Softlang Working Group
http://softlang.wikidot.com/

MegaL Traceabiltiy Recovery

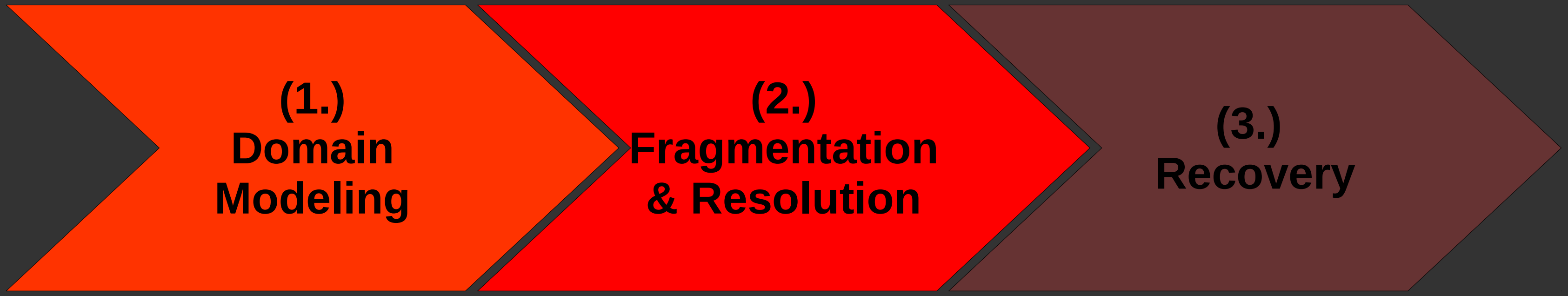
https://github.com/maxmeffert/megal-tr

An *ANTLR* Based Fragmentation API for MegaL

Meeting 2016-06-28

University of Koblenz-Landau
Maximilian Meffert

The Evaluation Process



(1.) Models the fragment *types* of the domain

(2.) Extracts the fragments from a domain instance (*Fragmentation*) & resolves further specializations if necessary (*Resolution*)

(3.) Recovers relationships between fragments
[not discussed now]

Example (Java)

```
JavaFragment < Fragment

// type declarations
JavaClass < JavaFragment
JavaInterface < JavaFragment
JavaEnum < JavaFragment

// member declarations
JavaInnerClass < JavaFragment
JavaMethod < JavaFragment
JavaConstructor < JavaMethod
JavaField < JavaFragment
JavaAnnotation < JavaFragment
```

```java
public class Foo {

    static public class Bar {

        private void getBar () {

        }

    }

    private String bar;

    public String getBar() {
        return bar;
    }

    public void setBar(String bar) {
        this.bar = bar;
    }

}
```

Example (Java)

```
aJavaFile: File
aJavaFile elementOf Java
aJavaFile = 'workspace:/org.softlang.megal.plugins/input/Foo.java'

aJavaFile.Foo#0: JavaClass
aJavaFile.Foo#0 partOf aJavaFile
aJavaFile.Foo#0 = 'file:/.../Foo.java#/0/Foo/JavaClass'

aJavaFile.Foo#0.Bar#0: JavaInnerClass
aJavaFile.Foo#0.Bar#0 partOf aJavaFile.Foo#0
aJavaFile.Foo#0.Bar#0 = 'file:/.../Foo.java#/0/Foo/JavaClass/0/Bar/JavaInnerClass'

aJavaFile.Foo#0.Bar#0.getBar#0: JavaMethod
aJavaFile.Foo#0.Bar#0.getBar#0 partOf aJavaFile.Foo#0.Bar#0
aJavaFile.Foo#0.Bar#0.getBar#0 = 'file:/.../Foo.java#/0/Foo/JavaClass/0/Bar/JavaInnerClass/0/getBar/JavaMethod'

aJavaFile.Foo#0.bar#1: JavaField
aJavaFile.Foo#0.bar#1 partOf aJavaFile.Foo#0
aJavaFile.Foo#0.bar#1 = 'file:/.../Foo.java#/0/Foo/JavaClass/1/bar/JavaField'

aJavaFile.Foo#0.getBar#2: JavaMethod
aJavaFile.Foo#0.getBar#2 partOf aJavaFile.Foo#0
aJavaFile.Foo#0.getBar#2 = 'file:/.../Foo.java#/0/Foo/JavaClass/2/getBar/JavaMethod'

aJavaFile.Foo#0.setBar#3: JavaMethod
aJavaFile.Foo#0.setBar#3 partOf aJavaFile.Foo#0
aJavaFile.Foo#0.setBar#3 = 'file:/.../Foo.java#/0/Foo/JavaClass/3/setBar/JavaMethod'
```

