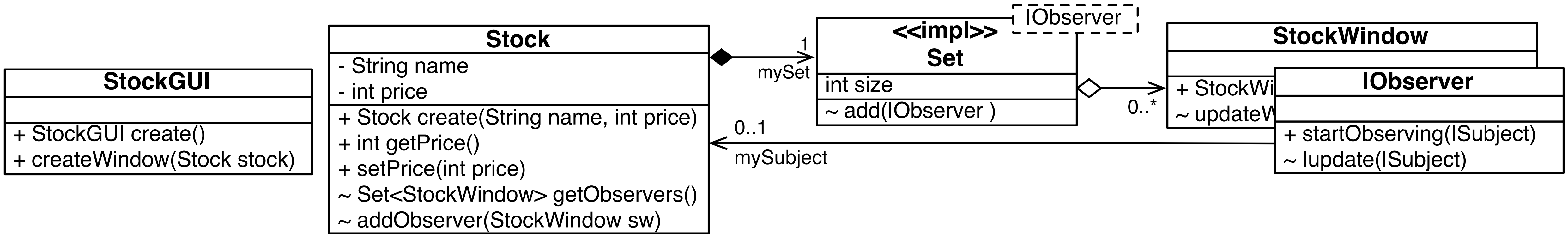
ISubject
IObserver
Imodify
Iupdate

RAM: Weaving


StockGUI
+ StockGUI create()
+ createWindow(Stock stock)
Stock
- String name
- int
+ Sto
+ int
+ set
ISubject
~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)
1
mySet
0..1
mySubject
<<impl>>
Set
IObserver
int size
~ add(IObserver)
0..*
StockWindow
+ StockWi
~ updateW
IObserver
+ startObserving(ISubject)
~ Iupdate(ISubject)


aspect Observer
structural view
ISubject
IObserver
Imodify
Iupdate
ISubject
~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)
1
mySet
mySubject
0..1
<<impl>>
Set
IObserver
int size
~ add(IObserver)
0..*
IObserver
+ startObserving(ISubject)
~ Iupdate(ISubject)

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price
+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)

1
mySet

0..1
mySubject

<<impl>>
Set

IObserver

int size

~ add(IObserver)

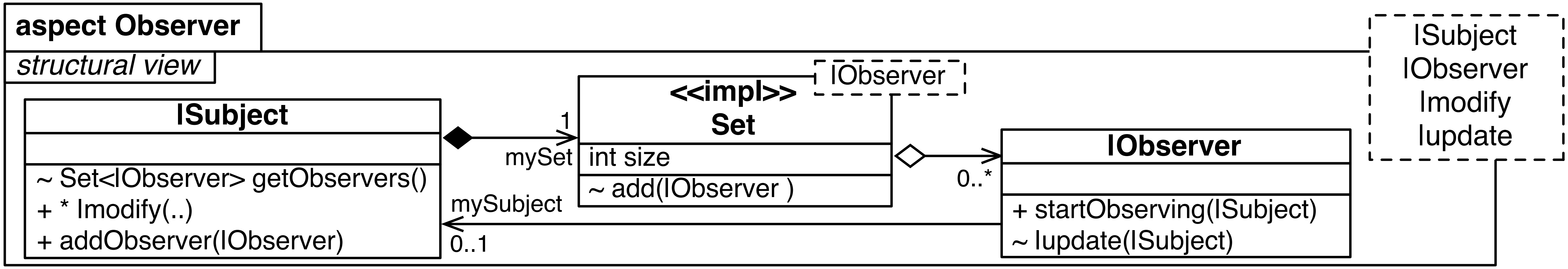
0..*

StockWindow

+ StockWi
~ updateW

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)



aspect Observer

structural view

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

1
mySet

mySubject
0..1

<<impl>>
Set

IObserver

int size

~ add(IObserver)

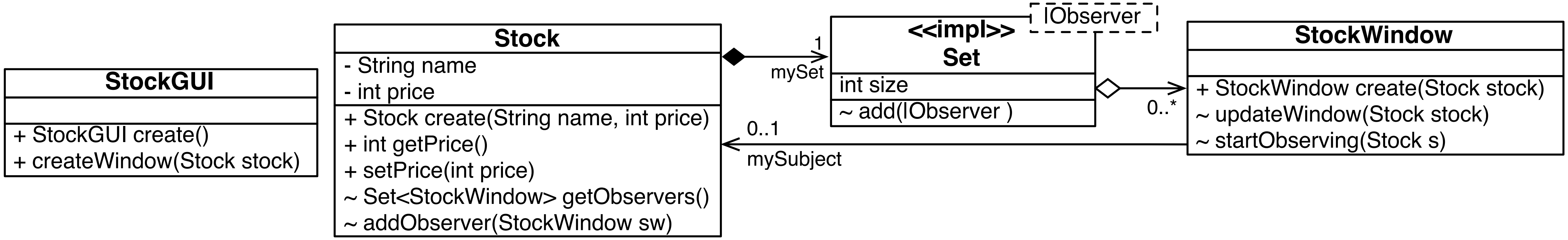
0..*

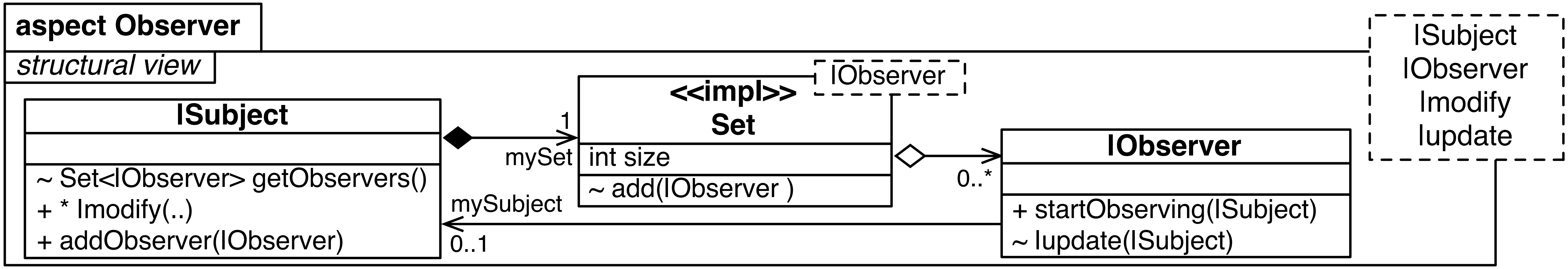
IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

