Analyzing Dart Language with Pharo: Report and early results

Nicolas Hlad, Benoit Verhaeghe, Mustapha Derras

Nicolas Hlad

Benoit Verhaeghe

Mustapha Derras

Presentation Plan

1. Industrial and Academic contexts

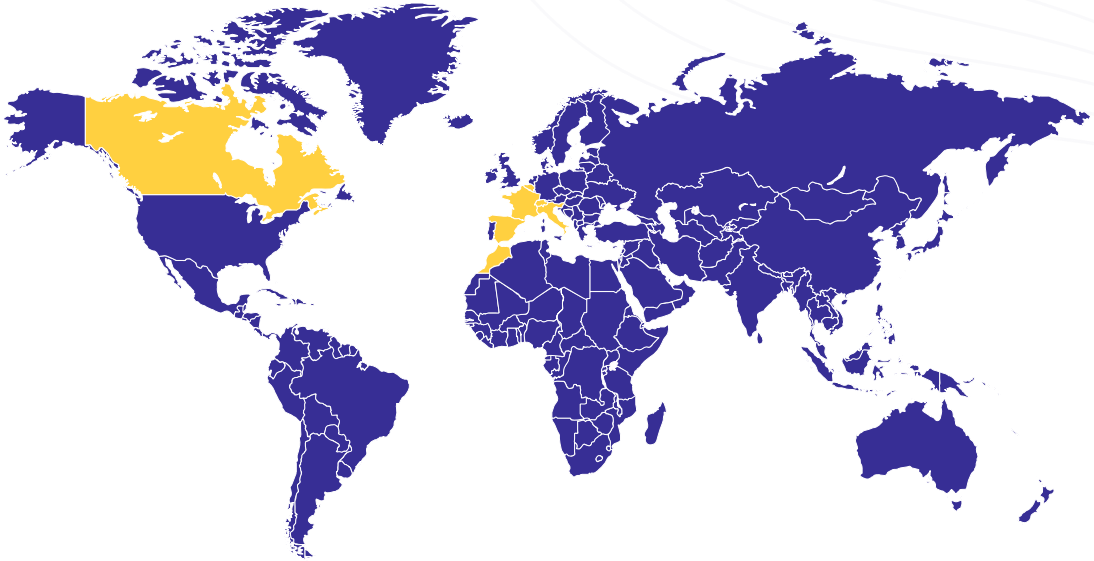
2. What is Dart ?

3. What is our analysis process of Dart ?

4. Conclusion

Berger-Levrault

- **created 1463** as a printing company
- **now** an international software editor



- Develop administrative software for cities administrations, industries, hospitals, etc.
- Since 2022, in collaboration with Evref
- Compagnies with divers technologies :

PowerBuilder Java TS Pharo Flutter

Flutter and its programming language Dart

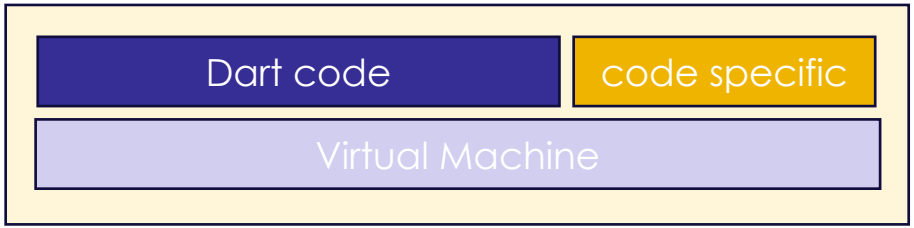
Flutter

since 2014

iOS & Android PC, MacOS, Linux Web*

Dart

since 2011



Code specific to the platform : *Java, Swift, C#, JS, etc.*

** On web, the Dart code is compiled to JS Compiler or WebAssembly*

Dart Grammar (quick overview)

i) Object oriented and function Programming

```
3   //class OOP
4 > class MyNumber {…
36
37  // function
38 > void execute() {…
```

