A practical approach to multi-modeling views composition

Andres Yie, Rubby Casallas, Dirk Deridder, Dennis Wagelaar

Universidad de los Andes - Vrije Universiteit Brussel

Universidad de los Andes

CSw

Grupo de Investigación en Construcción de Software

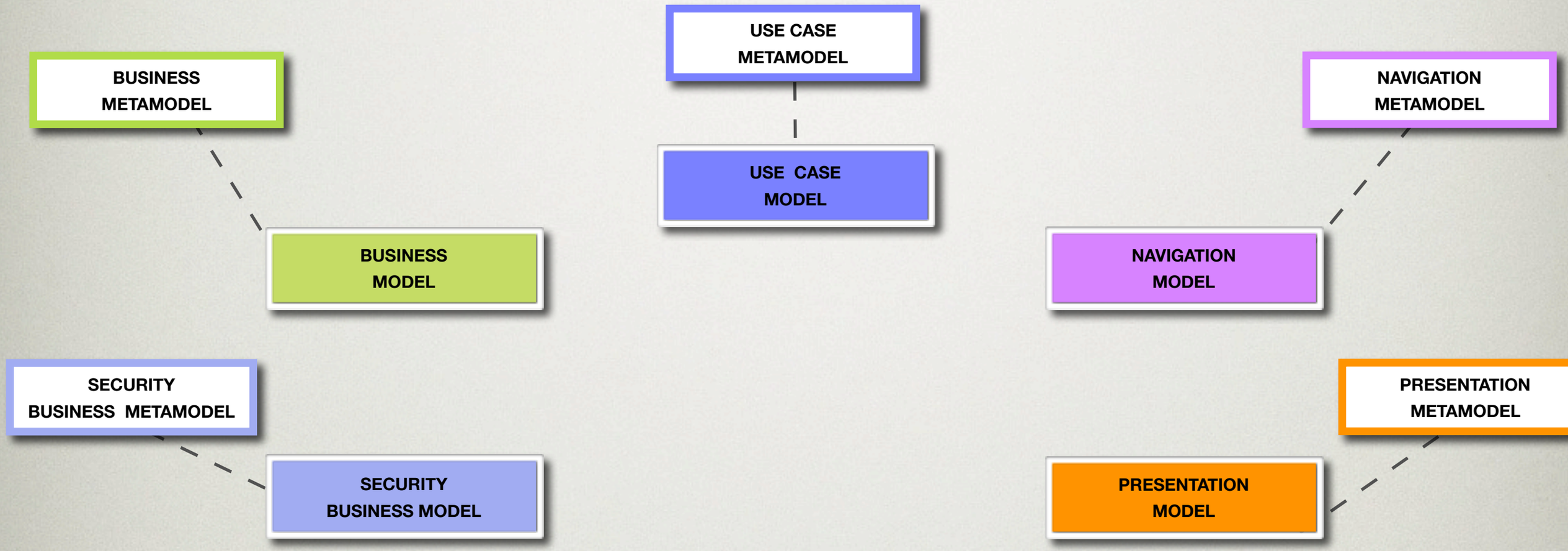
Vrije Universiteit Brussel

System and Software Engineering & Programming Technology Labs

Agenda

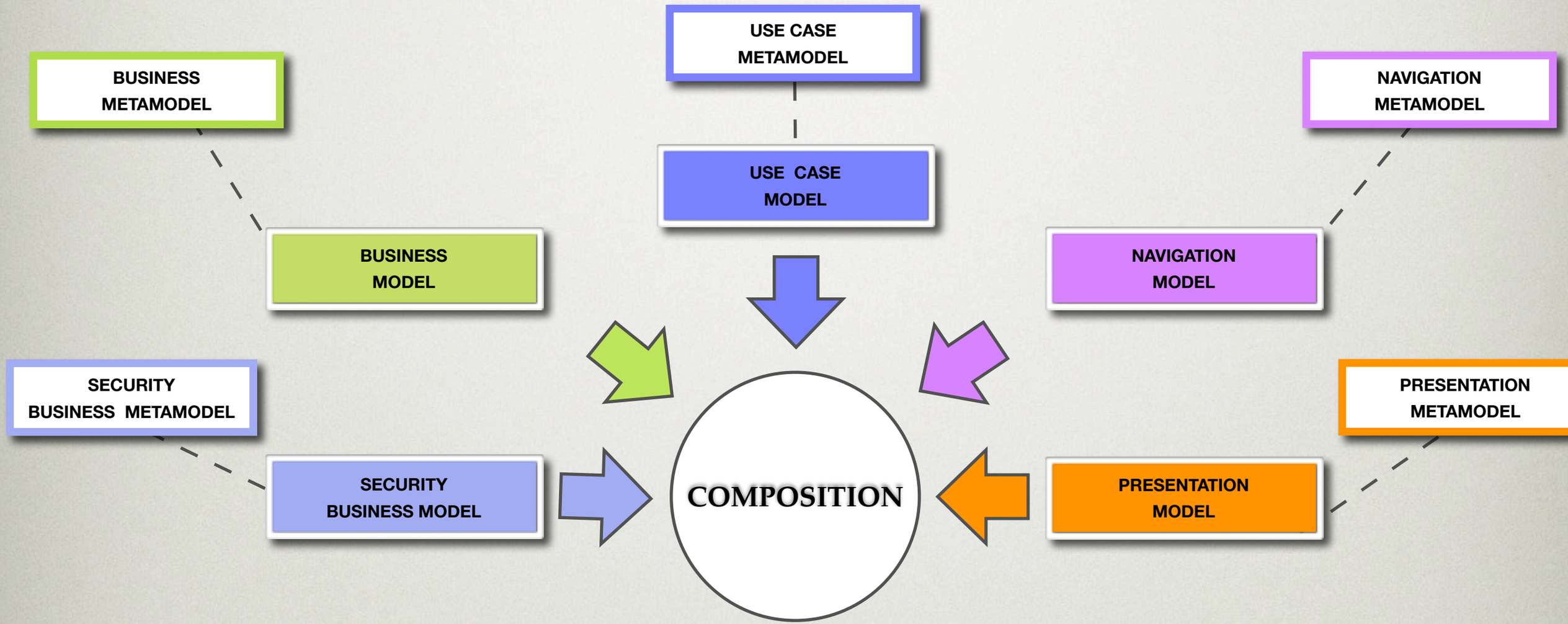
- Views & DSLs
- Heterogeneous composition
- Homogeneous composition
- Correspondence Model
- Derivation Model
- Composition
- Summary & Future work

Views and DSLs



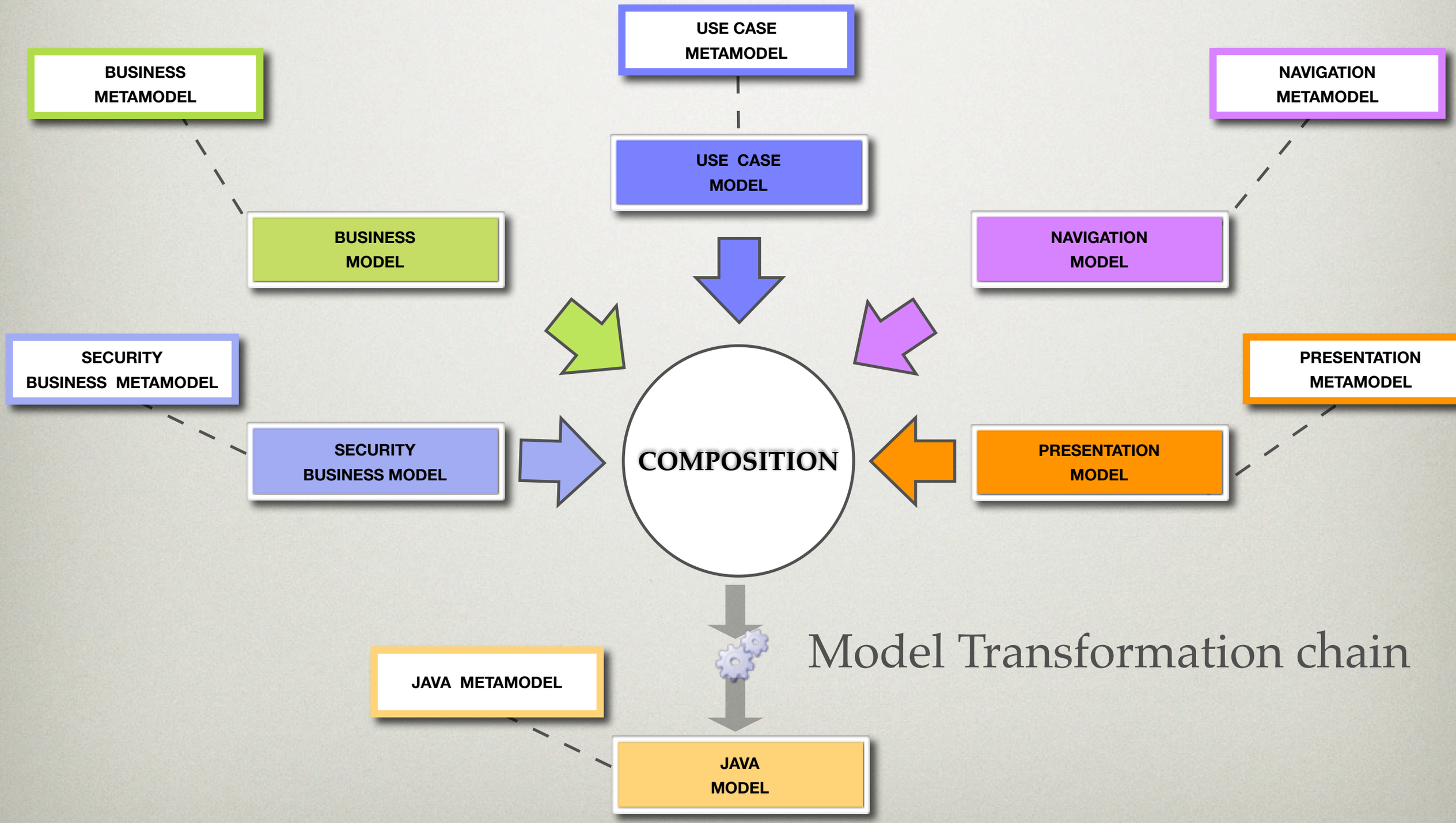
Model Transformation chain

Views and DSLs

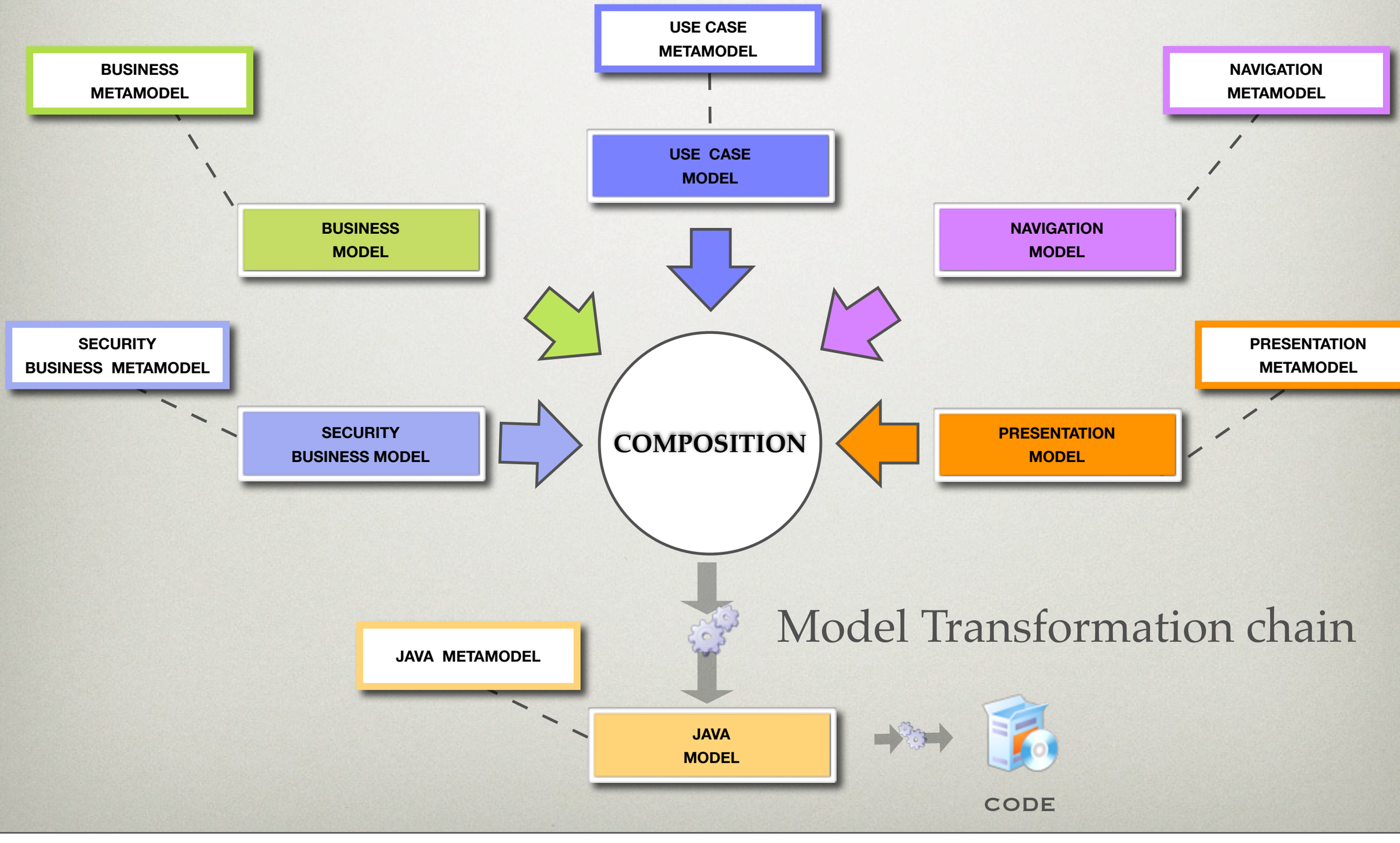


Model Transformation chain

Views and DSLs

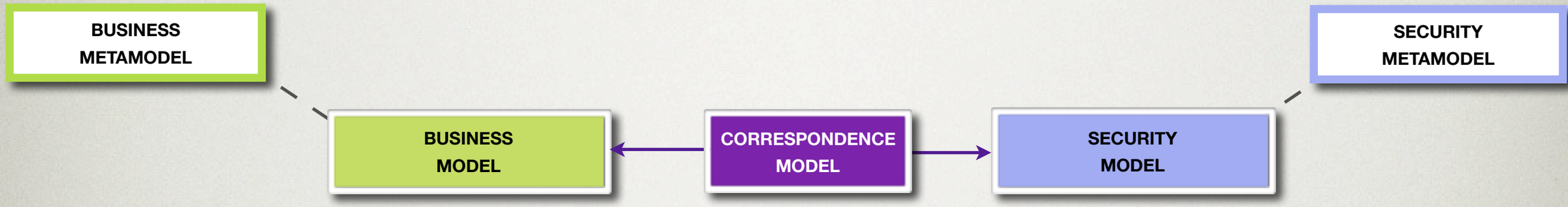


Views and DSLs



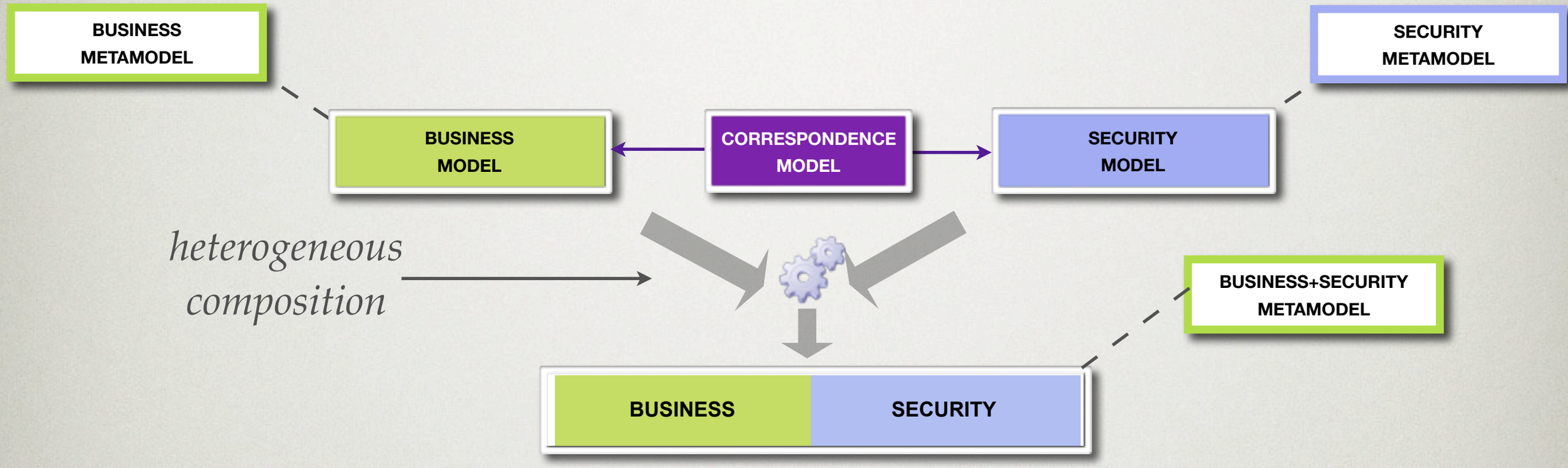
Model Transformation chain

Heterogeneous composition



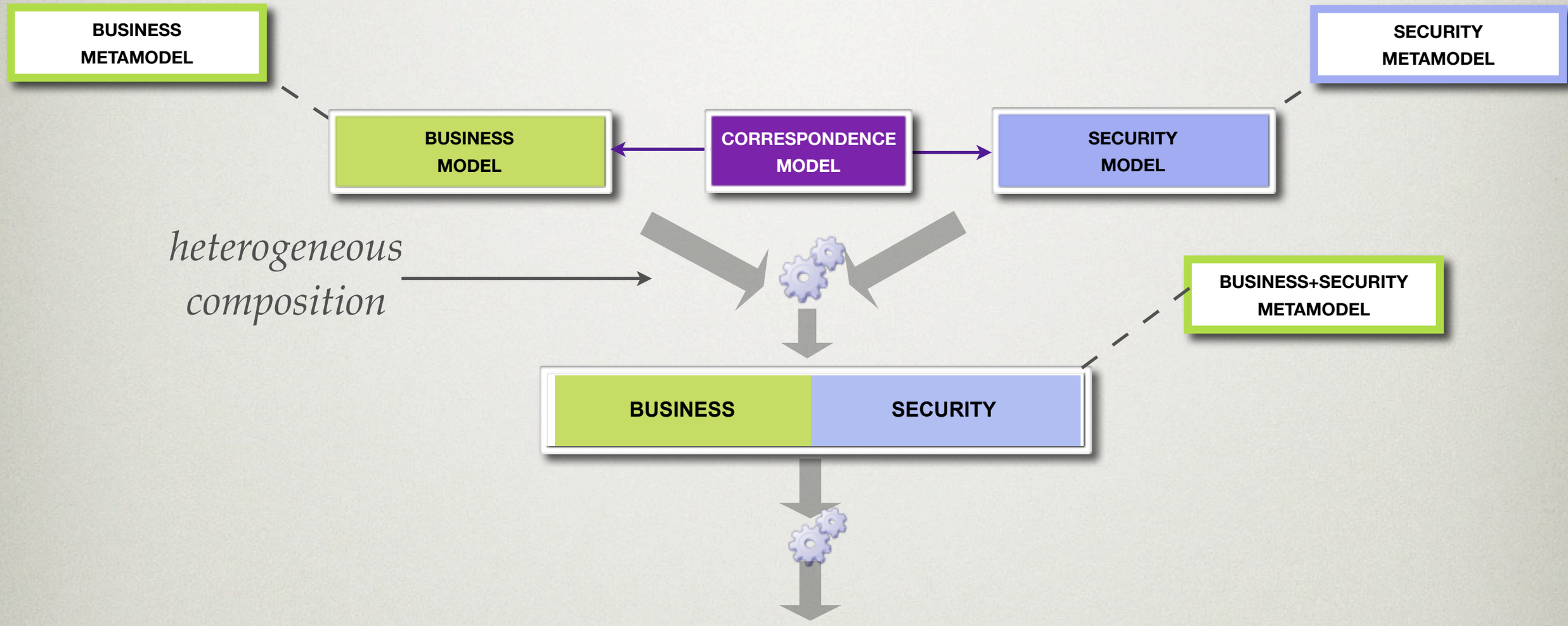
[Bezivin et al] A canonical scheme for model composition. ECMDA-FA (2006)

Heterogeneous composition



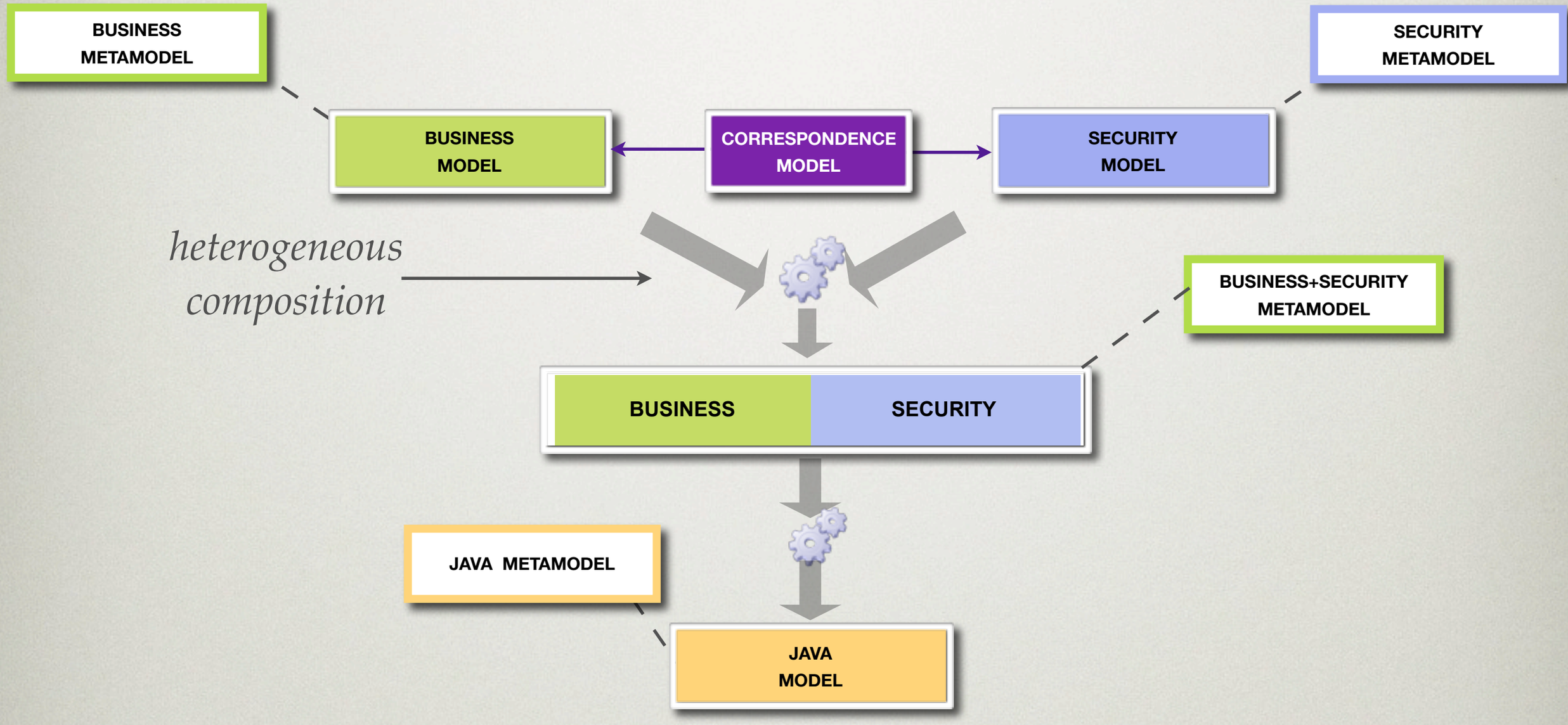
[Bezivin et al] A canonical scheme for model composition. ECMDA-FA (2006)

Heterogeneous composition



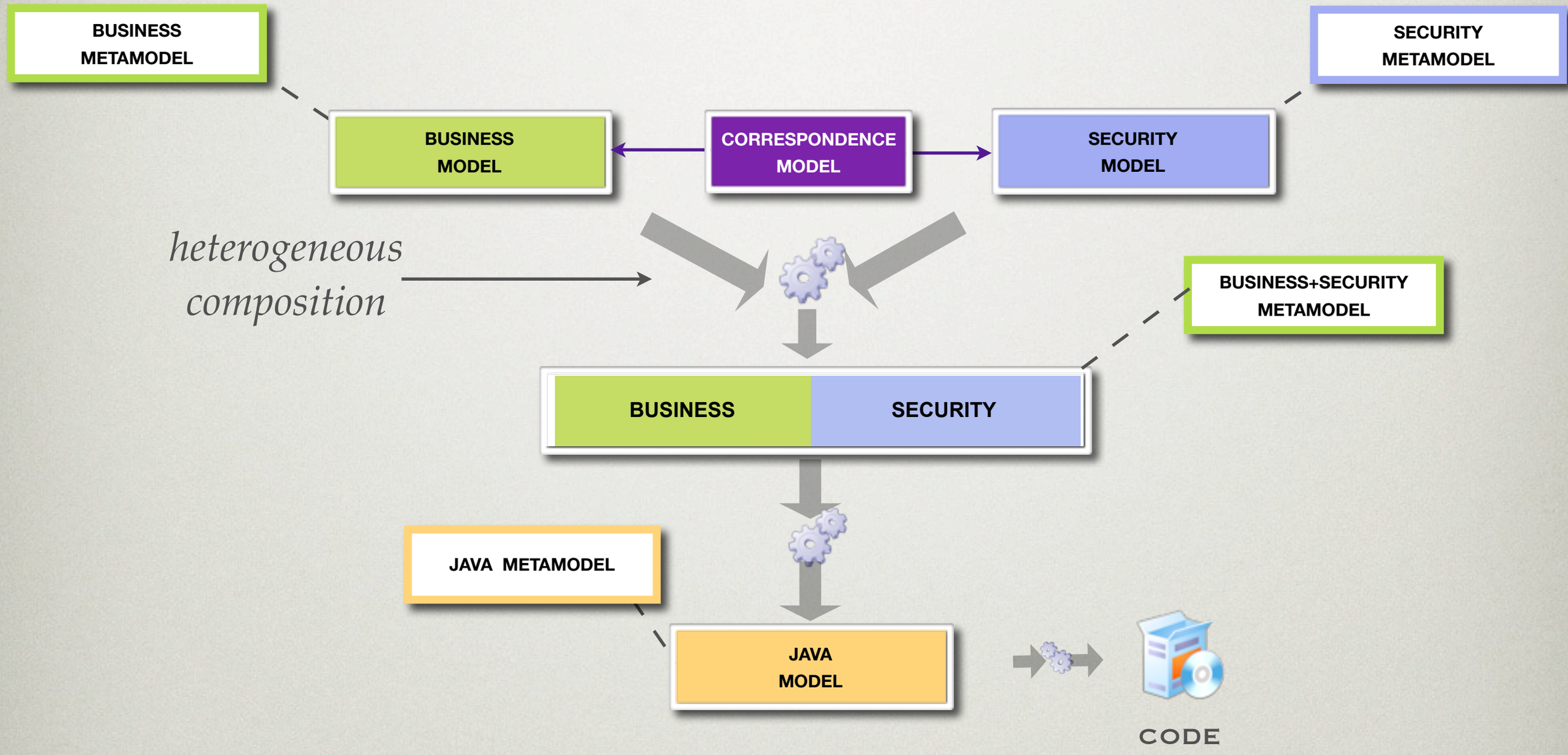
[Bezivin et al] A canonical scheme for model composition. ECMDA-FA (2006)

Heterogeneous composition



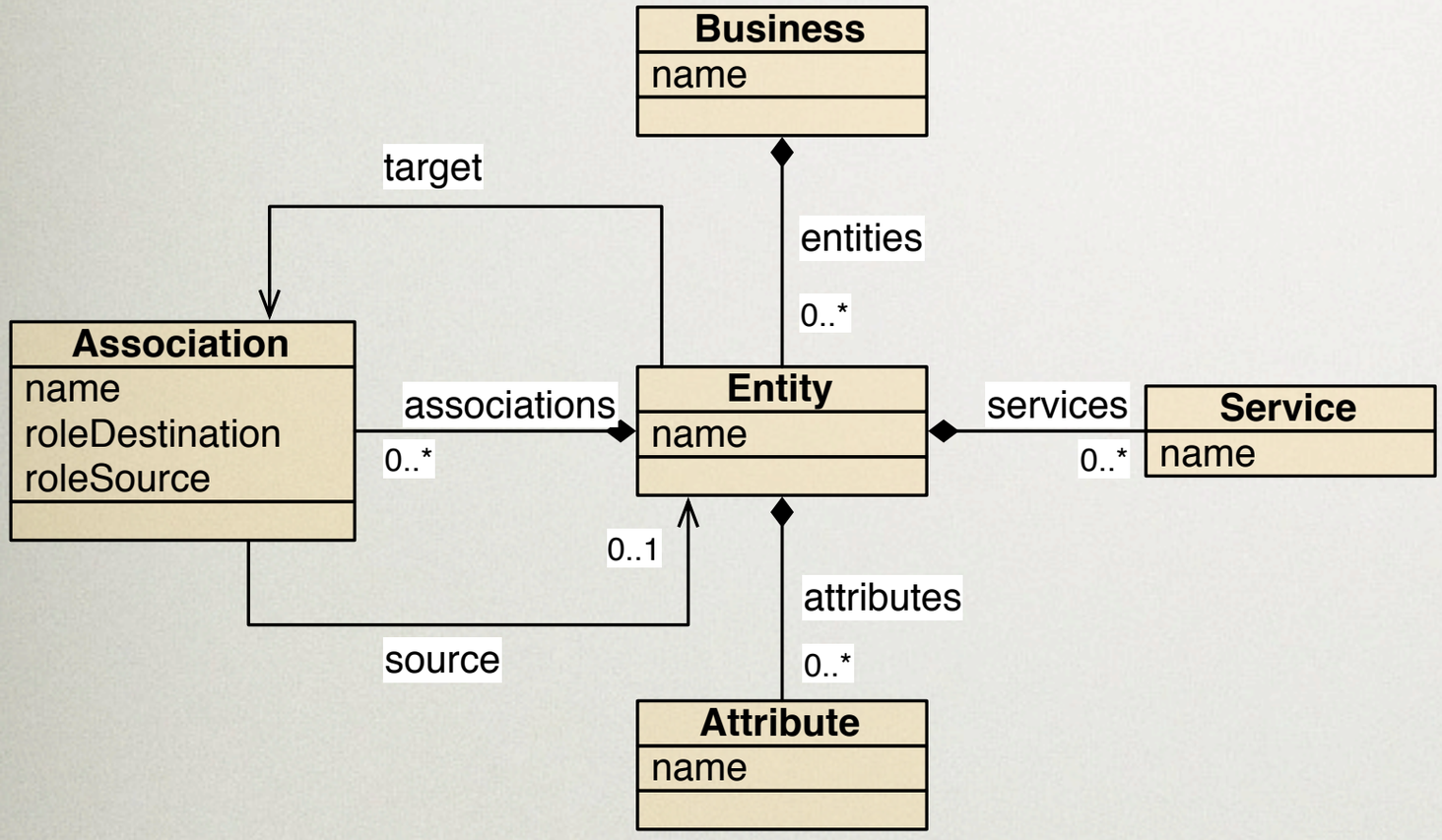
[Bezivin et al] A canonical scheme for model composition. ECMDA-FA (2006)