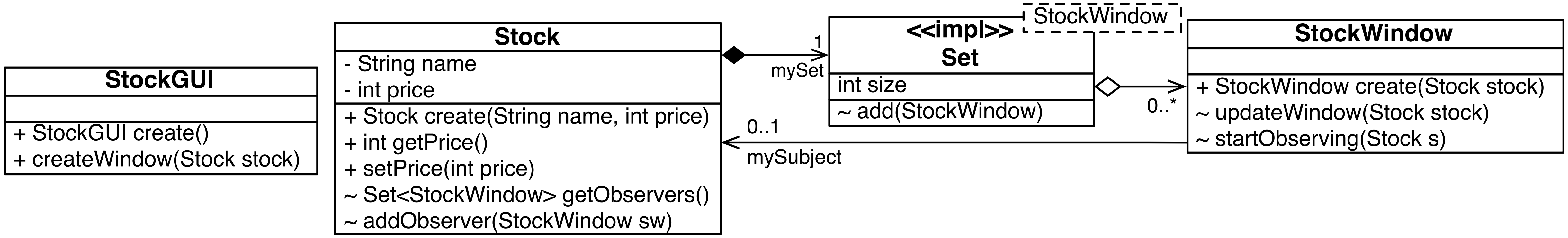
ISubject
IObserver
Imodify
Iupdate

RAM: Weaving


StockGUI
+ StockGUI create()
+ createWindow(Stock stock)
Stock
- String name
- int price
+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)
1
mySet
0..1
mySubject
IObserver
<<impl>>
Set
int size
~ add(IObserver)
0..*
StockWindow
+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)


aspect Observer
structural view
ISubject
~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)
1
mySet
mySubject
0..1
IObserver
<<impl>>
Set
int size
~ add(IObserver)
0..*
IObserver
+ startObserving(ISubject)
~ Iupdate(ISubject)
ISubject
IObserver
Imodify
Iupdate

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)

1
mySet

<<impl>>
Set

StockWindow

int size

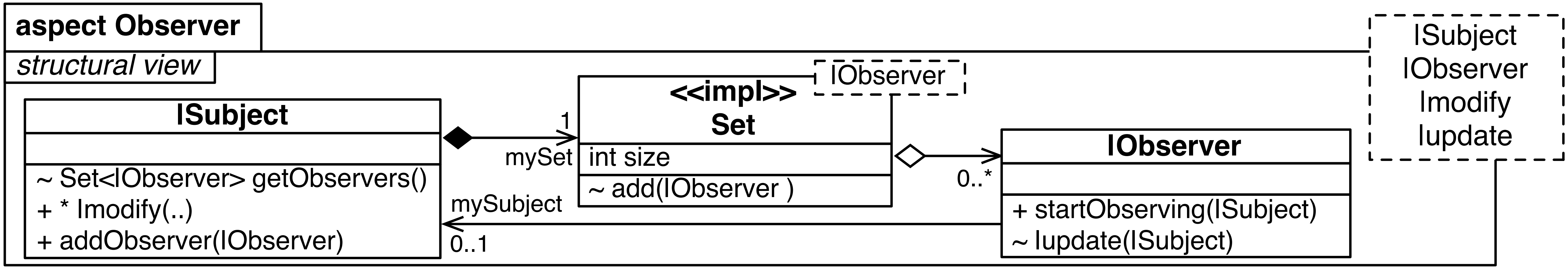
~ add(StockWindow)

0..*

0..1
mySubject

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)



aspect Observer

structural view

ISubject
IObserver
Imodify
Iupdate

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

1
mySet

mySubject
0..1

<<impl>>
Set

IObserver

int size

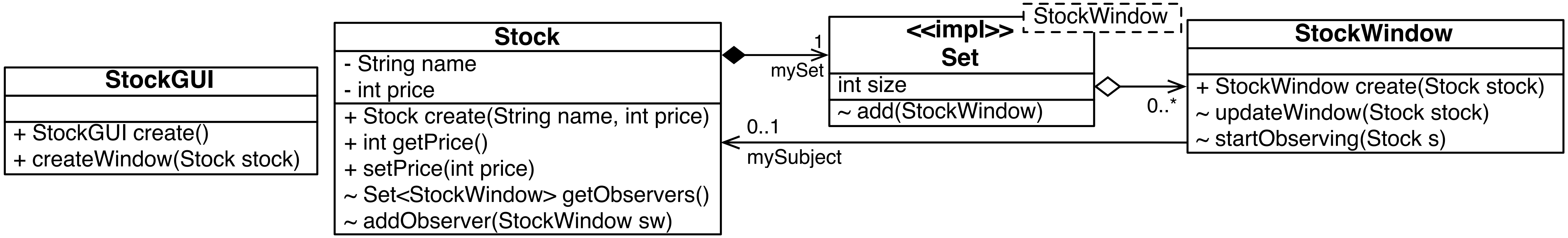
~ add(IObserver)

0..*

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)

1
mySet

StockWindow

<<impl>>
Set

int size

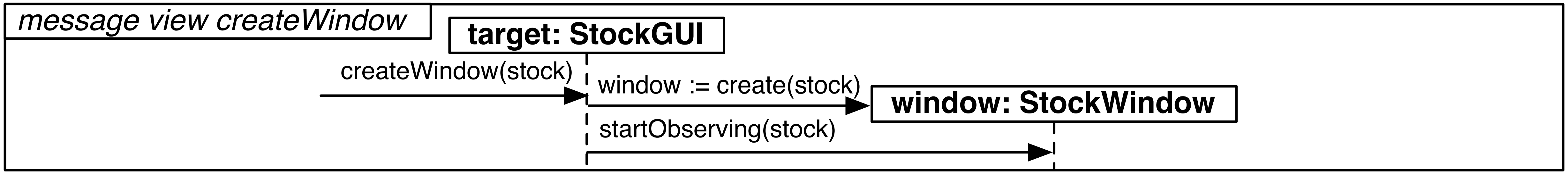
~ add(StockWindow)

0..*

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)

0..1
mySubject



message view createWindow

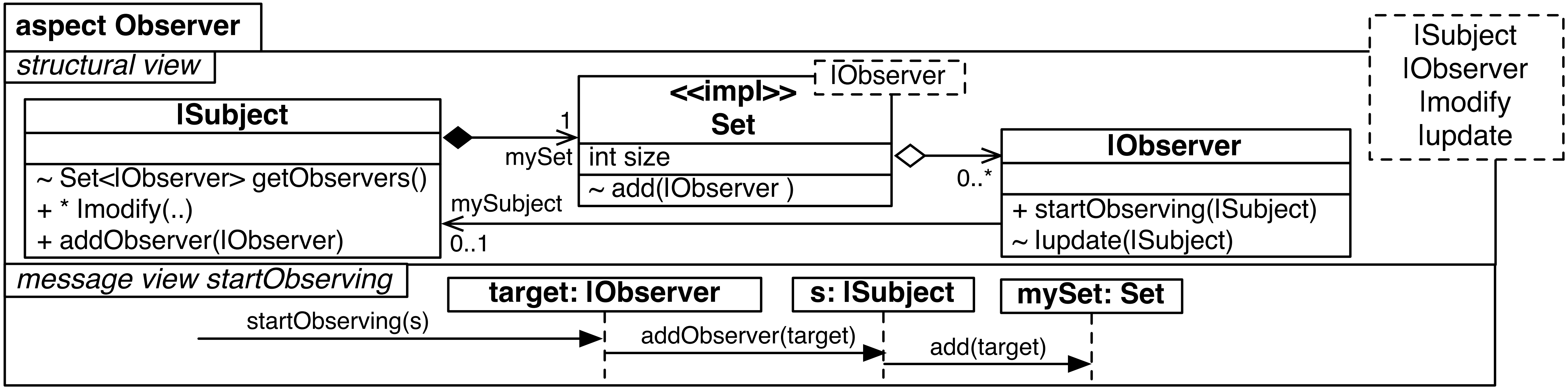
target: StockGUI

createWindow(stock)

window := create(stock)

window: StockWindow

startObserving(stock)



aspect Observer

structural view

ISubject
IObserver
Imodify
Iupdate

ISubject

~ Set<IObserver> getObservers()
+ * Imodify(..)
+ addObserver(IObserver)