ii) Language features and *syntax sugar*

```
4   class MyNumber {
5     late final int integer;
6
7     // parm 'value' marked as optional, with a default value of 0.
8     MyNumber({int value = 0}) : this.integer = value;
9     // LoC#9 is equivalent to :
10    // Number(int value = 0) {
11    //   this.integer = value;
12    // }
13    // Number() {
14    //   this.integer = 0;
15    // }
16
17    MyNumber add(int n) => MyNumber(value: n + integer);
18    // LoC#22 is equivalent to :
19    // int add(int n) { return new MyNumber(integer: n + integer); }
```

iii) A "loosely" Typed language

```
40  void execute() {
41    // <var> infers and locks Types at runtime
42    var x = 5; // x is an int
43    var y = MyNumber(); // y is a MyNumber
44    // Type can be specify
45    MyNumber z = MyNumber(value: x); // z is a MyNumber
46    // <dynamic> allows Type to change at runtime
47    dynamic i = 3; // i is an int
48    i = z.add(4); // i is now a MyNumber
```

Why do we need to analyze Dart ?

1

Main language for Flutter

With the deprecation of Xamarin, Flutter has become the main SDK for native multiplatform development on mobile (and more)

2

Growing popularity

According to *spectrum.ieee.org*, Dart is the 15th trending language in 2022

3

no Software Engineering tool yet

Outside Dart toolkit, we note an absence of academic or industrial tools to analyze Dart code. With Platforms like EMF/ECORE focusing on older language (e.g., Java, C/C++, C#, etc.)

Overview of our analysis of Dart

Goal

To exploit and **reuse the tools' suit of Pharo**, which has shown to be a **reliable static analyzer** of legacy languages like Java, Delphi, and C and which have **maturity over industrial** use case

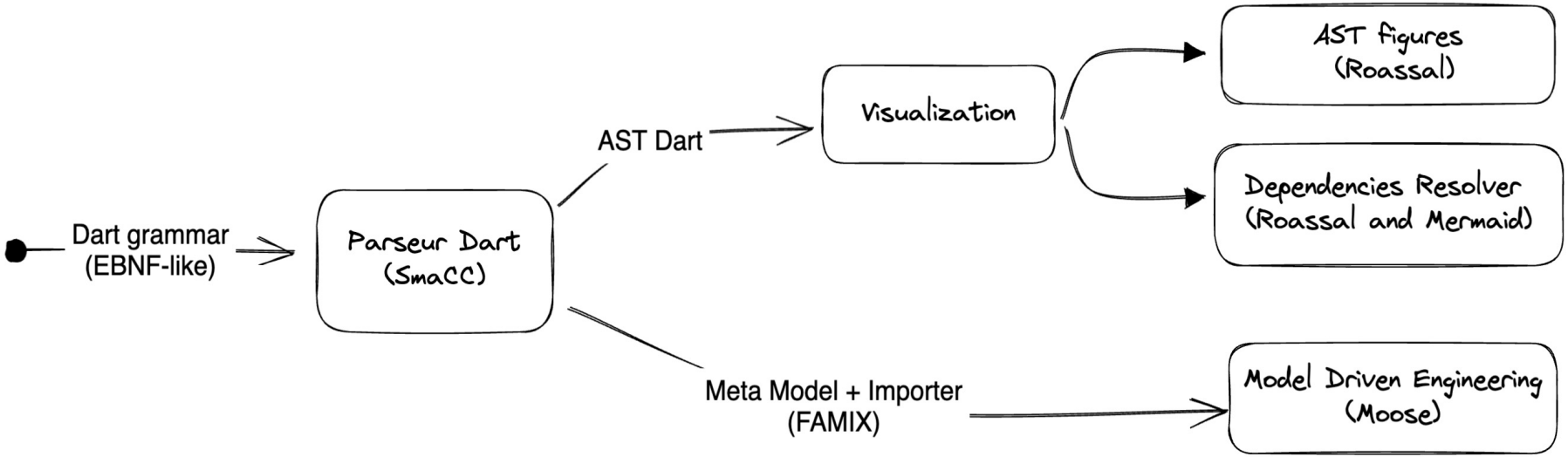


Figure – Analysis process set up for Dart in Pharo

Overview of our analysis of Dart

Goal

To exploit and **reuse the tools' suit of Pharo**, which has shown to be a **reliable static analyzer** of legacy languages like Java, Delphi, and C and which have **maturity over industrial** use case