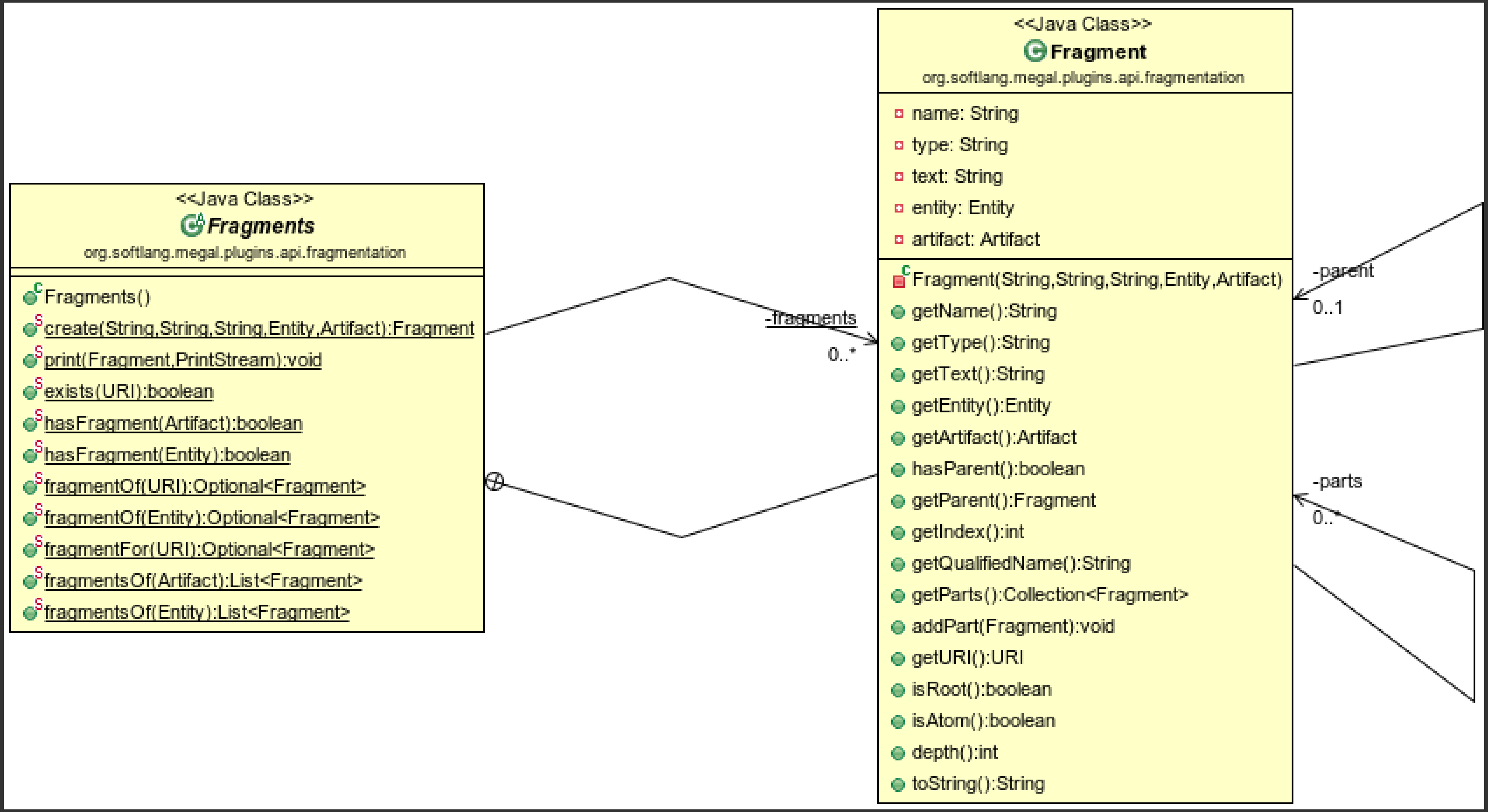
Fragmentation Result

Example (Java)

```
public class Foo {

    static public class Bar {

        private void getBar () {

        }

    }

    private String bar;

    public String getBar() {
        return bar;
    }

    public void setBar(String bar) {
        this.bar = bar;
    }

}
```

A computational fragment model should be loosely based on syntax trees.
Scope defines parthood.