Heterogeneous composition

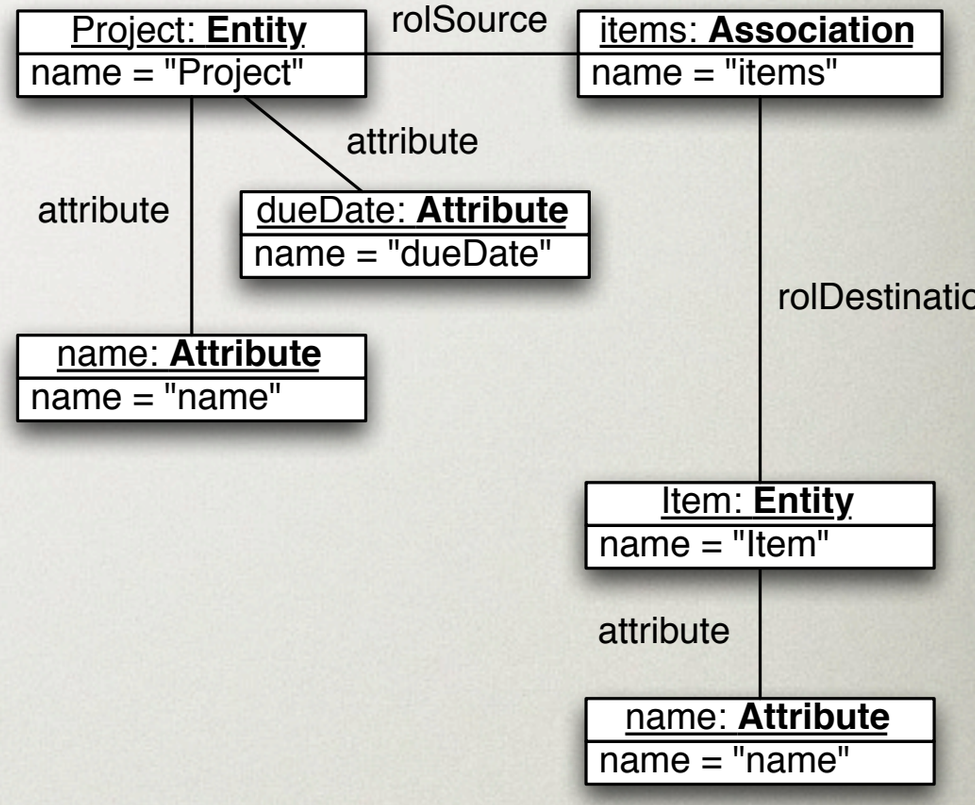


[Bezivin et al] A canonical scheme for model composition. ECMDA-FA (2006)

Business metamodel & model

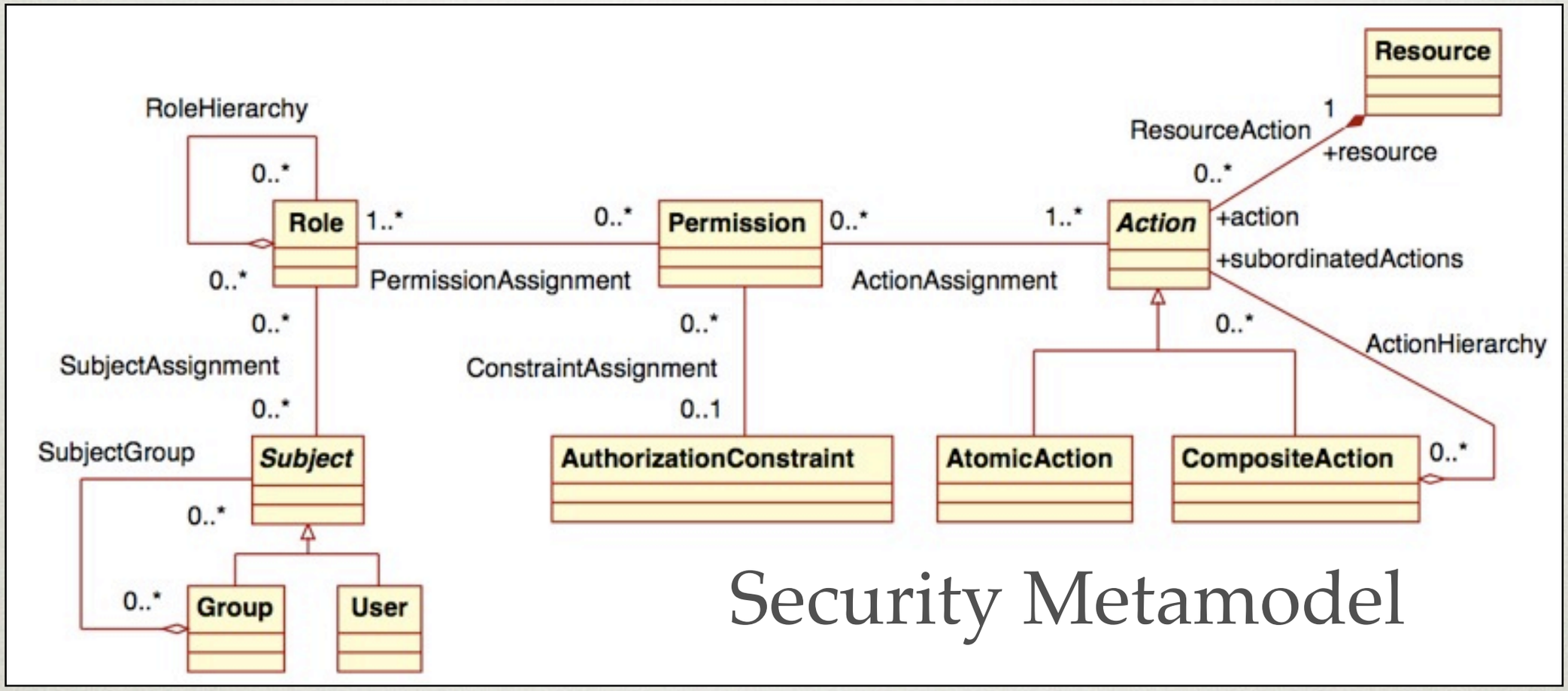


Business Metamodel

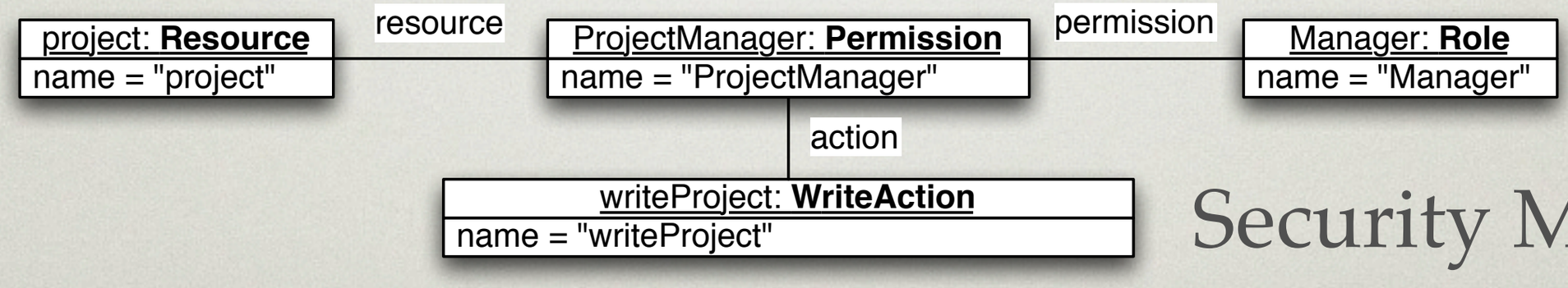


Business Model

Security Metamodel & Model



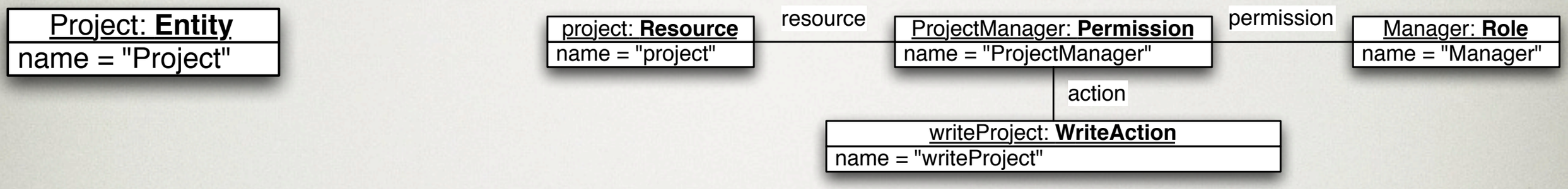
Security Metamodel



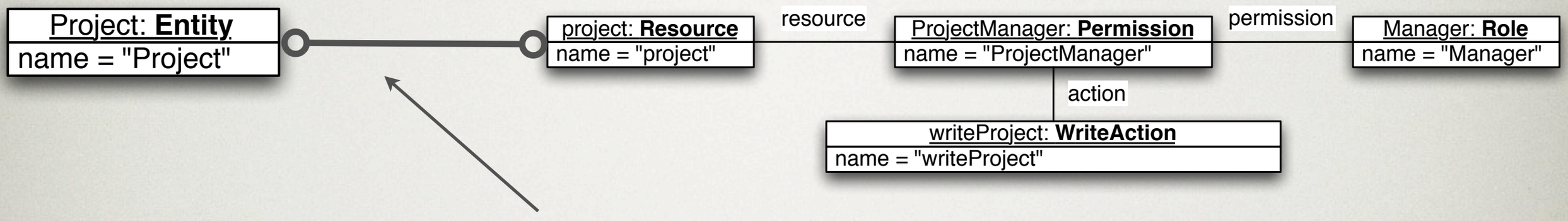
Security Model

[Lodderstedt et al] SecureUML: A UML-Based Modeling Language for Model-Driven Security. UML (2002) vol. 2460 pp. 426–441

Heterogeneous composition

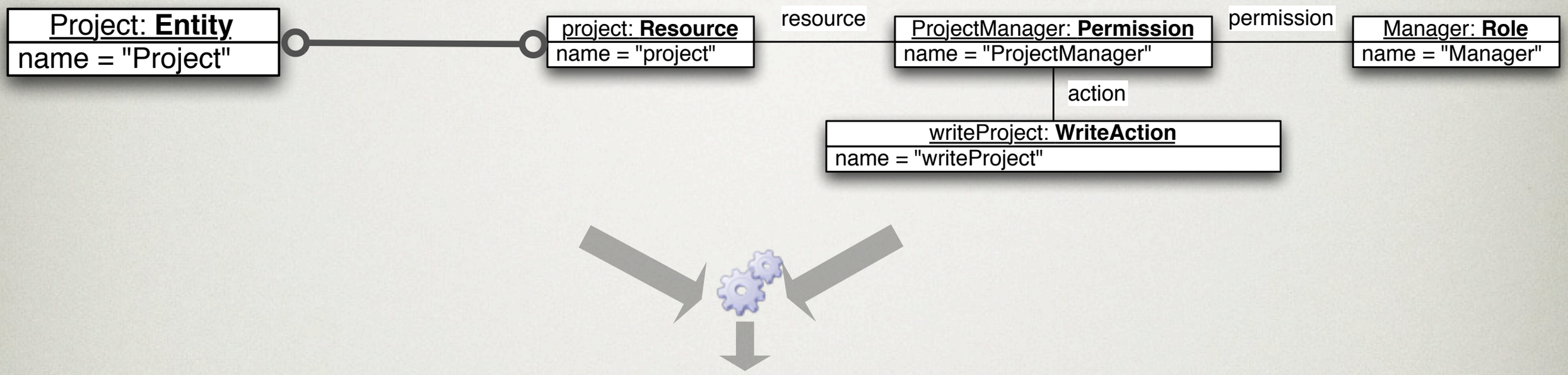


Heterogeneous composition

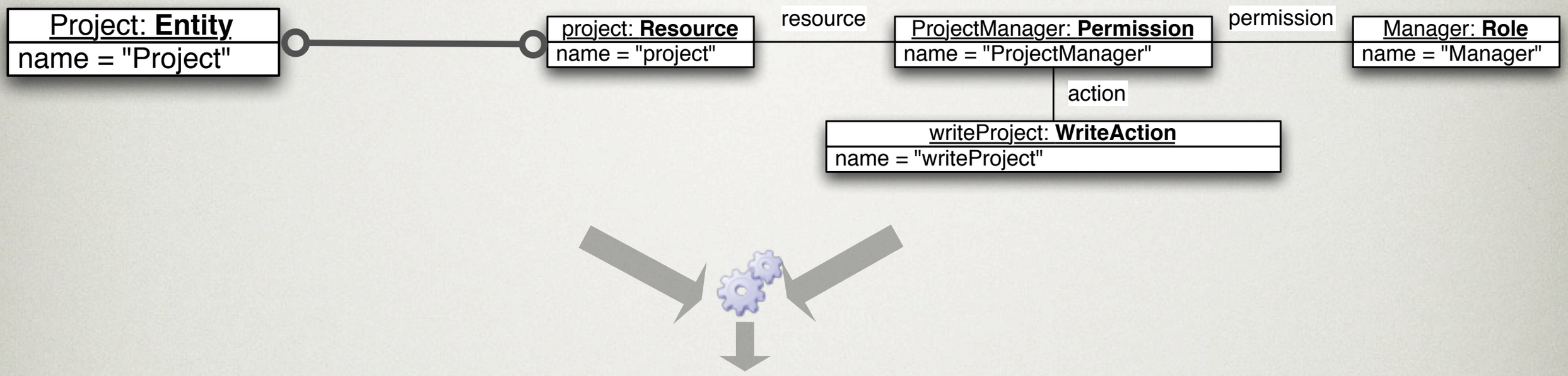


correspondence relationship

Heterogeneous composition



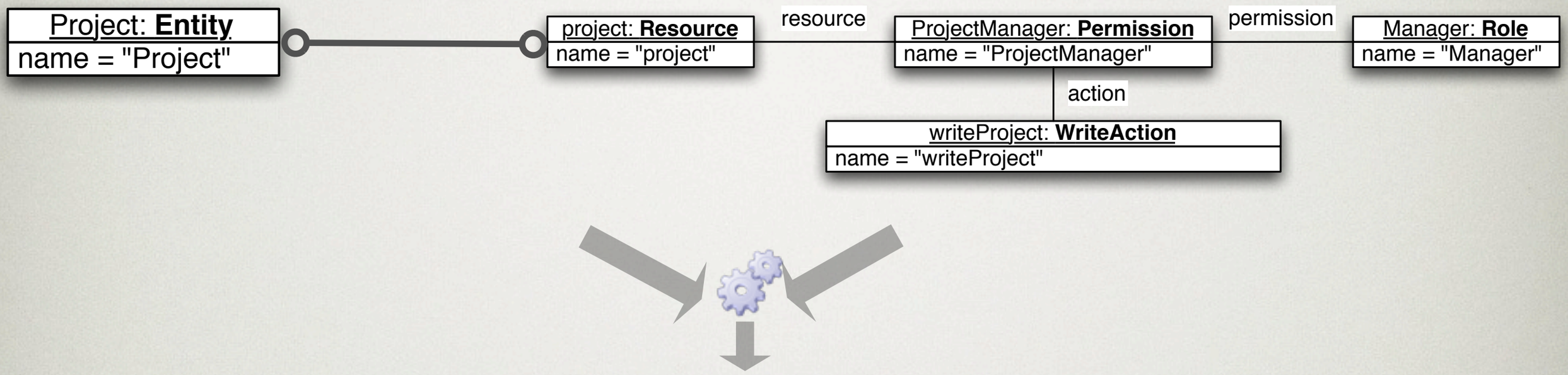
Heterogeneous composition

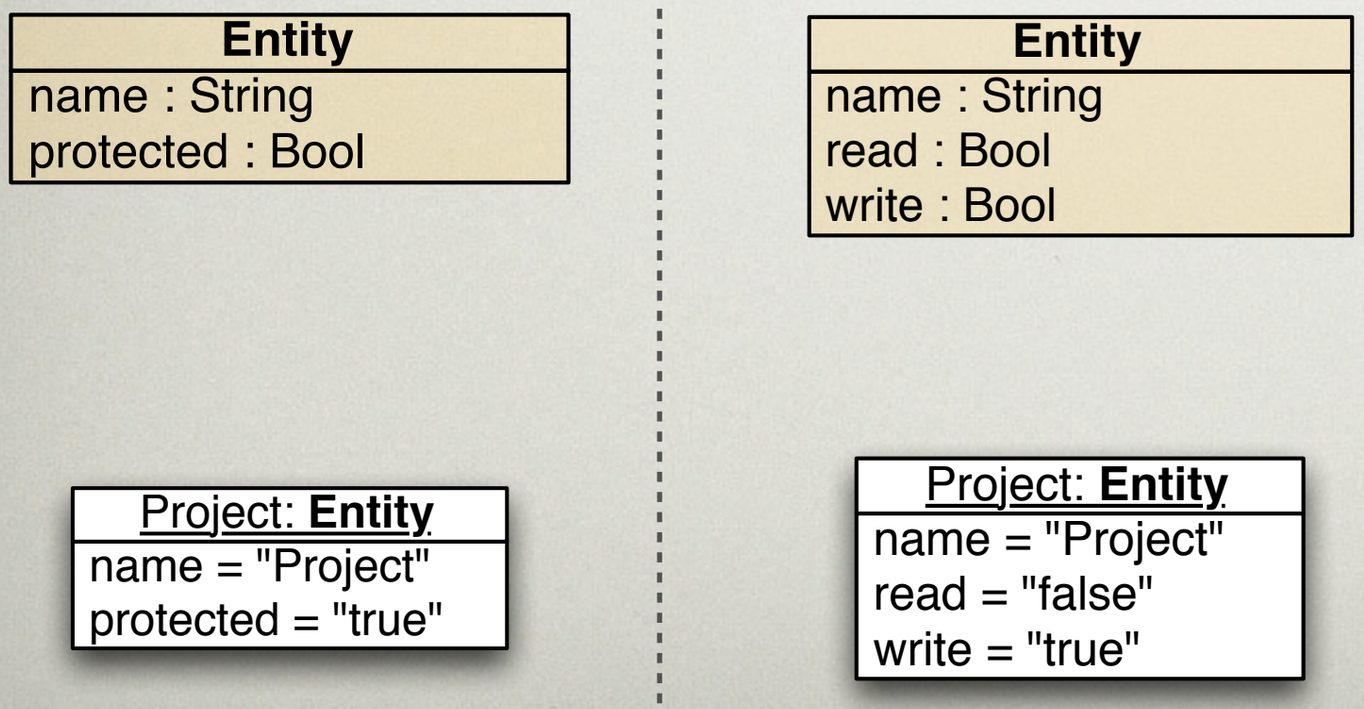


Entity
name : String protected : Bool

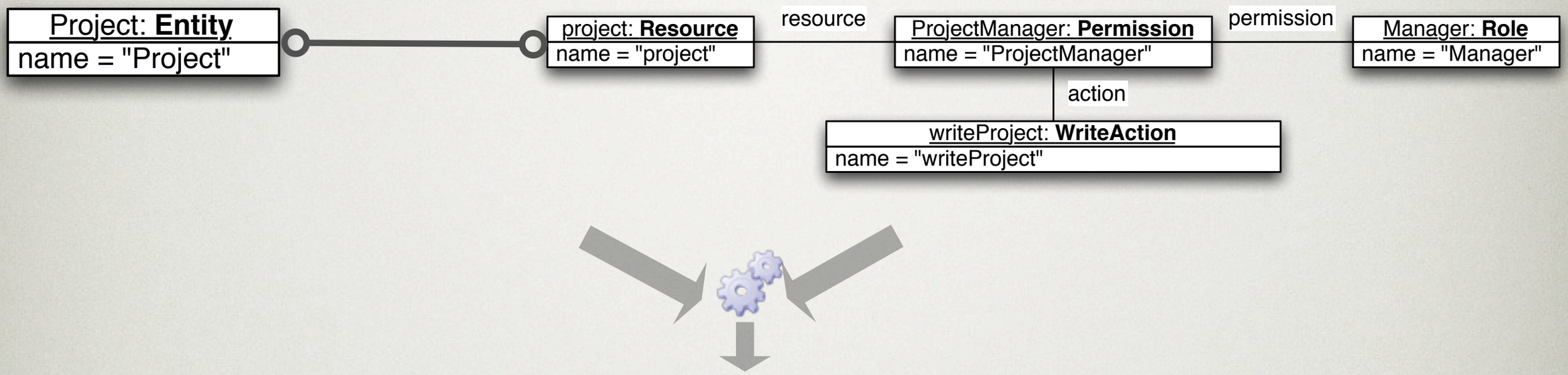
Project: Entity
name = "Project" protected = "true"

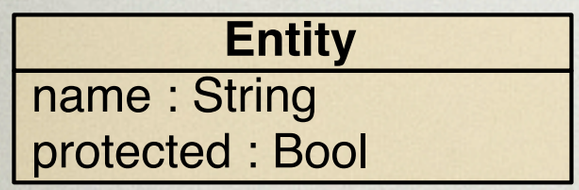
Heterogeneous composition

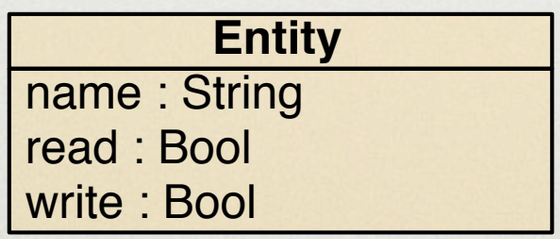


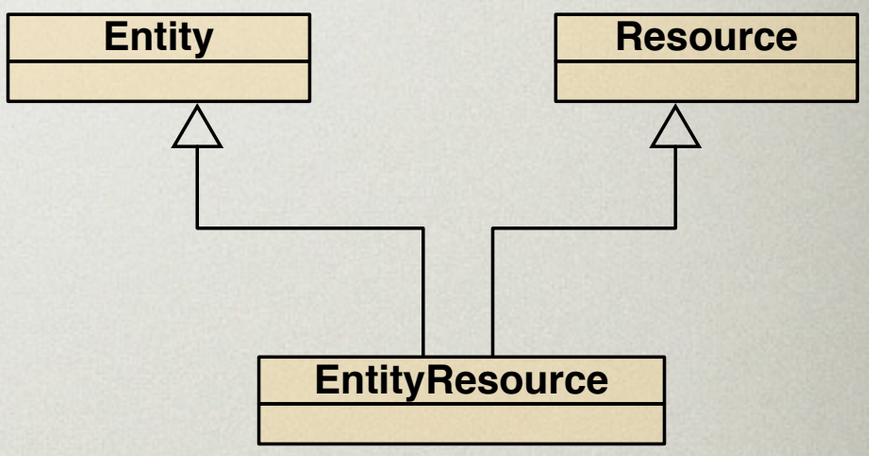


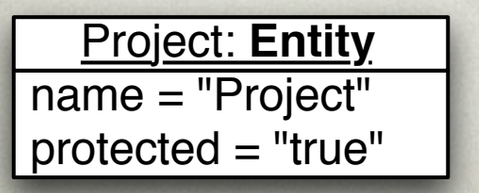
Heterogeneous composition

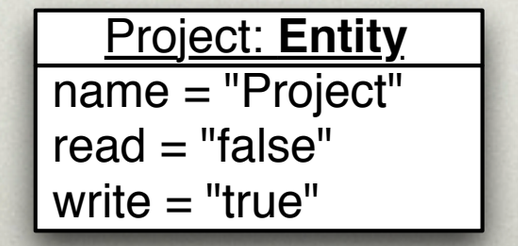




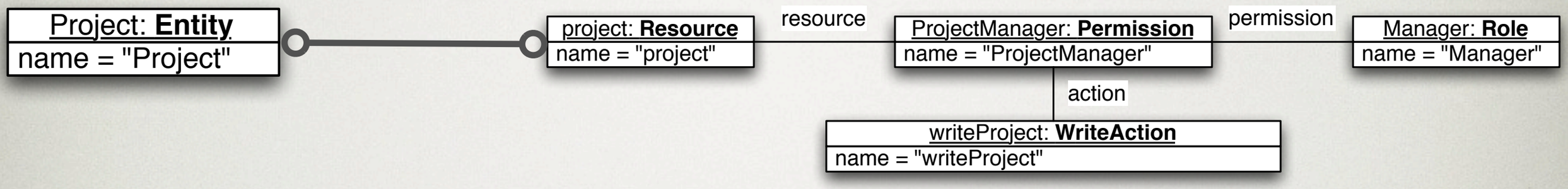




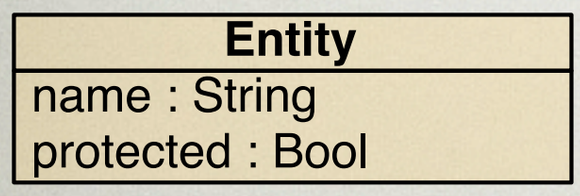


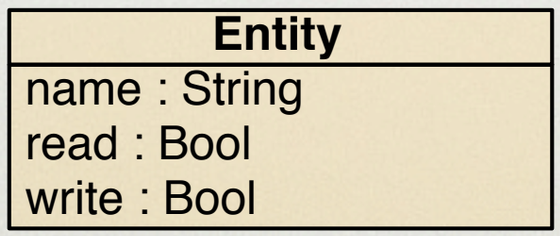


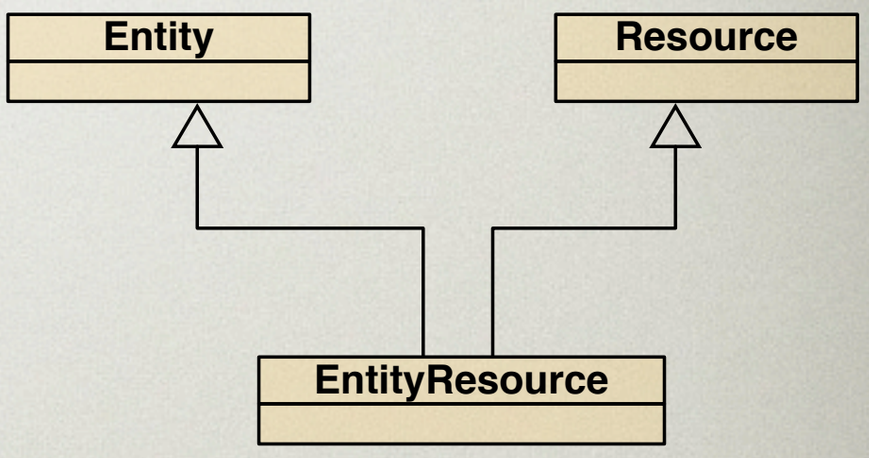
Heterogeneous composition

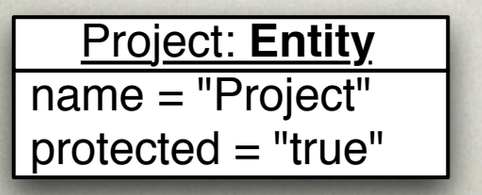


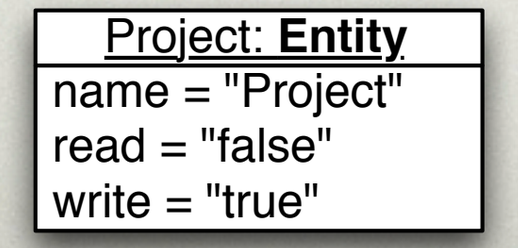
which is the correct composition?



