

Xtend

=> [large scale model transformations
with [it](#)]

Andreas Graf / itemis AG / EclipseCon 2017

Blurb:

Xtend is a flexible and expressive dialect of Java, which compiles into readable Java compatible source code. You can use any existing Java library seamlessly.

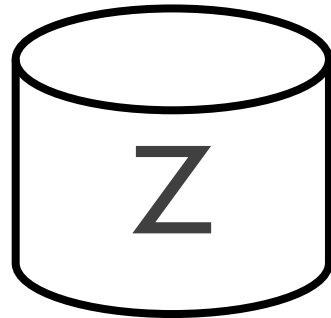
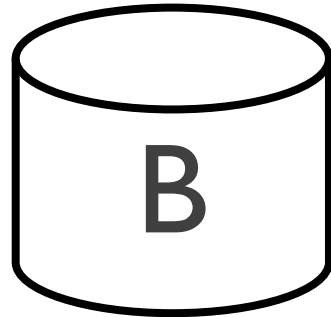
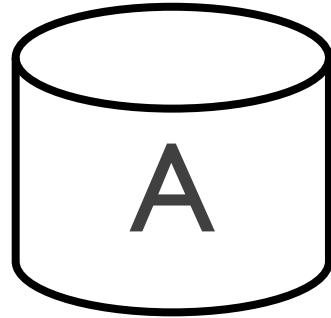
Blurb:

Xtend is a flexible and expressive dialect of Java, which compiles into readable Java compatible source code. You can use any existing Java library seamlessly.

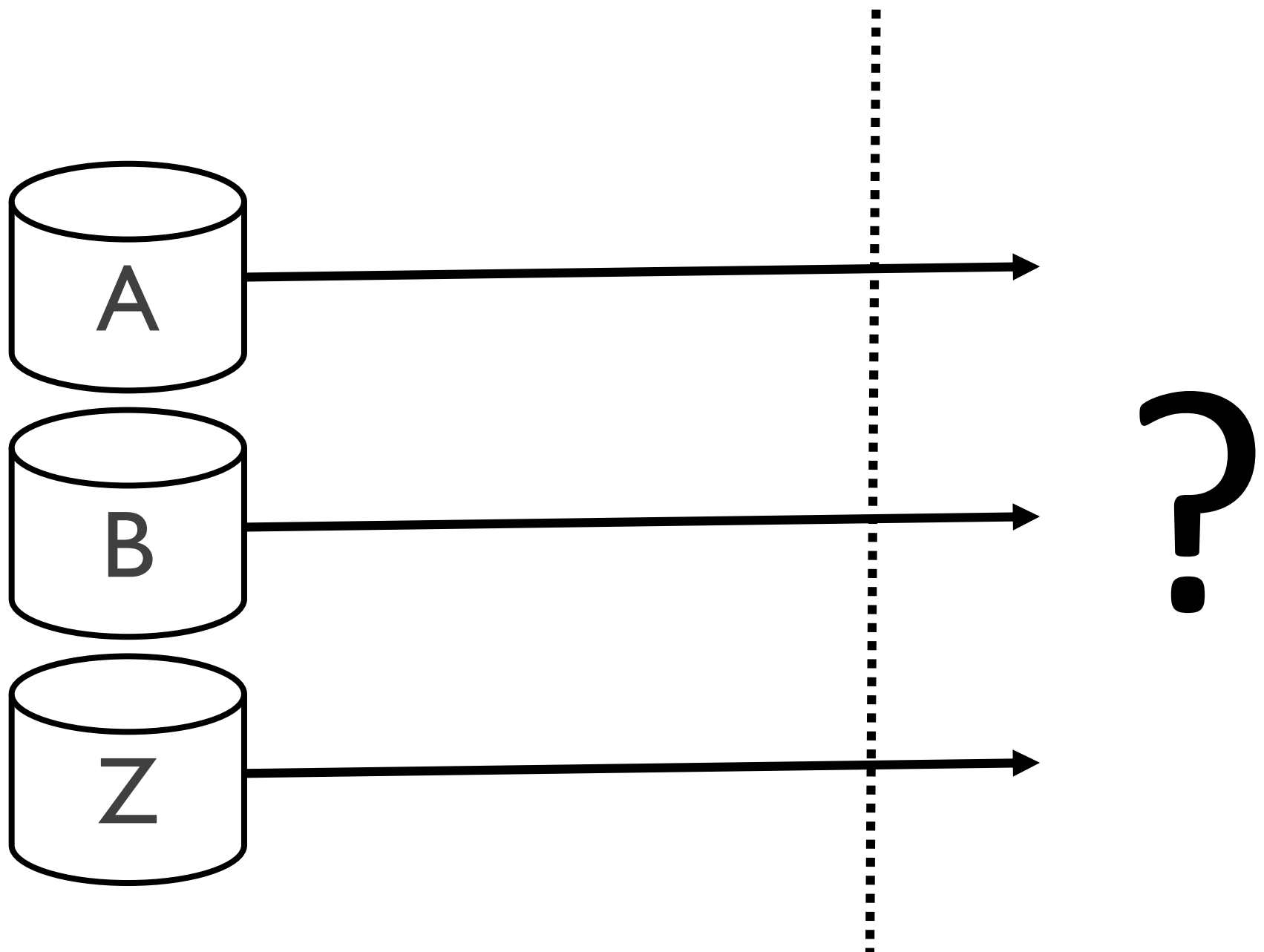
Cool:

- Model-To-Model
- Model-To-Text
- Lambdas **much nicer** than Java
- Active annotations
- Concise Syntax