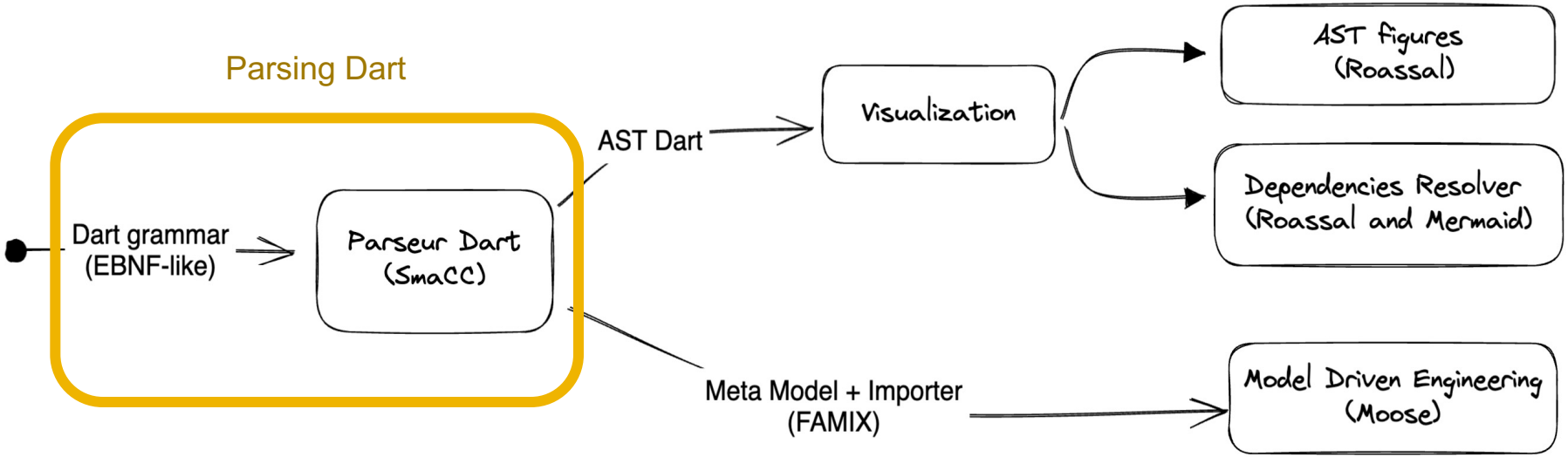


Figure – Analysis process set up for Dart in Pharo

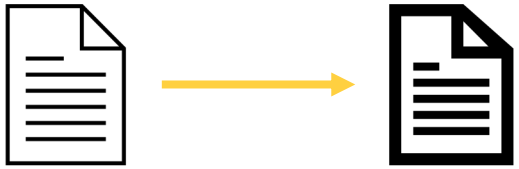
Generating a parser with SmaCC

EBNF of Dart
for ANTLR

ANTLR

```
classMemberDefinition
    :    methodSignature functionBody
    |    declaration  <SC>
```

Generating a parser with SmaCC



EBNF of Dart for ANTLR → EBNF of Dart for SmaCC

ANTLR

```
classMemberDefinition
    :    methodSignature functionBody
    |    declaration  <SC>
```