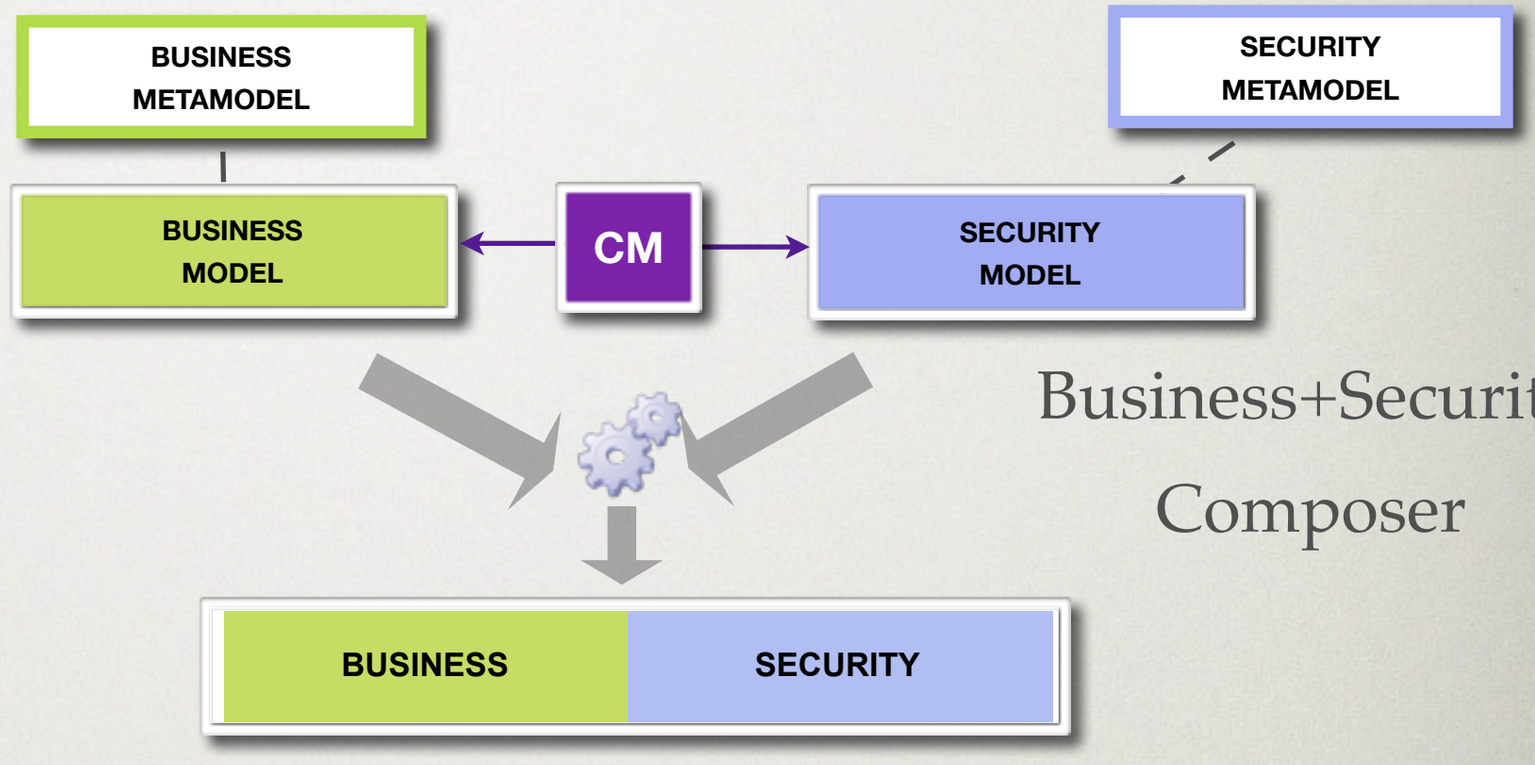






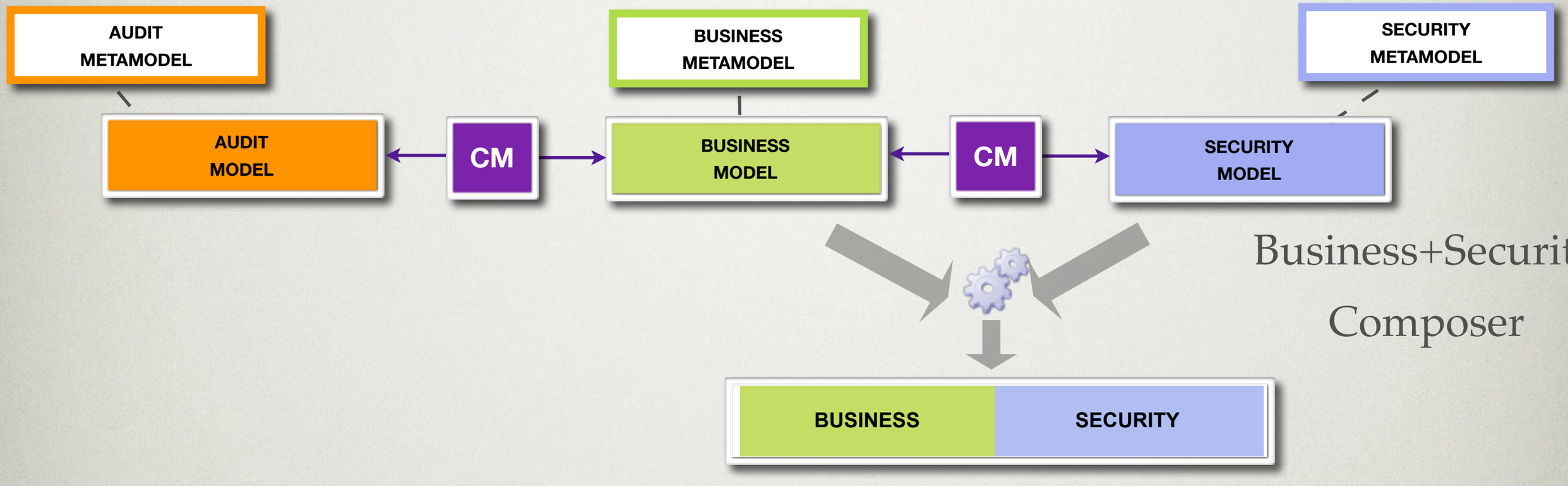


Heterogeneous composition

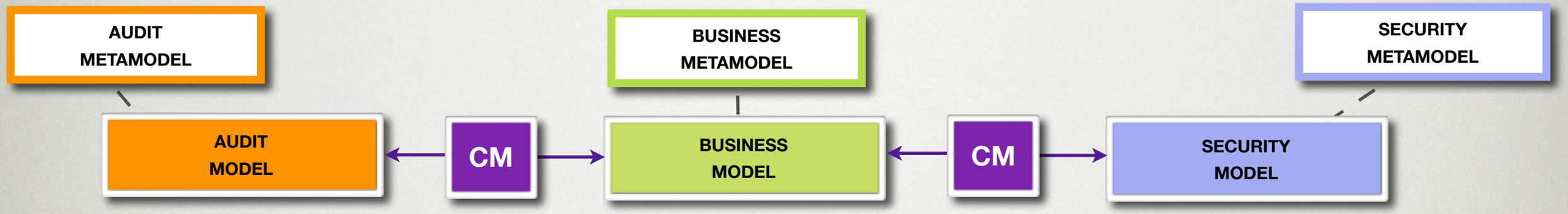


Business+Security Composer

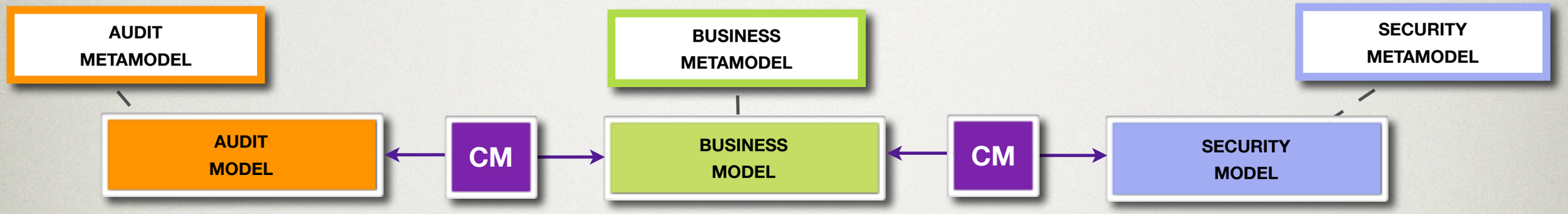
Heterogeneous composition



Heterogeneous composition

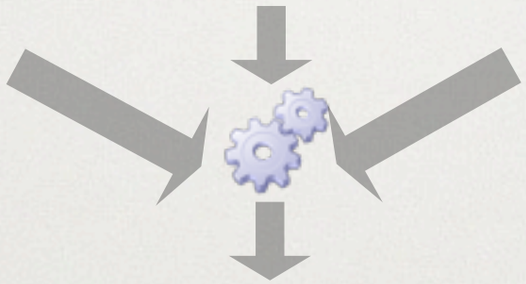


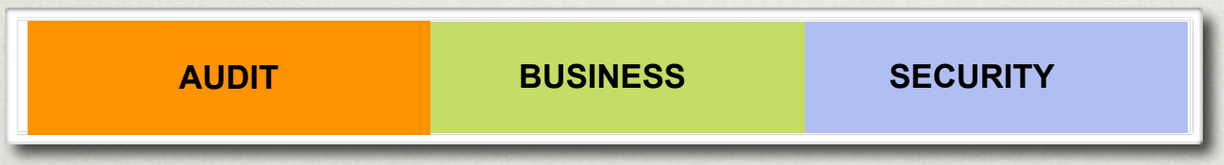
Heterogeneous composition



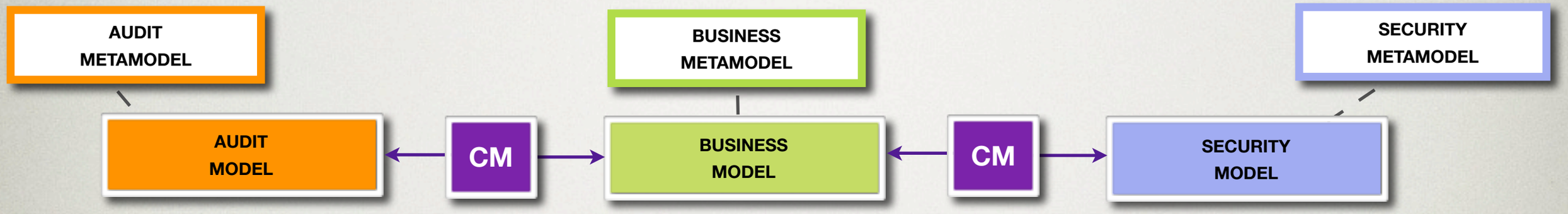
Business+Security+Audit

Composer

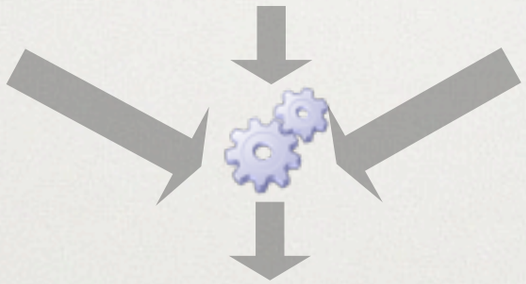


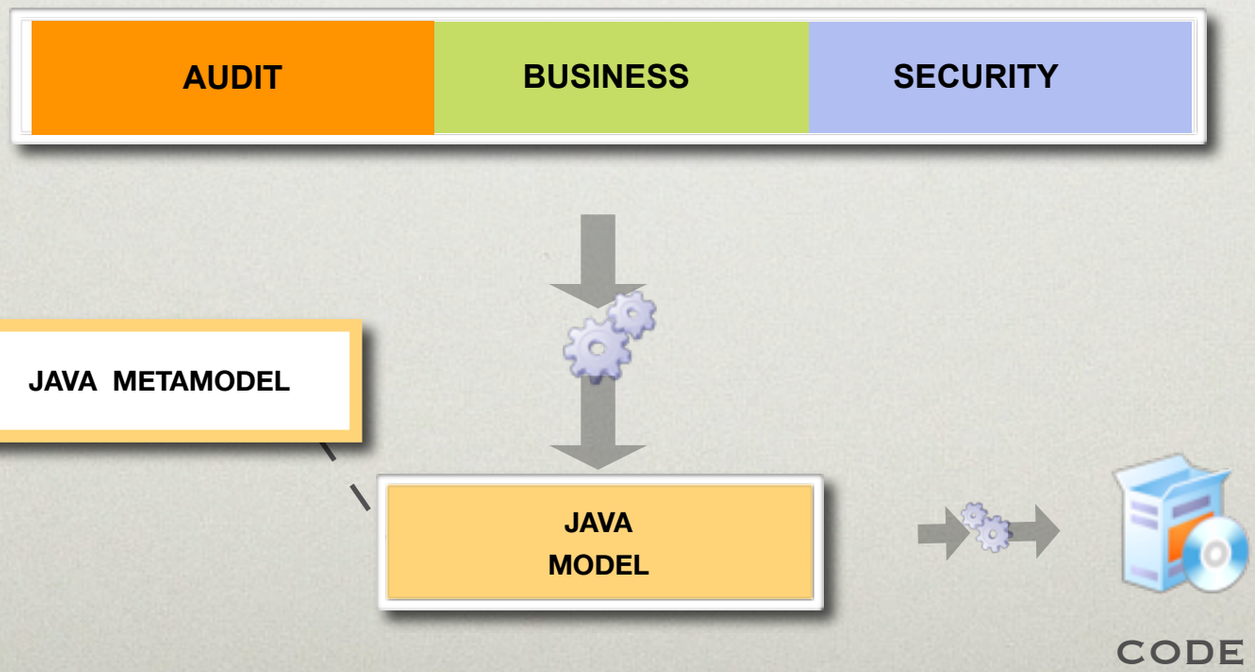


Heterogeneous composition

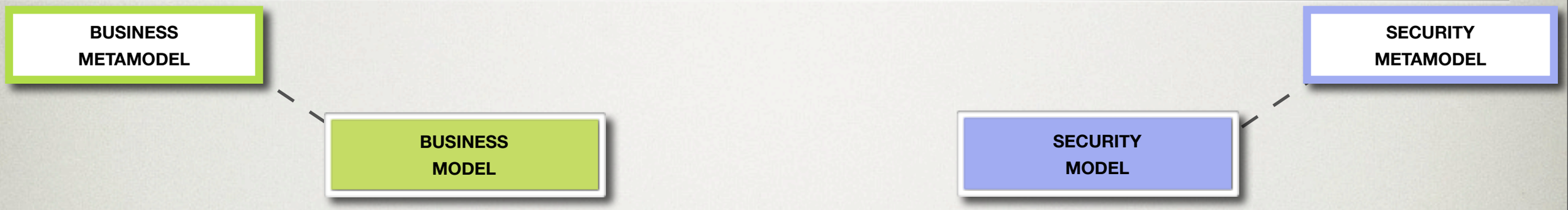


Business+Security+Audit Composer



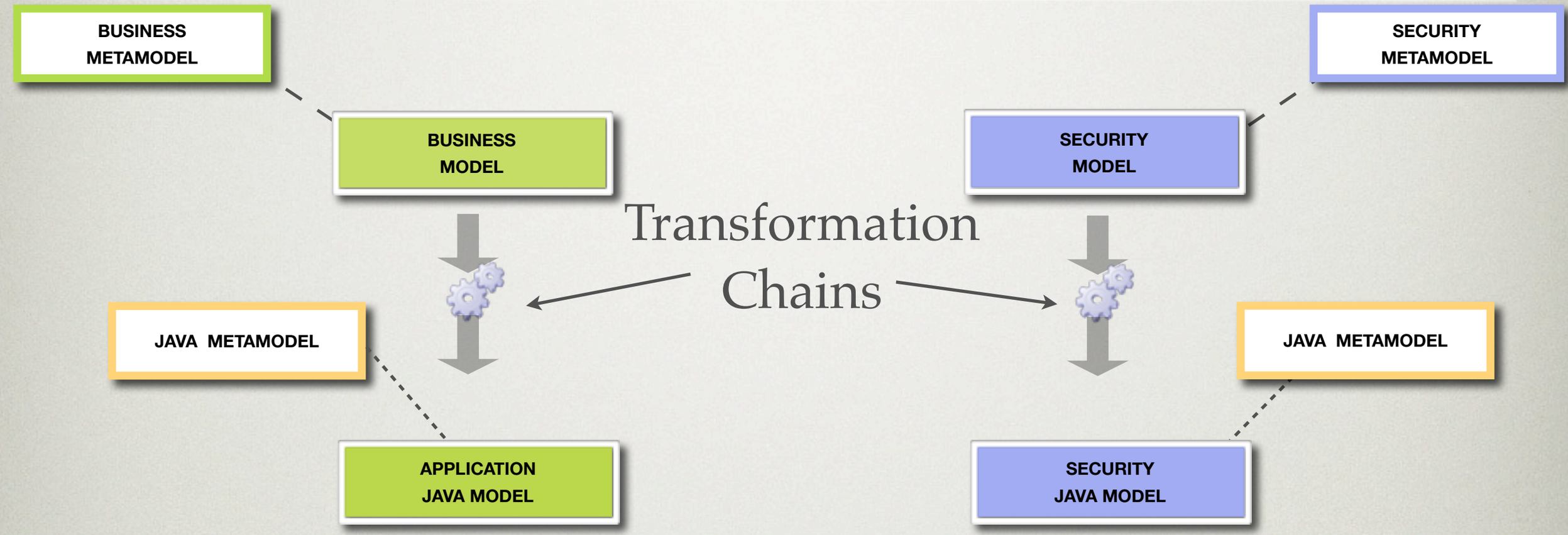


Homogeneous composition



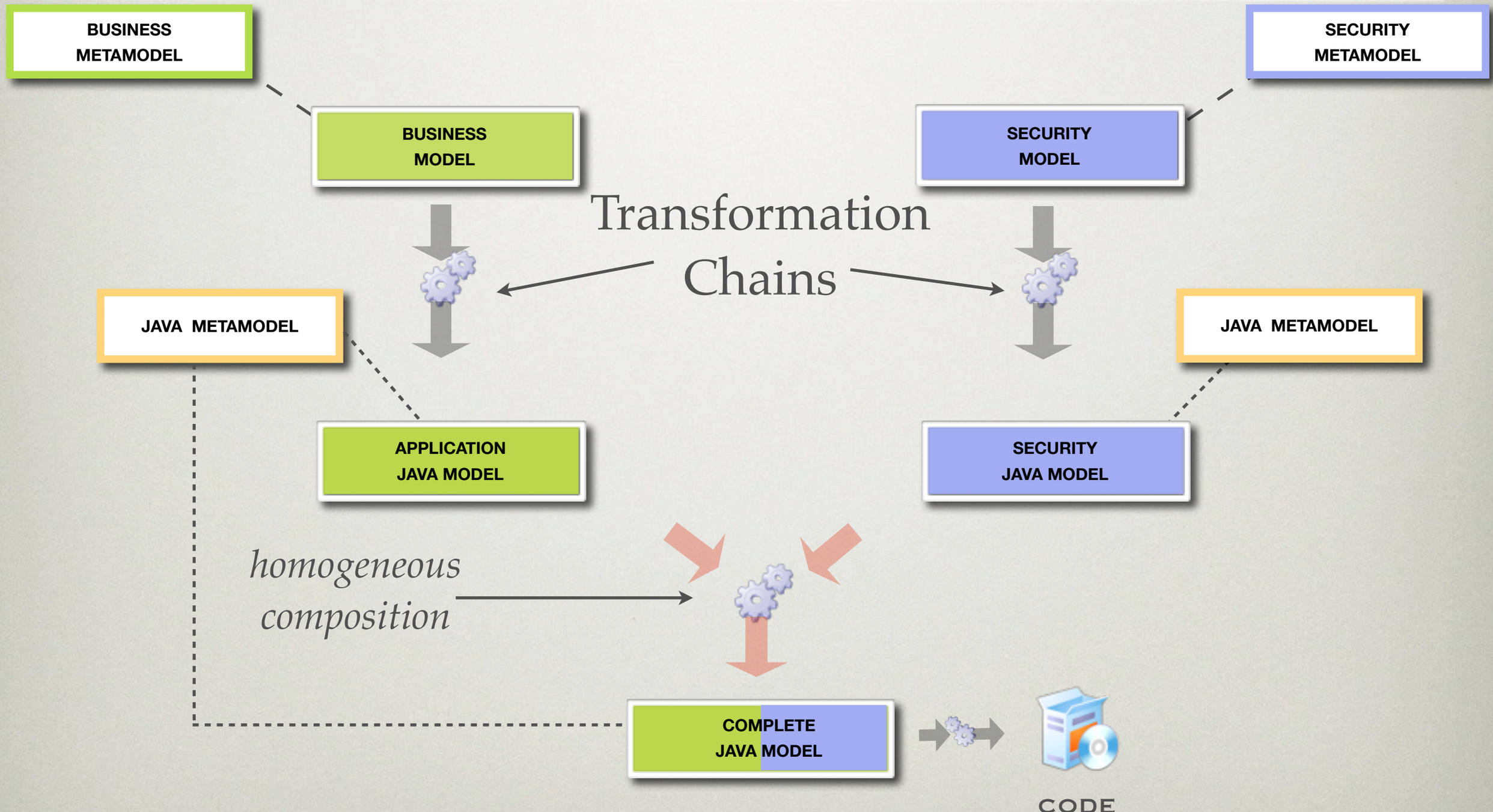
[Vangheluwe] DEVS as a common denominator for multi-formalism hybrid systems modelling. IEEE International Symposium on Computer-Aided Control System Design, 2000. CACSD 2000 (2000) pp. 129-134

Homogeneous composition



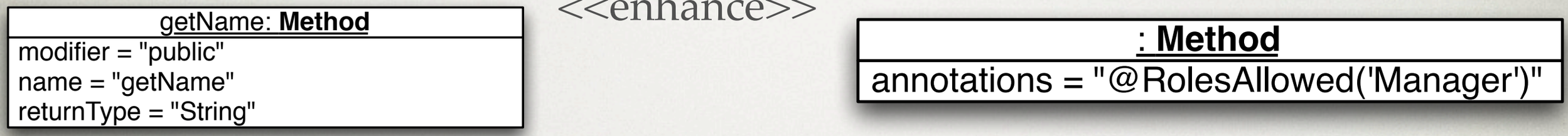
[Vangheluwe] DEVS as a common denominator for multi-formalism hybrid systems modelling. IEEE International Symposium on Computer-Aided Control System Design, 2000. CACSD 2000 (2000) pp. 129-134

Homogeneous composition

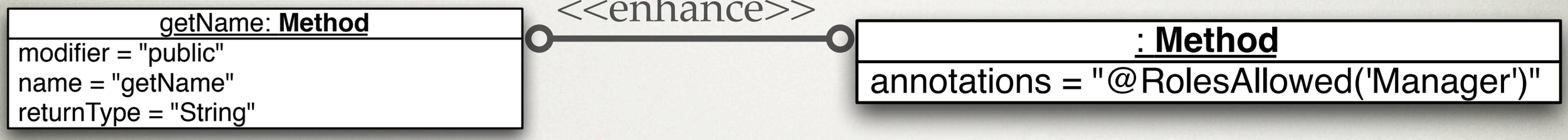


[Vangheluwe] DEVS as a common denominator for multi-formalism hybrid systems modelling. IEEE International Symposium on Computer-Aided Control System Design, 2000. CACSD 2000 (2000) pp. 129-134

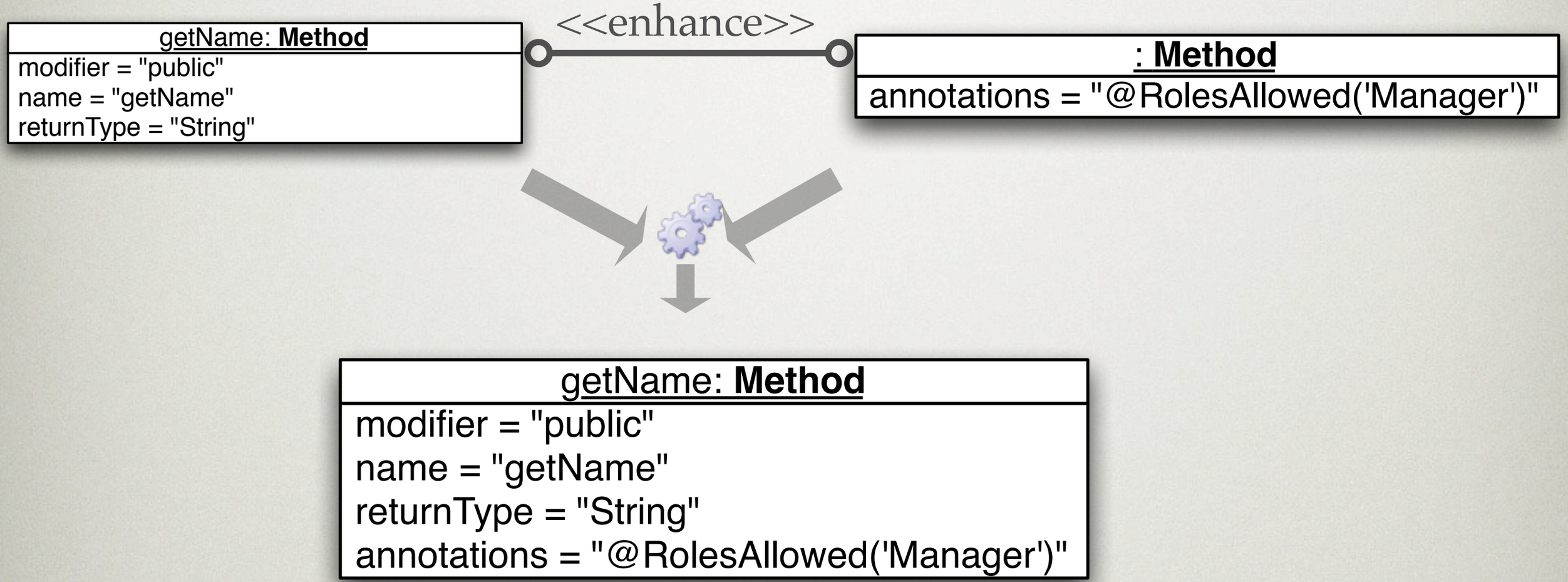
Homogeneous Composition



Homogeneous Composition



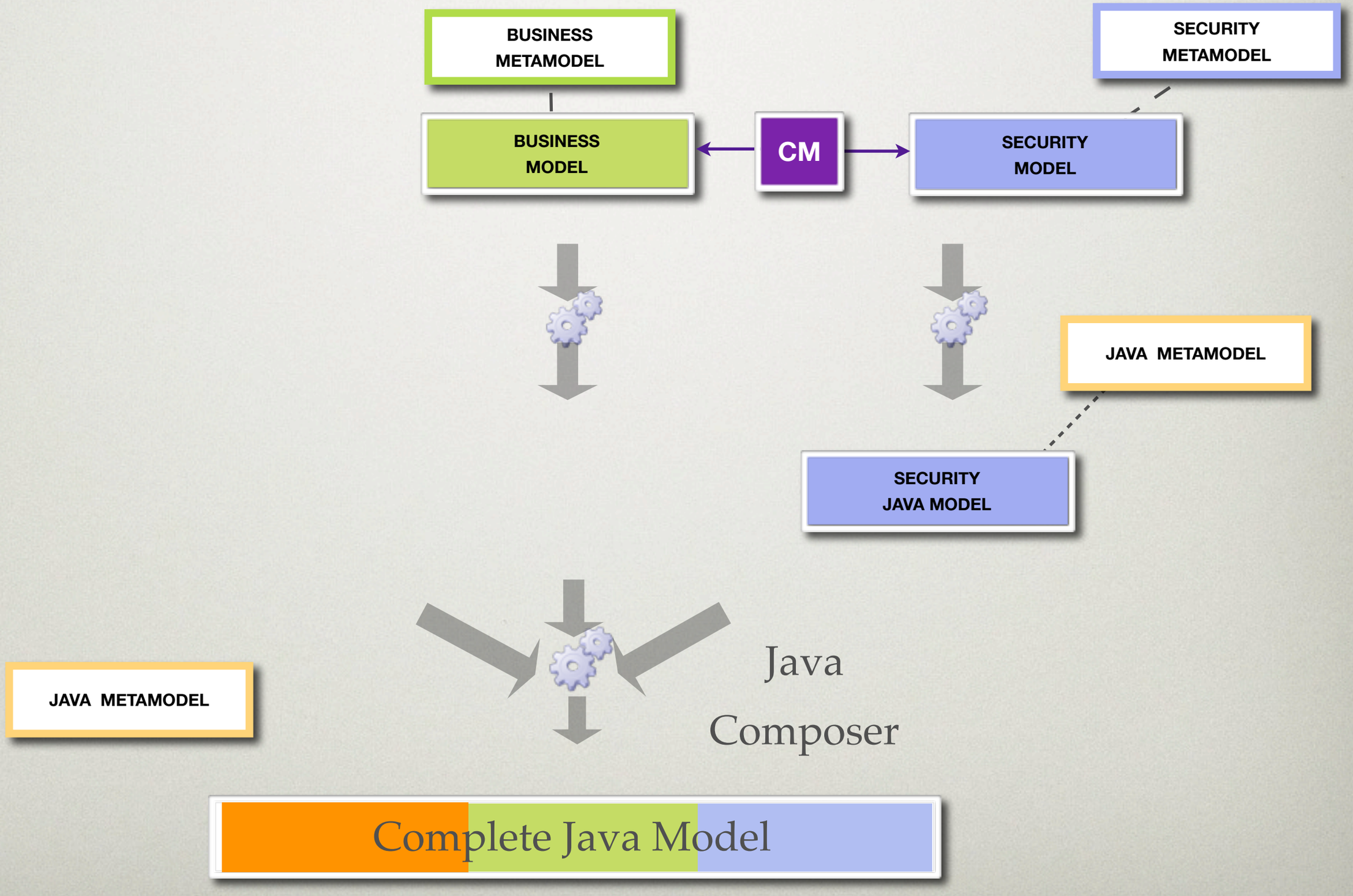
Homogeneous Composition



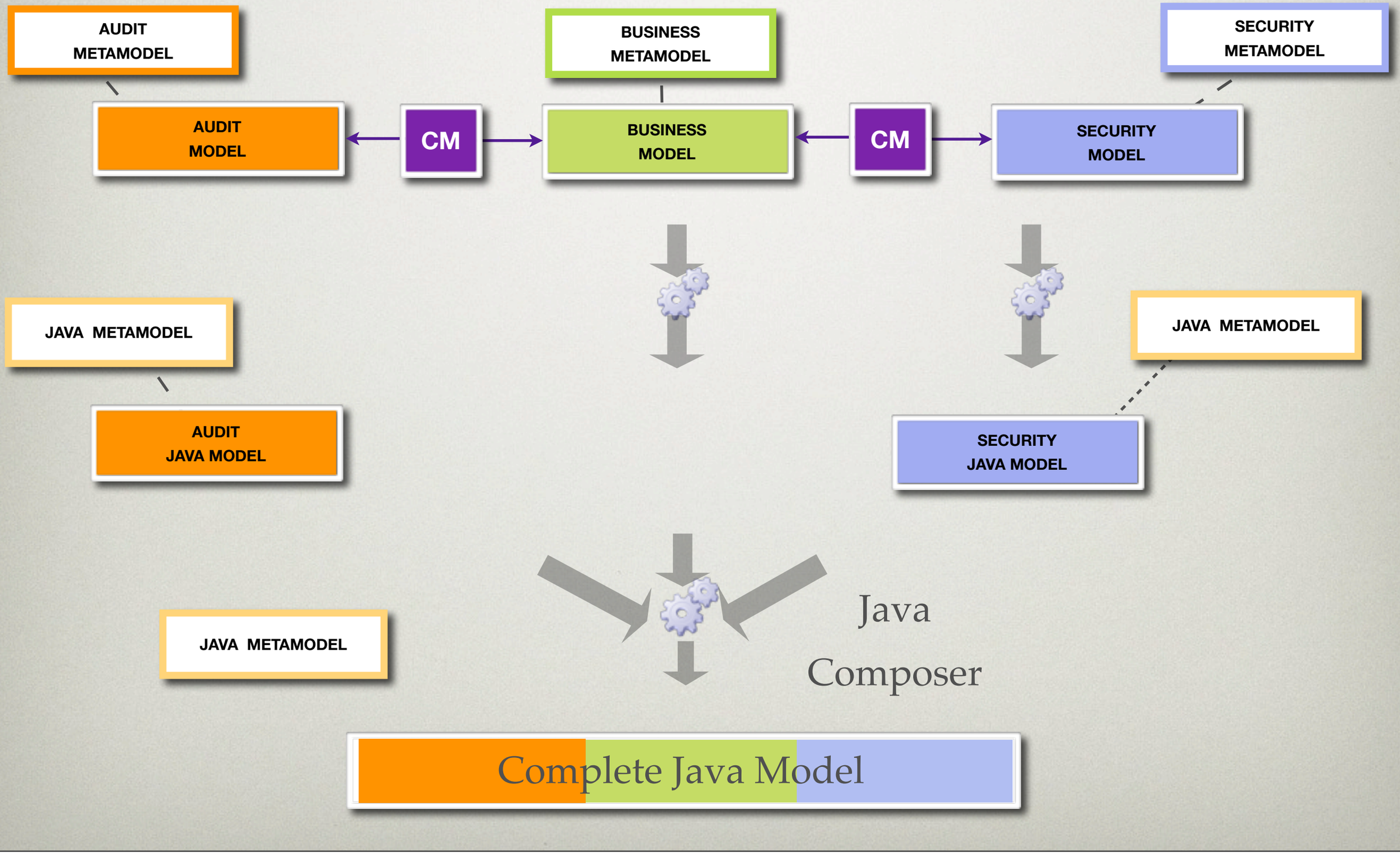
Delaying the composition

1. At low-level it is possible to perform an *homogeneous* composition *(class ⊕ class)*.
2. In a GPL based metamodel it is possible to model all the required concerns
3. Models richer in implementation details (allow fine grain composition)
4. Reuse of existing assets (Metamodels, Transformations and models)