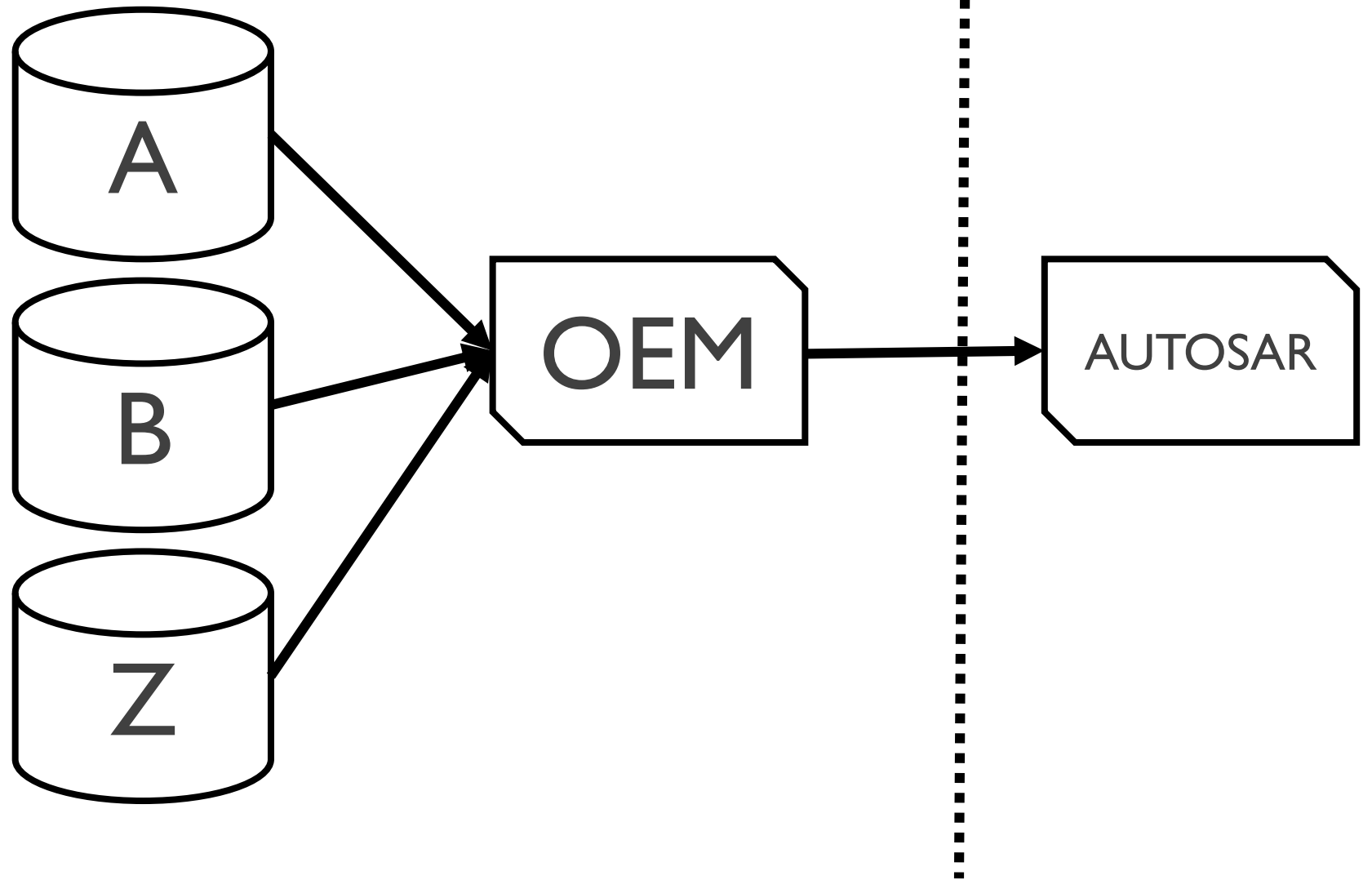
The Project



The Project

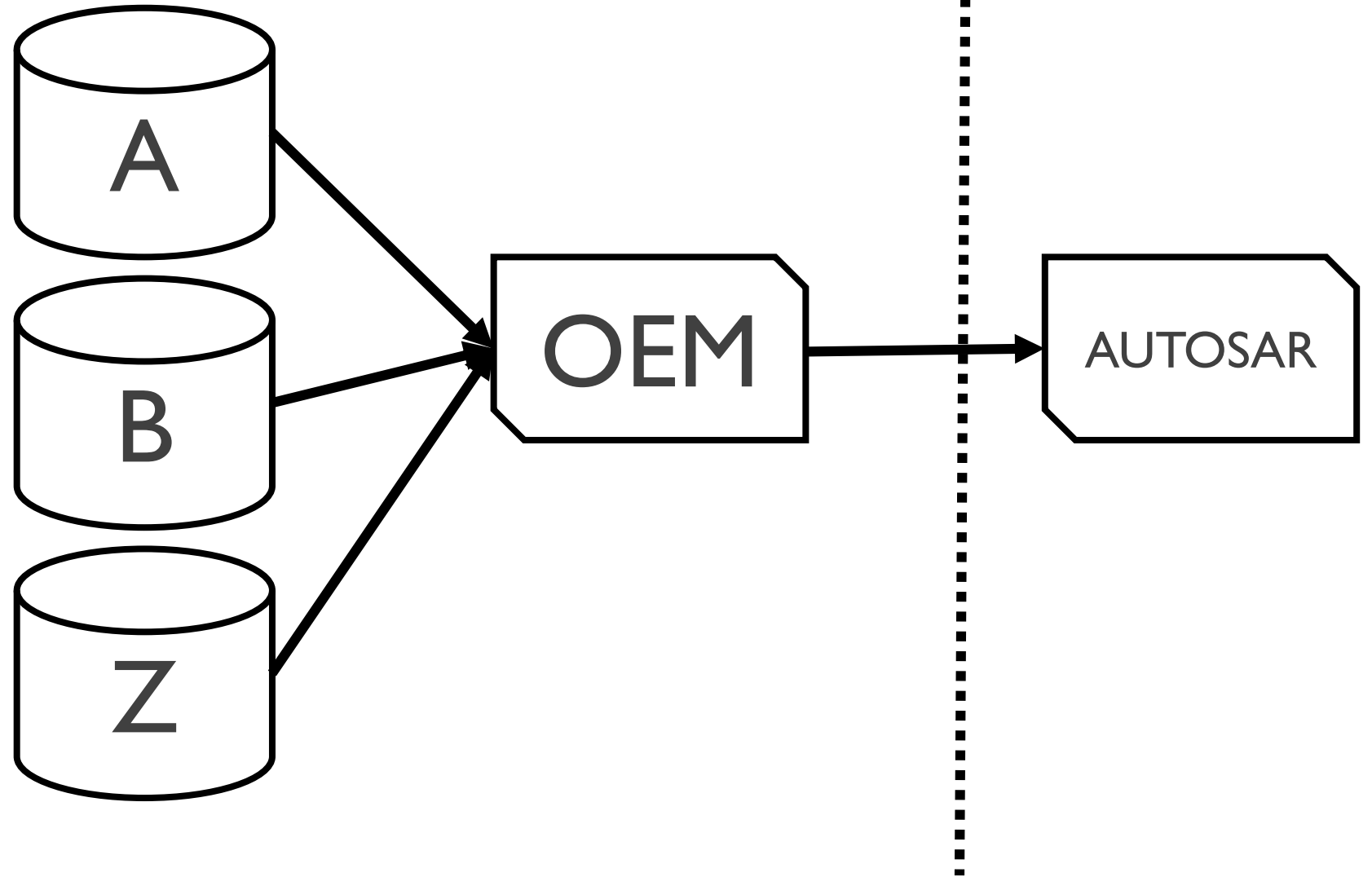


The Project



The Project

376,000 LOC



The Project

376,000 LOC
in Xtend!

```
TreeIterator<EObject> eAllContents = root.eAllContents();
while(eAllContents.hasNext()) {
    EObject ob = eAllContents.next();
    if(ob instanceof SwComponentType) {
        SwComponentType swc = (SwComponentType) ob;
        for(PortPrototype p : swc.getPorts()) {
            if(p instanceof PPortPrototype) {
                PortInterface portInterface = ((PPortPrototype) p).getProvidedInterface();
                if(portInterface instanceof SenderReceiverInterface) {
                    ((SenderReceiverInterface)portInterface).getDataElements();
                }
            }
        }
    }
}
```

```
root.eAllContents.filter(SwComponentType).map[ports].toList
    .flatten.filter(PPortPrototype).map[getProvidedInterface]
    .filter(SenderReceiverInterface).map[dataElements].toList.flatten
```