1
mySet

IObserver

<<impl>>
Set

int size

~ add(IObserver)

0..*

IObserver

+ startObserving(ISubject)
~ Iupdate(ISubject)

mySubject
0..1

message view startObserving

target: IObserver

s: ISubject

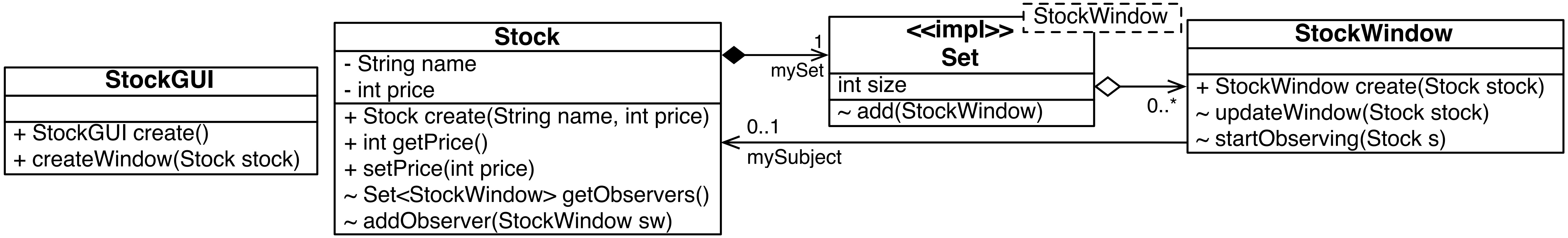
mySet: Set

startObserving(s)

addObserver(target)

add(target)

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)

1

mySet

<<impl>>
Set

StockWindow

int size

~ add(StockWindow)

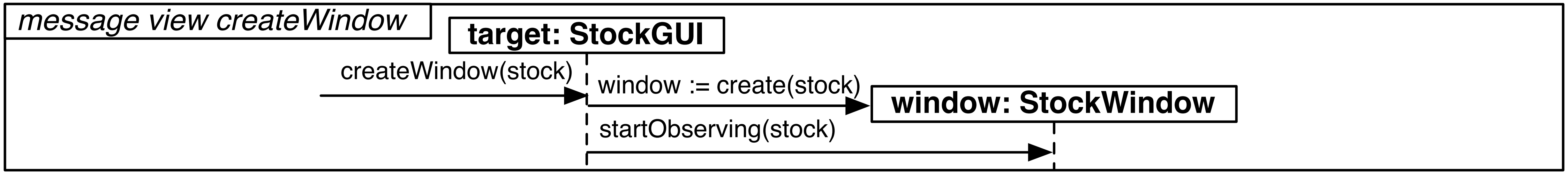
0..*

0..1

mySubject

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)



message view createWindow

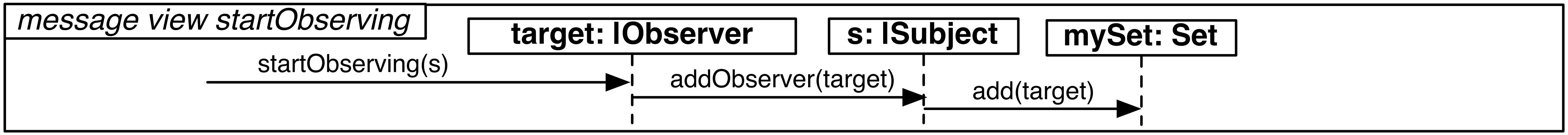
target: StockGUI

createWindow(stock)

window := create(stock)

window: StockWindow

startObserving(stock)



message view startObserving

target: IObserver

s: ISubject

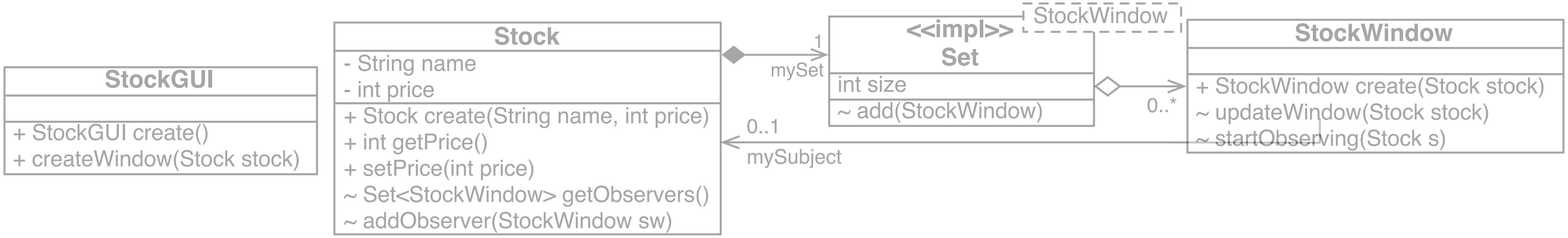
mySet: Set

startObserving(s)

addObserver(target)

add(target)

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)

StockWindow

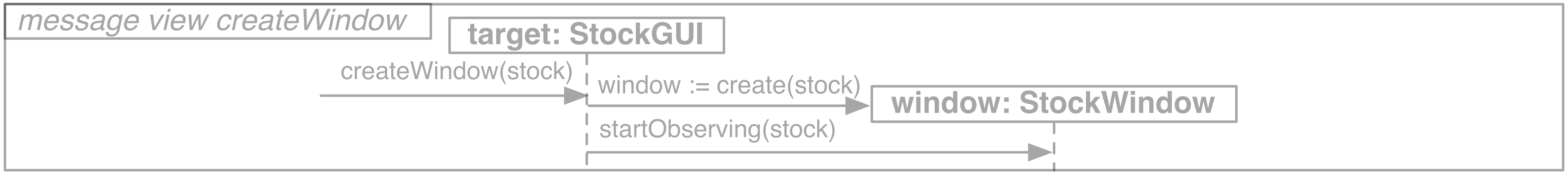
<<impl>>
Set

int size

~ add(StockWindow)

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)



message view createWindow

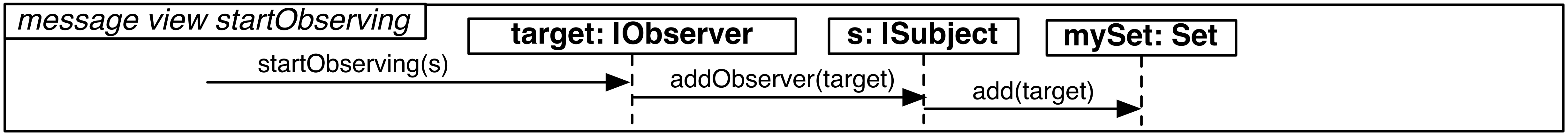
target: StockGUI

window: StockWindow

createWindow(stock)

window := create(stock)

startObserving(stock)



message view startObserving

target: IObserver

s: ISubject

mySet: Set

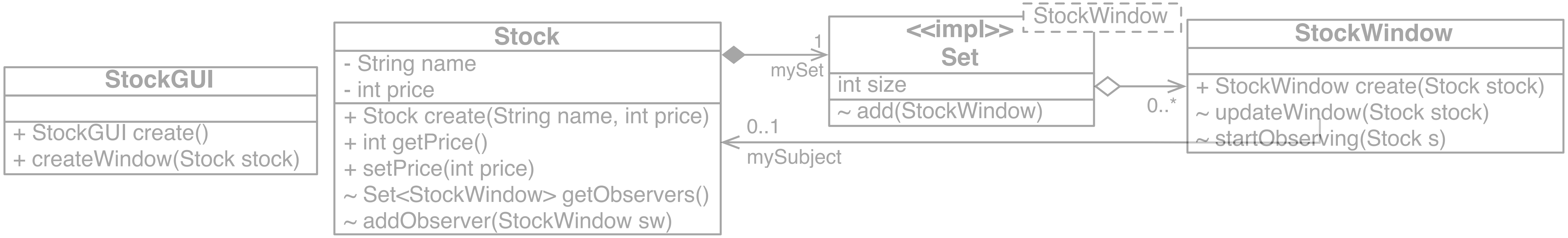
startObserving(s)