Computational Fragment Model & KB



<<Java Class>>
Fragments
org.softlang.megal.plugins.api.fragmentation
Fragments()
create(String,String,String,Entity,Artifact):Fragment
print(Fragment,PrintStream):void
exists(URI):boolean
hasFragment(Artifact):boolean
hasFragment(Entity):boolean
fragmentOf(URI):Optional<Fragment>
fragmentOf(Entity):Optional<Fragment>
fragmentFor(URI):Optional<Fragment>
fragmentsOf(Artifact):List<Fragment>
fragmentsOf(Entity):List<Fragment>
-fragments
0..*
<<Java Class>>
Fragment
org.softlang.megal.plugins.api.fragmentation
name: String
type: String
text: String
entity: Entity
artifact: Artifact
Fragment(String,String,String,Entity,Artifact)
getName():String
getType():String
getText():String
getEntity():Entity
getArtifact():Artifact
hasParent():boolean
getParent():Fragment
getIndex():int
getQualifiedName():String
getParts():Collection<Fragment>
addPart(Fragment):void
getURI():URI
isRoot():boolean
isAtom():boolean
depth():int
toString():String
-parent
0..1
-parts
0..*

Fragment Model & KB

- Fragments build a simple generic tree alongside the original AST
  - A leaf node is called ***atom***
  - A non-leaf node is called ***compound***
- A Fragment KB (*Fragments*) exists separately from the Megamodel KB during an evaluation process

Qualified Fragment Names

`aJavaFile . Bar#0 . […] . doStuff#666`

Name of the declared entity
Short name of the fragment
Index of the fragment in its container

Qualified Fragment Names

- Qualified Fragment Names are used as identifies for the derived entities
- Qualified Fragment Names depict *partOf* relationships
- Indexes depict the position of fragments in their respective composite

Fragment URIs

Generic URI Form:

```
scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
```

Fragment URI Form:

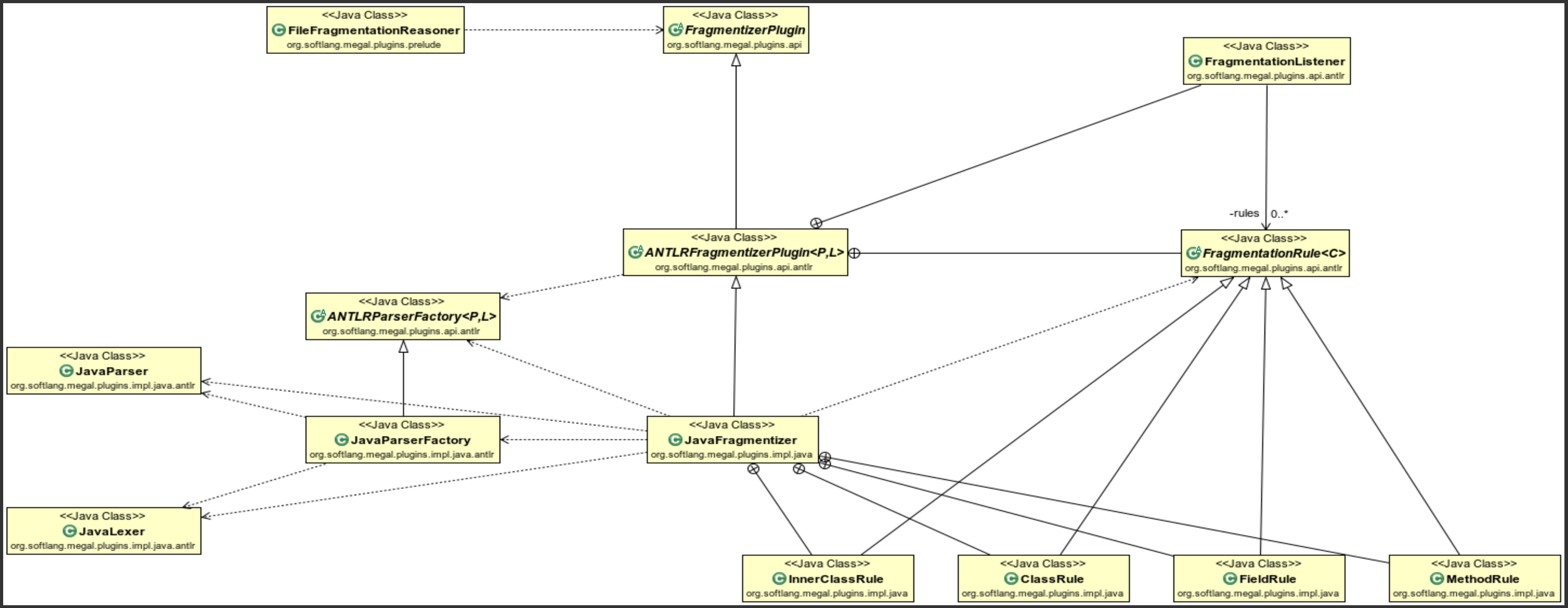
```
scheme://location#fragment
```

Where fragment matches:

```
Fragment   : ‘/’ INDEX ‘/’ NAME ‘/’ TYPE (‘/’ Fragment )*
INDEX      : \d+
NAME       : \w+
TYPE       : \w+
```

`file://path/to/Foo.java#/0/Foo/JavaClass/2/getBar/JavaMethod`

ANTLR Based Fragmentation API

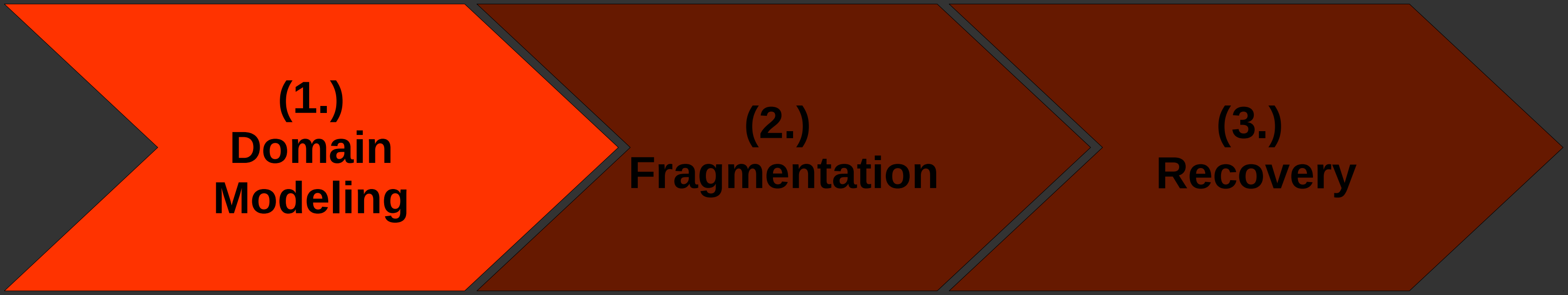


<<Java Class>> FileFragmentationReasoner org.softlang.megal.plugins.prelude
<<Java Class>> FragmentizerPlugin org.softlang.megal.plugins.api
<<Java Class>> FragmentationListener org.softlang.megal.plugins.api.antlr
-rules 0..*
<<Java Class>> ANTLRFragmentizerPlugin<P,L> org.softlang.megal.plugins.api.antlr
<<Java Class>> FragmentationRule<C> org.softlang.megal.plugins.api.antlr
<<Java Class>> ANTLRParserFactory<P,L> org.softlang.megal.plugins.api.antlr
<<Java Class>> JavaParser org.softlang.megal.plugins.impl.java.antlr
<<Java Class>> JavaParserFactory org.softlang.megal.plugins.impl.java.antlr
<<Java Class>> JavaFragmentizer org.softlang.megal.plugins.impl.java
<<Java Class>> JavaLexer org.softlang.megal.plugins.impl.java.antlr
<<Java Class>> InnerClassRule org.softlang.megal.plugins.impl.java
<<Java Class>> ClassRule org.softlang.megal.plugins.impl.java
<<Java Class>> FieldRule org.softlang.megal.plugins.impl.java
<<Java Class>> MethodRule org.softlang.megal.plugins.impl.java