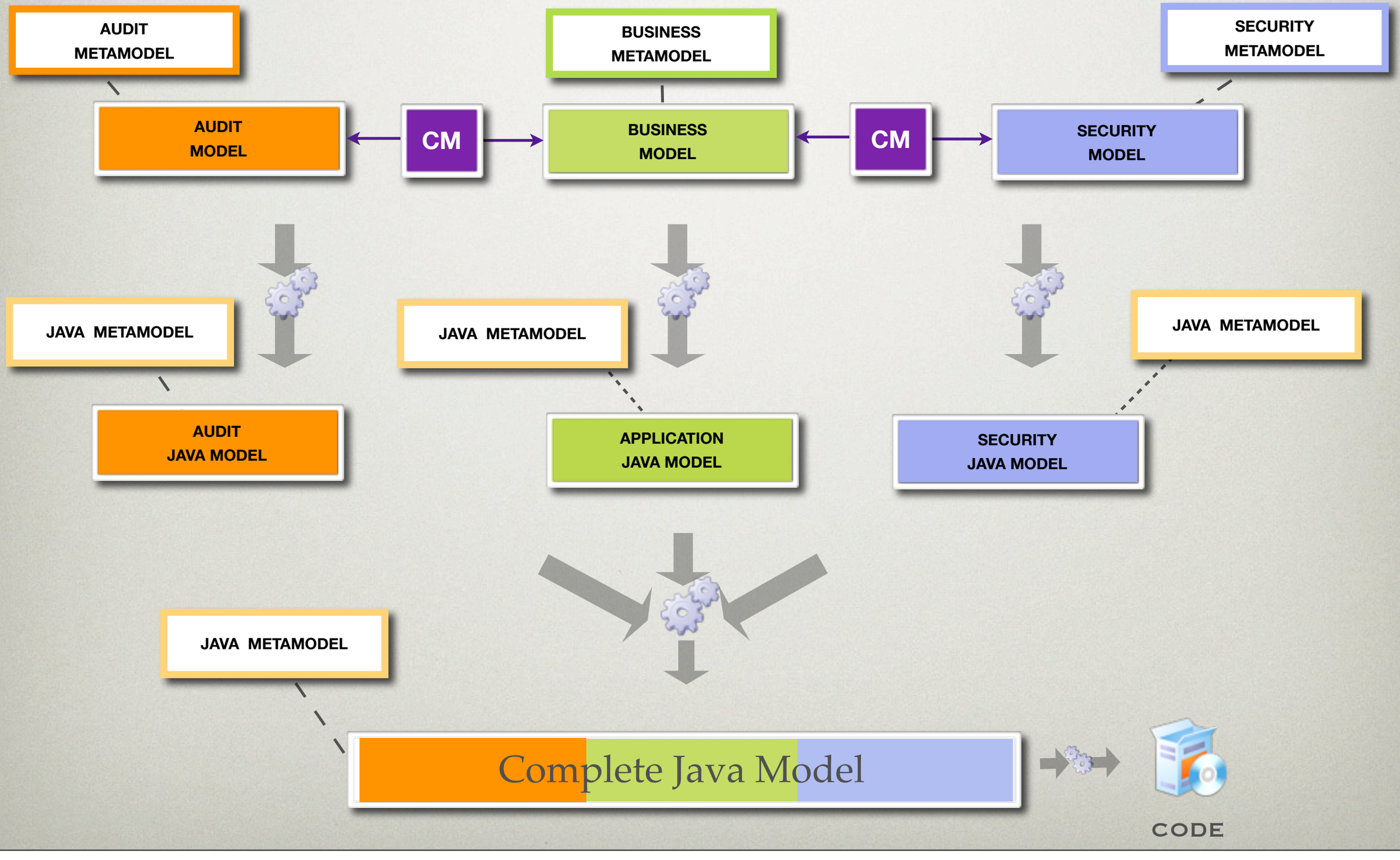
Multiple-views



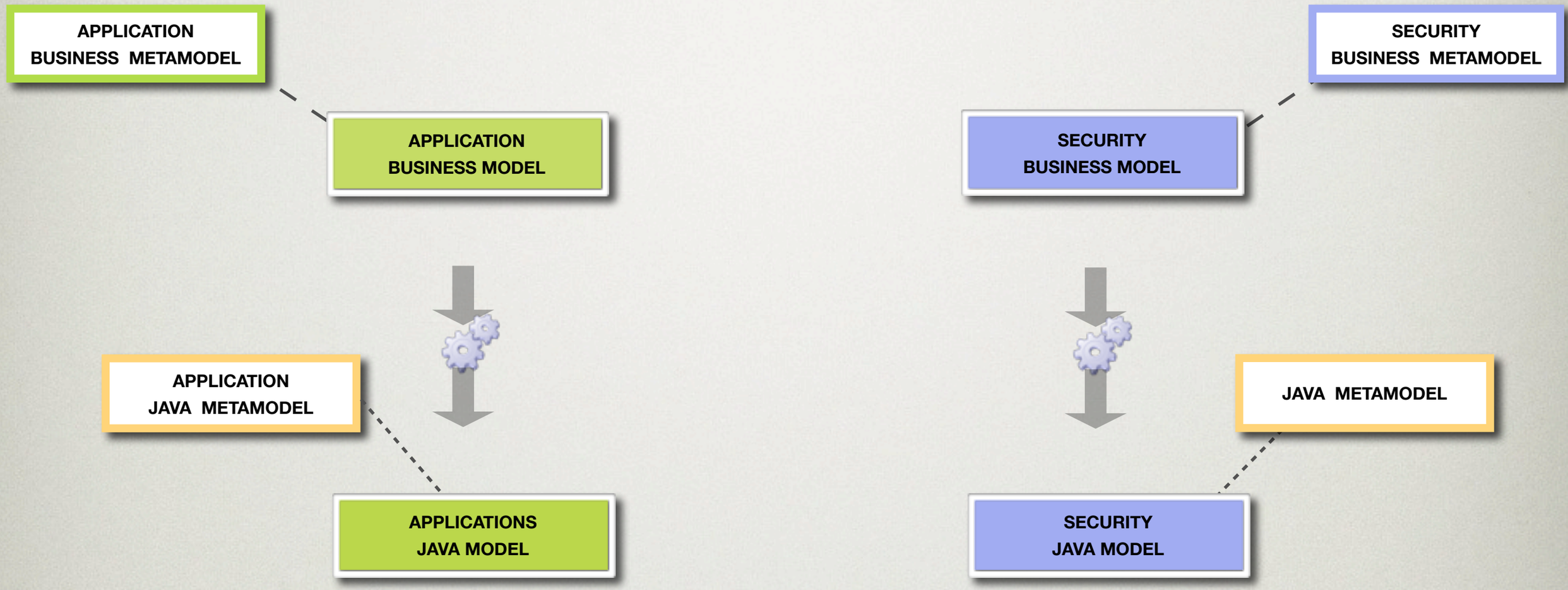
Multiple-views



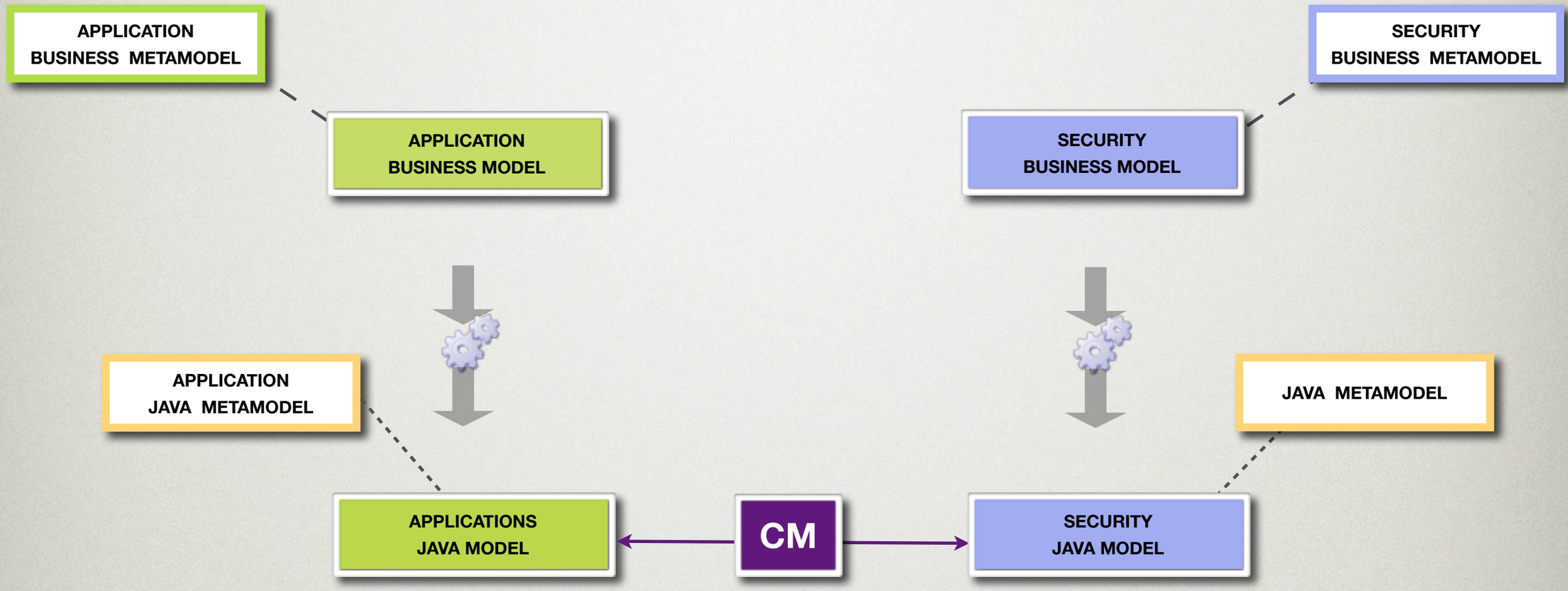
Multiple-views



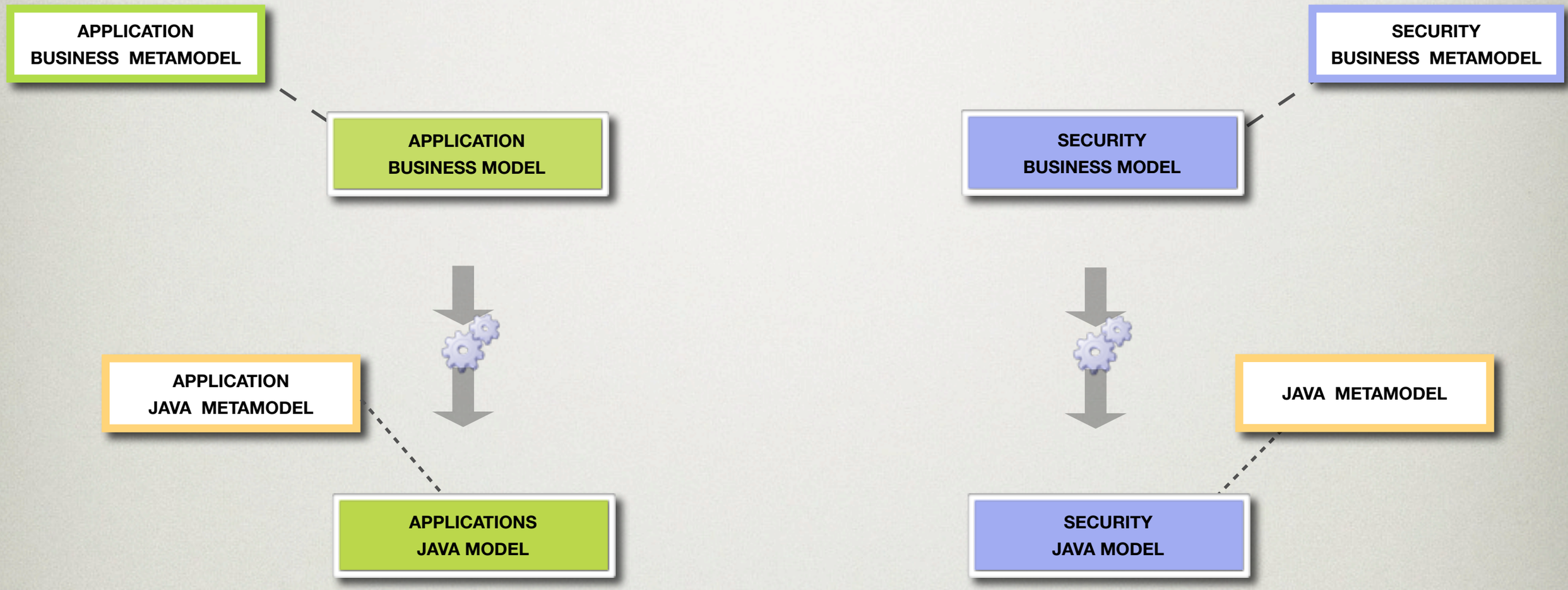
CORRESPONDENCE MODEL DERIVATION



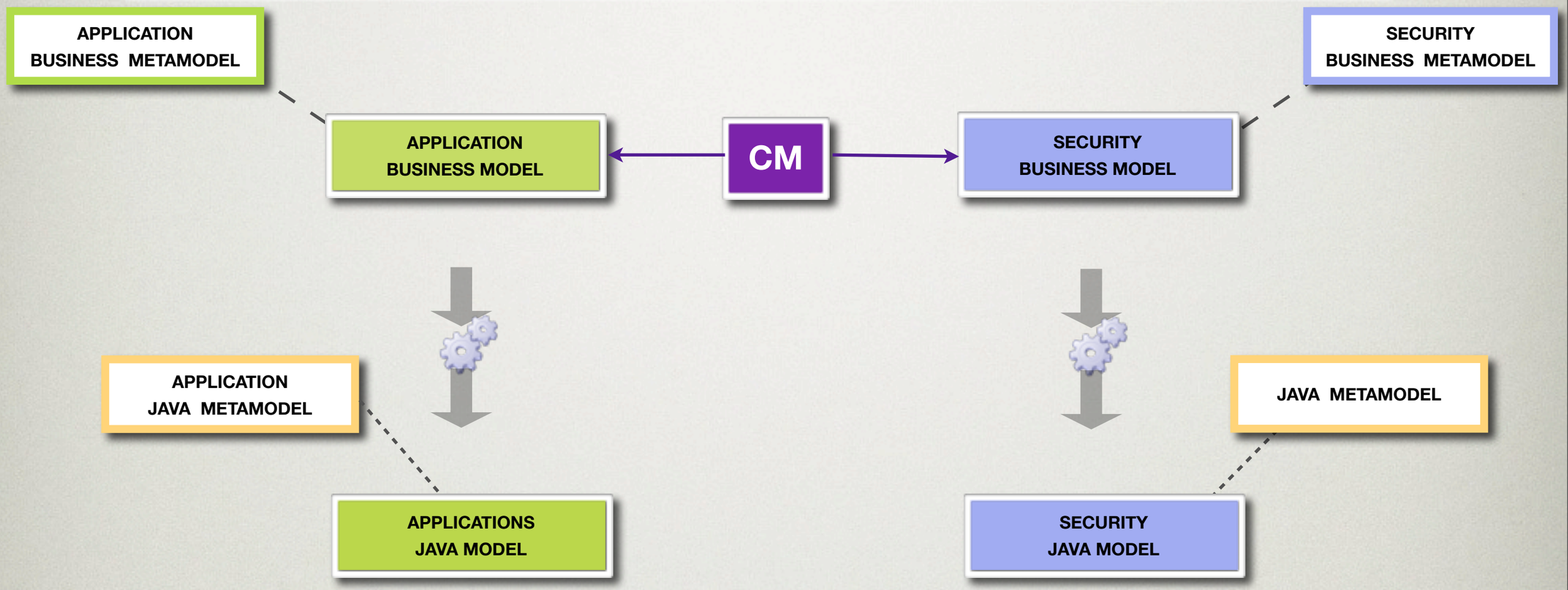
CORRESPONDENCE MODEL DERIVATION



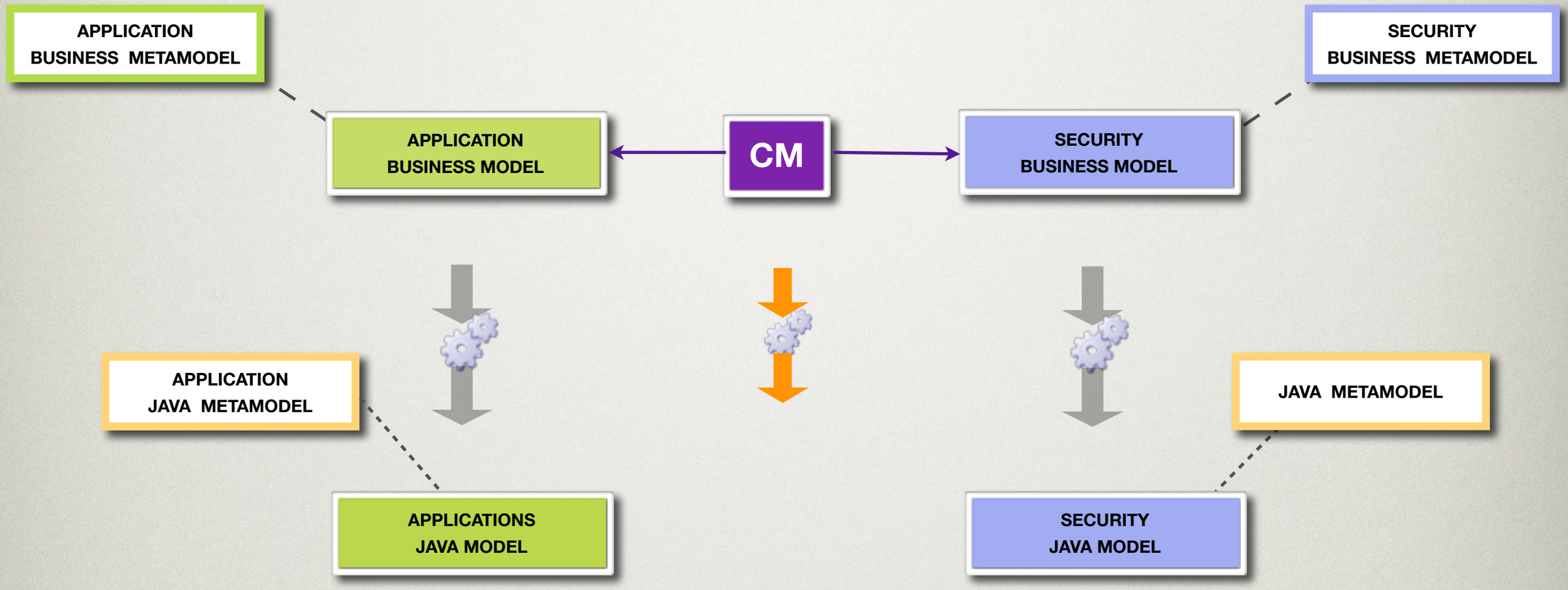
CORRESPONDENCE MODEL DERIVATION



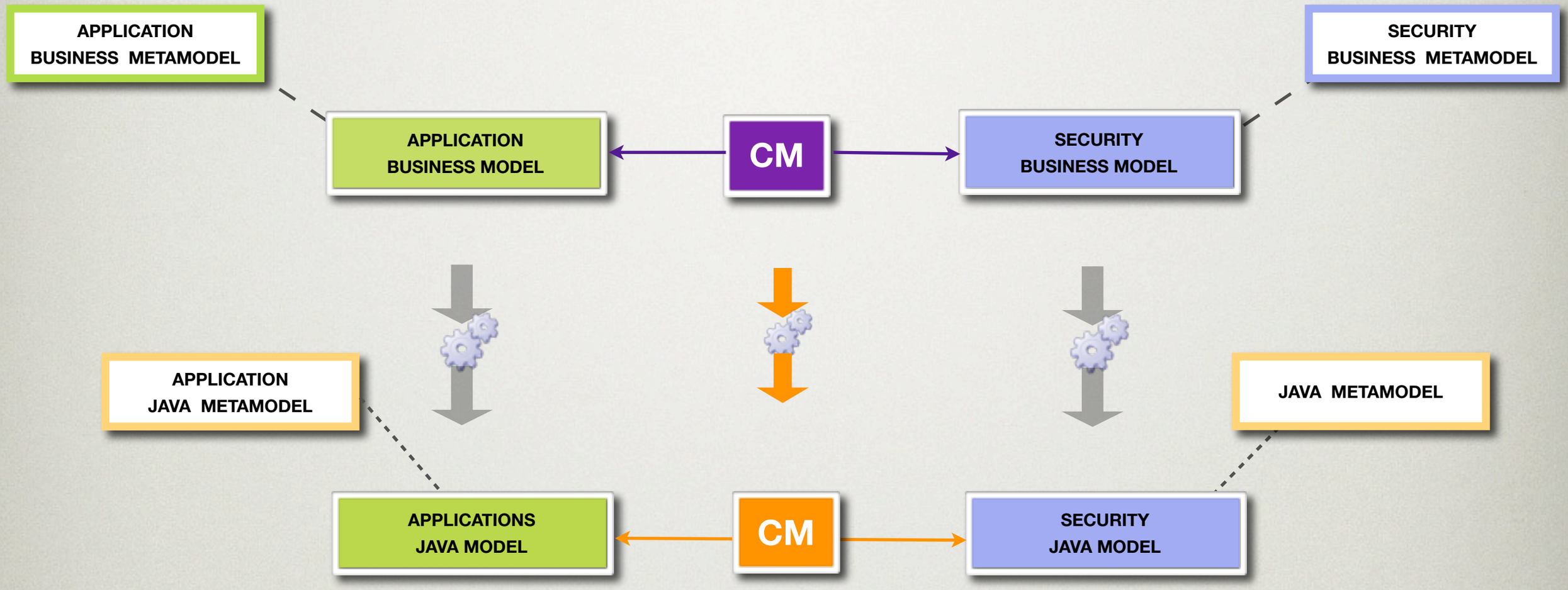
CORRESPONDENCE MODEL DERIVATION



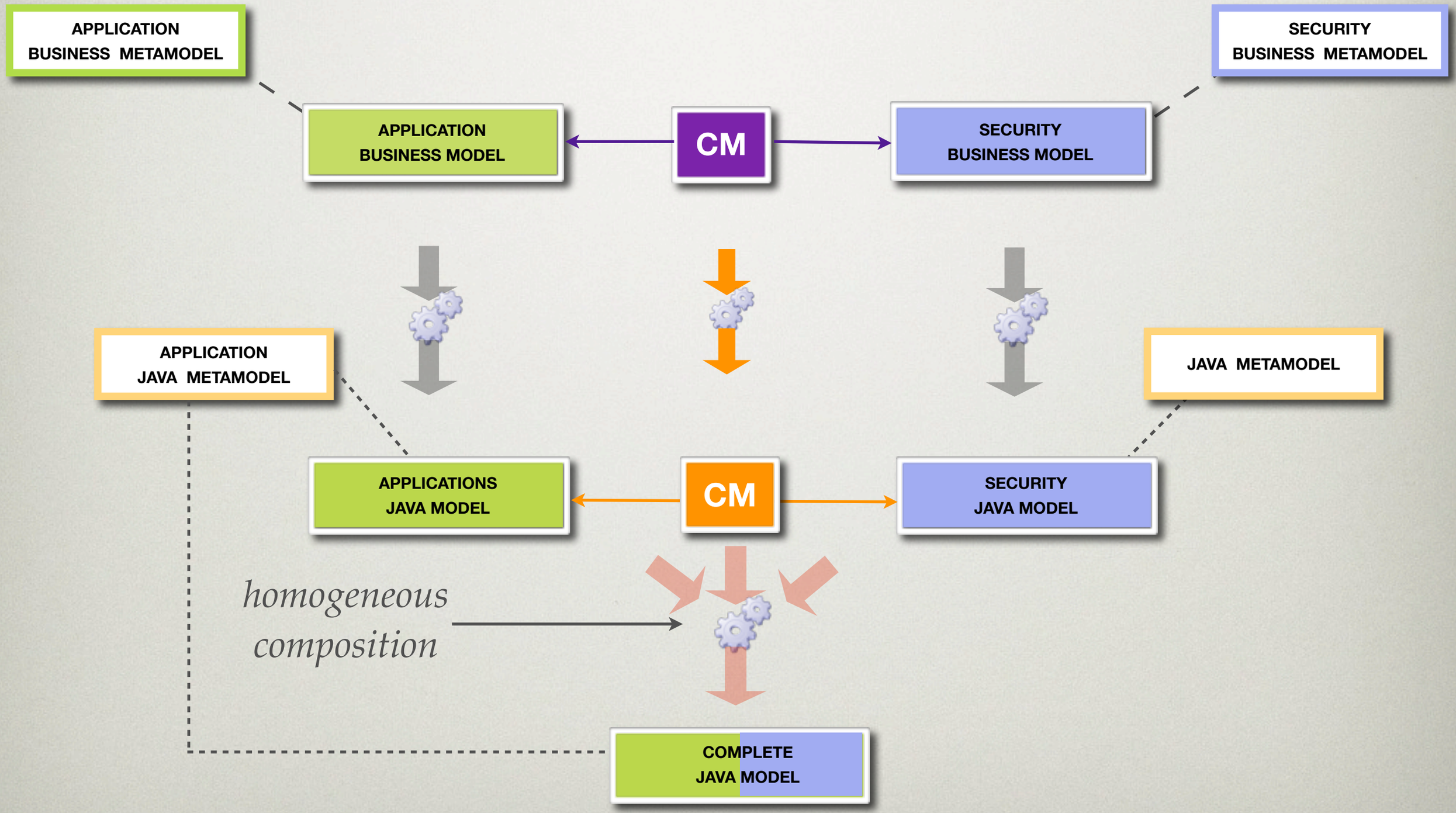
CORRESPONDENCE MODEL DERIVATION



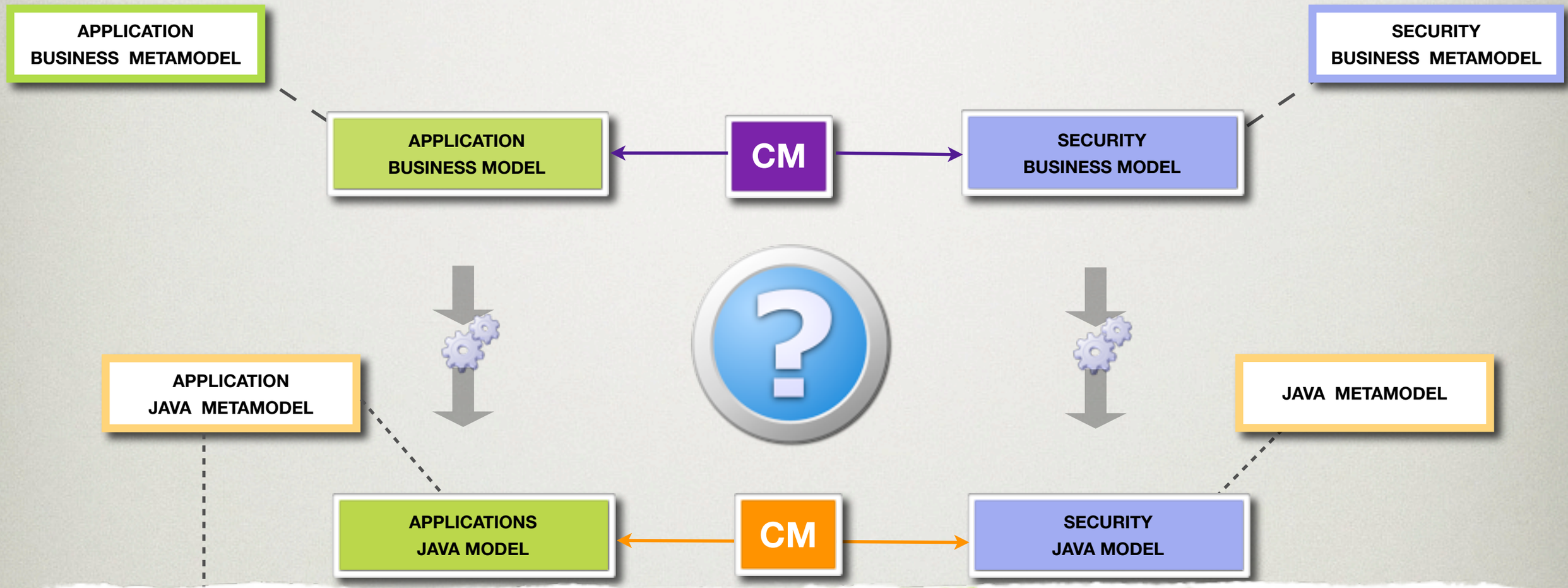
CORRESPONDENCE MODEL DERIVATION



CORRESPONDENCE MODEL DERIVATION

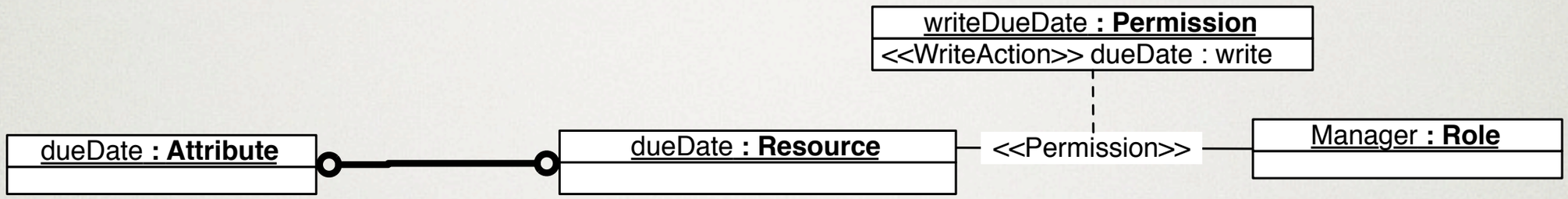


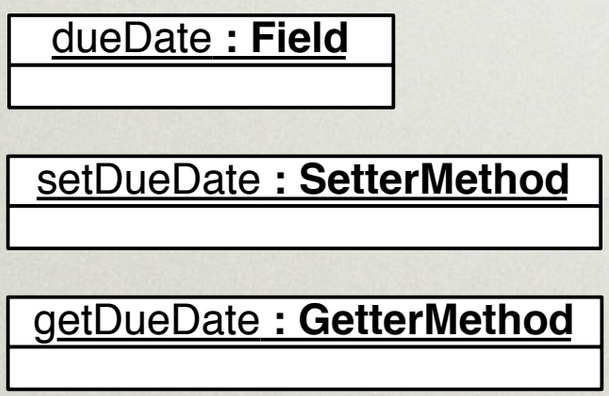
CORRESPONDENCE MODEL DERIVATION

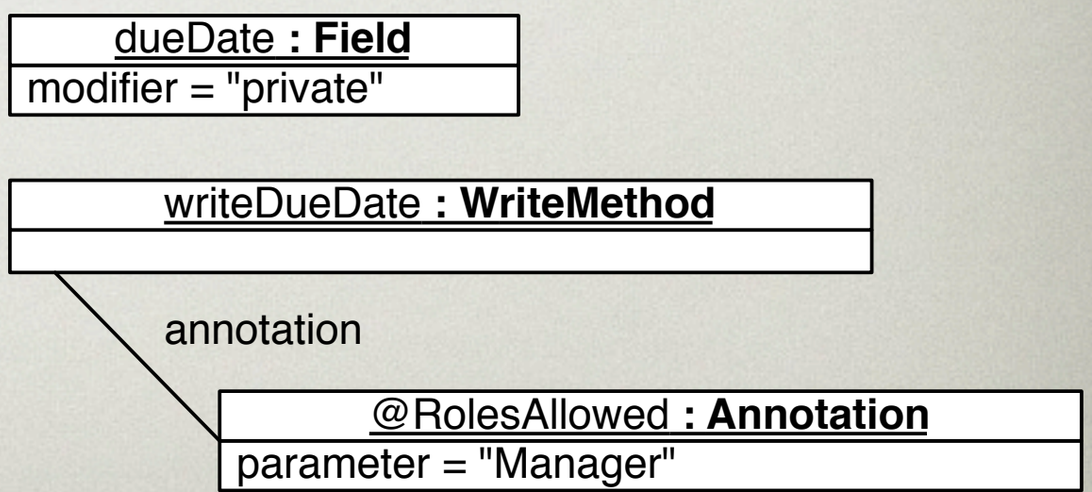


How to automatically derive a correspondence model

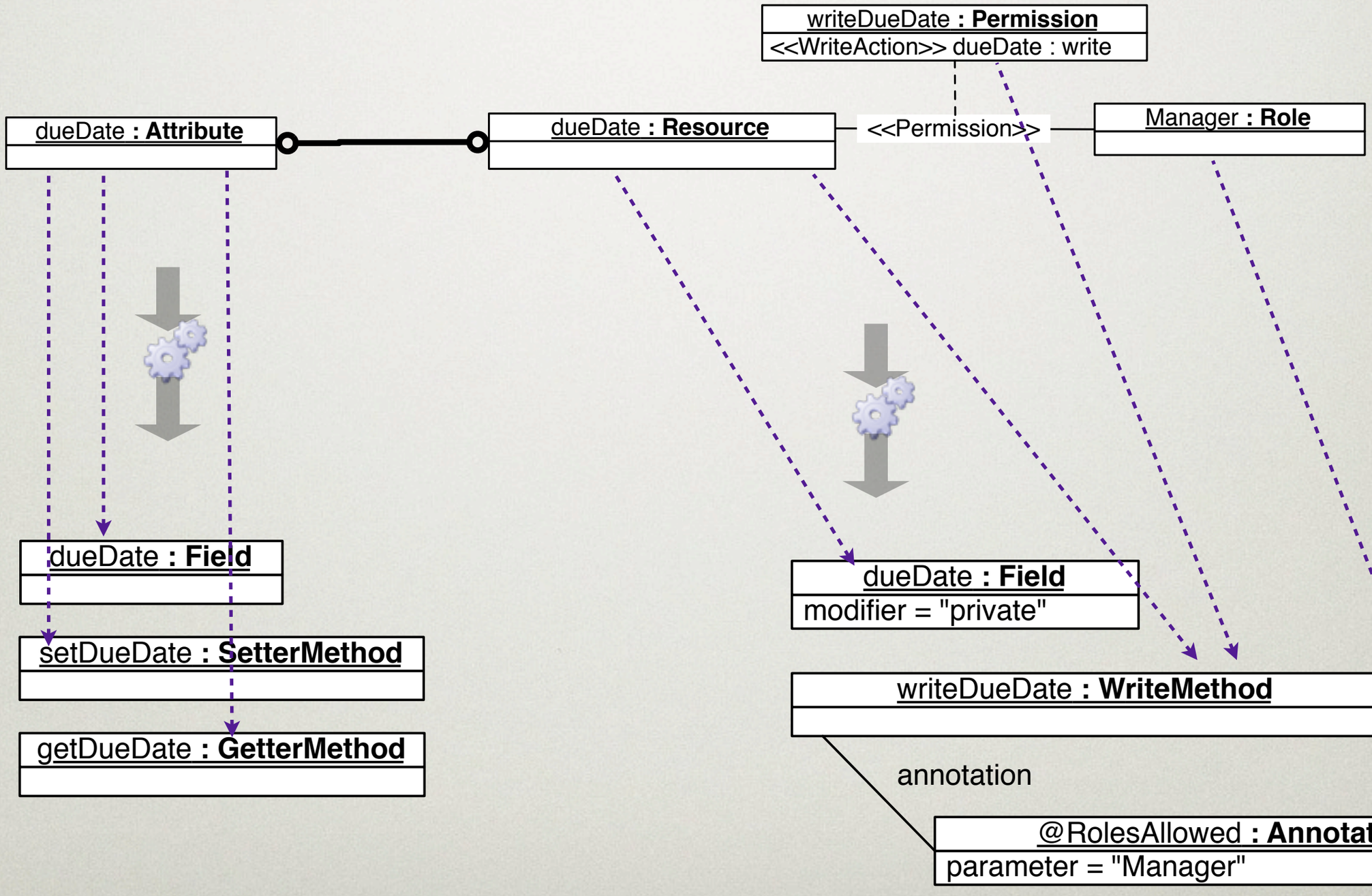
Correspondence model derivation



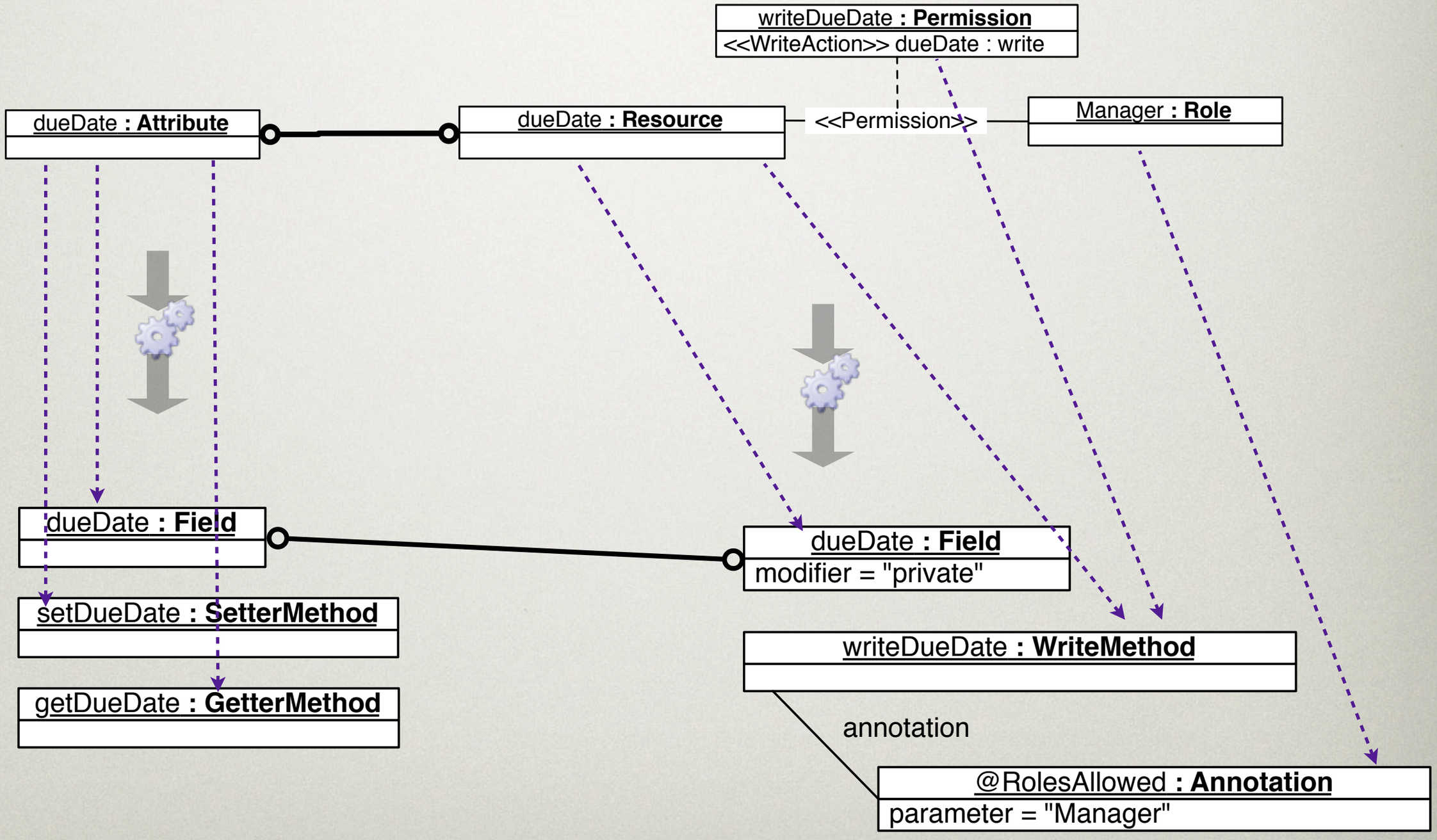




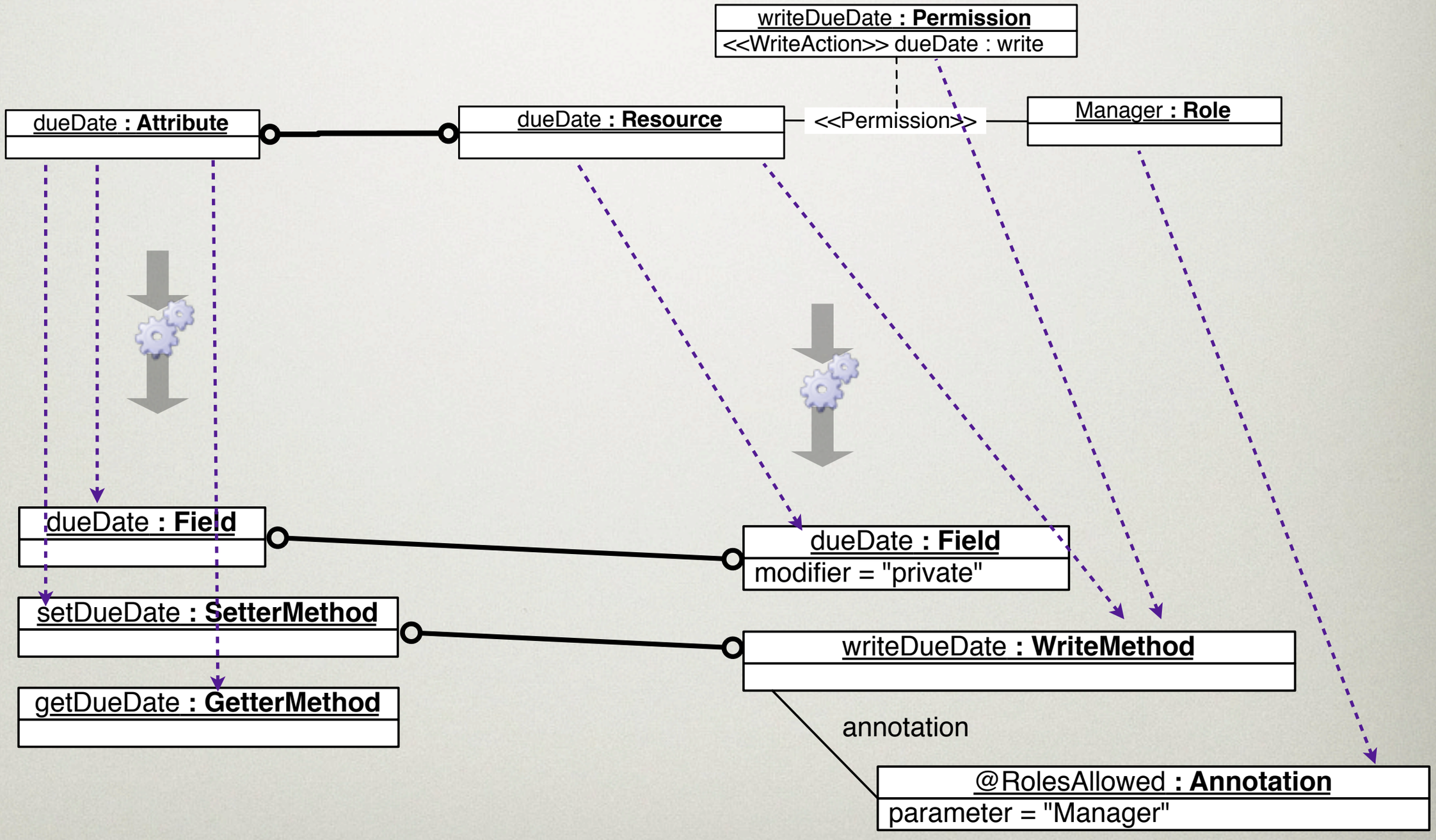
Correspondence model derivation



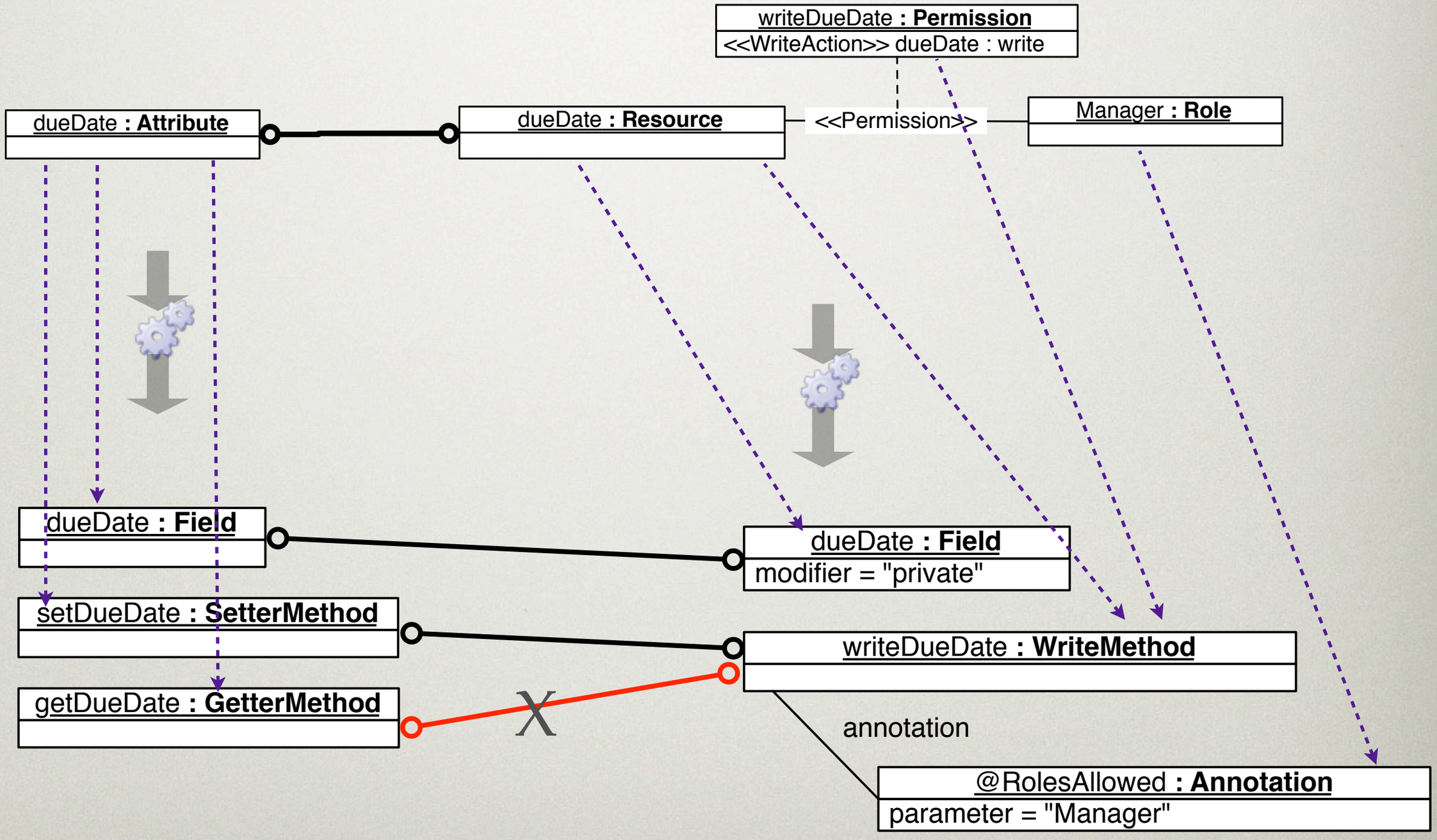
Correspondence model derivation

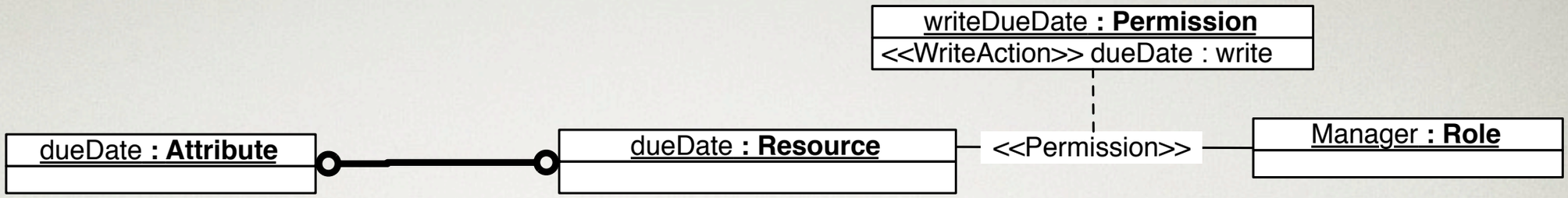


Correspondence model derivation



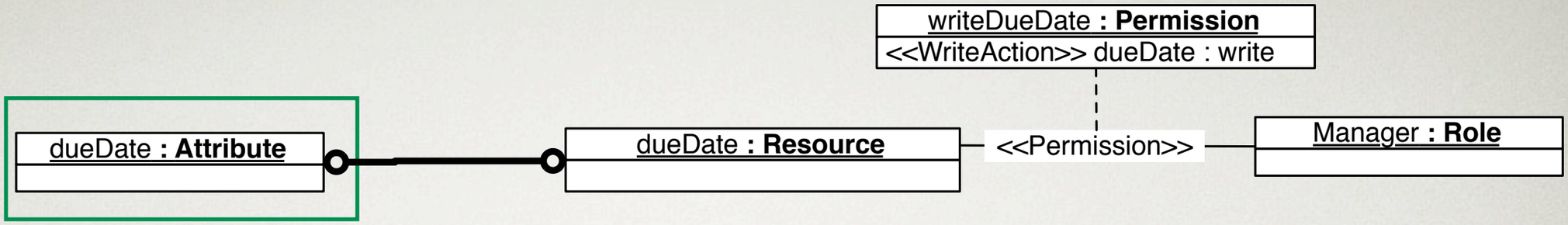
Correspondence model derivation





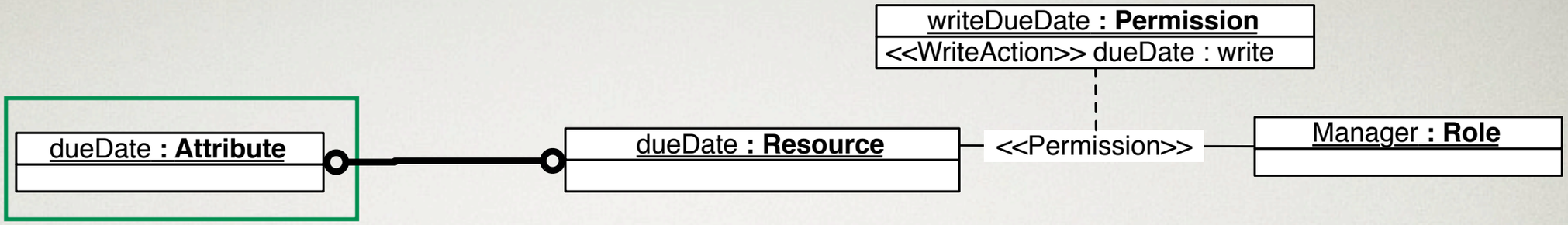

```
rule Attribute2Field {
	from
		a		:		EA!Attribute
	to
		s		:		JAVA!Field
		(
			name	<-	a.name
		),
		setter		:		JAVA!SetterMethod
		(
			name	<-	'set' + a.name
		),
		getter		:		JAVA!GetterMethod
		(
			name	<-	'get' + a.name
		)
}
```

```
rule Attribute2Field {
	from
		a			:Security!Resource
	to
		s			:	JAVA!Field
		(
			name		<-	a.name,
			modifier	<-	“private”
		),
		write		:	JAVA!WriteMethod
		(
			name		<-	'write' + a.write.name
			annotation	<-	ann,
		),
		ann			:	JAVA!Annotation
		(
			name		<-	'@RolesAllowed'
			parameters	<-	a.write.permission.role.name
		)
}
```



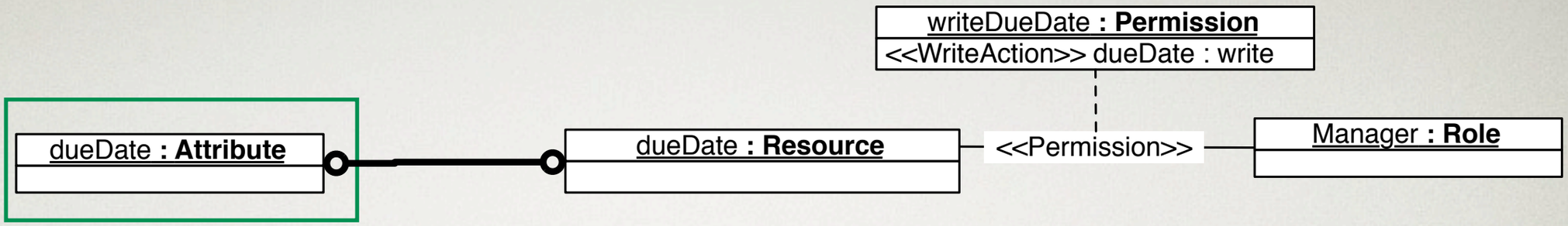

```
rule Attribute2Field {
	from
		a		:		EA!Attribute
	to
		s		:		JAVA!Field
		(
			name	<-	a.name
		),
		setter		:		JAVA!SetterMethod
		(
			name	<-	'set' + a.name
		),
		getter		:		JAVA!GetterMethod
		(
			name	<-	'get' + a.name
		)
}
```

```
rule Attribute2Field {
	from
		a			:Security!Resource
	to
		s			:	JAVA!Field
		(
			name		<-	a.name,
			modifier	<-	“private”
		),
		write		:	JAVA!WriteMethod
		(
			name		<-	'write' + a.write.name
			annotation	<-	ann,
		),
		ann			:	JAVA!Annotation
		(
			name		<-	'@RolesAllowed'
			parameters	<-	a.write.permission.role.name
		)
}
```



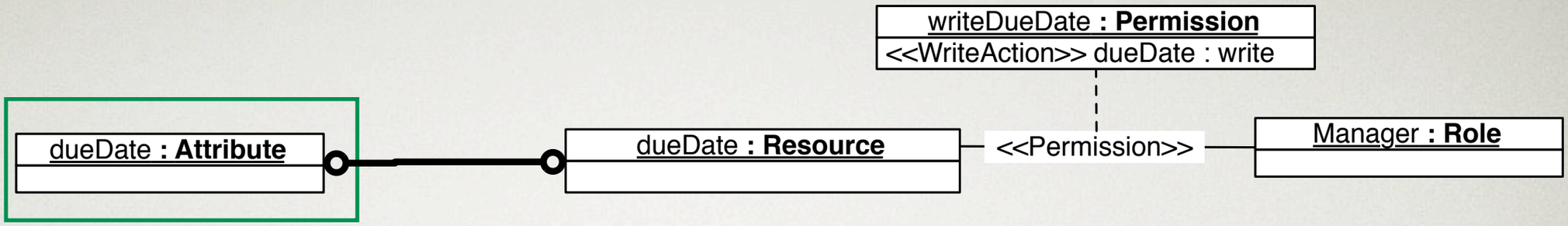

```
rule Attribute2Field {
	from
		a		:		EA!Attribute
	to
		s		:		JAVA!Field
		(
			name	<-	a.name
		),
		setter		:		JAVA!SetterMethod
		(
			name	<-	'set' + a.name
		),
		getter		:		JAVA!GetterMethod
		(
			name	<-	'get' + a.name