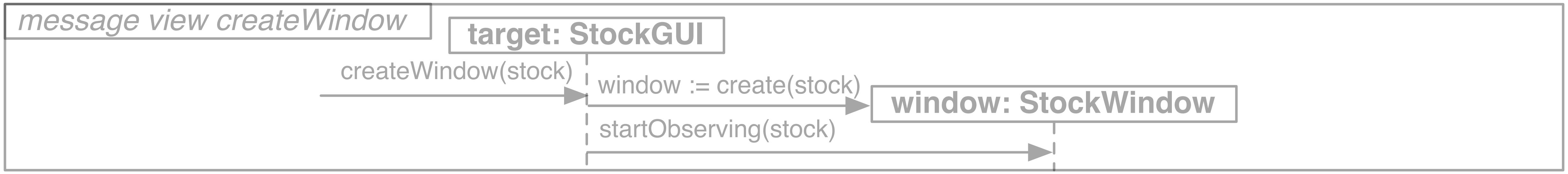
addObserver(target)

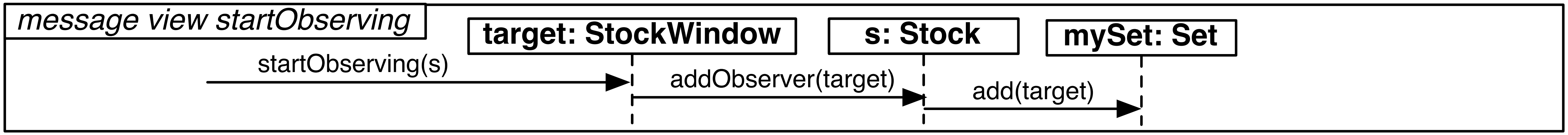
add(target)

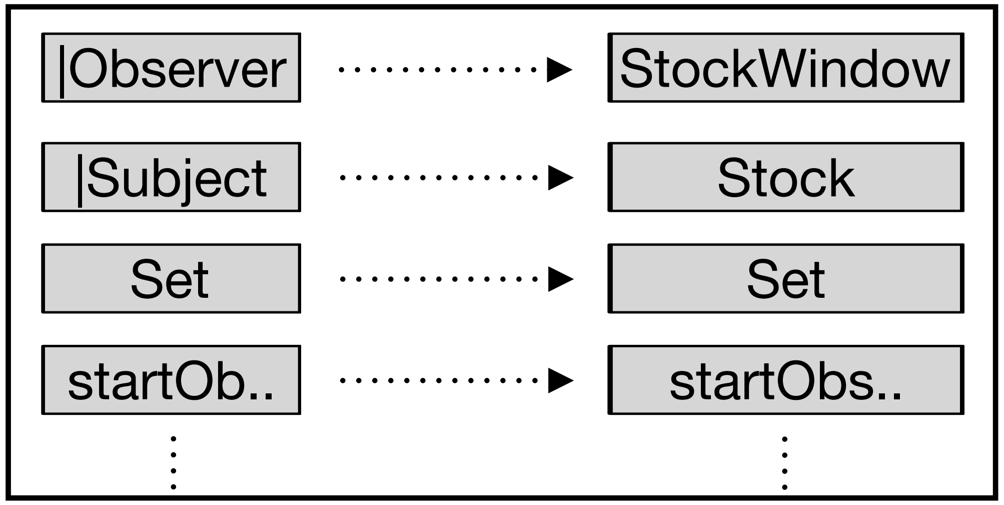
|Observer	StockWindow
|Subject	Stock
Set	Set
startOb..	startObs..
⋮	⋮

RAM: Weaving

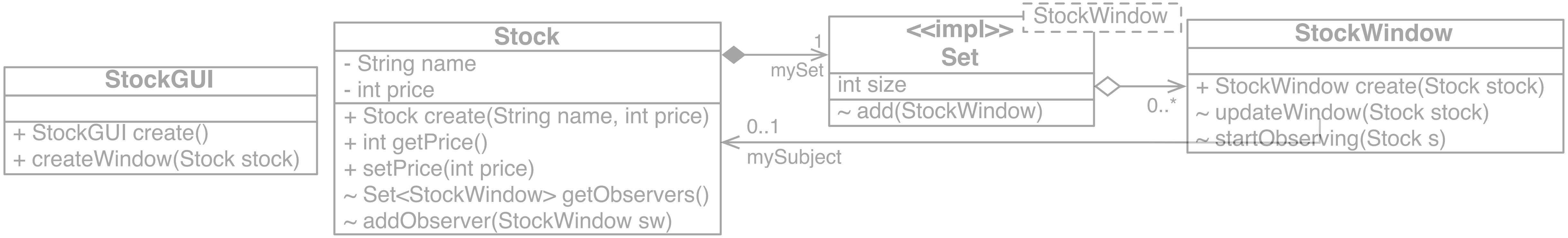

StockGUI
+ StockGUI create()
+ createWindow(Stock stock)
Stock
- String name
- int price
+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)
1
mySet
0..1
mySubject
StockWindow
<<impl>>
Set
int size
~ add(StockWindow)
0..*
StockWindow
+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)

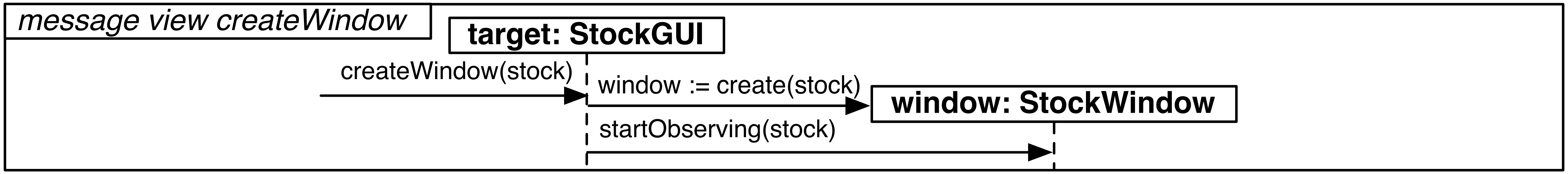

message view createWindow
target: StockGUI
createWindow(stock)
window := create(stock)
window: StockWindow
startObserving(stock)

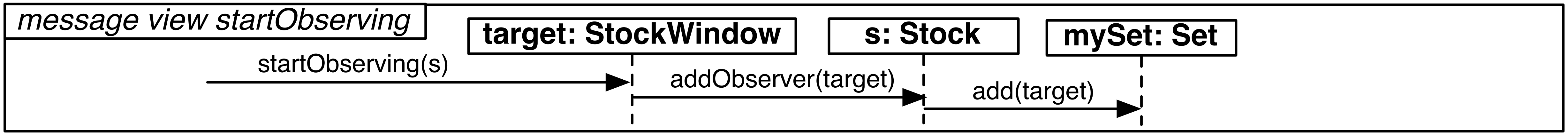

message view startObserving
target: StockWindow
s: Stock
mySet: Set
startObserving(s)
addObserver(target)
add(target)


|Observer
StockWindow
|Subject
Stock
Set
Set
startOb..
startObs..