The Project

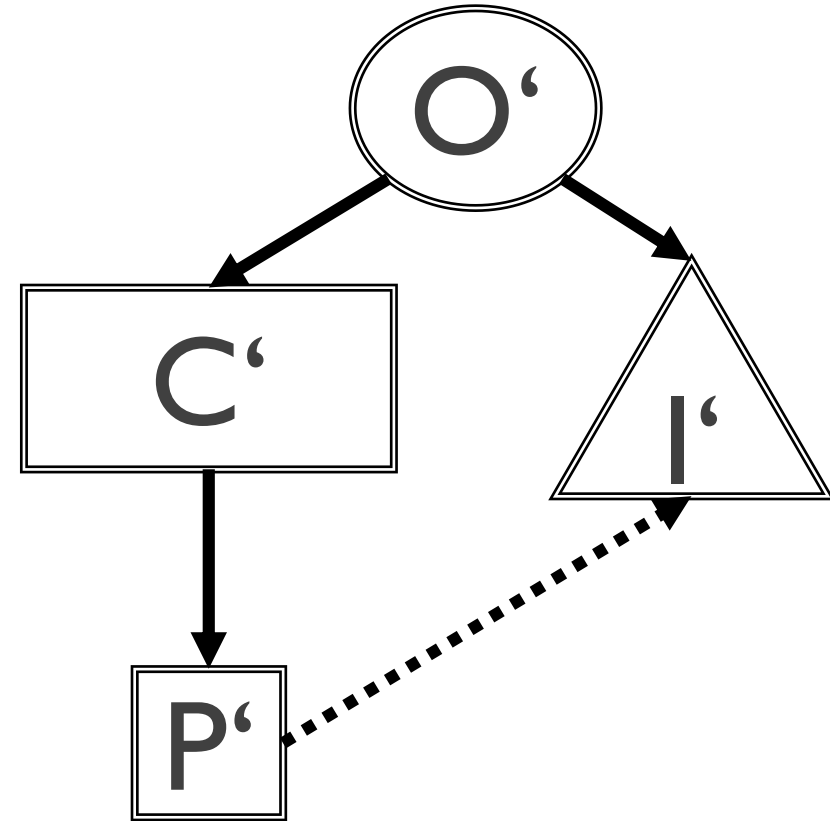
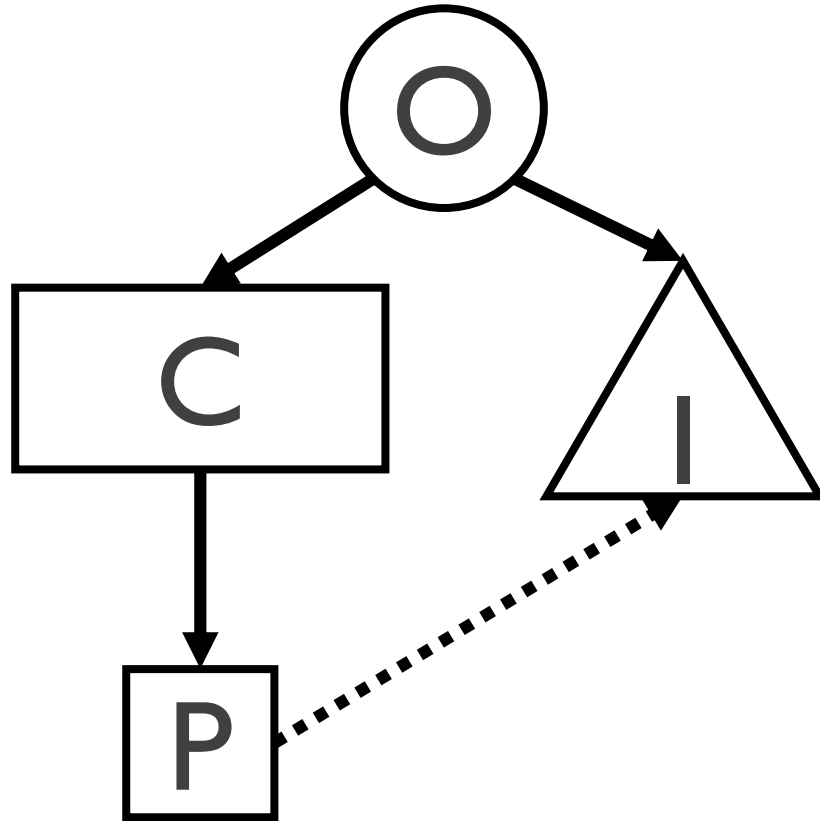
376,000 LOC
in Xtend!

```
Stream.generate(eAllContents::next).filter(SwComponentType.class::isInstance)
    .map(SwComponentType.class::cast).flatMap(sc -> sc.getPorts().stream())
    .filter(sc -> sc instanceof PPortPrototype).map(PPortPrototype.class::cast)
    .map(sc -> sc.getProvidedInterface())
    .filter(SenderReceiverInterface.class::isInstance).map(SenderReceiverInterface.class::cast)
    .flatMap(sc -> sc.getDataElements().stream())
    .collect(Collectors.toList())
;
```

```
root.eAllContents.filter(SwComponentType).map[ports].toList
    .flatten.filter(PPortPrototype).map[getProvidedInterface]
    .filter(SenderReceiverInterface).map[dataElements].toList.flatten
```