		)
}
```

```
rule Attribute2Field {
	from
		a			:Security!Resource
	to
		s			:	JAVA!Field
		(
			name		<-	a.name,
			modifier	<-	“private”
		),
		write		:	JAVA!WriteMethod
		(
			name		<-	'write' + a.write.name
			annotation	<-	ann,
		),
		ann			:	JAVA!Annotation
		(
			name		<-	'@RolesAllowed'
			parameters	<-	a.write.permission.role.name
		)
}
```

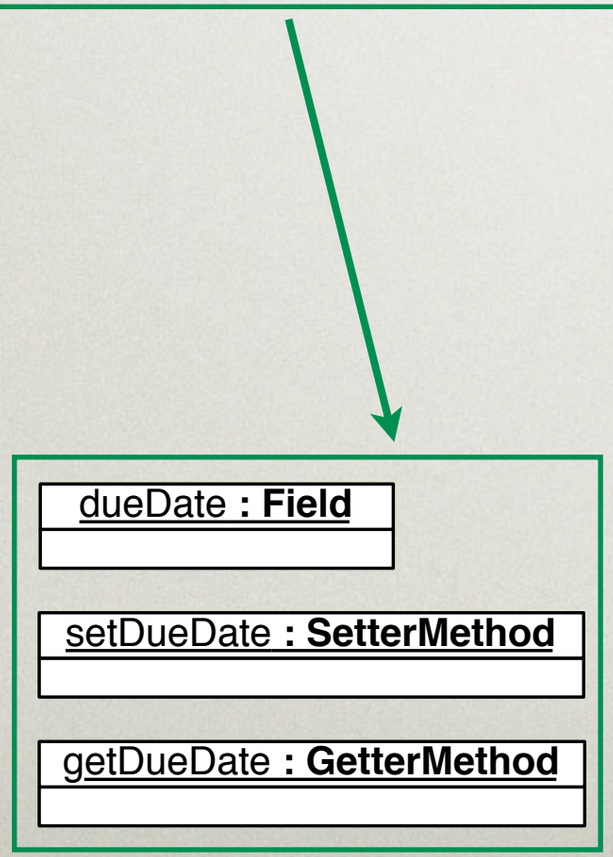



```
rule Attribute2Field {
	from
		a		:		EA!Attribute
	to
		s		:		JAVA!Field
		(
			name	<-		a.name
		),
		setter		:		JAVA!SetterMethod
		(
			name	<-		'set' + a.name
		),
		getter		:		JAVA!GetterMethod
		(
			name	<-		'get' + a.name
		)
}
```

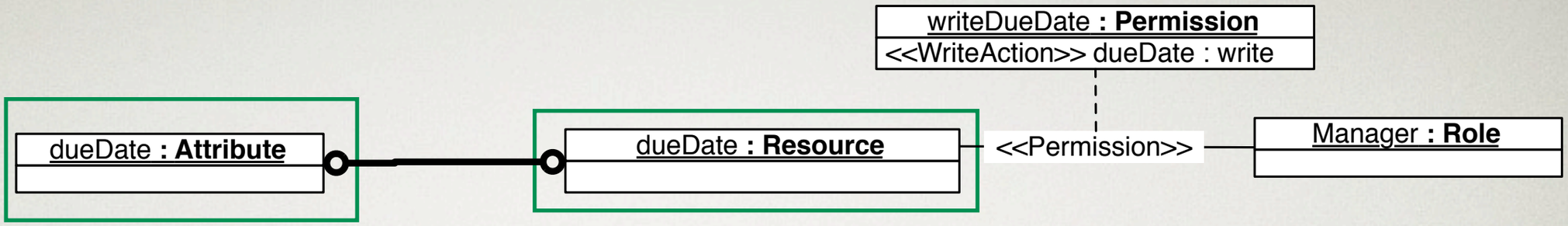
```
rule Attribute2Field {
	from
		a			:Security!Resource
	to
		s			:		JAVA!Field
		(
			name		<-		a.name,
			modifier	<-		“private”
		),
		write		:		JAVA!WriteMethod
		(
			name		<-		'write' + a.write.name
			annotation	<-		ann,
		),
		ann			:		JAVA!Annotation
		(
			name		<-		'@RolesAllowed'
			parameters	<-		a.write.permission.role.name
		)
}
```



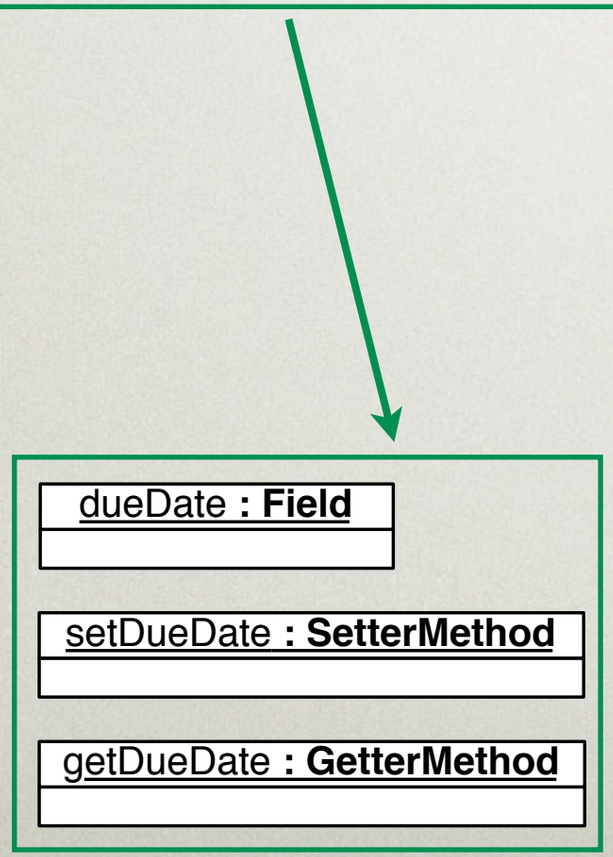

```
rule Attribute2Field {
    from
        a               :               EA!Attribute
    to
        s               :               JAVA!Field
        (
            name  <-        a.name
        ),
        setter          :               JAVA!SetterMethod
        (
            name  <-        'set' + a.name
        ),
        getter          :               JAVA!GetterMethod
        (
            name  <-        'get' + a.name
        )
}
```



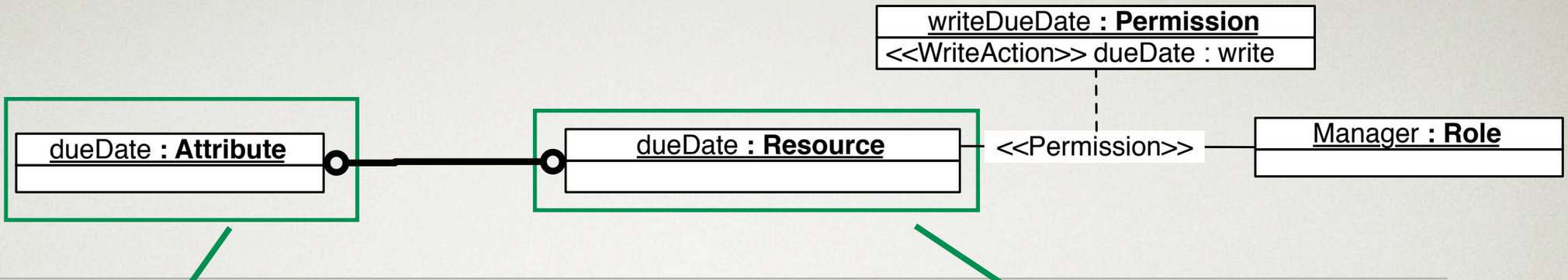

```
rule Attribute2Field {
    from
        a                   :Security!Resource
    to
        s                   :       JAVA!Field
        (
            name        <-      a.name,
            modifier    <-      “private”
        ),
        write       :       JAVA!WriteMethod
        (
            name        <-      'write' + a.write.name
            annotation  <-      ann,
        ),
        ann                 :       JAVA!Annotation
        (
            name        <-      '@RolesAllowed'
            parameters  <-      a.write.permission.role.name
        )
}
```




```
rule Attribute2Field {
	from
		a		:		EA!Attribute
	to
		s		:		JAVA!Field
		(
			name	<-	a.name
		),
		setter		:		JAVA!SetterMethod
		(
			name	<-	'set' + a.name
		),
		getter		:		JAVA!GetterMethod
		(
			name	<-	'get' + a.name
		)
}
```

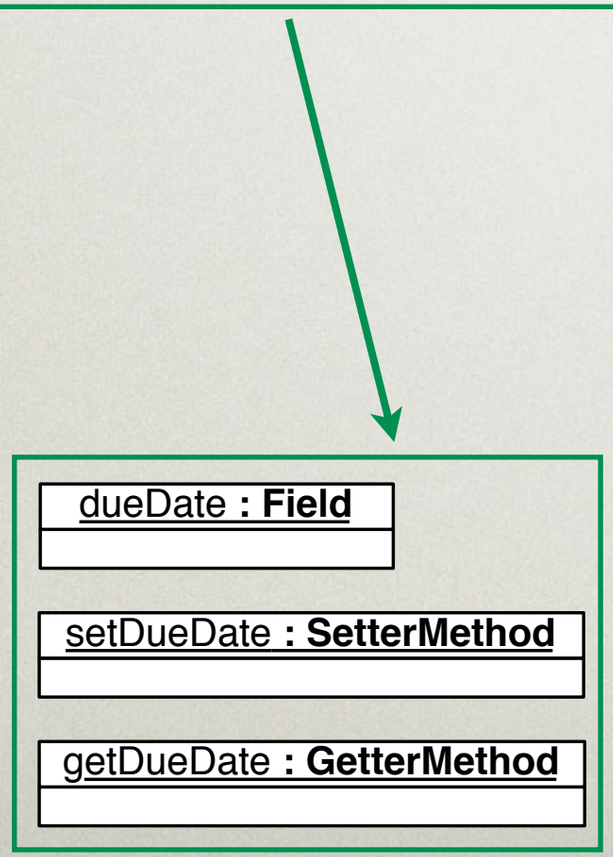


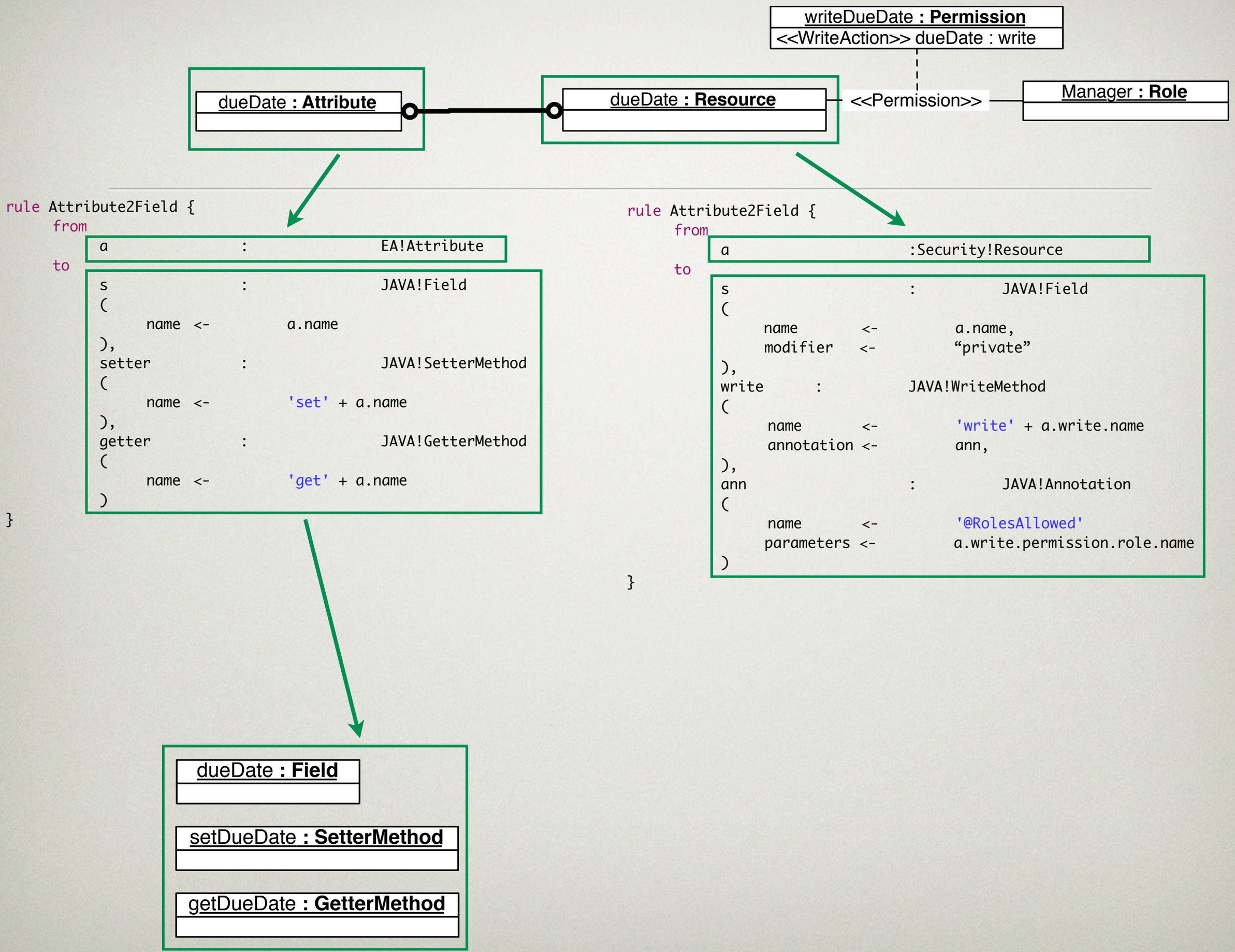

```
rule Attribute2Field {
	from
		a			:Security!Resource
	to
		s			:	JAVA!Field
		(
			name		<-	a.name,
			modifier	<-	“private”
		),
		write		:	JAVA!WriteMethod
		(
			name		<-	'write' + a.write.name
			annotation	<-	ann,
		),
		ann			:	JAVA!Annotation
		(
			name		<-	'@RolesAllowed'
			parameters	<-	a.write.permission.role.name
		)
}
```