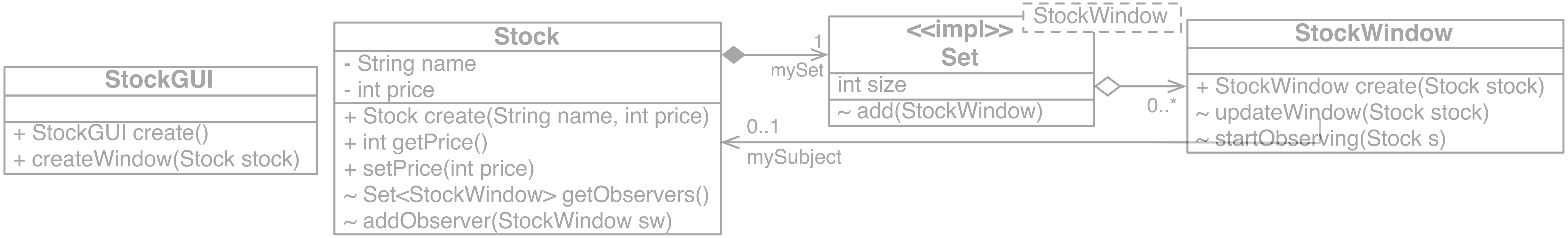
RAM: Weaving


StockGUI
+ StockGUI create()
+ createWindow(Stock stock)
Stock
- String name
- int price
+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)
1
mySet
<<impl>>
Set
StockWindow
int size
~ add(StockWindow)
0..*
0..1
mySubject
StockWindow
+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)


message view createWindow
target: StockGUI
createWindow(stock)
window := create(stock)
window: StockWindow
startObserving(stock)


message view startObserving
target: StockWindow
s: Stock
mySet: Set
startObserving(s)
addObserver(target)
add(target)

RAM: Weaving



StockGUI

+ StockGUI create()

+ createWindow(Stock stock)

Stock

- String name

- int price

+ Stock create(String name, int price)

+ int getPrice()

+ setPrice(int price)

~ Set<StockWindow> getObservers()

~ addObserver(StockWindow sw)

1

mySet

<<impl>>
Set

StockWindow

int size

~ add(StockWindow)

0..*

0..1

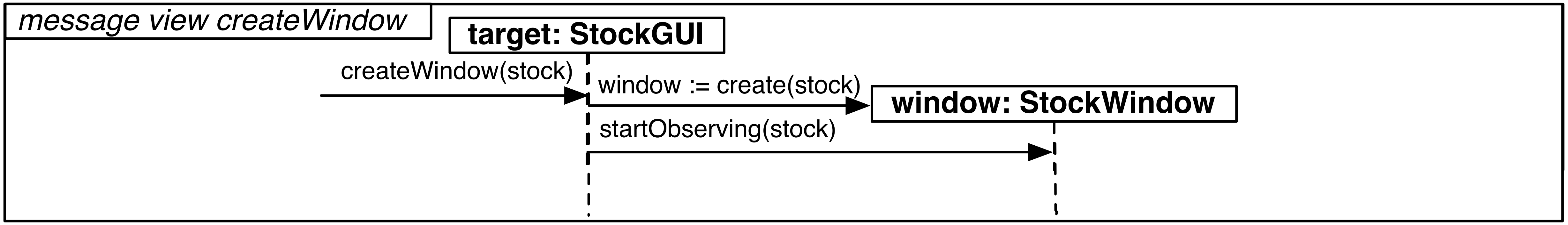
mySubject

StockWindow

+ StockWindow create(Stock stock)

~ updateWindow(Stock stock)

~ startObserving(Stock s)



message view createWindow

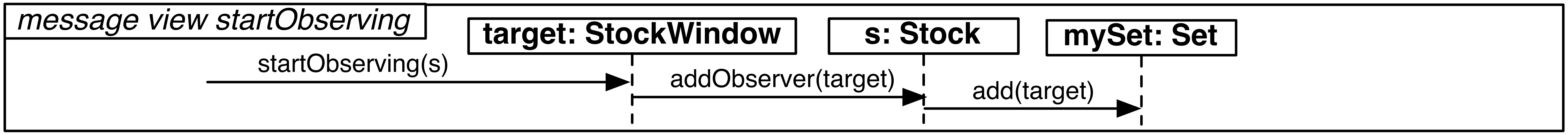
target: StockGUI

createWindow(stock)

window := create(stock)

window: StockWindow

startObserving(stock)



message view startObserving

target: StockWindow

s: Stock

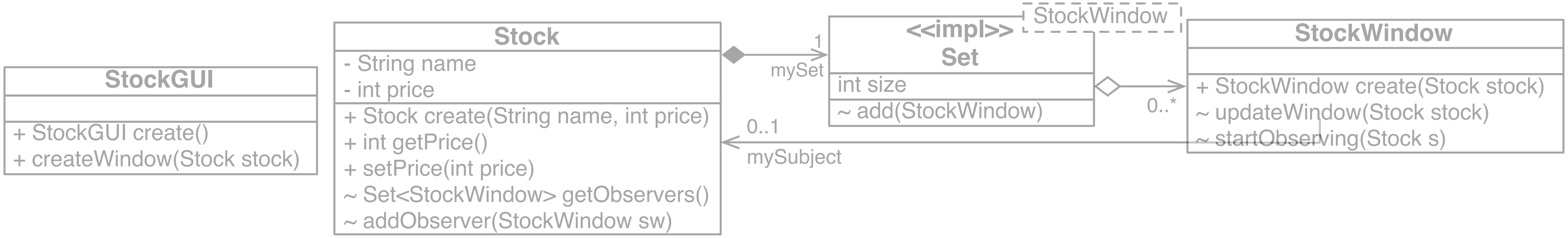
mySet: Set

startObserving(s)

addObserver(target)

add(target)

RAM: Weaving



StockGUI

+ StockGUI create()
+ createWindow(Stock stock)

Stock

- String name
- int price

+ Stock create(String name, int price)
+ int getPrice()
+ setPrice(int price)
~ Set<StockWindow> getObservers()
~ addObserver(StockWindow sw)