M2M

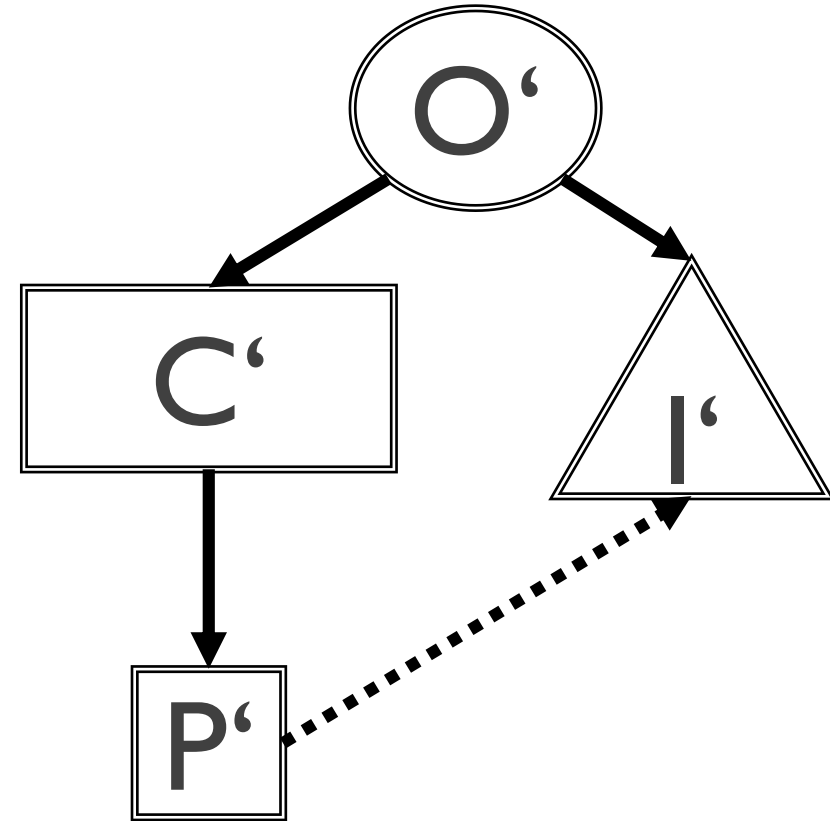
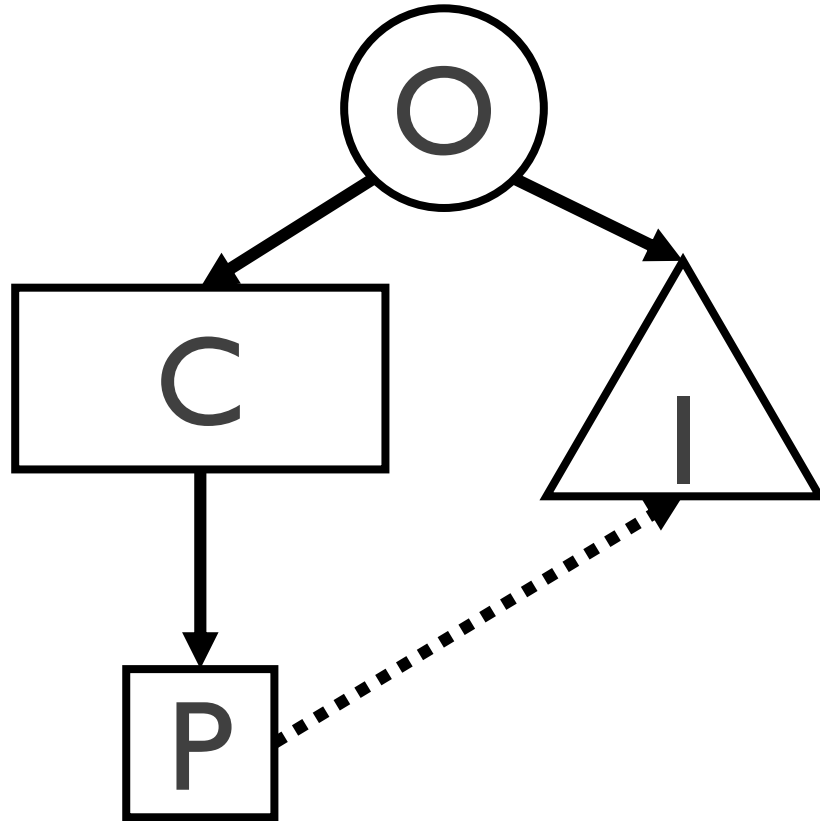
The Reference Problem



