From Spoon to Repairnator: hands on!

Simon Urli

11th, July 2018

Spirals Team Meeting 2018

Program

Goal: present Spoon and discuss with its potential usage on your own works. It's meant to be interactive!

- Overview of Spoon (10-15 min max)
- Spoon demo (10 min)
- Hands on! (35-40 min)

Disclaimer: title is cool, but we won't really have time to really talk about Spoon usage on repair tools ;)

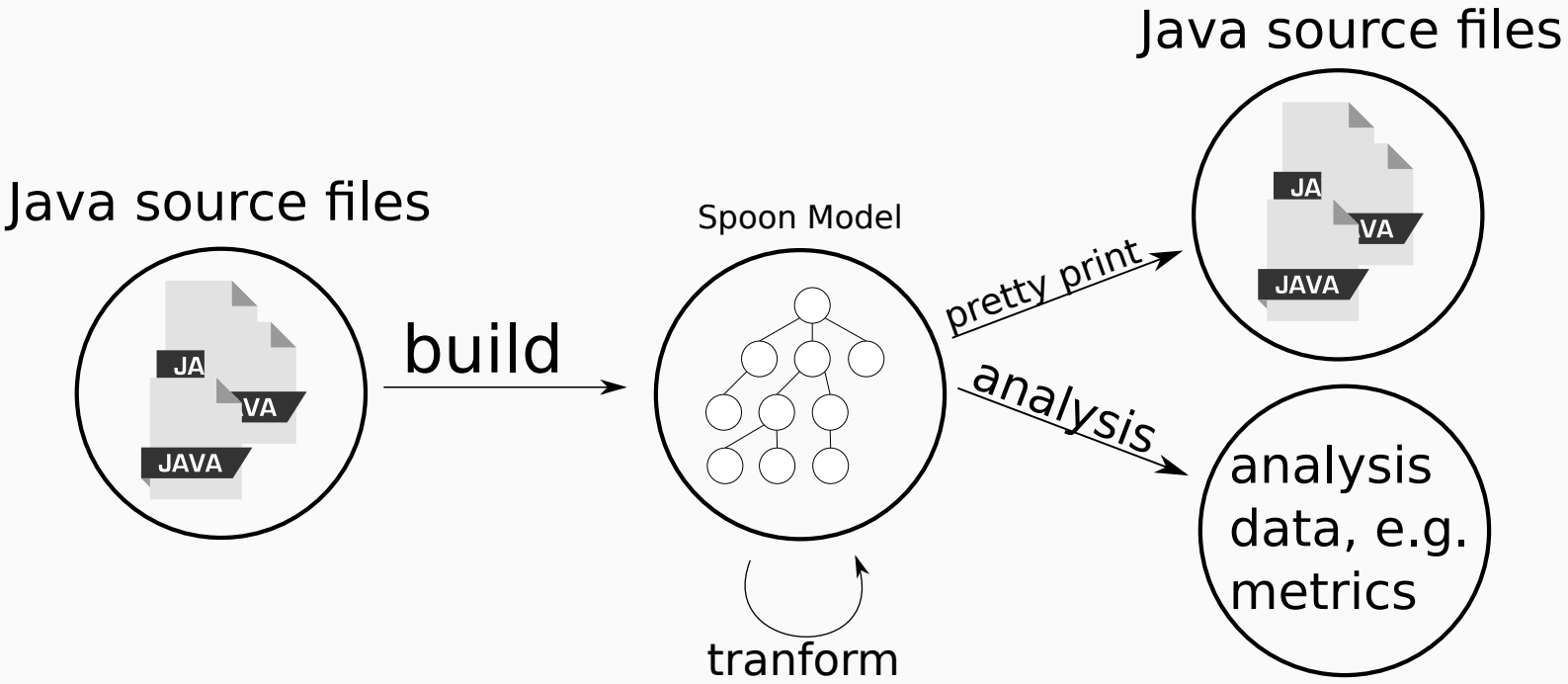
How Spoon is used in the team?

- Mixin creations and component architecture optimization: Lionel
- Software repair tools: Thomas, Benjamin [`Nopol`, `NPEFix`, `AssertFixer`, `Astor`, `...`]
- Test amplification: Benjamin [`DSpot`]
- Bad smells detection: Sarra [`Paprika`]
- JavaScript documentation generation from Android API: Romain, Lakhdar [`APISense`]
- (WIP: Dynamic Software Product Line: Clément)
- (Create state diagrams for java programs: Simon B.)
- (Change data structure in program to measure energy consumption changes: Chakib)

Spoon 101

Spoon in a nutshell

A library to write your own analysis and transformation in java.