1
mySet

0..1
mySubject

<<impl>>
Set

StockWindow

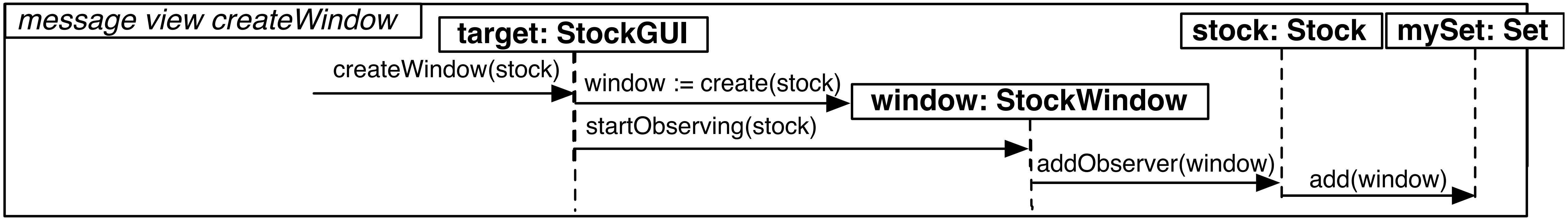
int size

~ add(StockWindow)

0..*

StockWindow

+ StockWindow create(Stock stock)
~ updateWindow(Stock stock)
~ startObserving(Stock s)



message view createWindow

target: StockGUI

stock: Stock

mySet: Set

createWindow(stock)

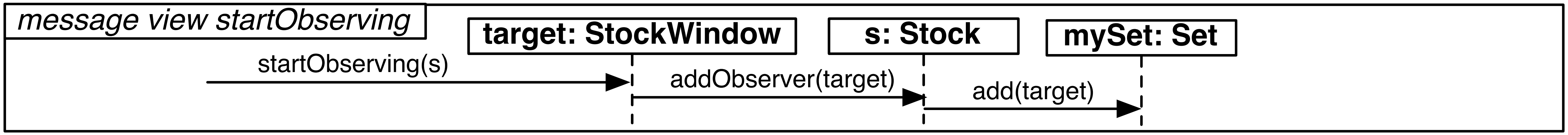
window := create(stock)

window: StockWindow

startObserving(stock)

addObserver(window)

add(window)



message view startObserving

target: StockWindow

s: Stock

mySet: Set

startObserving(s)

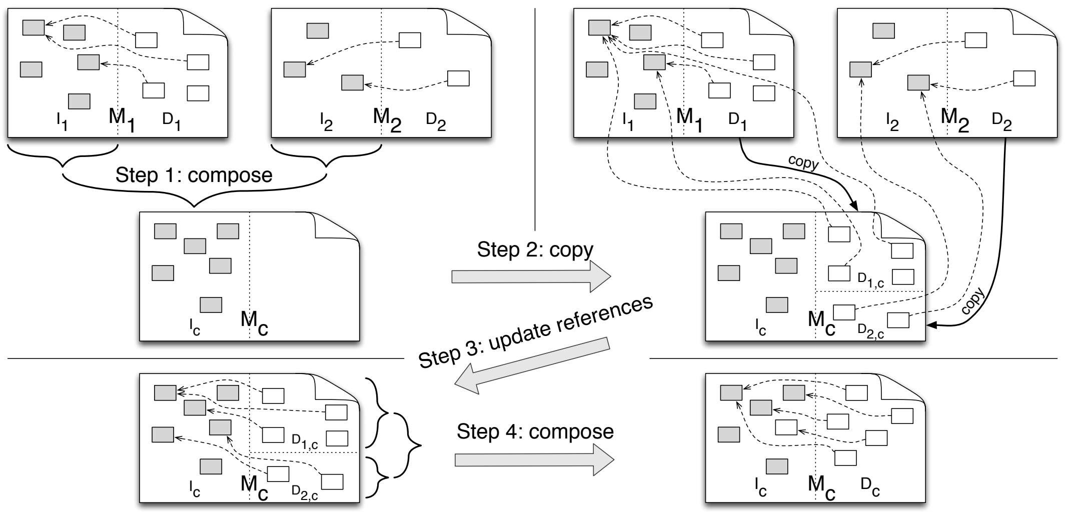
addObserver(target)

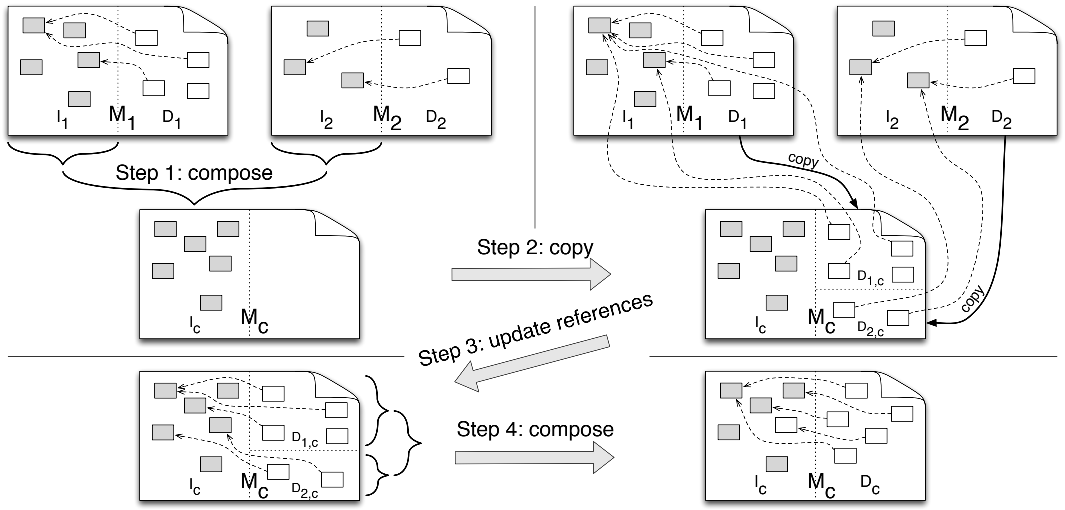
add(target)

Challenges

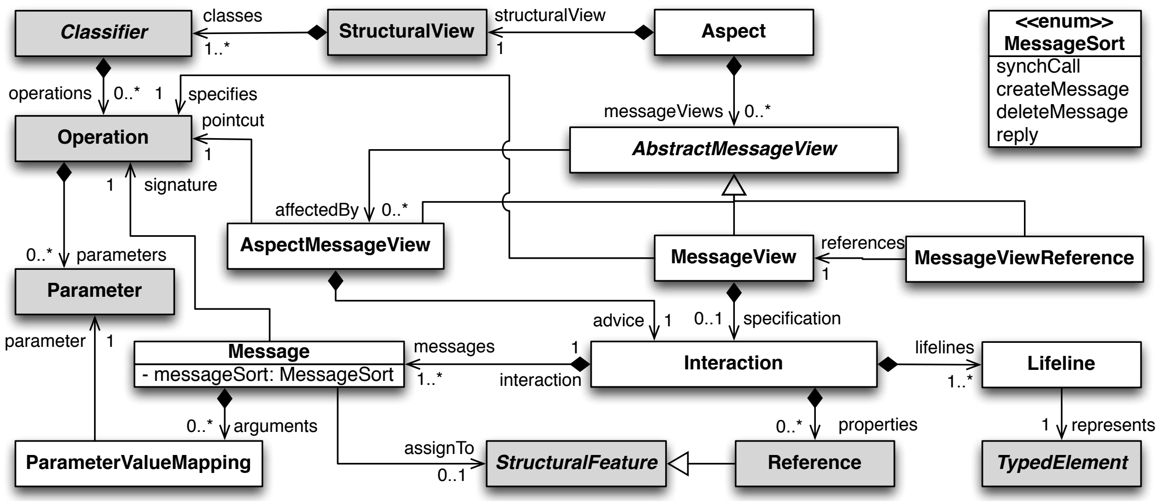
- Complexity of integration depending on shared concepts
    - MM_I might have to be modified
- High dependencies
- Testing very important