M2M

The Reference Problem

Multi-Pass
Custom Cache



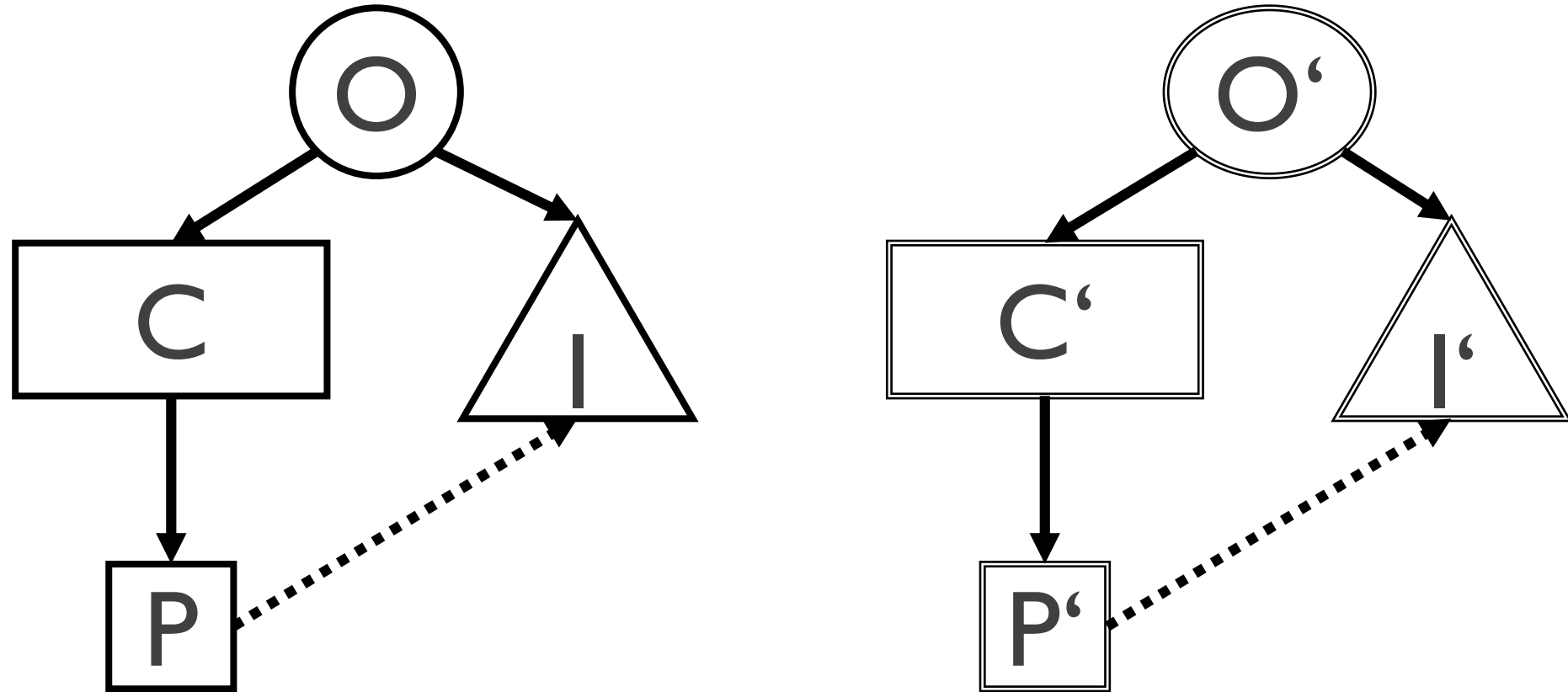
M2M

The Reference Problem

~~Multi-Pass~~

~~Custom Cache~~

Create Method

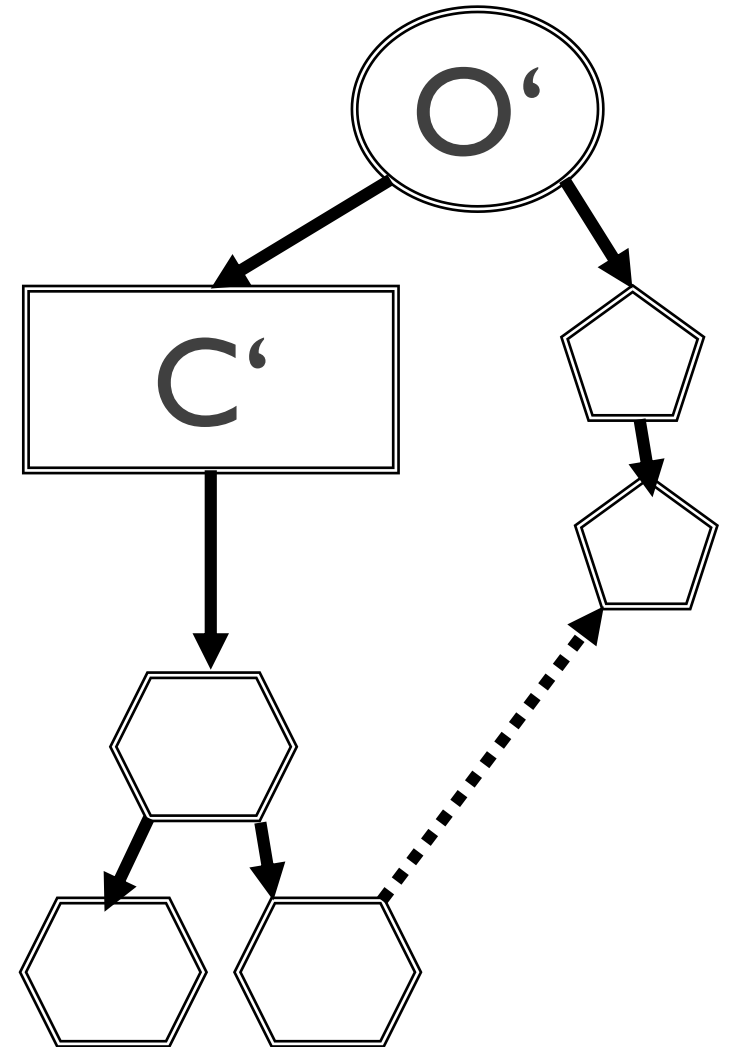
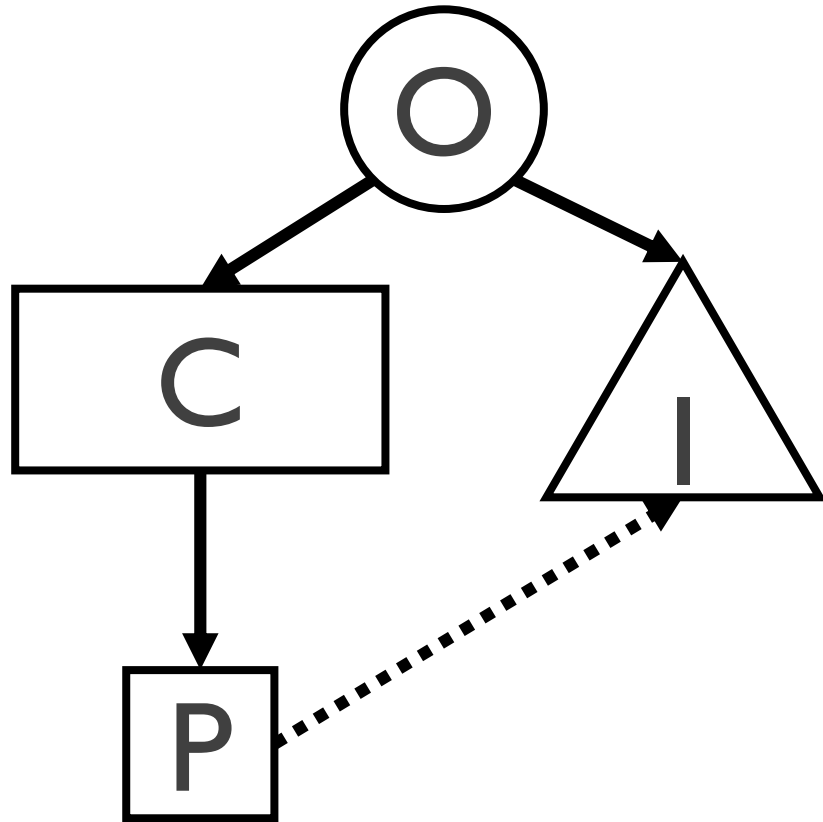


```
def create target: new T_Interface trafo(S_Interface s) {  
  target.shortName = s  
}
```

```
t_p.interface = s_p.iface.trafo; t_o.owned+=s_o.elements.map[trafo]
```