Usage examples:

- test improvement
- transpilation, e.g. Java to Javascript
- detection of bad smells
- automatic refactoring

Spoon standard process

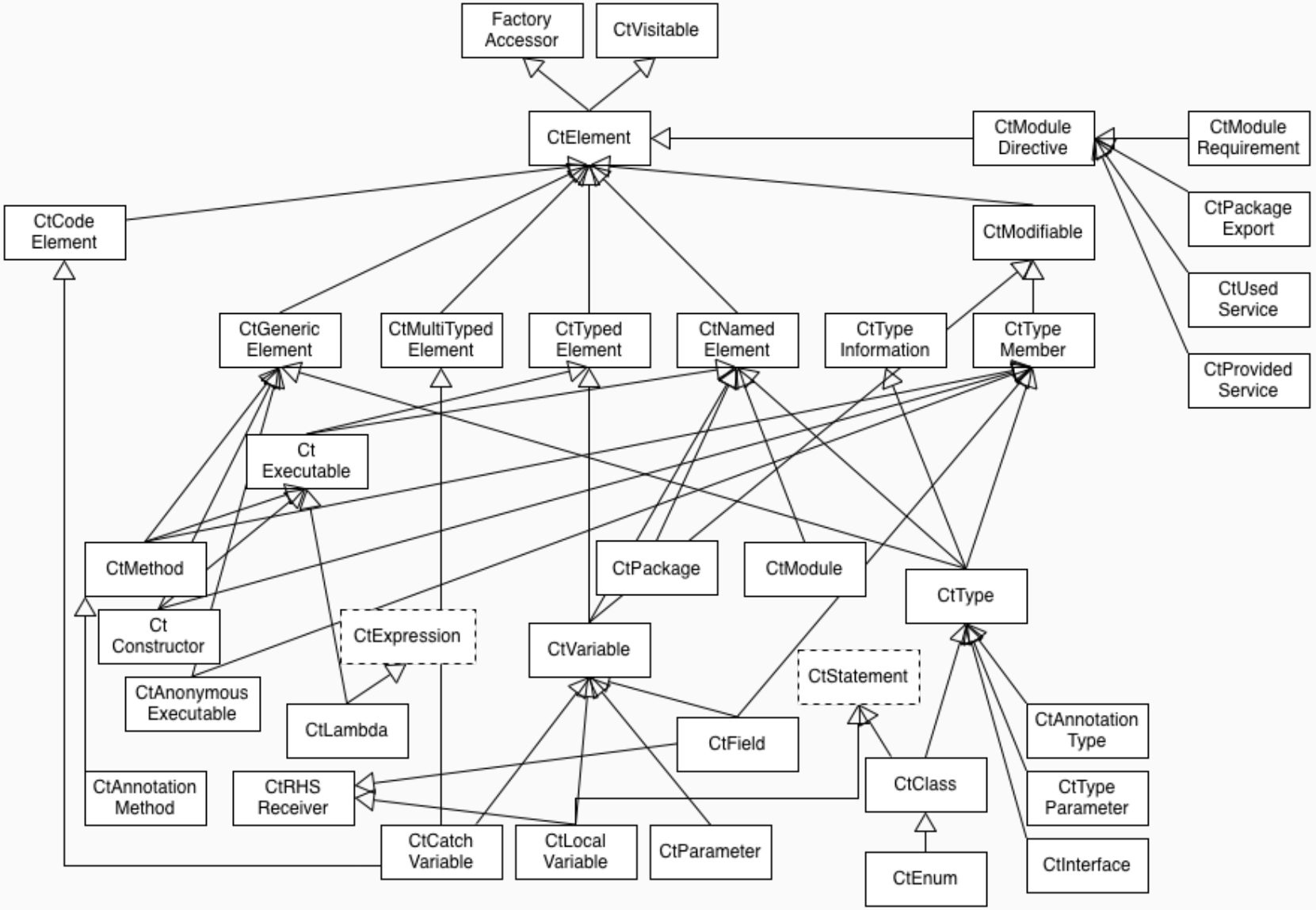
1. Build a model of your project
2. Query and analyze interesting parts
3. Transform what needs to be transformed
4. Output a transformed source code