Demo

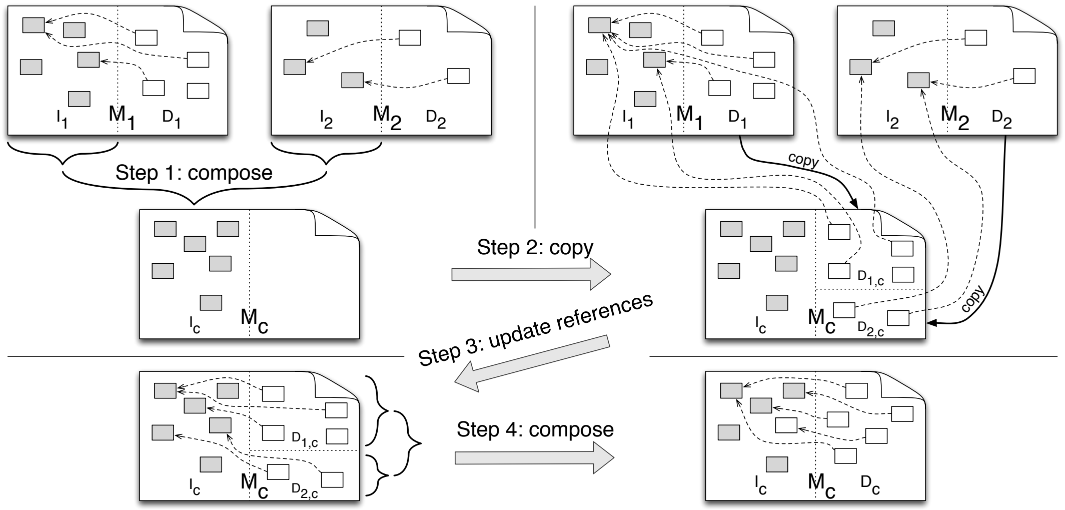

I_1 M_1 D_1
I_2 M_2 D_2
Step 1: compose
I_c M_c
Step 2: copy
copy
$D_{1,c}$
$D_{2,c}$
copy
Step 3: update references
Step 4: compose
D_c



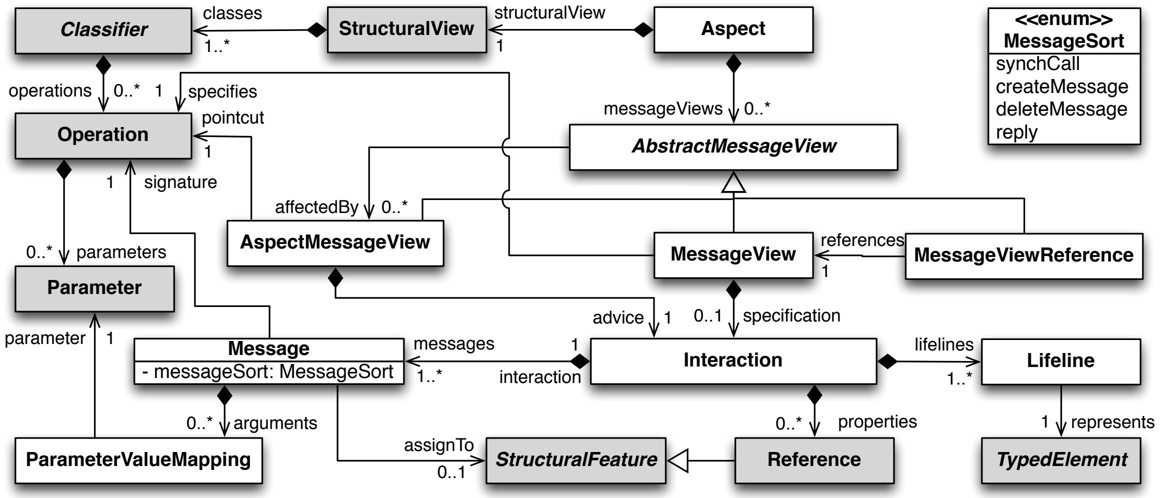
I1
M1
D1
I2
M2
D2
Step 1: compose
Ic
Mc
Step 2: copy
copy
D1,c
D2,c
copy
Step 3: update references
Step 4: compose
Dc



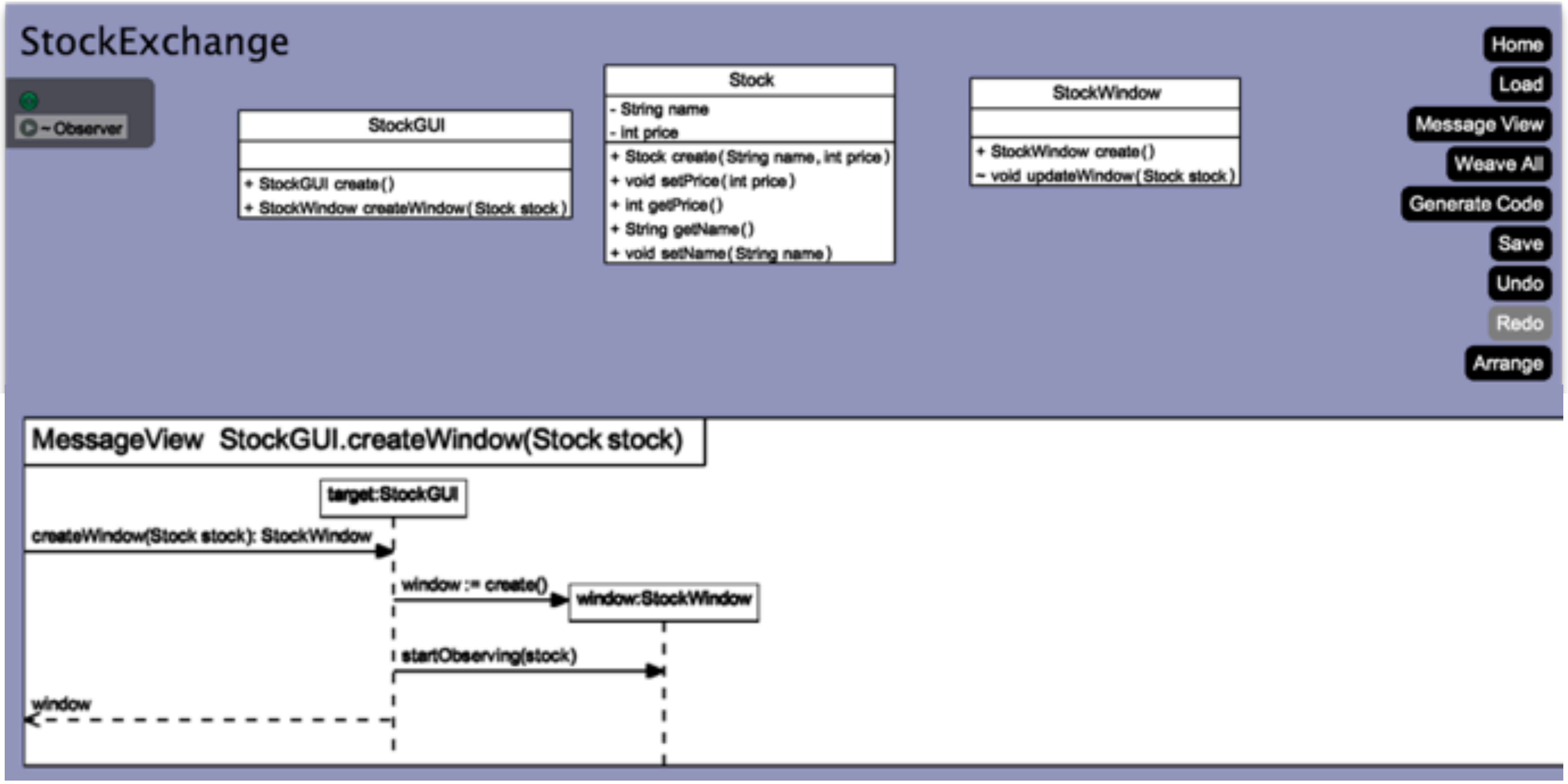
Classifier
classes
1..*
StructuralView
structuralView
1
Aspect
<<enum>>
MessageSort
synchCall
createMessage
deleteMessage
reply
operations
0..*
1
specifies
pointcut
1
Operation
messageViews
0..*
AbstractMessageView
1
signature
affectedBy
0..*
AspectMessageView
MessageView
references
1
MessageViewReference
0..*
parameters
Parameter
advice
1
0..1
specification
parameter
1
Message
- messageSort: MessageSort
messages
1
1..*
interaction
Interaction
lifelines
1..*
Lifeline
0..*
arguments
0..*
properties
1
represents
ParameterValueMapping
assignTo
0..1
StructuralFeature
Reference
TypedElement