M2M

Analyzing



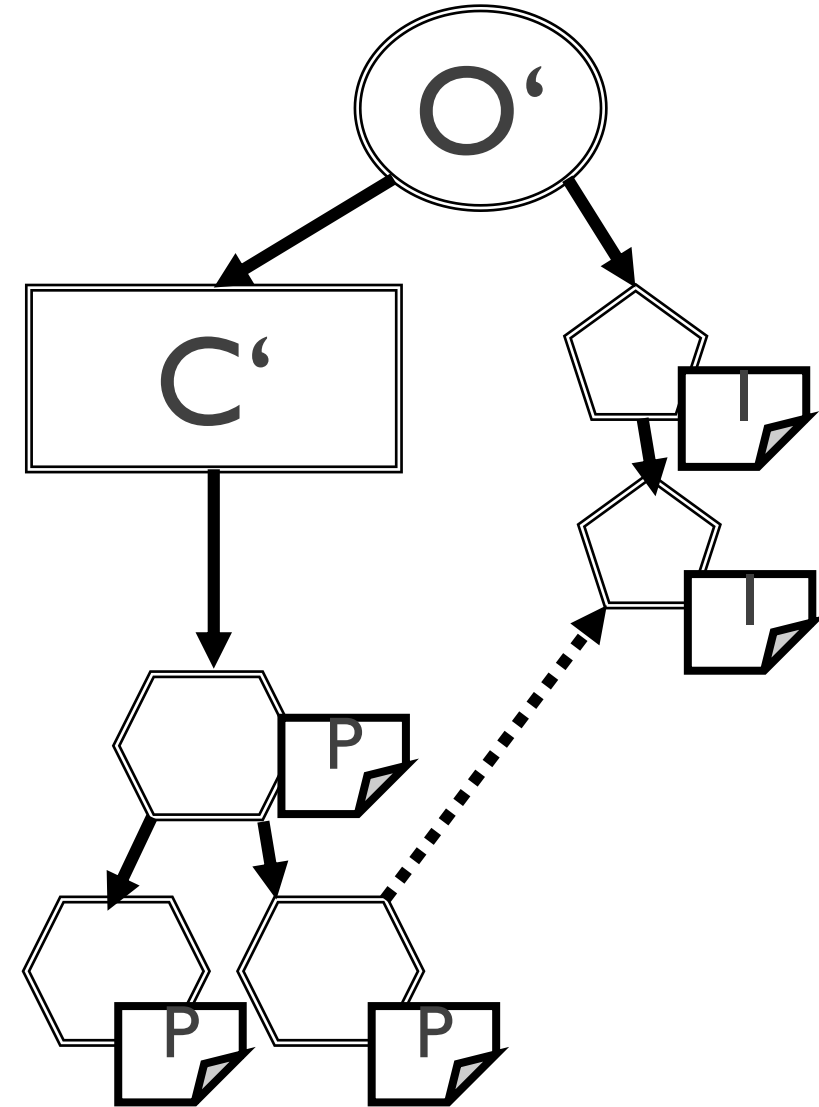
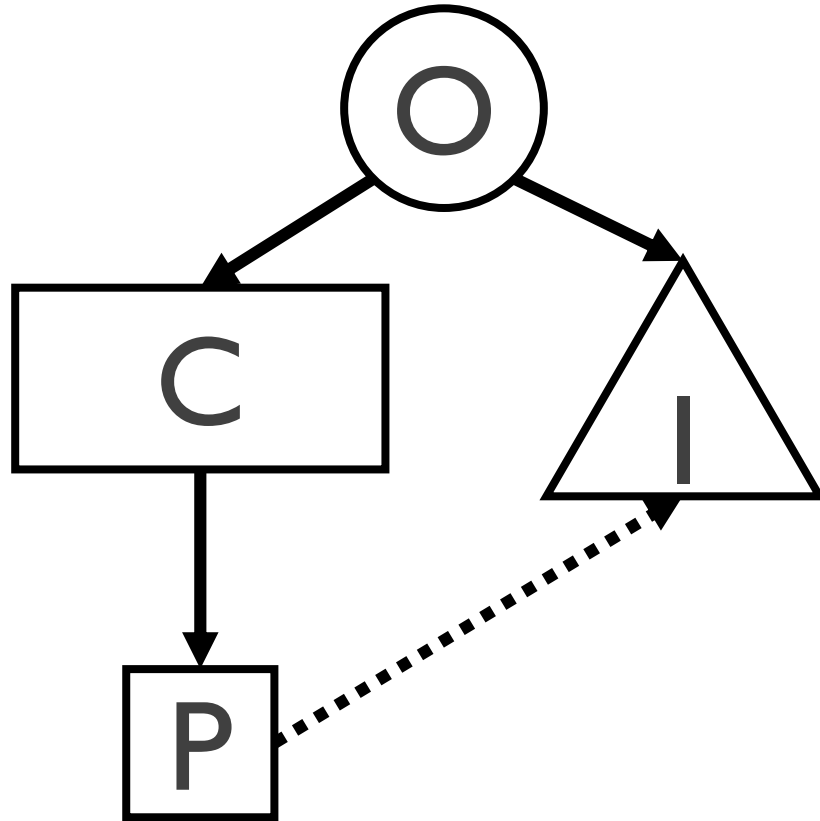
M2M

Analyzing

Tracing Info:
Find all caches of
create methods
(reflection)

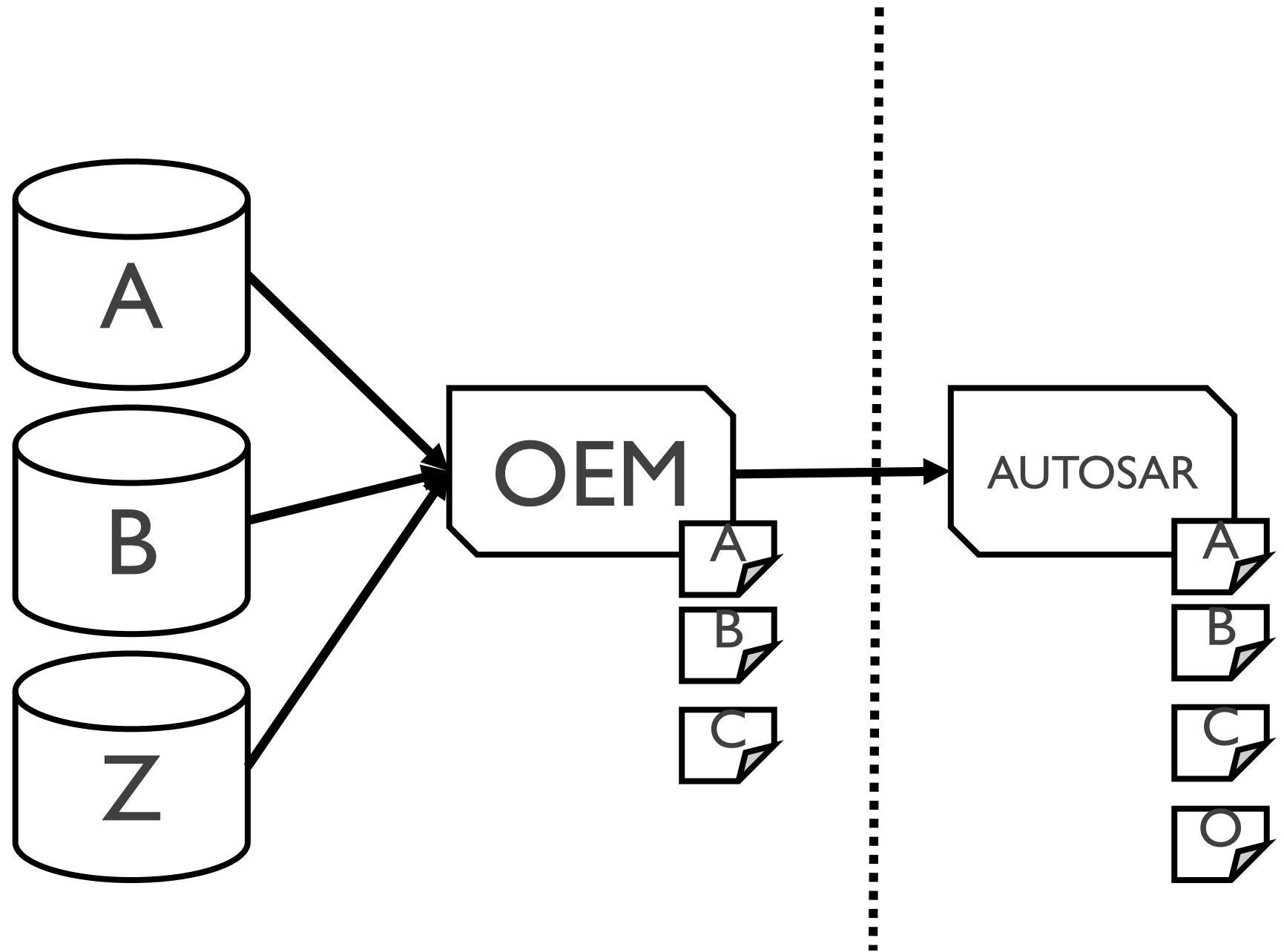
Attach information
about source element
from cache

(Dependency Injection
makes this much
easier)