Spoon AST Metamodel

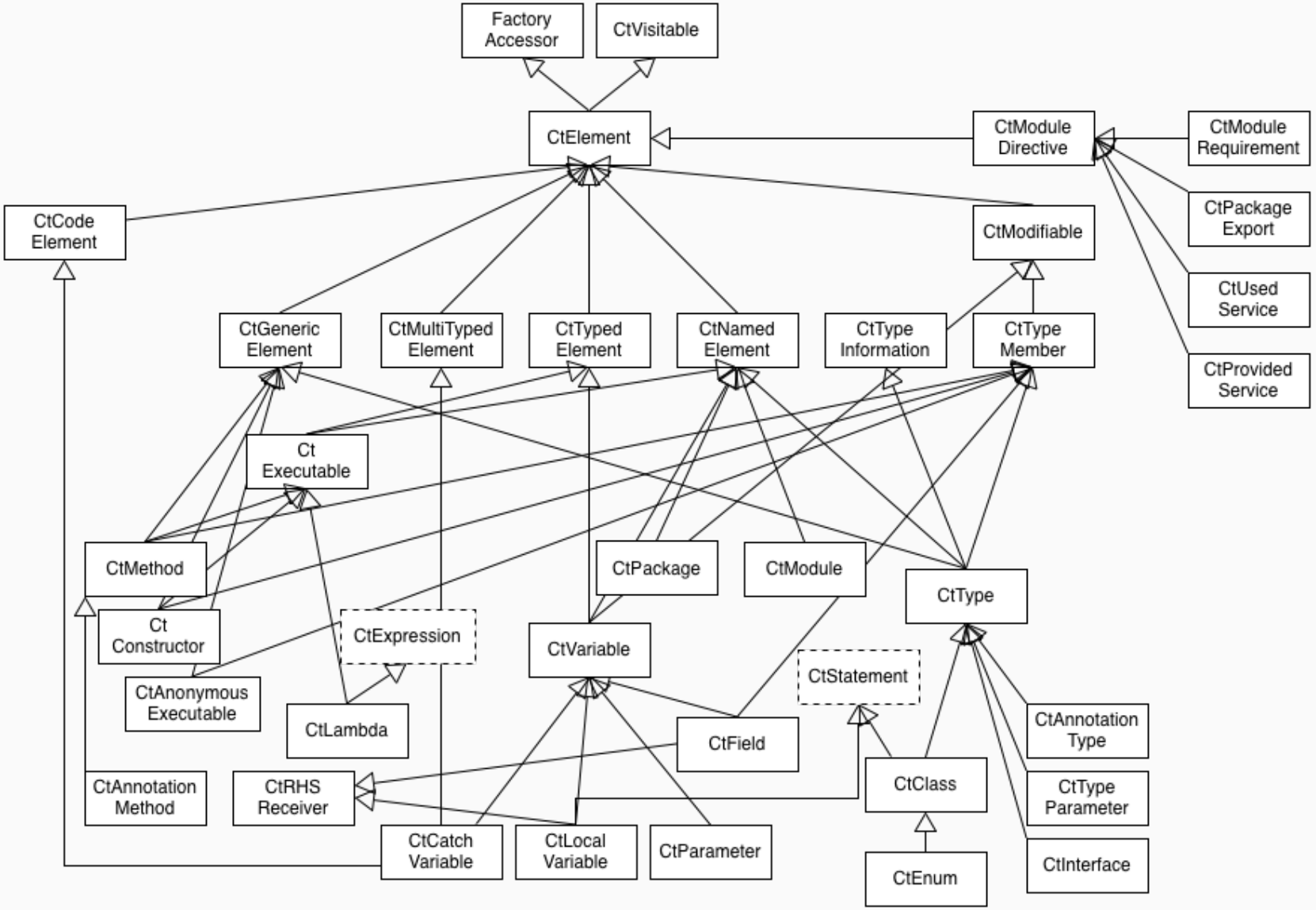
Three categories:

- structural elements: class, package, ...
- code elements: if, for, while, ...
- **references elements**: when something already declared is used (types, methods, fields, ...)