XML-Dialect Resolution

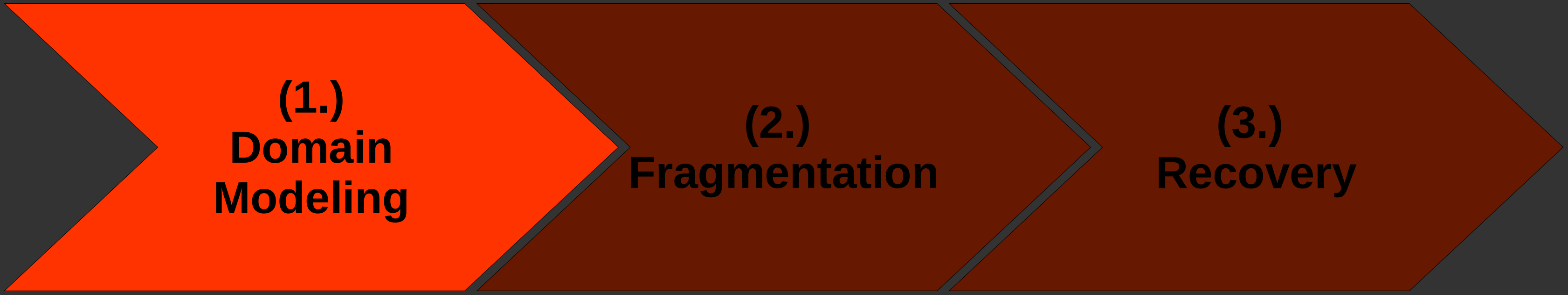
XML Fragmentation

Motivation



A domain model is a collection of (axiomatic) *statements* over entity-types and entities.

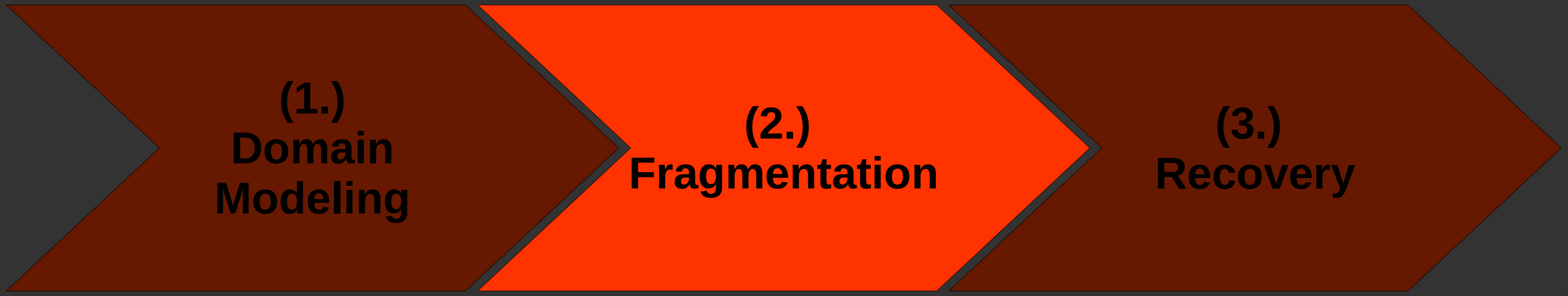
Motivation



```
Foo < Entity
Bar < Entity

partOf < Foo * Bar
```

...

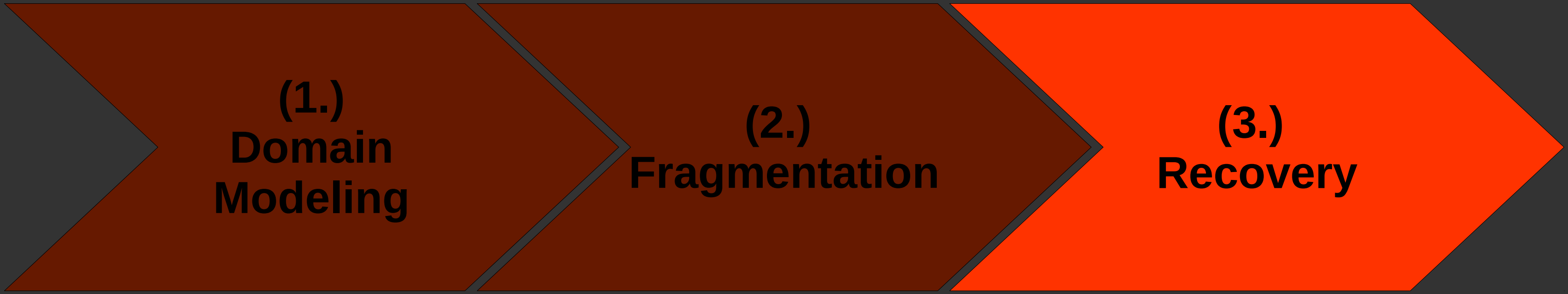
Motivation



Extends a KB with entities found in Manifestations.

Fragmentation is done by several plugins, specifically tailored to a domain.

Motivation



Can we provide singular plugins for recovery?

TODO