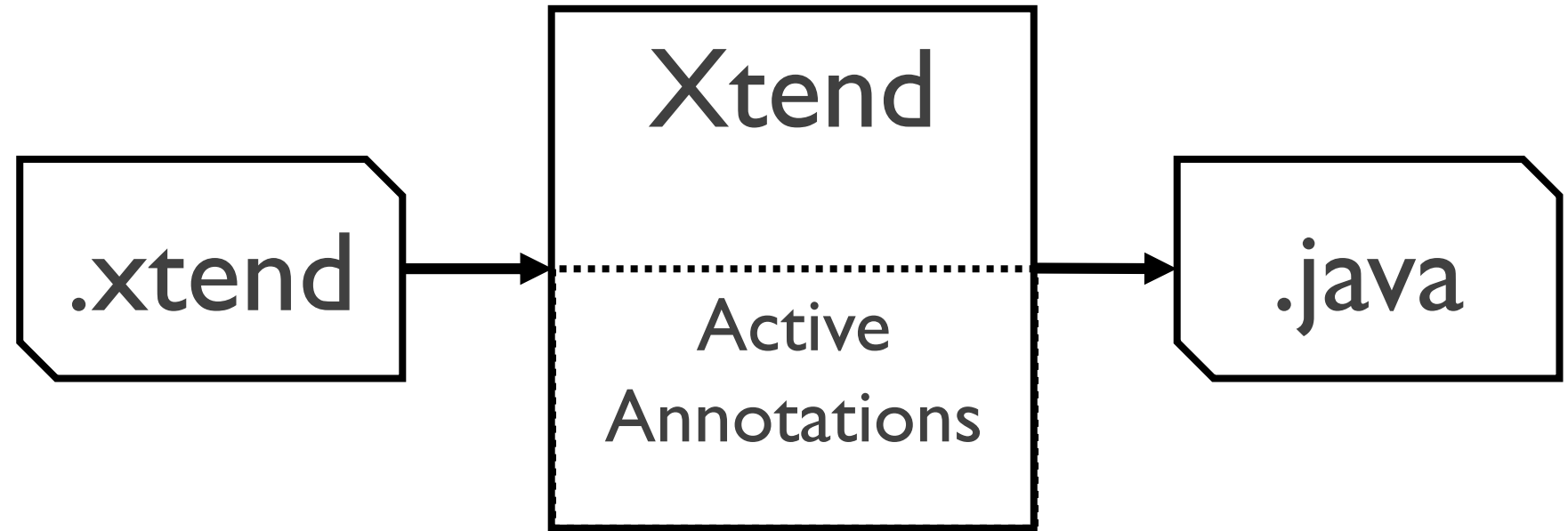
M2M

Analyzing



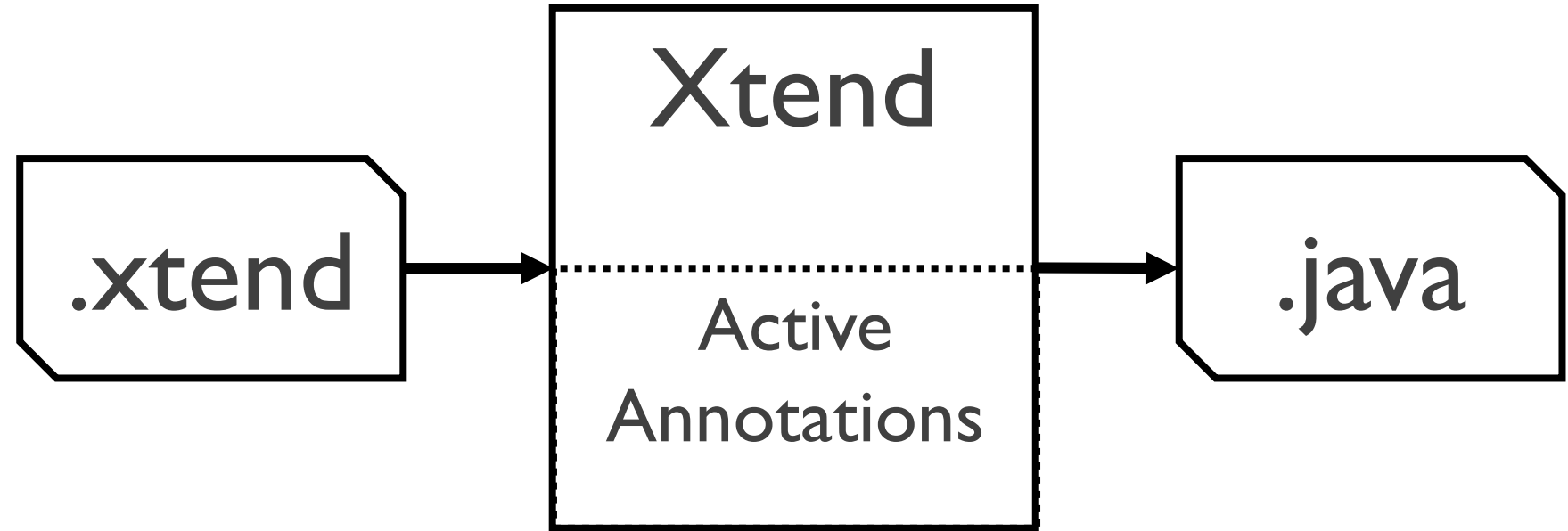
M2M

Annotations



M2M

Annotations



```
annotation Log {  
}
```

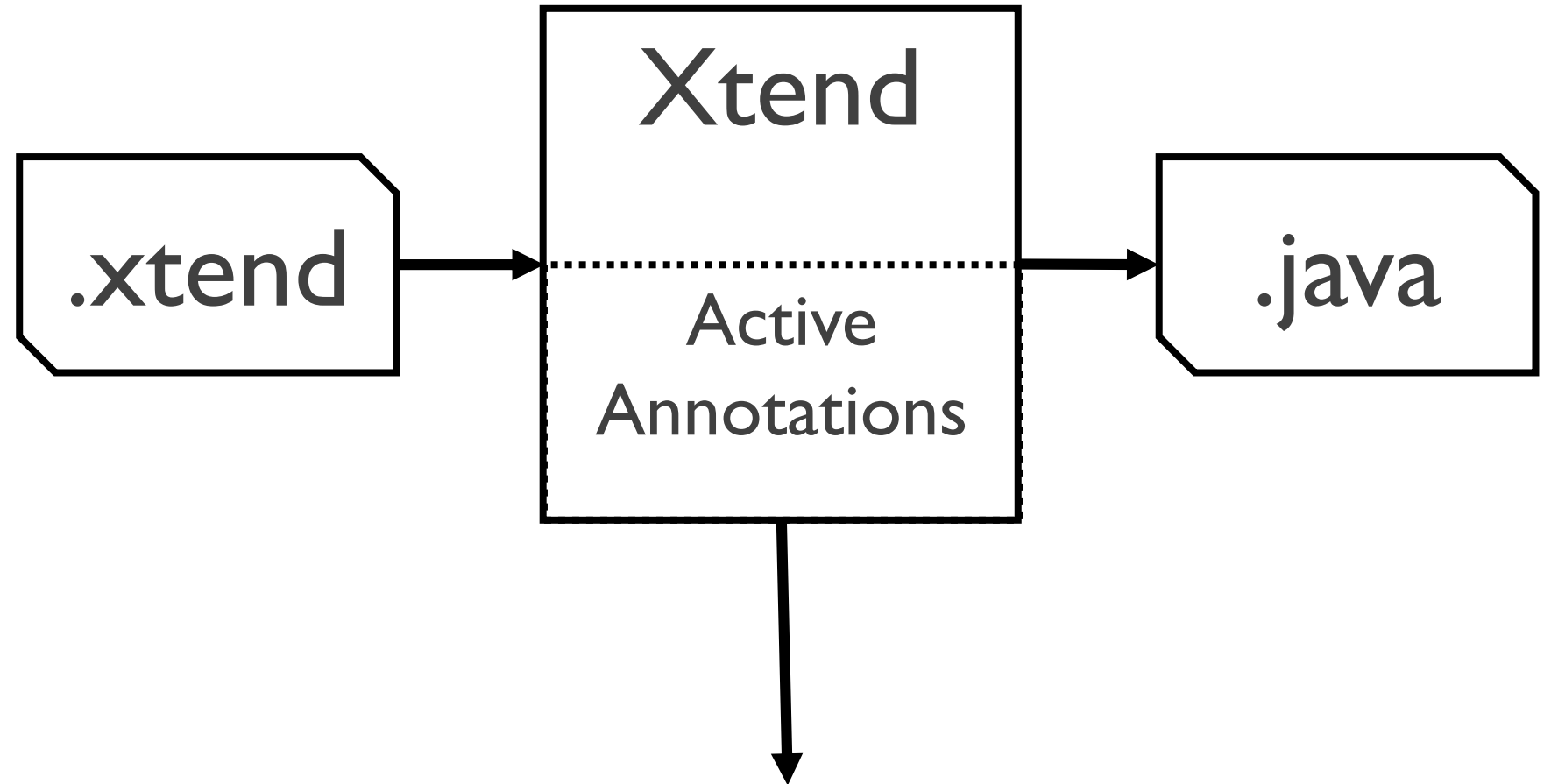
```
class LogProcessor extends AbstractClassProcessor {  
    public static String LOGGER_NAME = "LOGGER"
```

```
    override doTransform(MutableClassDeclaration annotatedClass, extension TransformationContext context)  
        annotatedClass.addField(LOGGER_NAME) [  
            visibility = Visibility.PRIVATE  
            static = true  
            final = true  
            type = Logger.newTypeReference  
            initializer = '''« Logger».create(«annotatedClass.simpleName».class)'''  
        ];  
};
```

M2M

Annotations

@Transformation



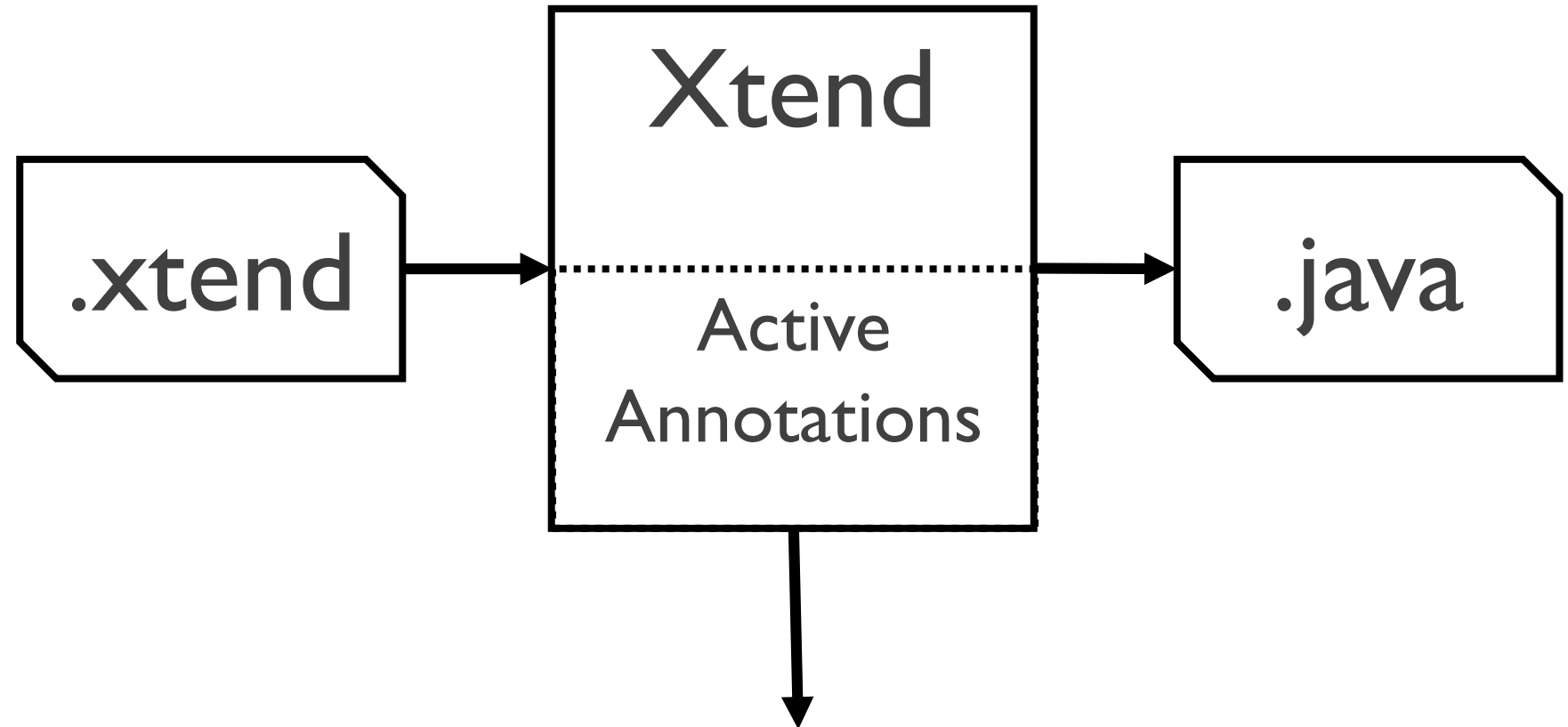
Created Element	Method	Parameters	...
T_Component	trafoC	S_Component	
T_Component	trafoC_String	String	Integer
T_Port	trafoPort	S_Port	Integer
T_SomeElement	blaFasel	S_Interface	T_Port

M2M

Annotations

@Req

```
@Req(  
reqs=#[„REQ_12“,  
„REQ_B5C“])
```



Requirement	Class	Method
REQ_12	Trafo1	compoTransform
REQ_12	Trafo2	
REQ_B5C	Trafo1	portTransform
REQ_B5C	Trafo1	portTransformSpecial

Blurb:

Xtend is a flexible and expressive dialect of Java, which compiles into readable Java 5 compatible source code. You can use any existing Java library seamlessly.

~~Blurb:~~ Cool:

Xtend is a flexible and expressive dialect of Java, which compiles into readable Java 5 compatible source code. You can use any existing Java library seamlessly.

Profiling JUnit Code Coverage Javadoc
Memory Analysis mvn
Class Swapping AspectJ
Java Libraries
EMF Support Dep. Injection