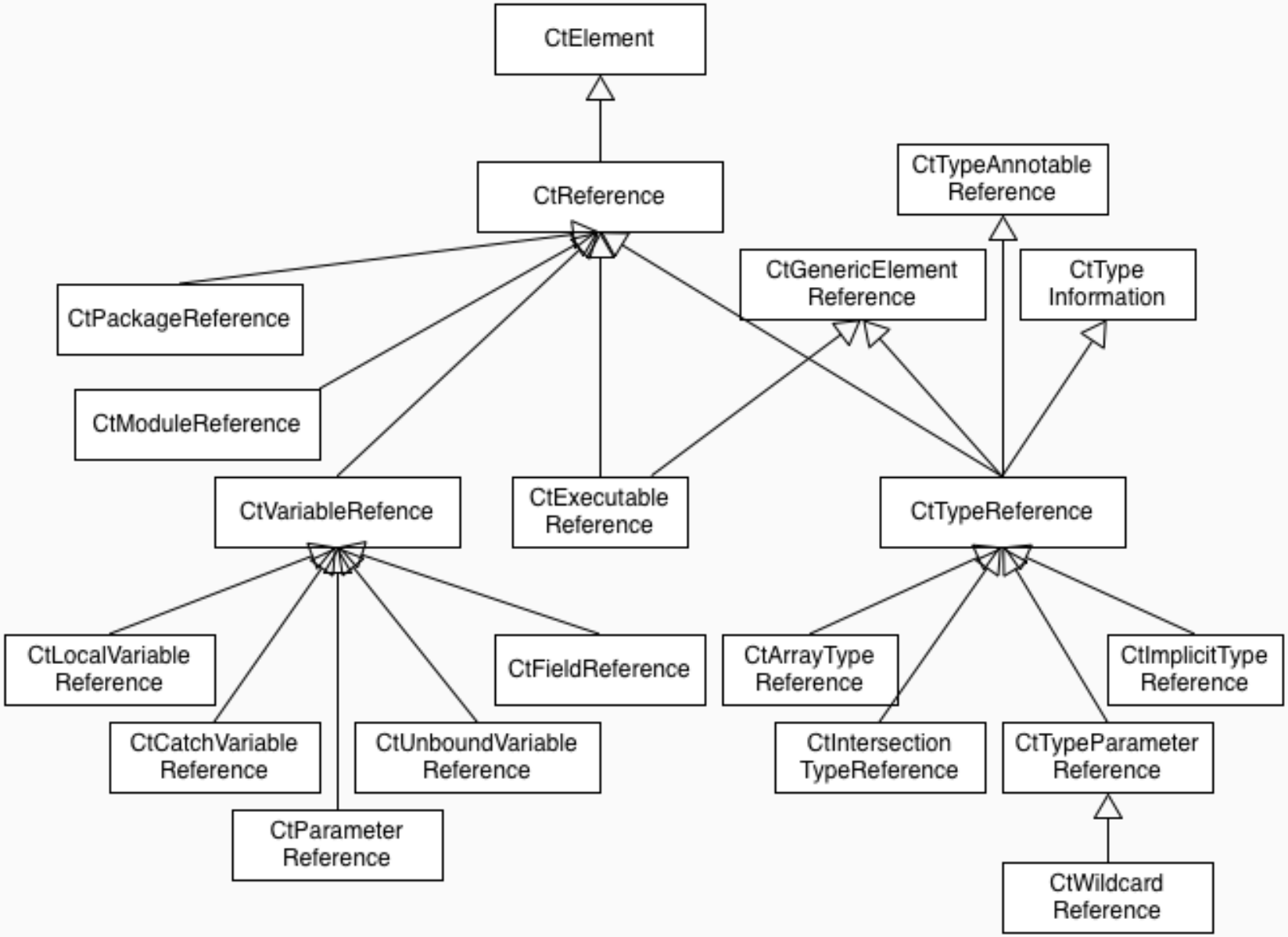
Spoon AST Metamodel (structural elements)



Spoon AST Metamodel (code elements)



Spoon AST Metamodel (references elements)



Spoon Model

Some hints about Spoon Model:

- it's an Abstract Syntax **Tree**: you have access to parents and children at each node
- only references might appear multiple times in the model
- everything's not necessarily named
- the tree is processed in depth first when using processors

CRUD on the model

Transforming model involve performing basic CRUD operations on the nodes of the model.

CRUD on the model

Transforming model involve performing basic CRUD operations on the nodes of the model.

- Create a node

```
Factory factory = launcher.getFactory();
CtClass aClass = factory.createClass( qualifiedName: "my.org.MyClass");
```

CRUD on the model

Transforming model involve performing basic CRUD operations on the nodes of the model.

- Create a node
- Retrieve a node or a property

```
namedElement.getSimpleName();
List<CtMethod> elements = ctClass.getElements(new TypeFilter<>(CtMethod.class));
```

CRUD on the model

Transforming model involve performing basic CRUD operations on the nodes of the model.

- Create a node
- Retrieve a node or a property
- Update a node or a property

```
aClass.setSimpleName("myNewName");
CtMethod myMethod = factory.createMethod();
aClass.addMethod(myMethod);
```

CRUD on the model

Transforming model involve performing basic CRUD operations on the nodes of the model.

- Create a node
- Retrieve a node or a property
- Update a node or a property
- Delete a node

```
aClass.removeMethod(myMethod);
```

Demo

Spoon Demo

- Present the project
- present Spoon model of the project (GUI)

Hands on!

What you have to do?

- Rename `BadNameCounter` using `RenameLinearCountingProcessor`
- Create a processor to detect types of `privateapi` package that are available through `publicapi` package
- (if you still have time...) Create a processor to transform setters of `Store` to a builder pattern