I1 M1 D1
I2 M2 D2
Step 1: compose
Ic Mc
Step 2: copy
copy
copy
D1,c
D2,c
Step 3: update references
Step 4: compose
Dc

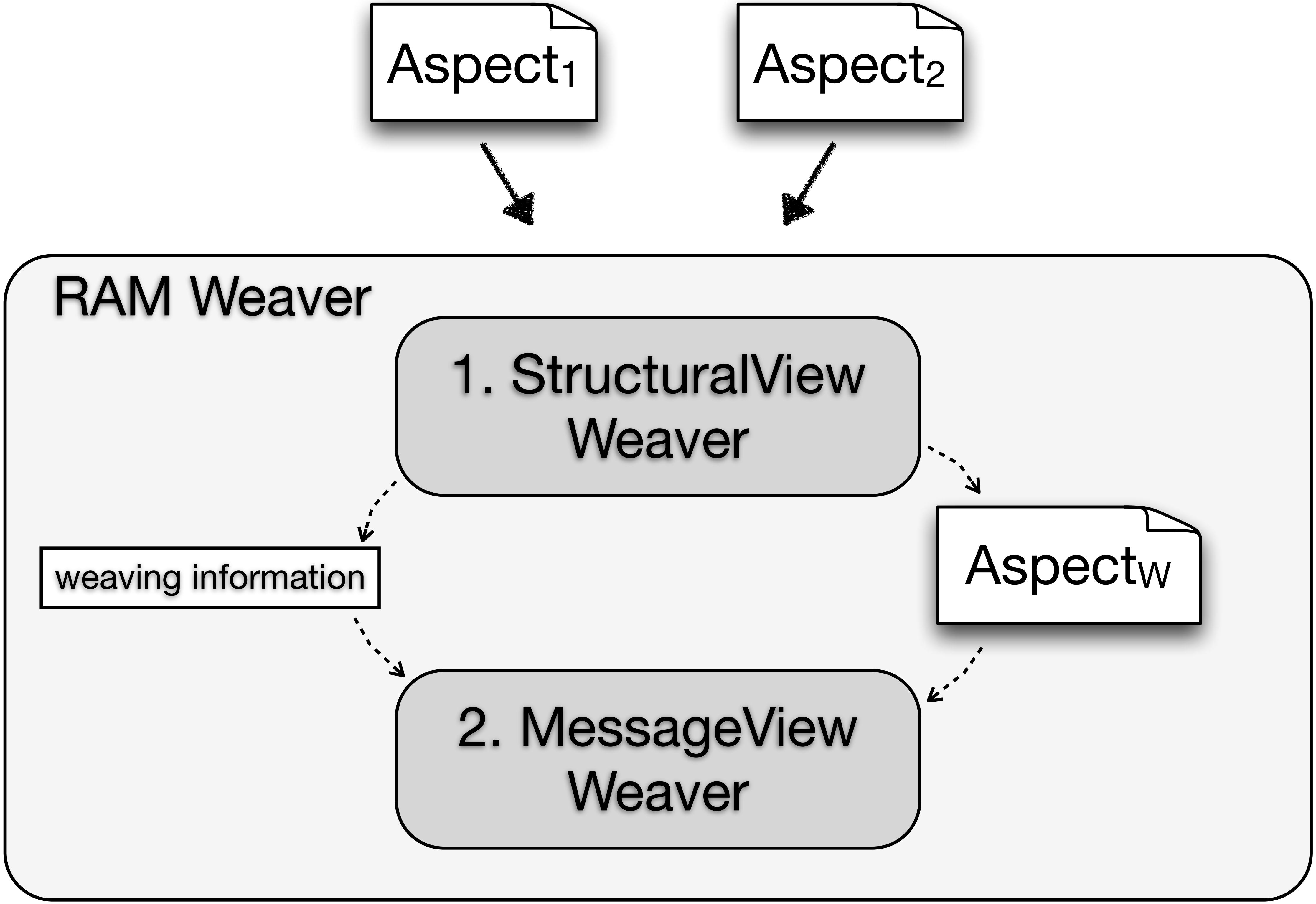


Classifier
classes
1..*
StructuralView
structuralView
1
Aspect
<<enum>>
MessageSort
synchCall
createMessage
deleteMessage
reply
operations
0..*
1
specifies
pointcut
1
Operation
messageViews
0..*
AbstractMessageView
1
signature
affectedBy
0..*
AspectMessageView
MessageView
references
1
MessageViewReference
0..*
parameters
Parameter
advice
1
0..1
specification
parameter
1
Message
- messageSort: MessageSort
messages
1
1..*
interaction
Interaction
lifelines
1..*
Lifeline
0..*
arguments
ParameterValueMapping
assignTo
0..1
StructuralFeature
0..*
properties
Reference
1
represents
TypedElement



StockExchange
~ Observer
StockGUI
+ StockGUI create()
+ StockWindow createWindow(Stock stock)
Stock
- String name
- int price
+ Stock create(String name, int price)
+ void setPrice(int price)
+ int getPrice()
+ String getName()
+ void setName(String name)
StockWindow
+ StockWindow create()
~ void updateWindow(Stock stock)
Home
Load
Message View
Weave All
Generate Code
Save
Undo
Redo
Arrange
MessageView StockGUI.createWindow(Stock stock)
target:StockGUI
createWindow(Stock stock): StockWindow
window := create()
window:StockWindow
startObserving(stock)
window

RAM: Weaving



Aspect$_1$
Aspect$_2$
RAM Weaver
1. StructuralView Weaver
weaving information
Aspect$_W$
2. MessageView Weaver