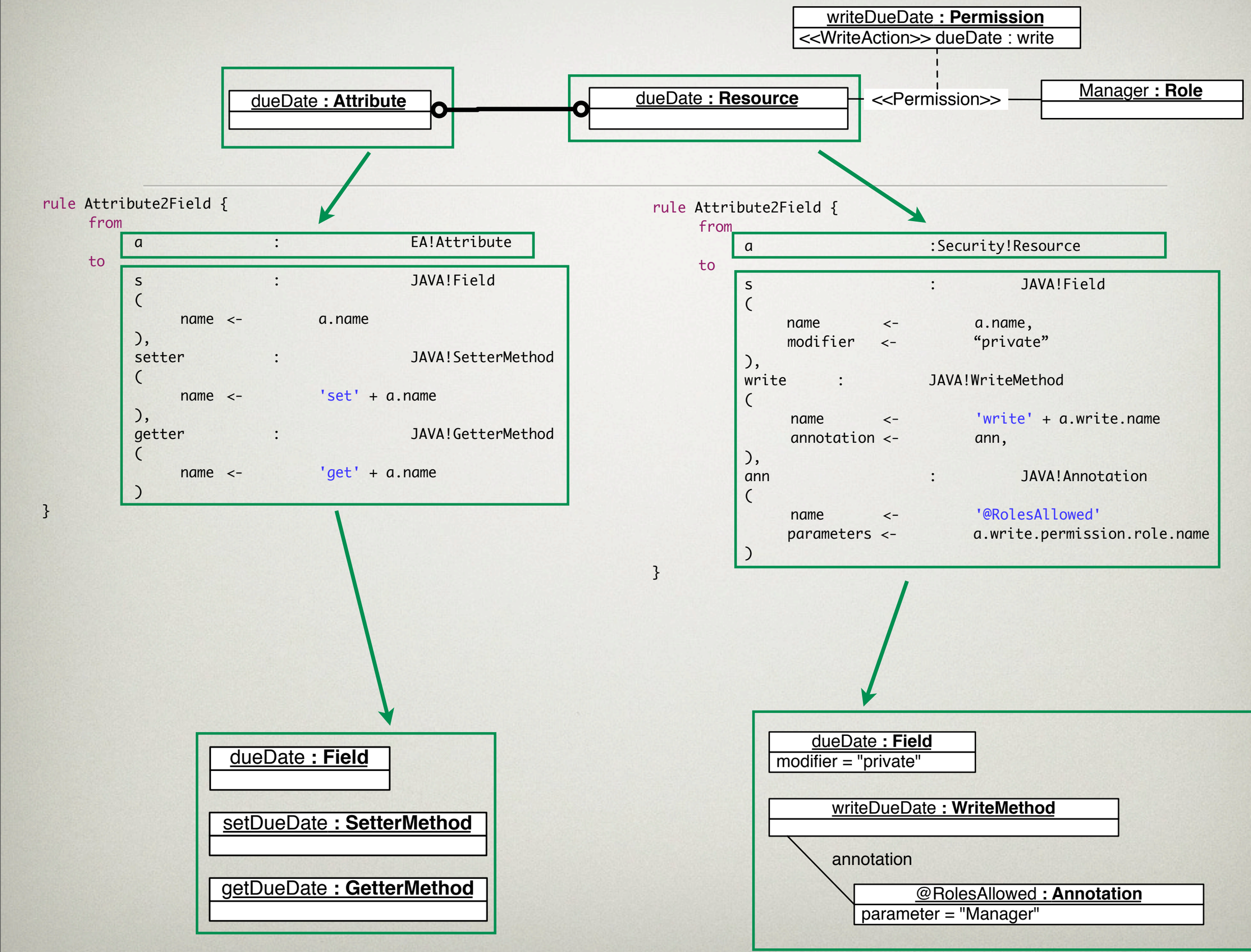


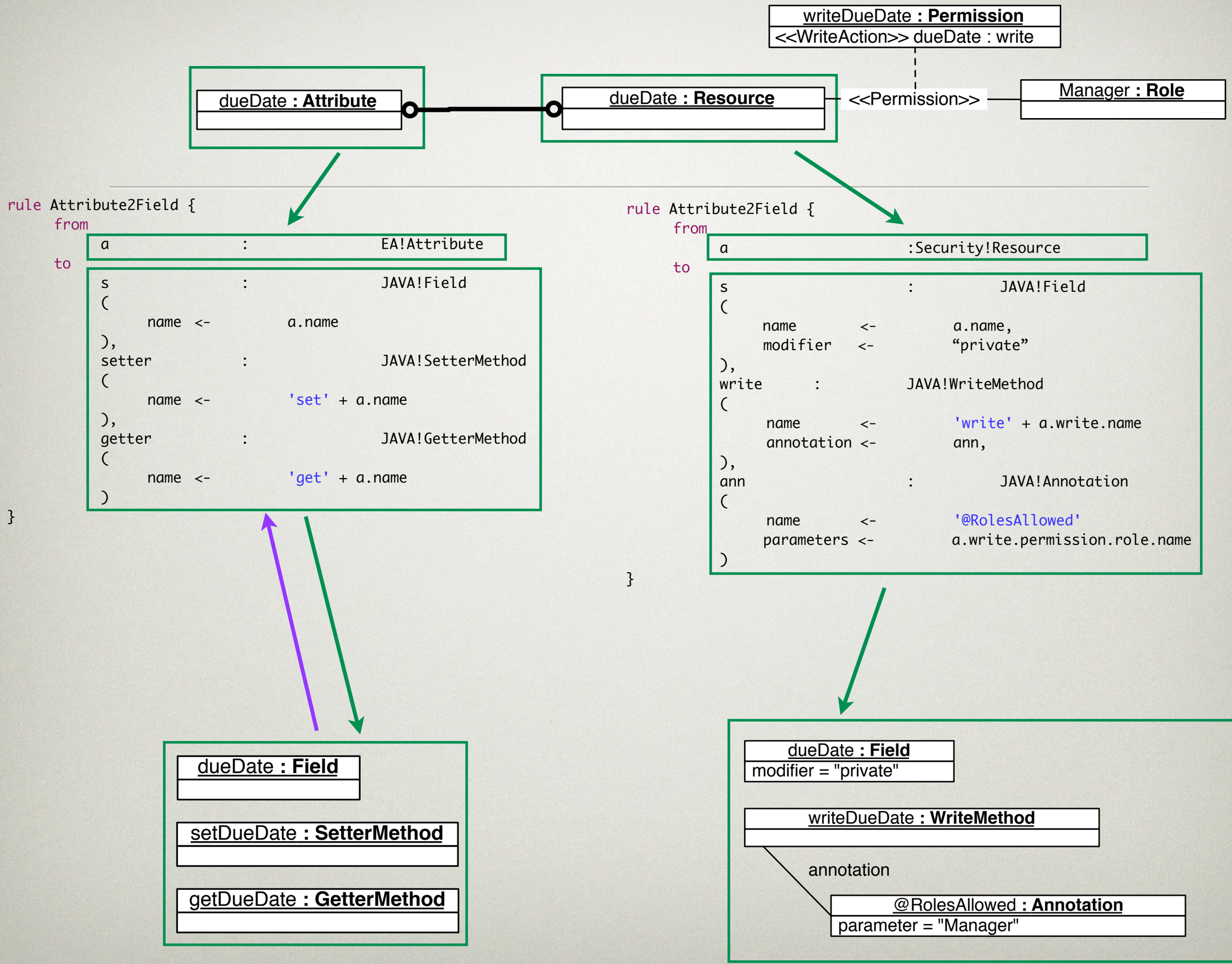

```
rule Attribute2Field {
    from
        a                   :                   EA!Attribute
    to
        s                   :                   JAVA!Field
        (
            name    <-          a.name
        ),
        setter              :                   JAVA!SetterMethod
        (
            name    <-          'set' + a.name
        ),
        getter              :                   JAVA!GetterMethod
        (
            name    <-          'get' + a.name
        )
}
```

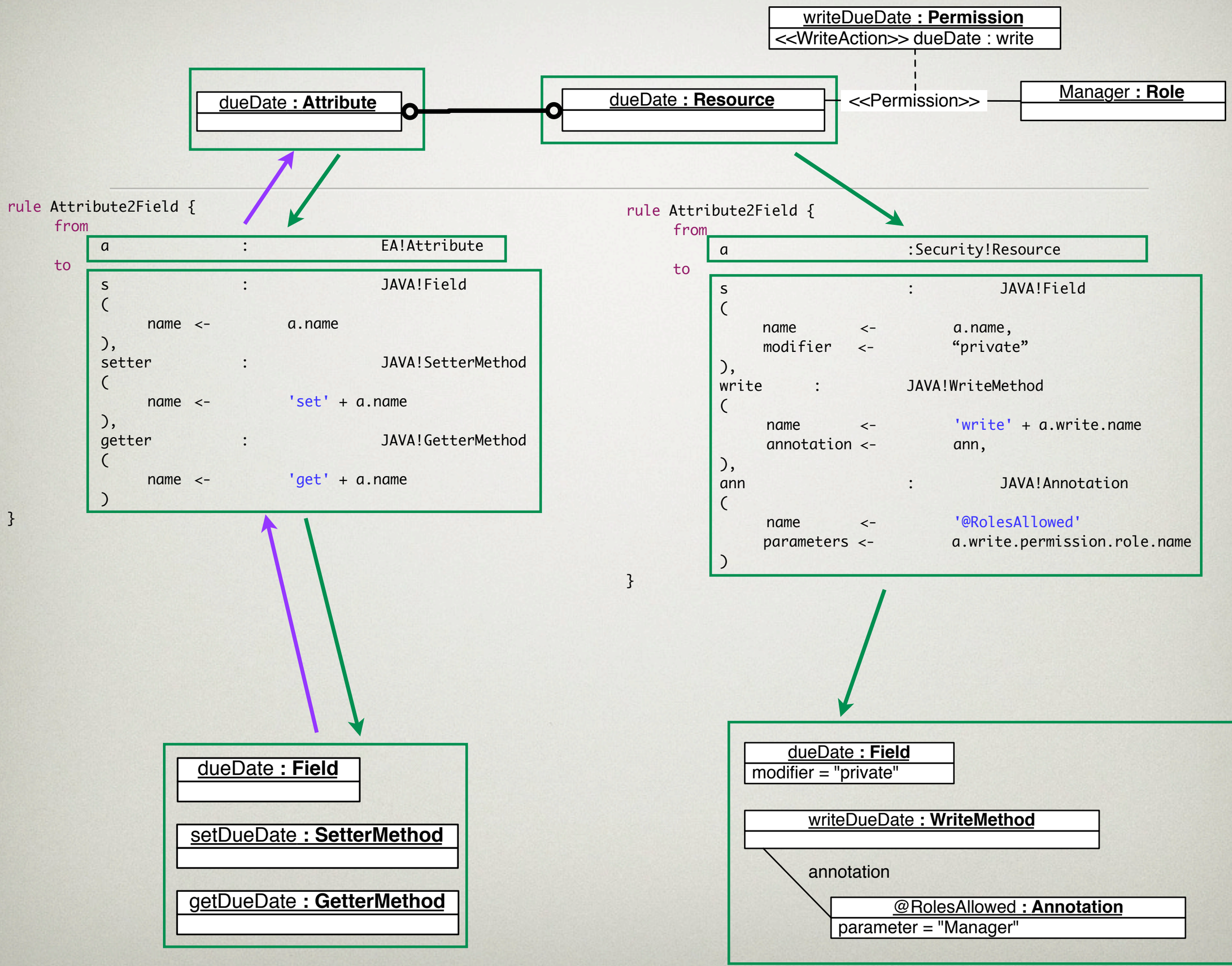
```
rule Attribute2Field {
    from
        a                       :Security!Resource
    to
        s                       :           JAVA!Field
        (
            name        <-          a.name,
            modifier    <-          “private”
        ),
        write       :           JAVA!WriteMethod
        (
            name        <-          'write' + a.write.name
            annotation  <-          ann,
        ),
        ann                     :           JAVA!Annotation
        (
            name        <-          '@RolesAllowed'
            parameters  <-          a.write.permission.role.name
        )
}
```

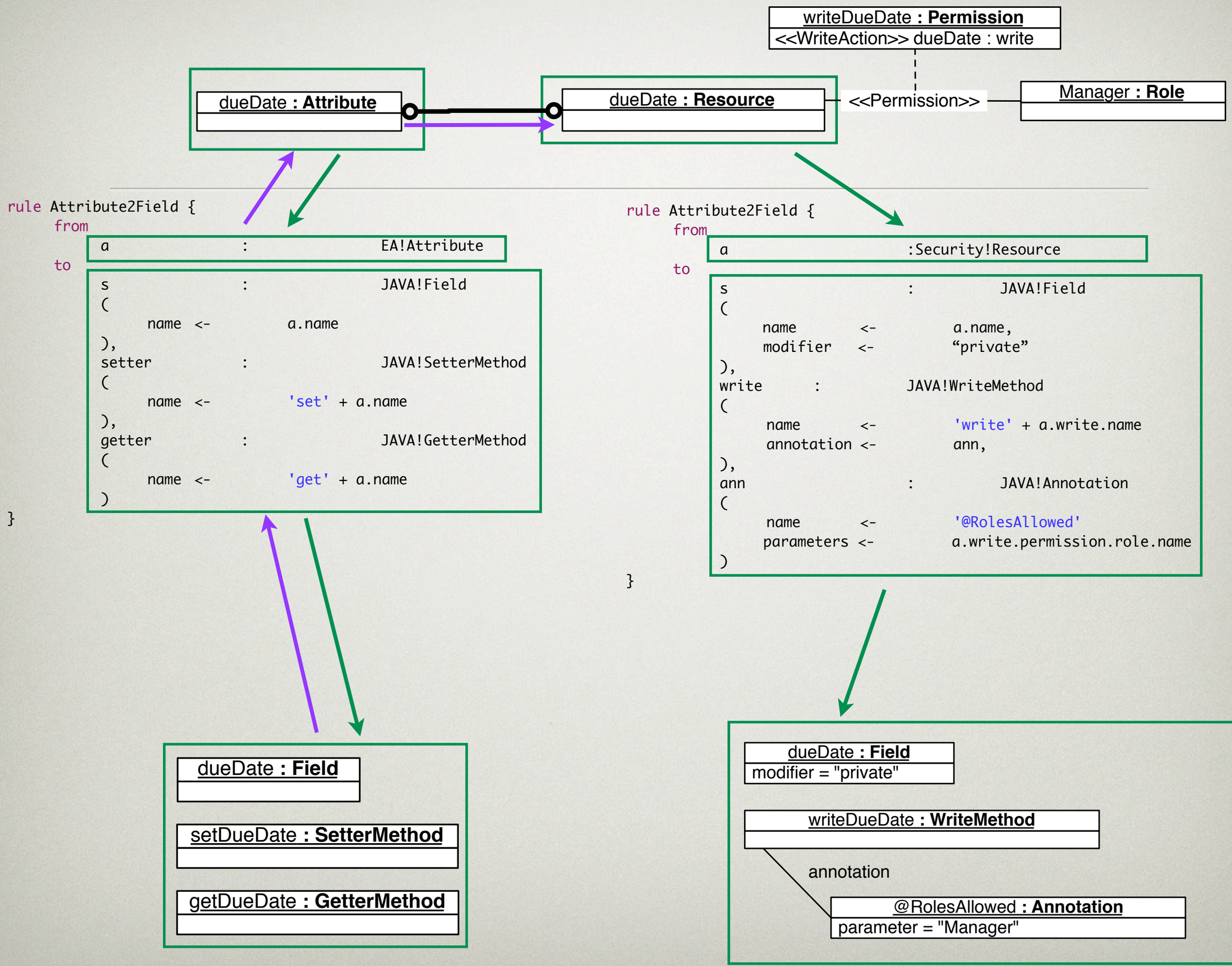


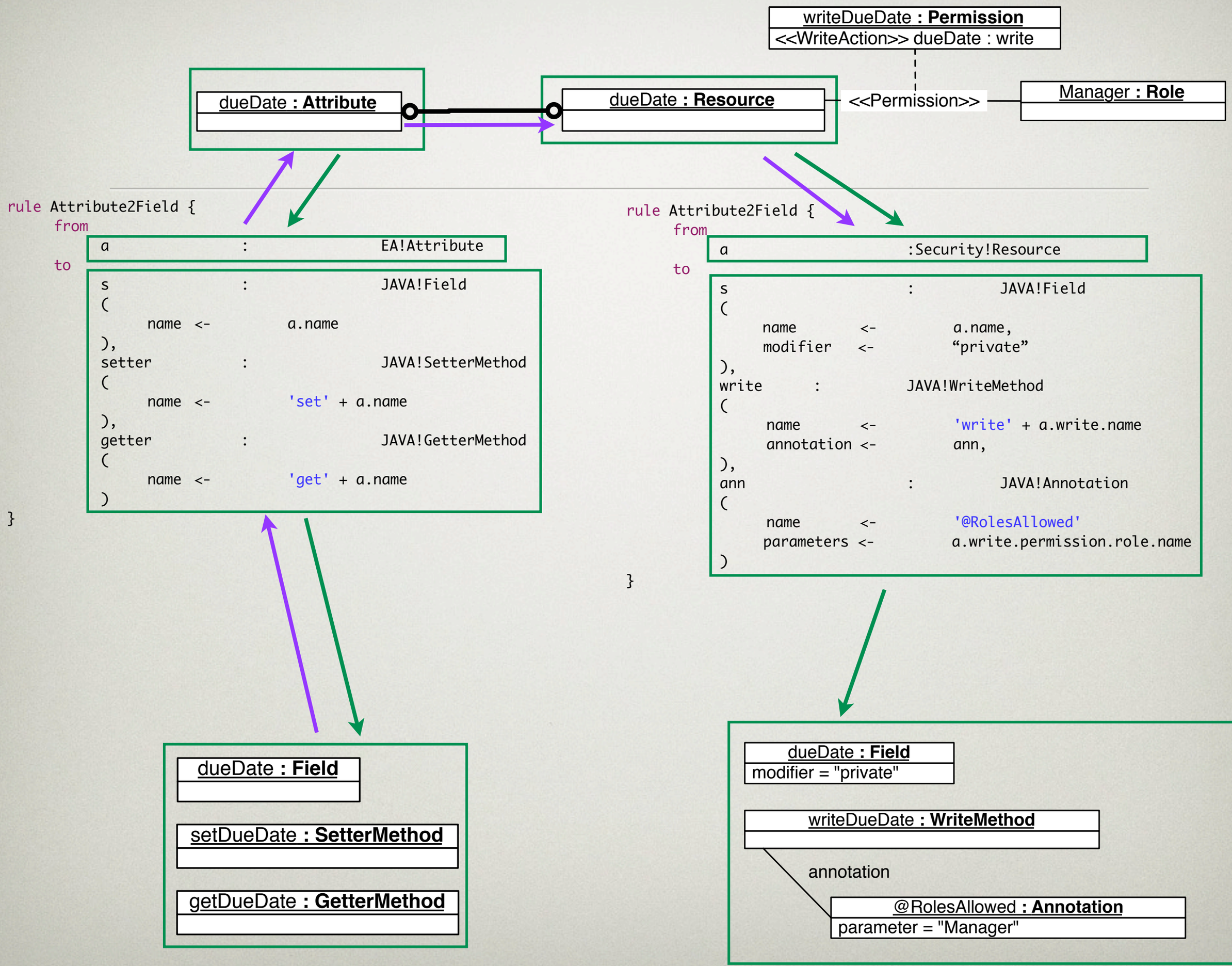

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod

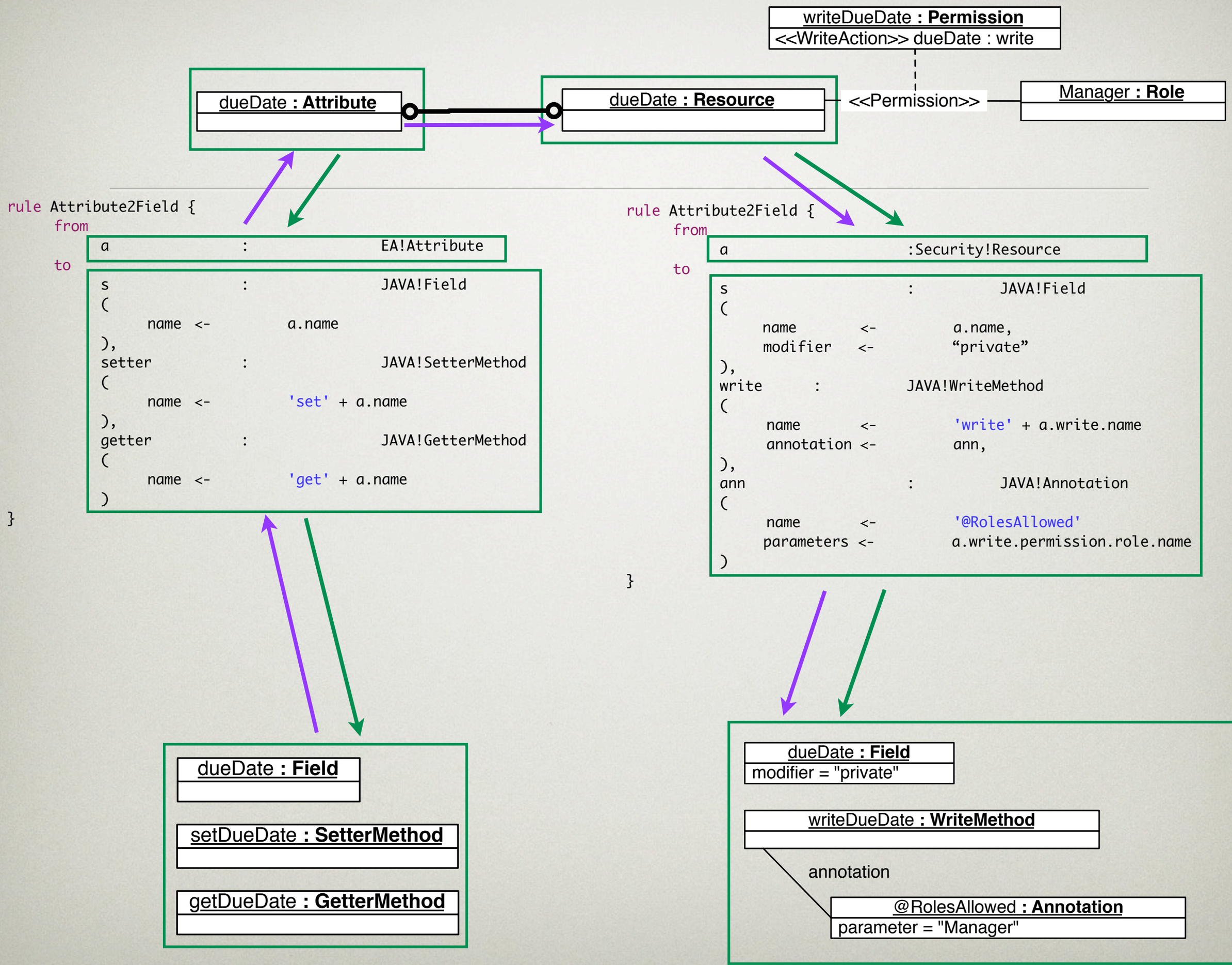

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"

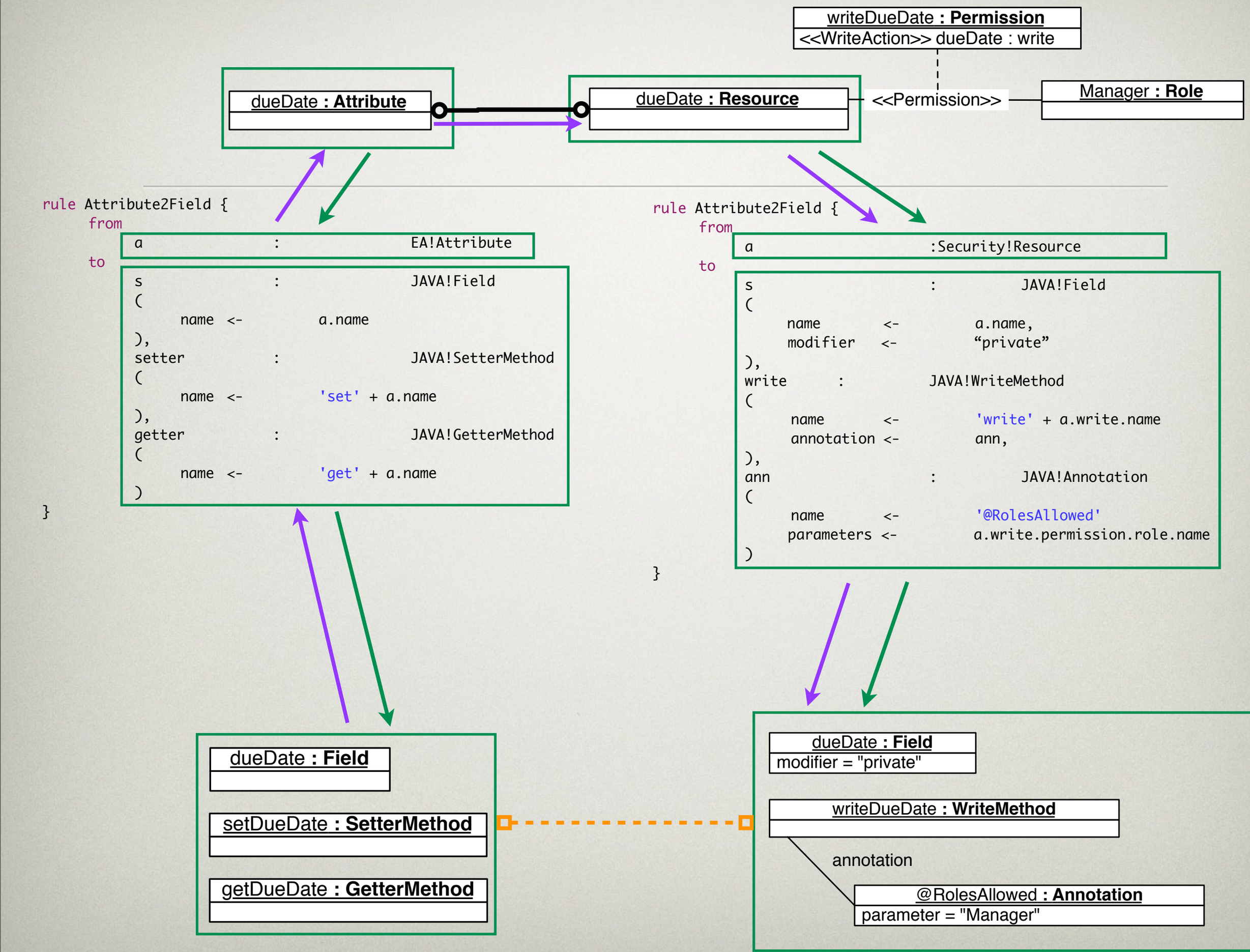

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"

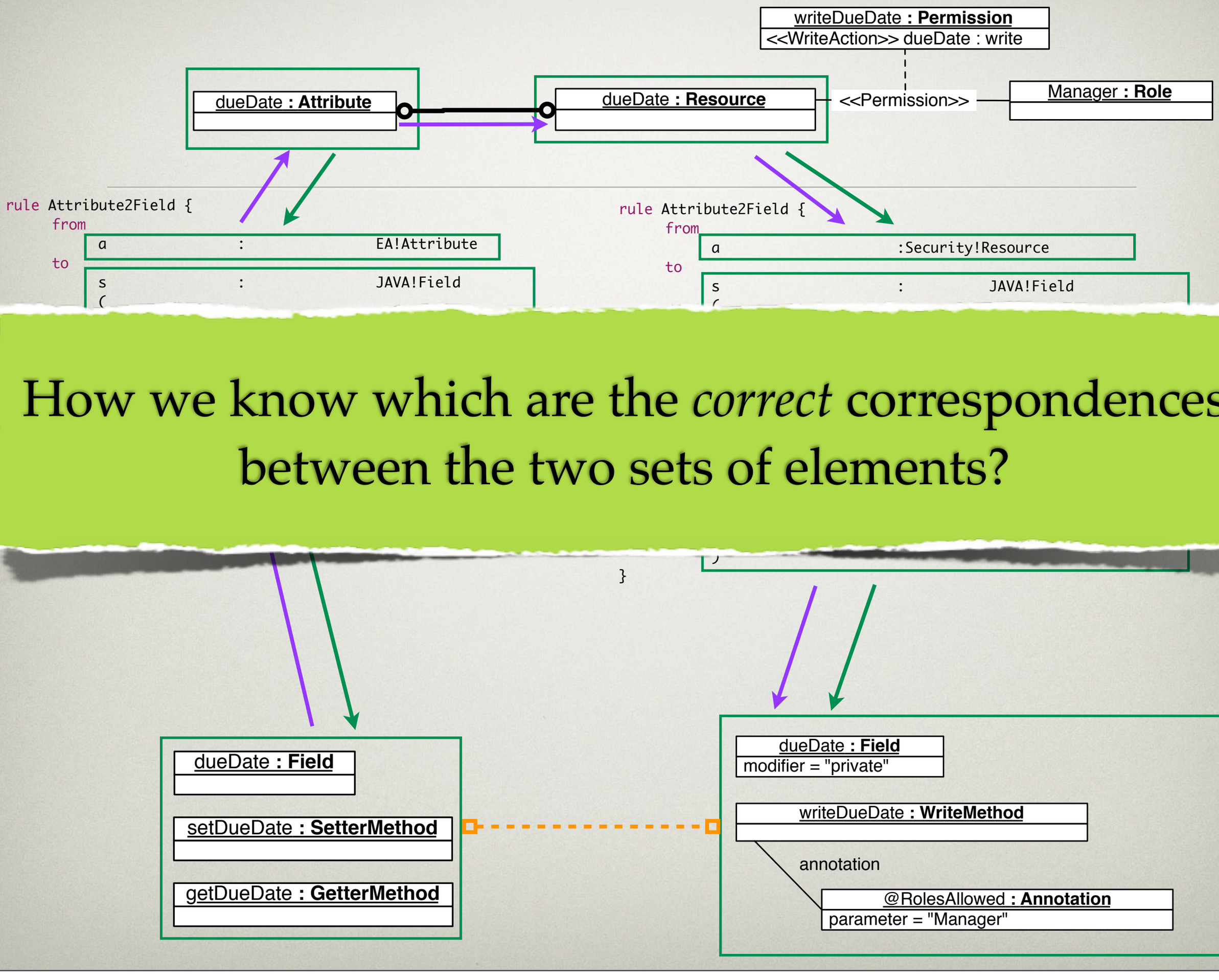

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"


writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"

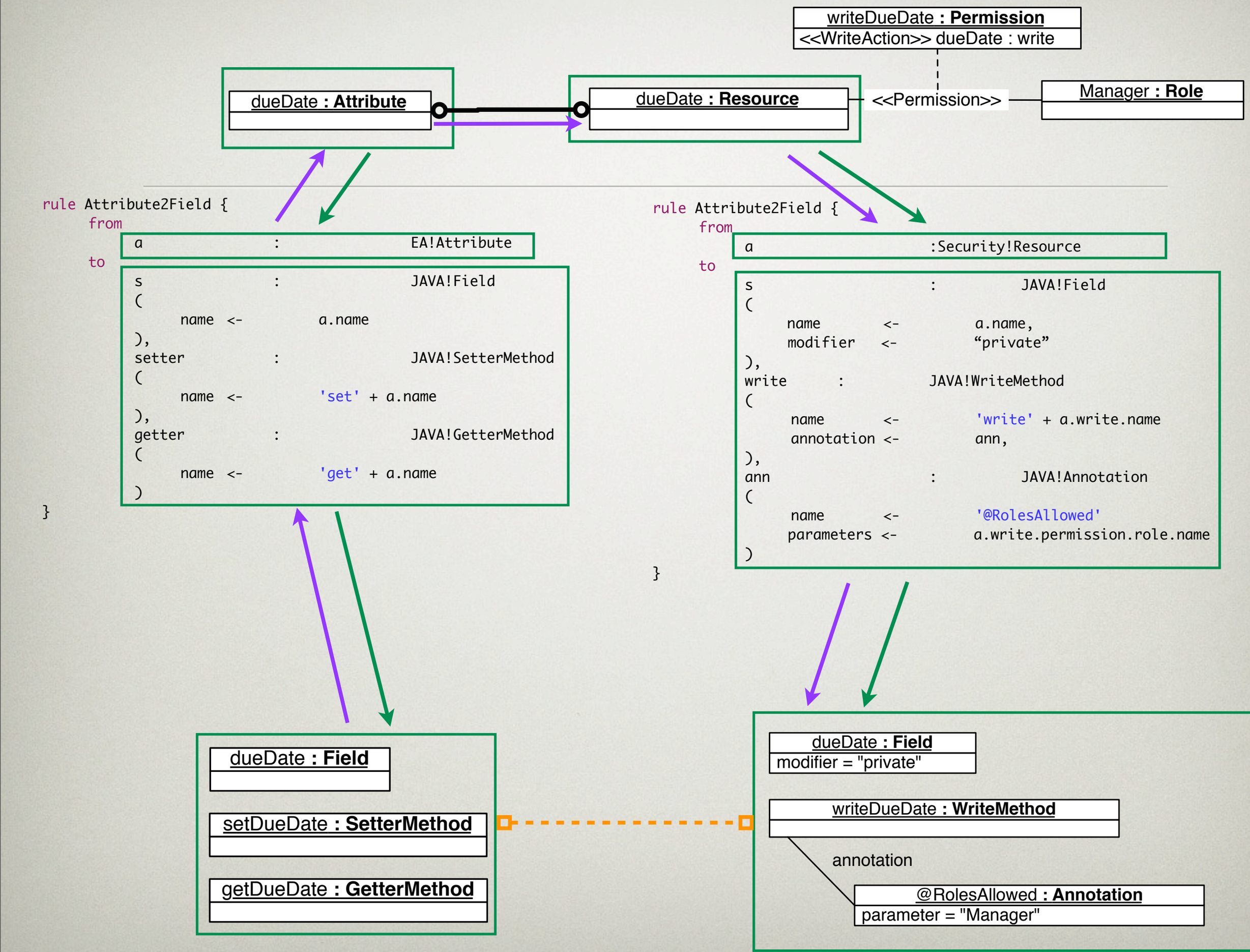

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"

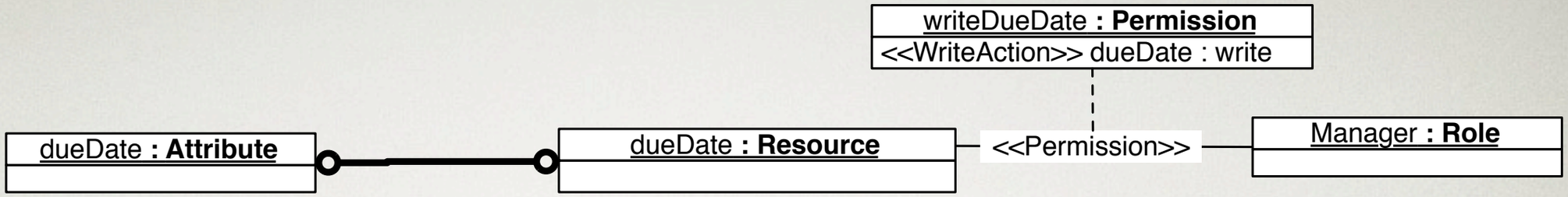

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"


writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"



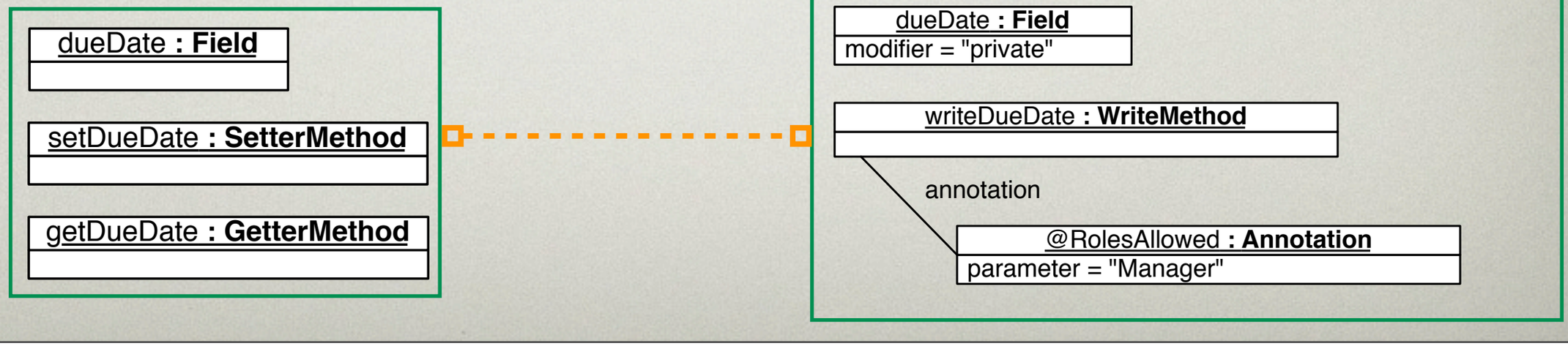
How we know which are the *correct* correspondences between the two sets of elements?

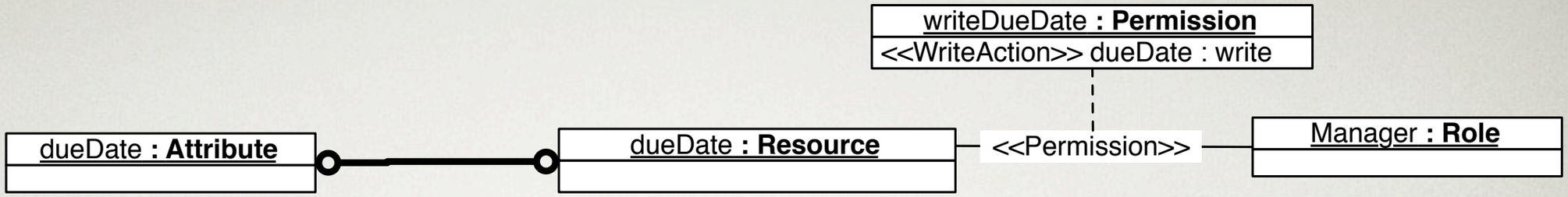

writeDueDate : Permission
<<WriteAction>> dueDate : write
dueDate : Attribute
dueDate : Resource
<<Permission>>
Manager : Role
rule Attribute2Field {
from
a : EA!Attribute
to
s : JAVA!Field
(
name <- a.name
),
setter : JAVA!SetterMethod
(
name <- 'set' + a.name
),
getter : JAVA!GetterMethod
(
name <- 'get' + a.name
)
}
rule Attribute2Field {
from
a :Security!Resource
to
s : JAVA!Field
(
name <- a.name,
modifier <- “private”
),
write : JAVA!WriteMethod
(
name <- 'write' + a.write.name
annotation <- ann,
),
ann : JAVA!Annotation
(
name <- '@RolesAllowed'
parameters <- a.write.permission.role.name
)
}
dueDate : Field
setDueDate : SetterMethod
getDueDate : GetterMethod
dueDate : Field
modifier = "private"
writeDueDate : WriteMethod
annotation
@RolesAllowed : Annotation
parameter = "Manager"




```
rule Attribute2Field {
	from
		a		:		EA!Attribute
	to
		s		:		JAVA!Field
		(
			name	<-	a.name
		),
		setter		:		JAVA!SetterMethod
		(
			name	<-	'set' + a.name
		),
		getter		:		JAVA!GetterMethod
		(
			name	<-	'get' + a.name
		)
}
```

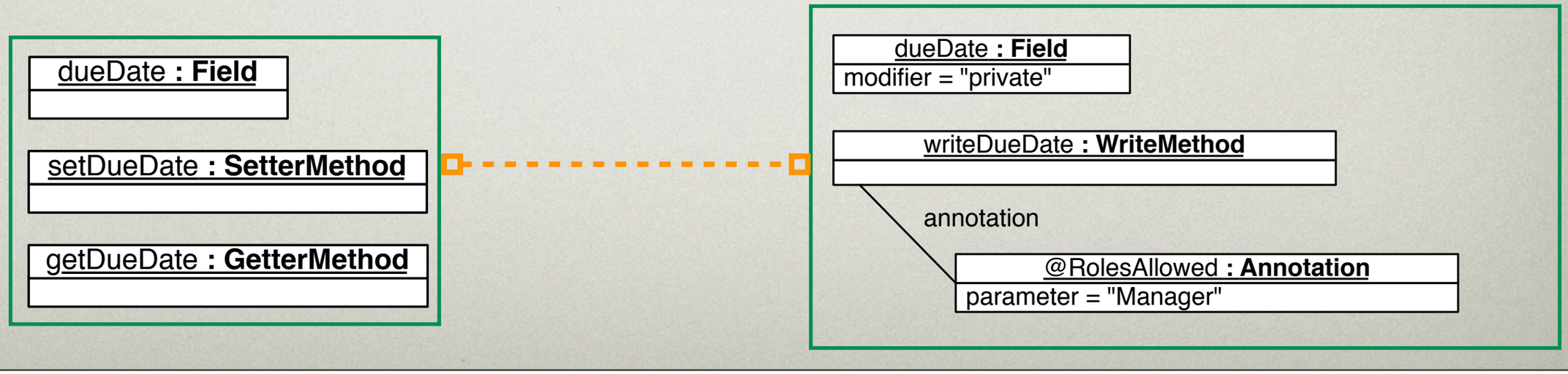
```
rule Attribute2Field {
	from
		a			:Security!Resource
	to
		s			:	JAVA!Field
		(
			name		<-	a.name,
			modifier	<-	“private”
		),
		write		:	JAVA!WriteMethod
		(
			name		<-	'write' + a.write.name
			annotation	<-	ann,
		),
		ann			:	JAVA!Annotation
		(
			name		<-	'@RolesAllowed'
			parameters	<-	a.write.permission.role.name
		)
}
```

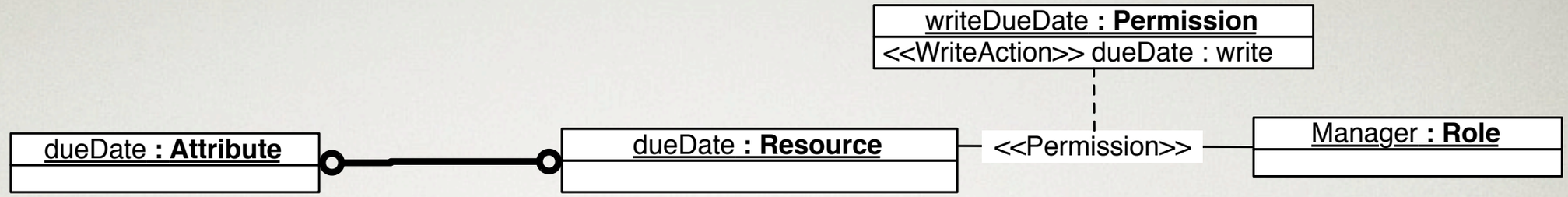





```
rule Attribute2Field {
	from
		a 		: 		EA!Attribute
	to
		s 		: 		JAVA!Field
		(
			name 	<- 		a.name
		),
		setter 		: 		JAVA!SetterMethod
		(
			name 	<- 		'set' + a.name
		),
		getter 		: 		JAVA!GetterMethod
		(
			name 	<- 		'get' + a.name
		)
}
```

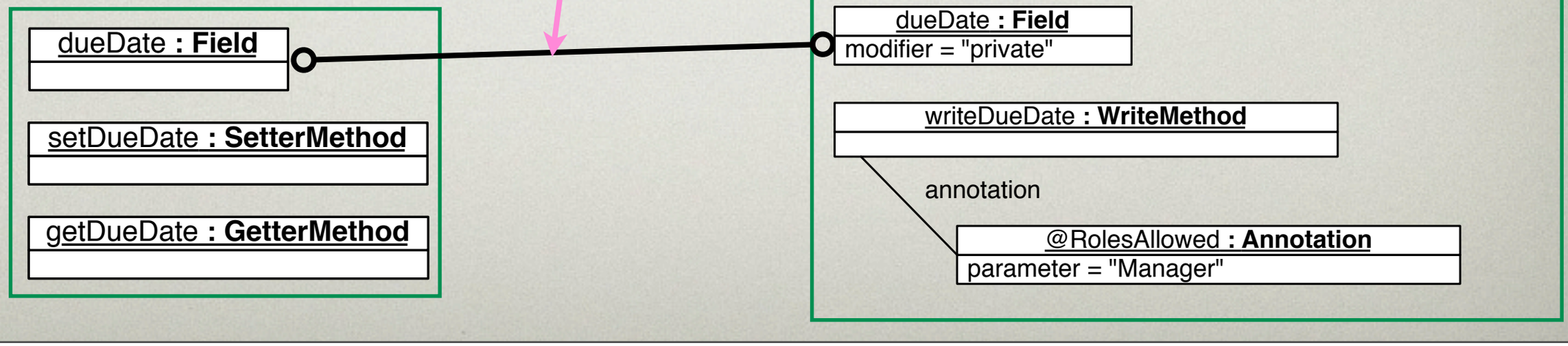
```
rule Attribute2Field {
	from
		a 			:Security!Resource
	to
		s 			: 		JAVA!Field
		(
			name 		<- 		a.name,
			modifier 	<- 		“private”
		),
		write 		: 		JAVA!WriteMethod
		(
			name 		<- 		'write' + a.write.name
			annotation 	<- 		ann,
		),
		ann 			: 		JAVA!Annotation
		(
			name 		<- 		'@RolesAllowed'
			parameters 	<- 		a.write.permission.role.name
		)
}
```

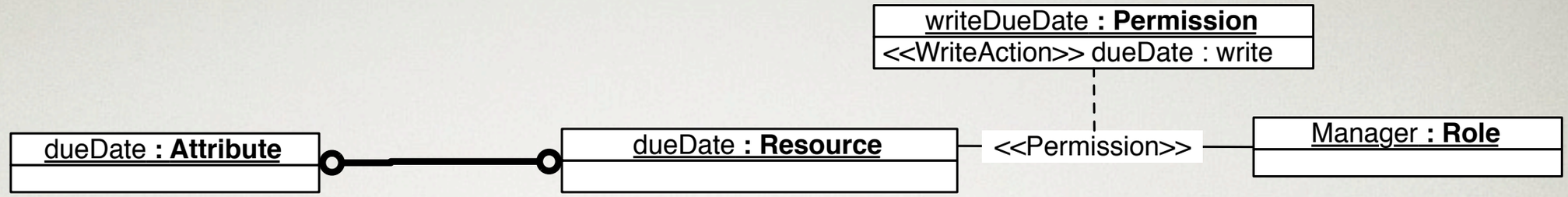





```
rule Attribute2Field {
	from
		a 		: 		EA!Attribute
	to
		s 		: 		JAVA!Field
		(
			name 	<- 		a.name
		),
		setter 		: 		JAVA!SetterMethod
		(
			name 	<- 		'set' + a.name
		),
		getter 		: 		JAVA!GetterMethod
		(
			name 	<- 		'get' + a.name
		)
}
```

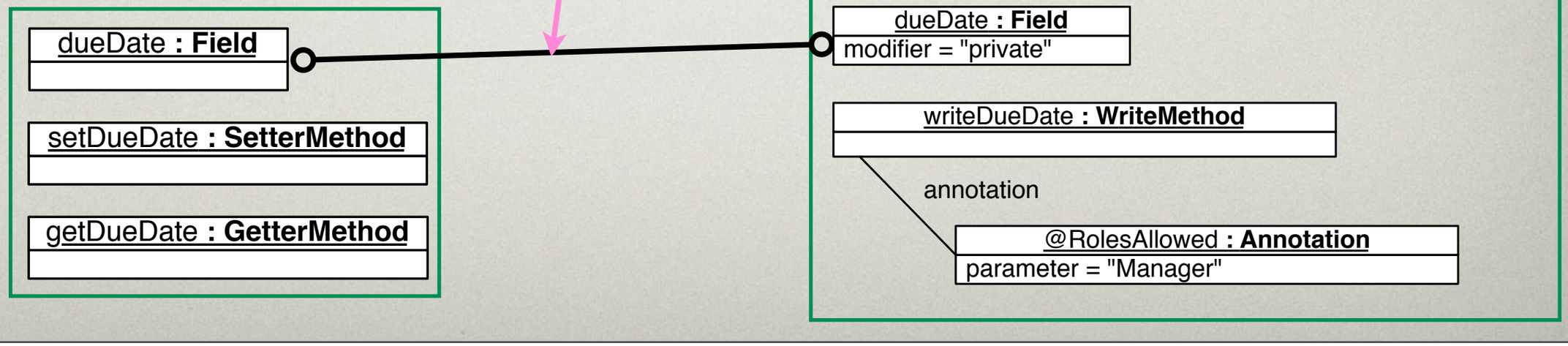
```
rule Attribute2Field {
	from
		a 			:Security!Resource
	to
		s 			: 		JAVA!Field
		(
			name 		<- 		a.name,
			modifier 	<- 		“private”
		),
		write 		: 		JAVA!WriteMethod
		(
			name 		<- 		'write' + a.write.name
			annotation 	<- 		ann,
		),
		ann 			: 		JAVA!Annotation
		(
			name 		<- 		'@RolesAllowed'
			parameters 	<- 		a.write.permission.role.name
		)
}
```

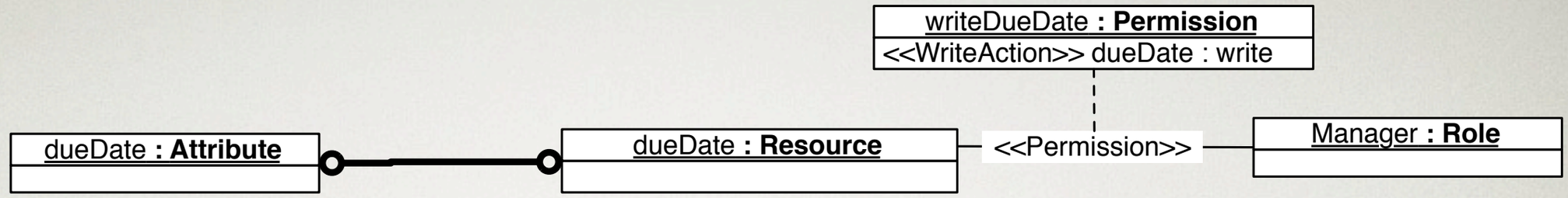





```
rule Attribute2Field {
    from
        a               :               EA!Attribute
    to
        s               :               JAVA!Field
        (
            name  <-        a.name
        ),
        setter          :               JAVA!SetterMethod
        (
            name  <-        'set' + a.name
        ),
        getter          :               JAVA!GetterMethod
        (
            name  <-        'get' + a.name
        )
}
```

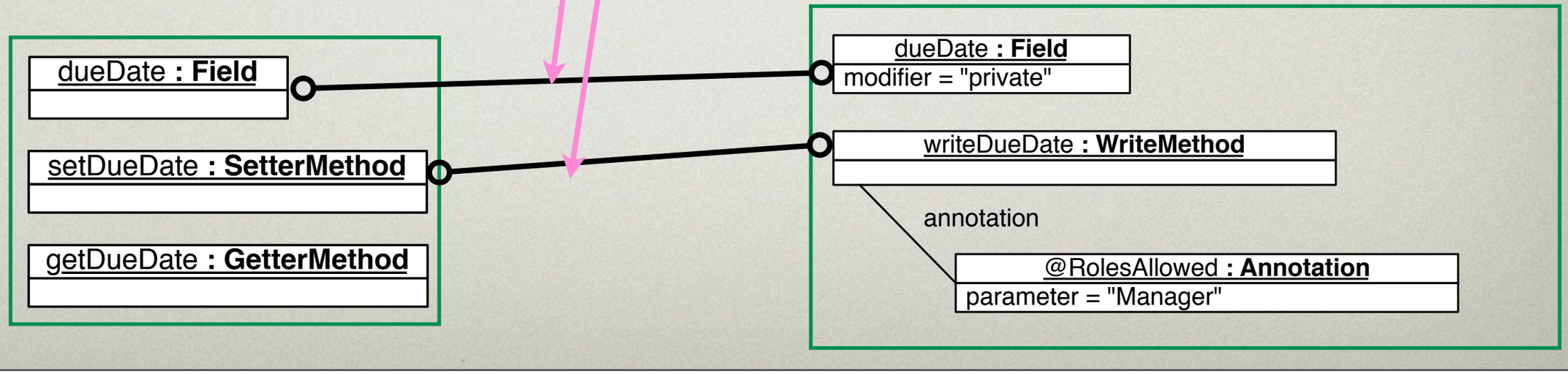
```
rule Attribute2Field {
    from
        a                       :Security!Resource
    to
        s                       :       JAVA!Field
        (
            name        <-      a.name,
            modifier    <-      “private”
        ),
        write       :           JAVA!WriteMethod
        (
            name        <-      'write' + a.write.name
            annotation  <-      ann,
        ),
        ann                     :       JAVA!Annotation
        (
            name        <-      '@RolesAllowed'
            parameters  <-      a.write.permission.role.name
        )
}
```

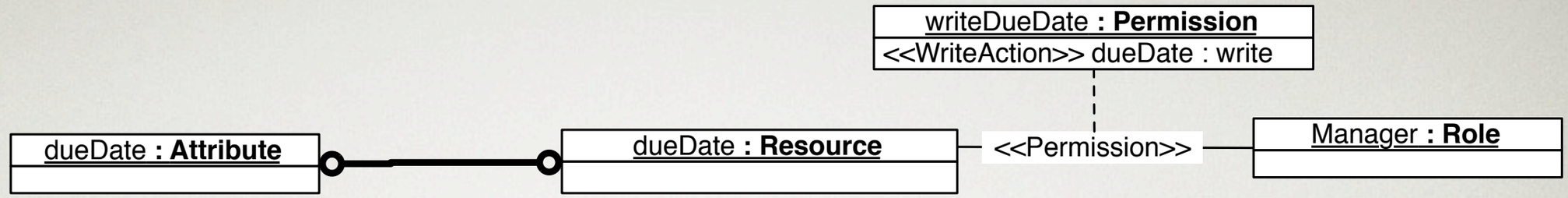





```
rule Attribute2Field {
    from
        a               :               EA!Attribute
    to
        s               :               JAVA!Field
        (
            name  <-        a.name
        ),
        setter          :               JAVA!SetterMethod
        (
            name  <-        'set' + a.name
        ),
        getter          :               JAVA!GetterMethod
        (
            name  <-        'get' + a.name
        )
}
```

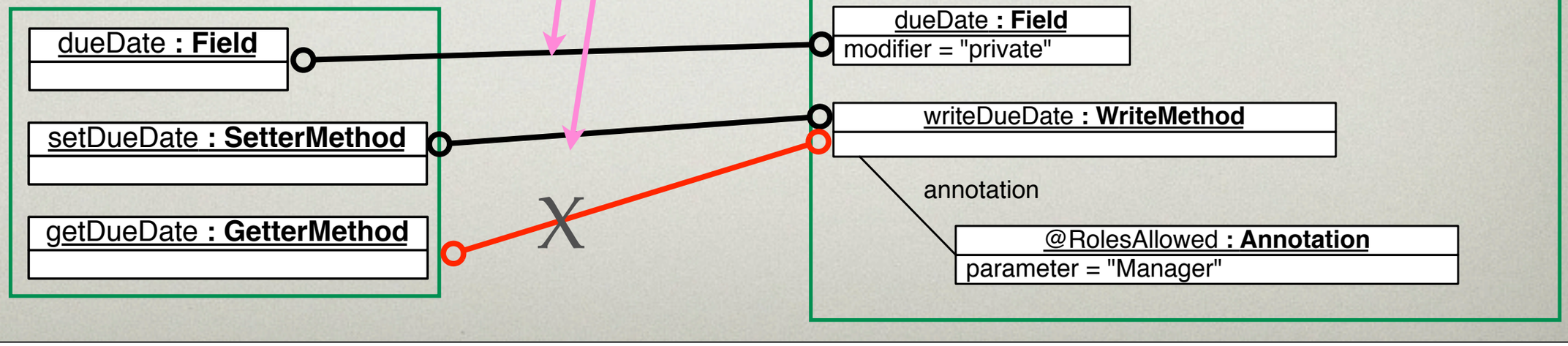
```
rule Attribute2Field {
    from
        a                       :Security!Resource
    to
        s                       :       JAVA!Field
        (
            name        <-      a.name,
            modifier    <-      “private”
        ),
        write       :           JAVA!WriteMethod
        (
            name        <-      'write' + a.write.name
            annotation  <-      ann,
        ),
        ann                     :       JAVA!Annotation
        (
            name        <-      '@RolesAllowed'
            parameters  <-      a.write.permission.role.name
        )
}
```



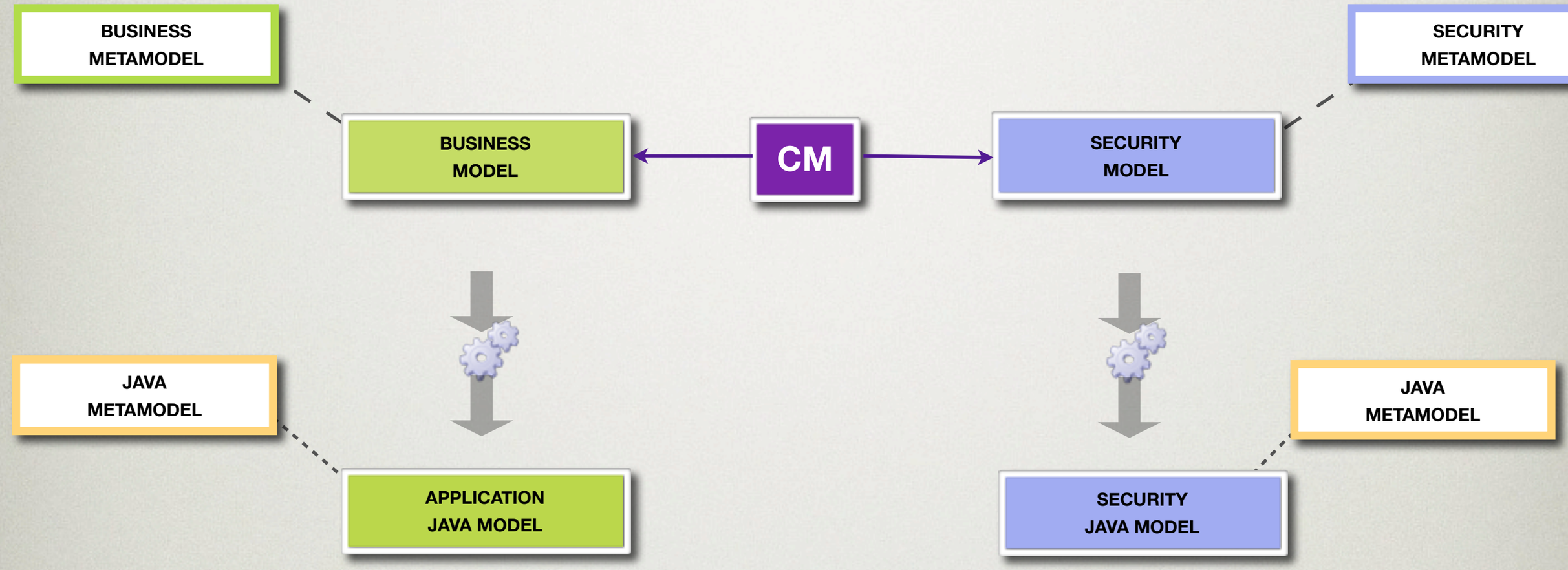



```
rule Attribute2Field {
    from
        a               :               EA!Attribute
    to
        s               :               JAVA!Field
        (
            name  <-        a.name
        ),
        setter          :               JAVA!SetterMethod
        (
            name  <-        'set' + a.name
        ),
        getter          :               JAVA!GetterMethod
        (
            name  <-        'get' + a.name
        )
}
```

```
rule Attribute2Field {
    from
        a                       :Security!Resource
    to
        s                       :       JAVA!Field
        (
            name        <-      a.name,
            modifier    <-      “private”
        ),
        write       :           JAVA!WriteMethod
        (
            name        <-      'write' + a.write.name
            annotation  <-      ann,
        ),
        ann                     :       JAVA!Annotation
        (
            name        <-      '@RolesAllowed'
            parameters  <-      a.write.permission.role.name
        )
}
```



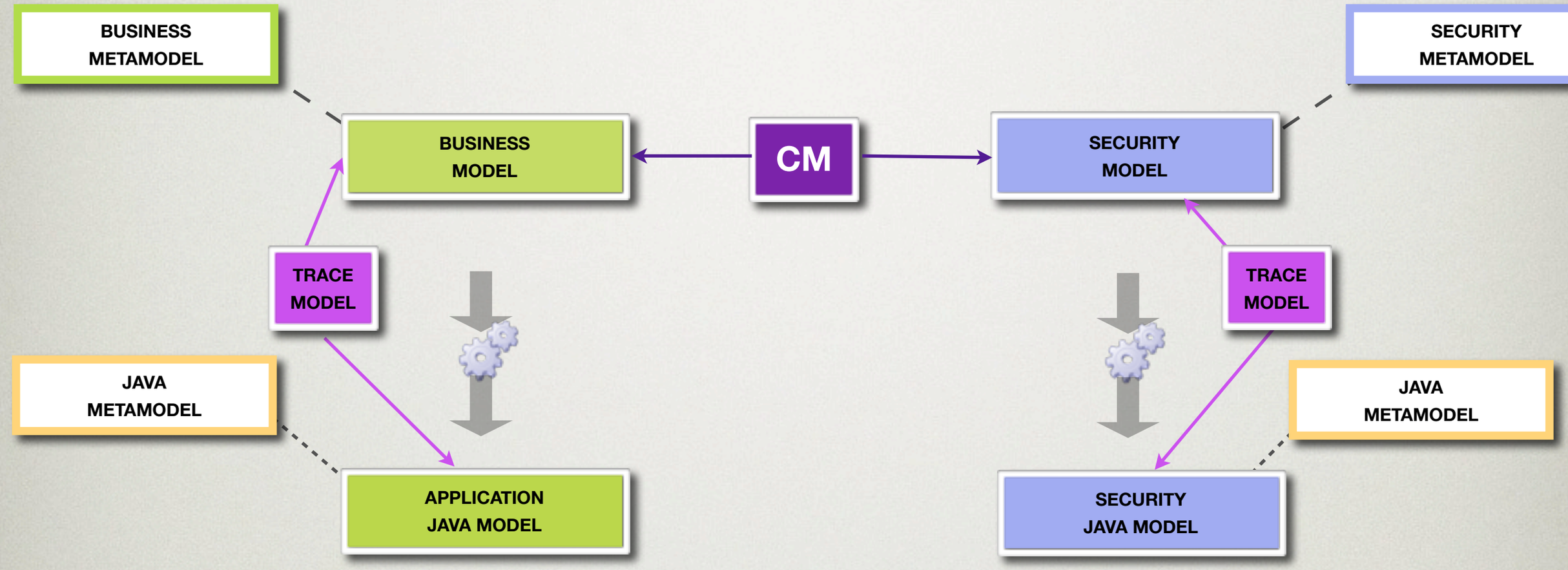
Correspondence Model Derivation



[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)

[Yie et Al] Advanced traceability for ATL. MtATL (2009)

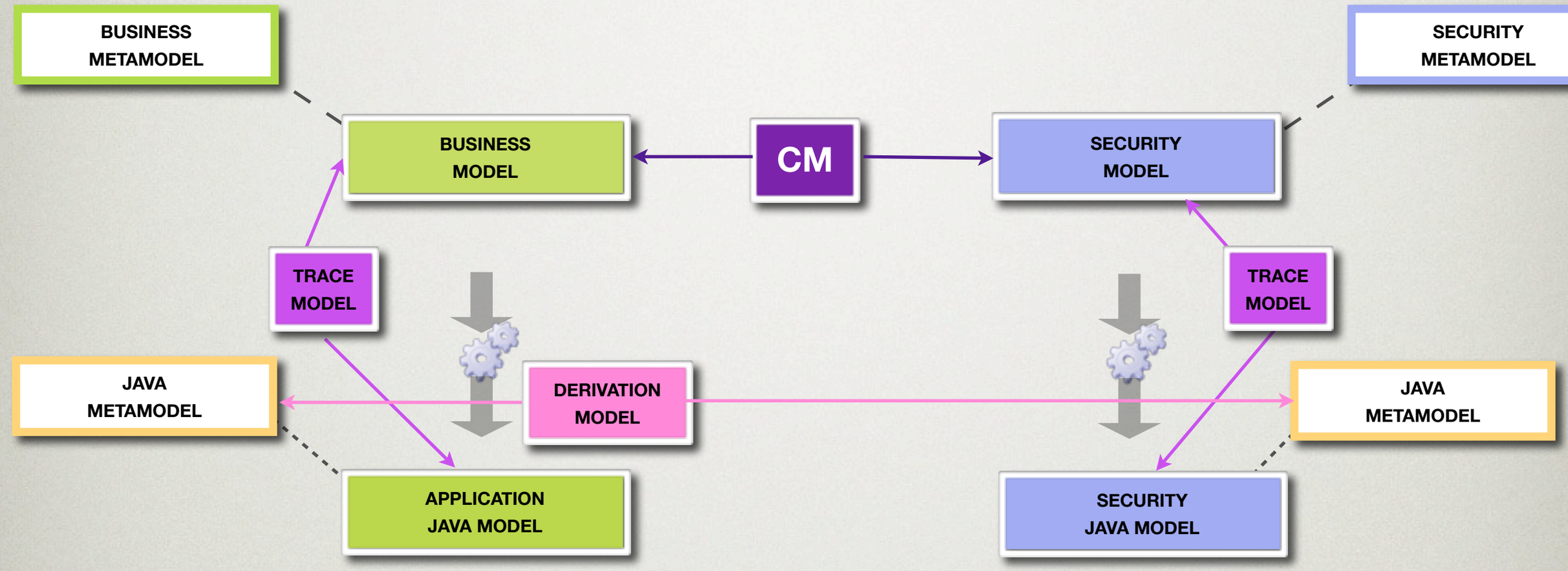
Correspondence Model Derivation



With *trace models* it is possible to detect which target elements are generated from a pair of correspondent source elements

[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)
[Yie et Al] Advanced traceability for ATL. MtATL (2009)

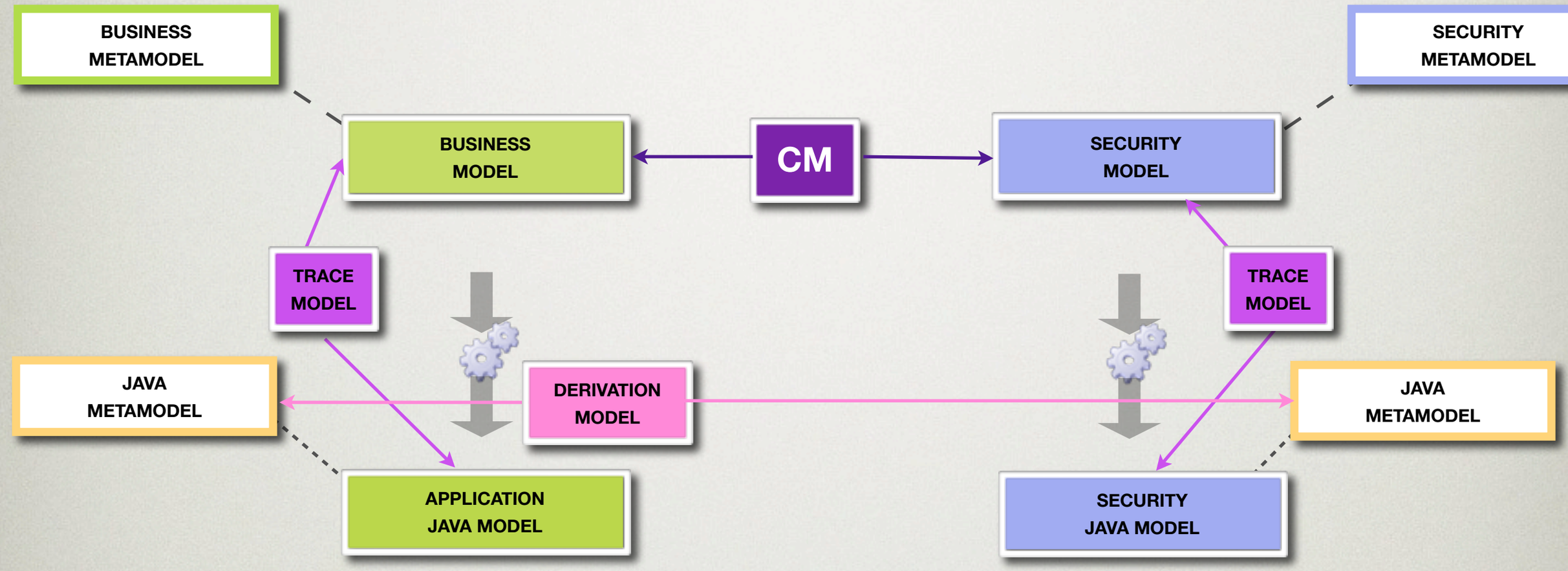
Correspondence Model Derivation



The *Derivation Model* specifies constraints that allow or reject correspondence relationships

[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)
[Yie et Al] Advanced traceability for ATL. MtATL (2009)

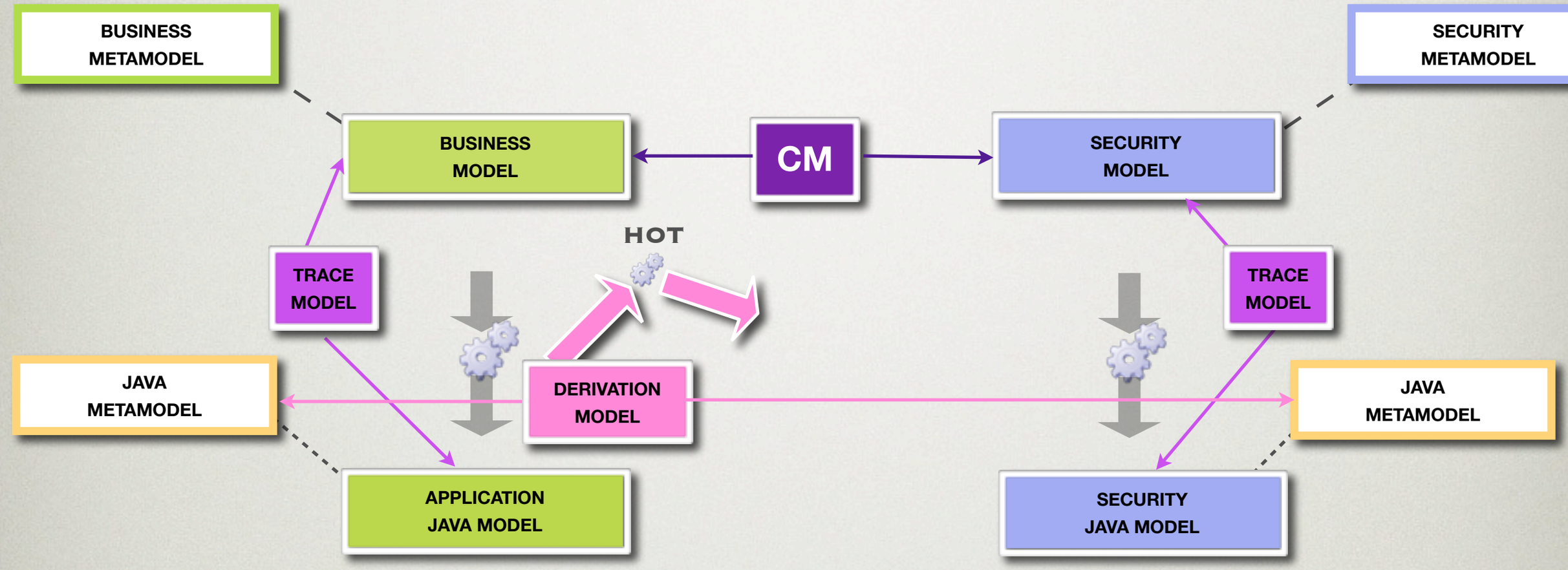
Correspondence Model Derivation



[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)

[Yie et Al] Advanced traceability for ATL. MtATL (2009)

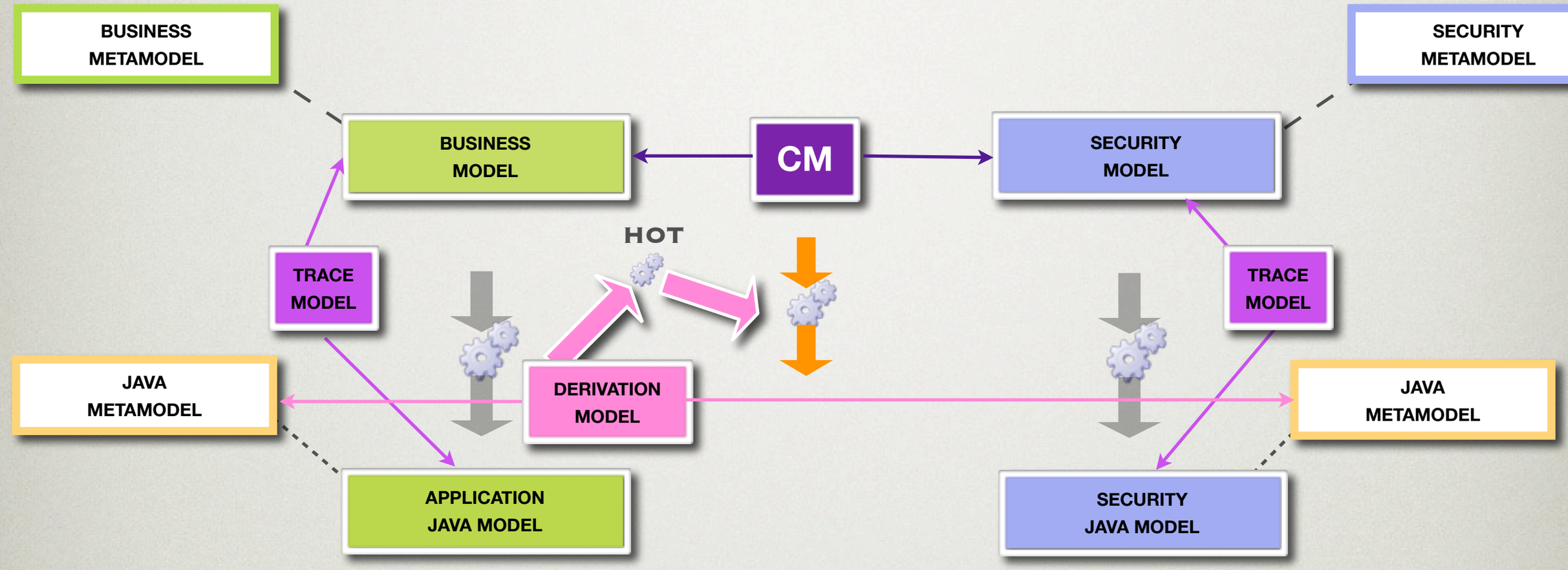
Correspondence Model Derivation



It is necessary to apply a High Order Transformation (HOT) to use the Derivation Model constraints at model level

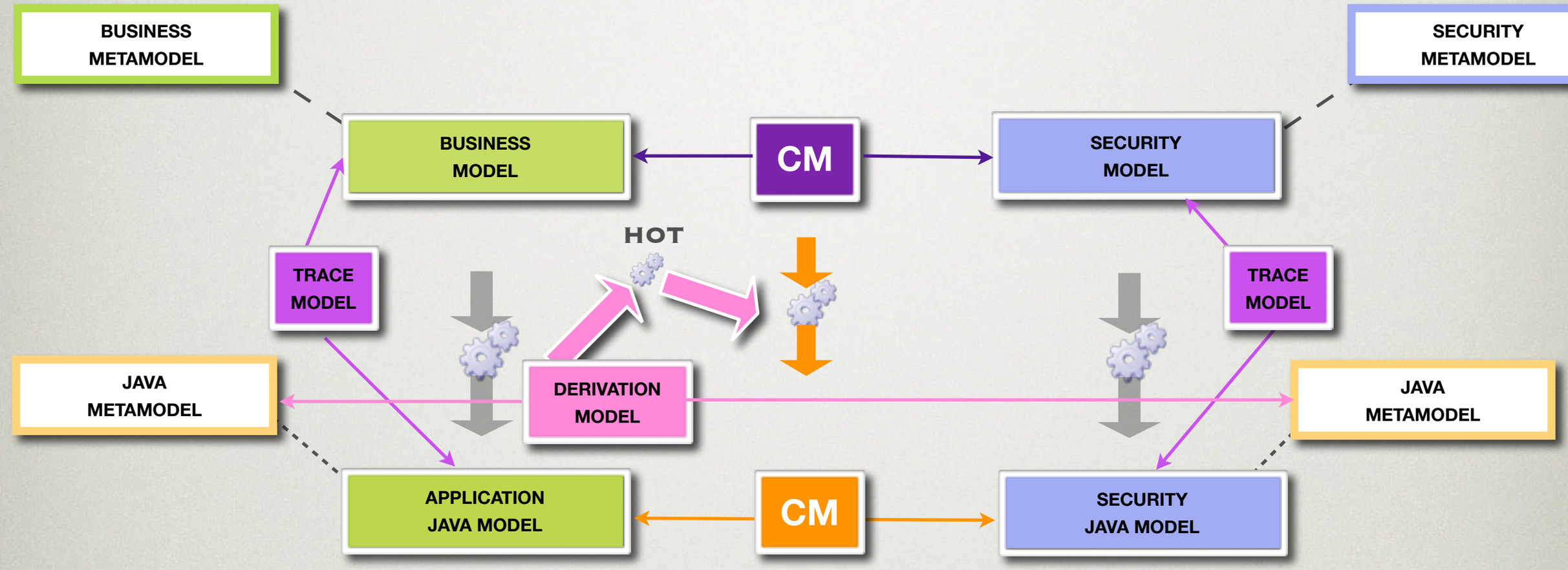
[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)
[Yie et Al] Advanced traceability for ATL. MtATL (2009)

Correspondence Model Derivation



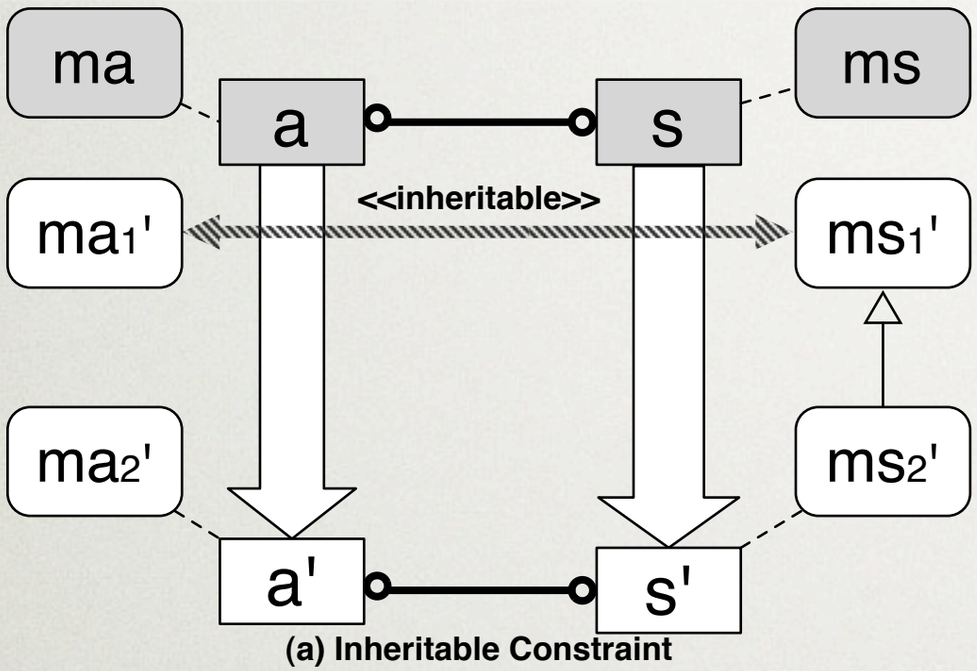
[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)
[Yie et Al] Advanced traceability for ATL. MtATL (2009)

Correspondence Model Derivation

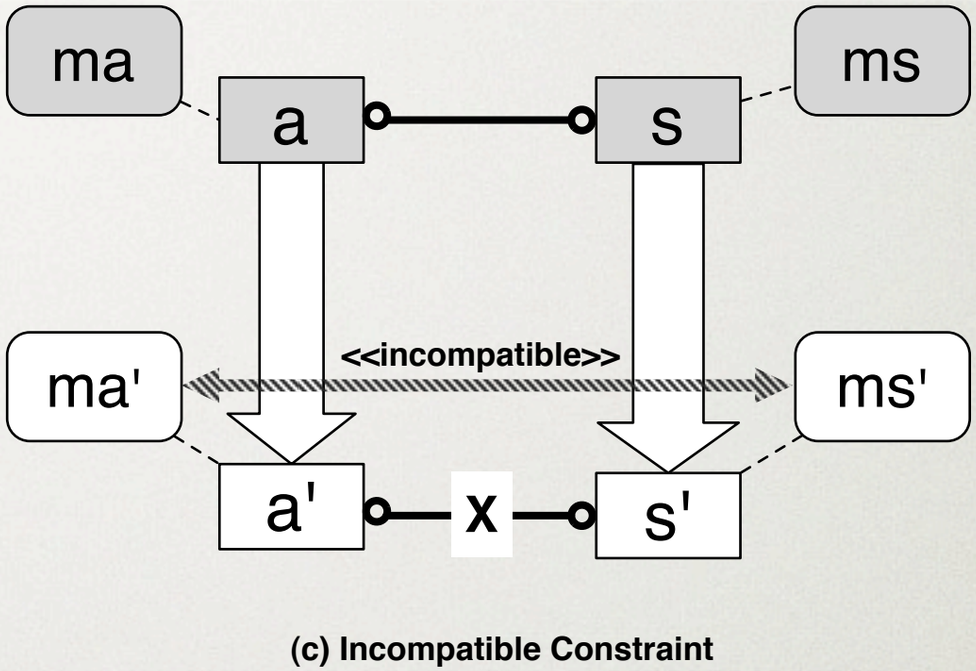


[Aizenbud-Reshef] Model traceability. IBM Systems Journal (2006)
[Yie et Al] Advanced traceability for ATL. MtATL (2009)

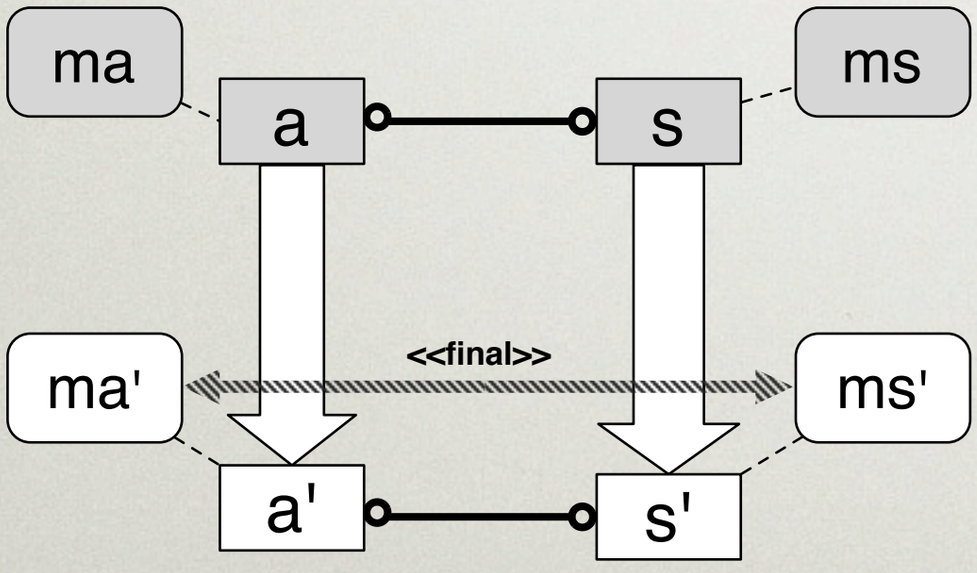
Derivation Model



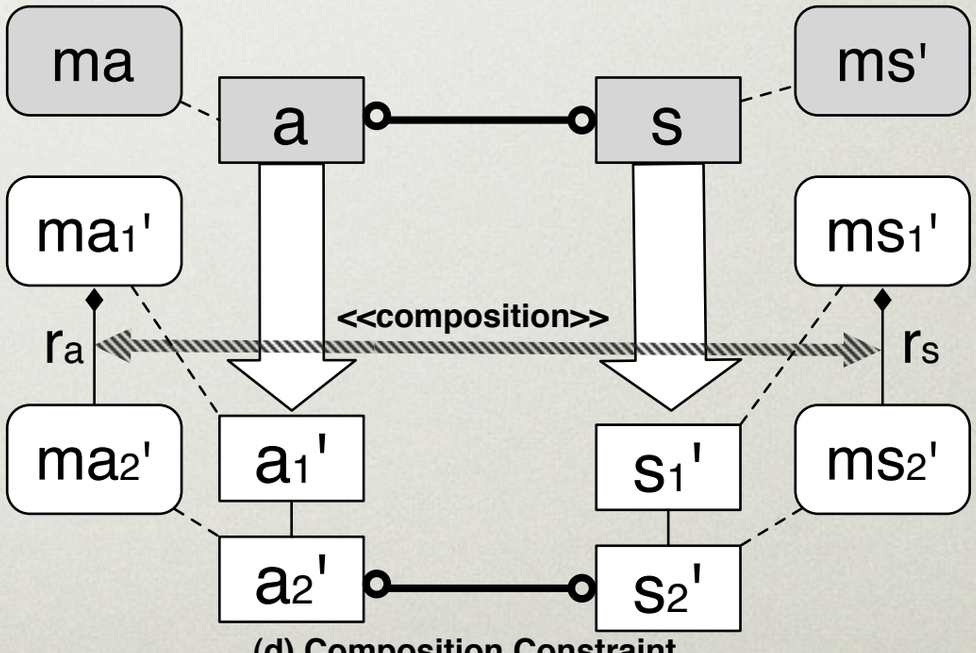
(a) Inheritable Constraint



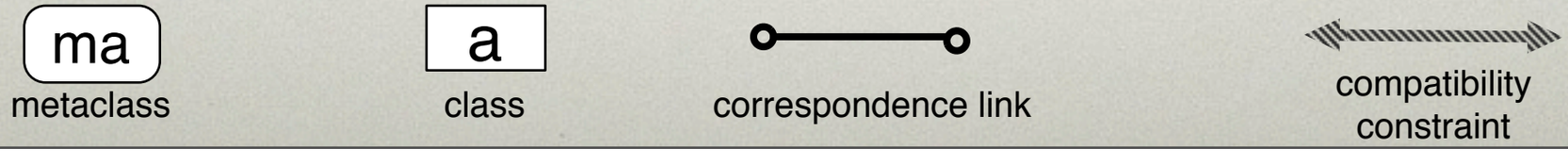
(c) Incompatible Constraint

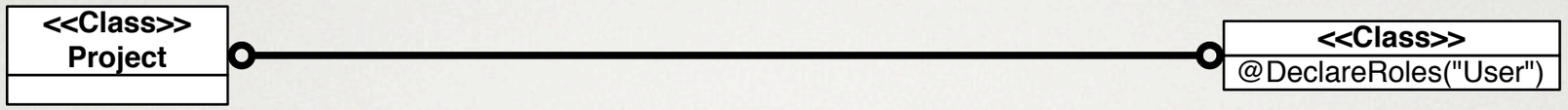


(b) Final Constraint



(d) Composition Constraint

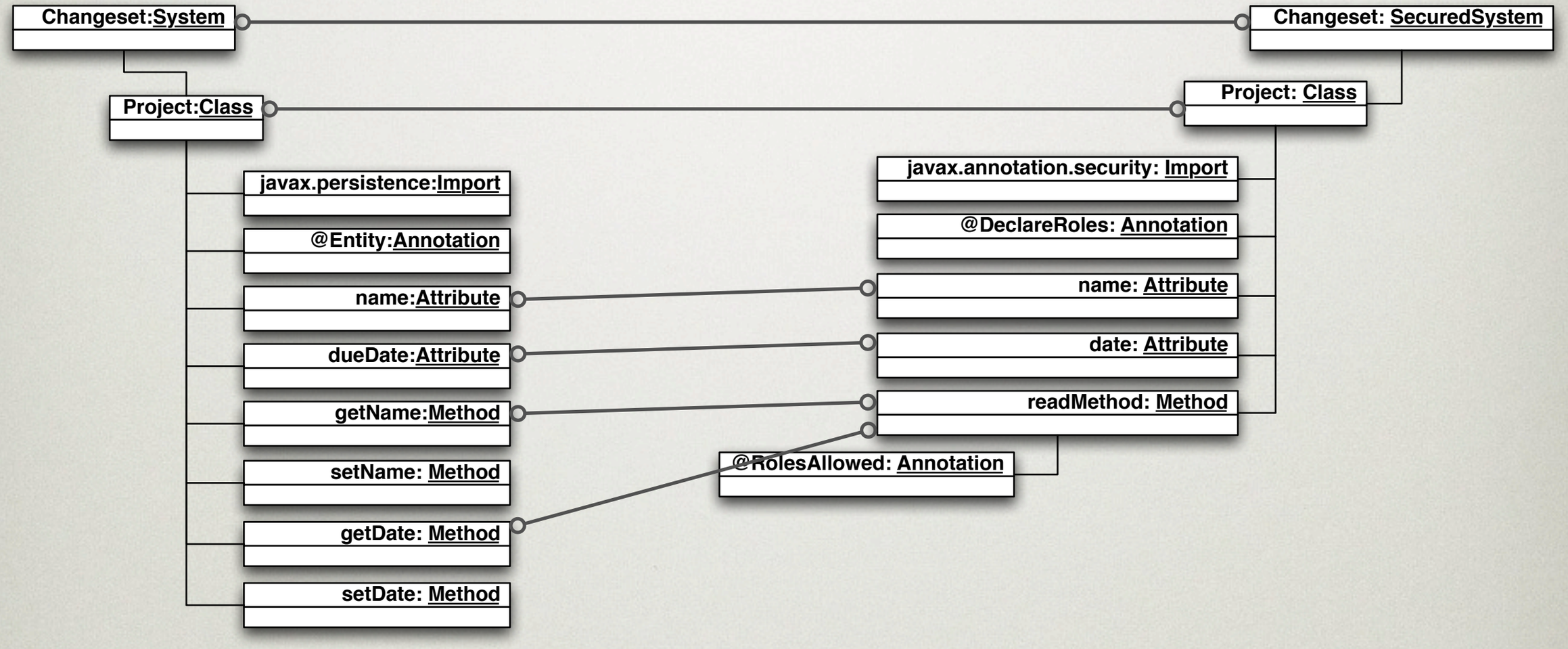


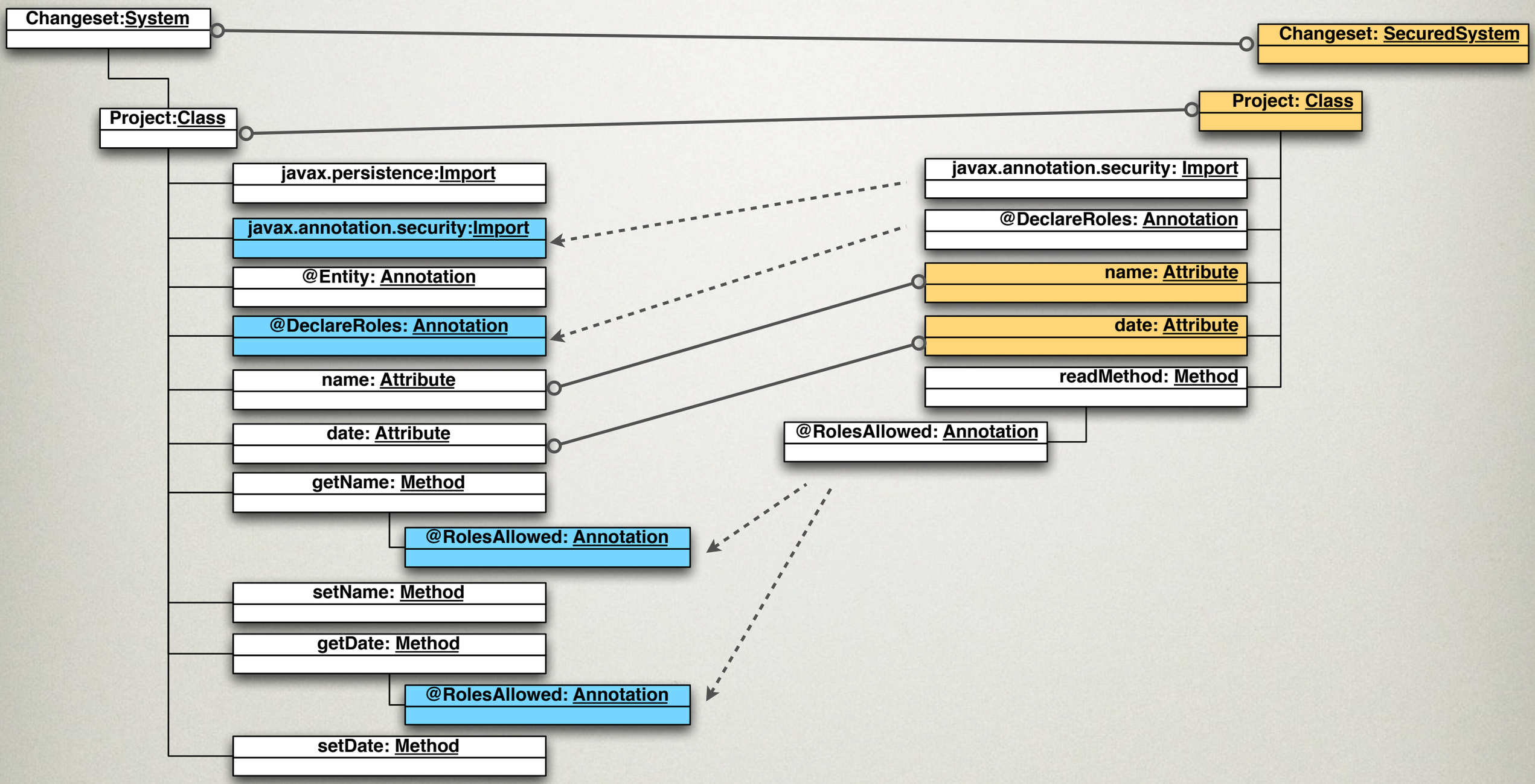


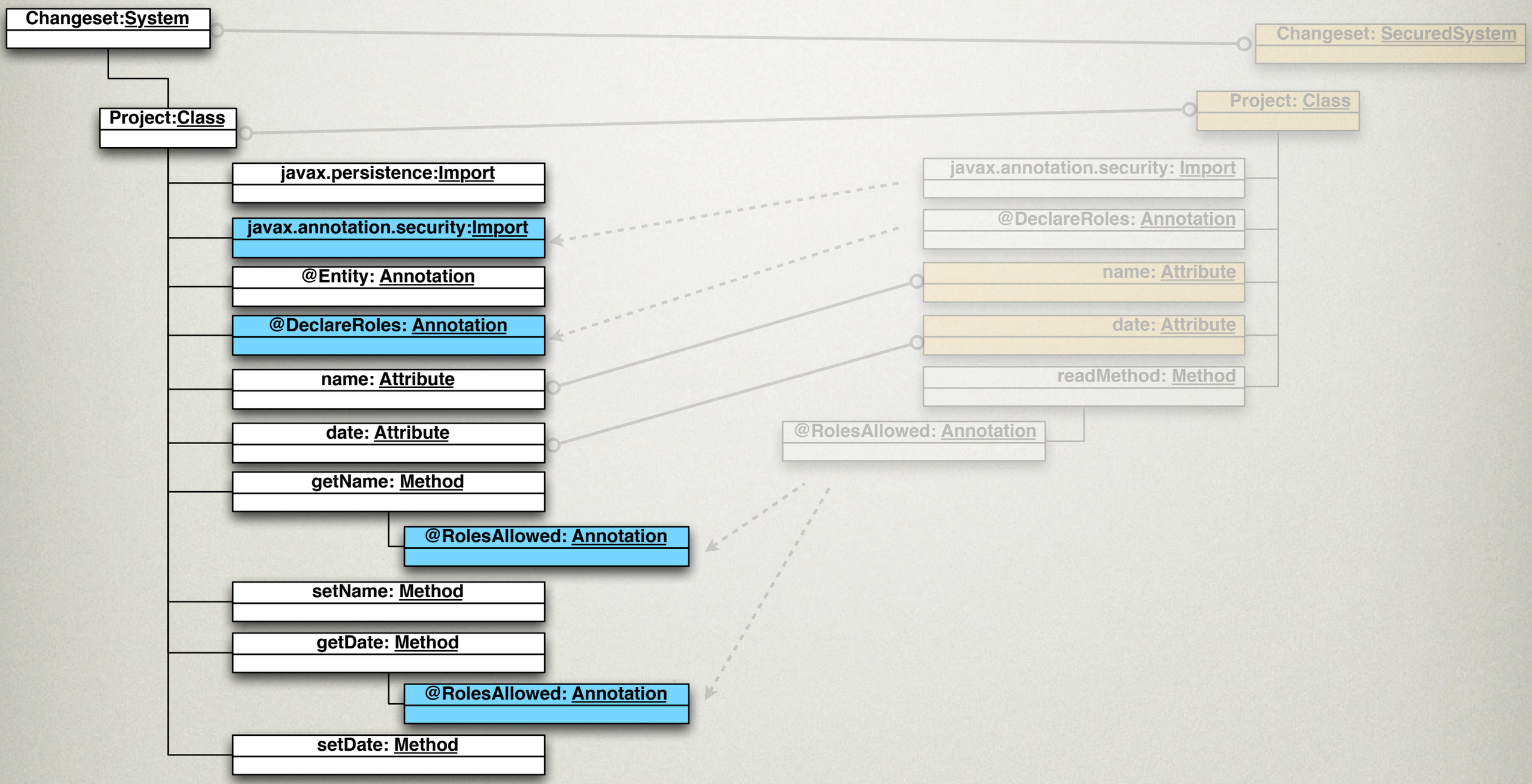

```
rule CompositionClassDeclaration {
    from
        s       :   JAVA!"j2se5::ClassDeclaration" (thisModule.inElements->includes(s)
                        and thisModule.incompleteLeftElements->includes(s))
    using {
        link    :   MATCHMM!MatchLink = thisModule.getLink(s);
    }
    to
        t       :   JAVA!"j2se5::ClassDeclaration"
        (
            __xmiID__               <-  s.__xmiID__,
            name                    <-  s.name,
            proxy                   <-  s.proxy,
            originalRank            <-  s.originalRank,
            originalFileContent     <-  s.originalFileContent,
            comments                <-  s.comments,
            annotations             <-  s.annotations->union(link.right.ref.annotations),
            modifier                <-  s.modifier,
            imports                 <-  s.imports,
            bodyDeclarations        <-  s.bodyDeclarations,
            superInterfaces         <-  s.superInterfaces,
            commentsBeforeBody      <-  s.commentsBeforeBody,
            commentsAfterBody       <-  s.commentsAfterBody,
            typeParameters          <-  s.typeParameters,
            superclass              <-  s.superclass
    )
}
```

The composition statements can be added by a HOT to a copy rule

Composition




Changeset:System
Changeset: SecuredSystem
Project:Class
Project: Class
javax.persistence:Import
javax.annotation.security:Import
javax.annotation.security: Import
@Entity: Annotation
@DeclareRoles: Annotation
@DeclareRoles: Annotation
name: Attribute
name: Attribute
date: Attribute
date: Attribute
readMethod: Method
@RolesAllowed: Annotation
@RolesAllowed: Annotation
getName: Method
@RolesAllowed: Annotation
setName: Method
getDate: Method
@RolesAllowed: Annotation
setDate: Method


Changeset:System
Project:Class
javax.persistence:Import
javax.annotation.security:Import
@Entity: Annotation
@DeclareRoles: Annotation
name: Attribute
date: Attribute
getName: Method
@RolesAllowed: Annotation
setName: Method
getDate: Method
@RolesAllowed: Annotation
setDate: Method
Changeset: SecuredSystem
Project: Class
javax.annotation.security: Import
@DeclareRoles: Annotation
name: Attribute
date: Attribute
readMethod: Method
@RolesAllowed: Annotation

Summary

- The composition is postponed to the lowest level of abstraction
- The use of a GPL metamodel allows the specification of the involved concerns
- The automatic derivation of correspondence models allows to easily reuse our proposal in a pair of target MTC
- Richer metamodels are required

Current Work

- To compose multiple concerns at the same time. (Security, Business, Navigation, Presentation, Use Cases)
- To use the low-level correspondence model to:
  - Check consistency
  - Generate interoperability bridges.

Questions?

Questions?