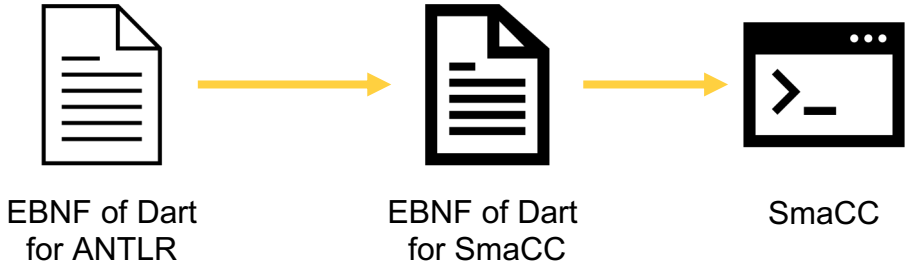
SmaCC

```
classMemberDefinition
    :    methodSignature 'methodSignature' functionBody 'functionBody' {{ClassMemberDefinition}}
    |    declaration 'declaration' <SC> {{ClassMemberDefinition}}
    ;
```

'Attribut_of_the_node'

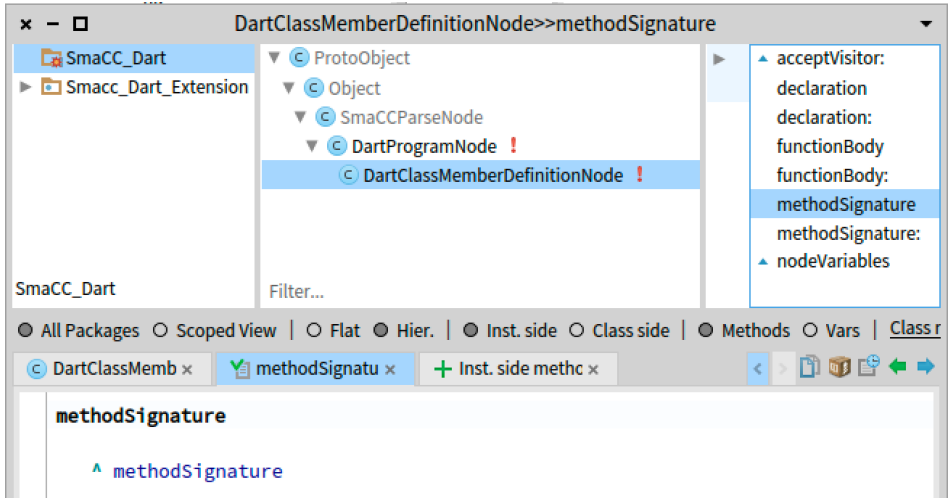
{{declared_SmaCC_Node}}

Generating a parser with SmaCC

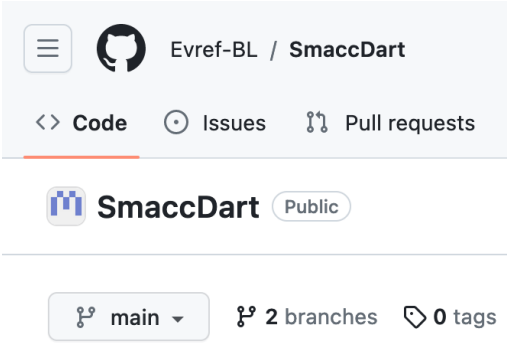



```
definition := ('path\to\grammar\dart.g' asFileReference contents).
grammarCompiler := SmaCCGrammarCompiler new.
grammarCompiler codeGenerator defaultCategory: 'SmaCC_Dart'.
grammarCompiler
        buildDefinition: definition;
        compileInto: 'DartScanner' andParser: 'DartParser'.
```

Generating a parser with SmaCC



https://github.com/Evref-BL/SmaccDart
parser + AST visualisation & more…



limitations of SmaCCDart

1. Limited to the Dart2 grammar (by ANTLR)
2. No official EBNF grammar by Google a its specification (with ambiguities reported)
3. We use code-refactoring to handle ***multiline string interpolations*** and *performance issues*

```
10  void main(List<String> args) {
11    var hello = 'hello';
12    var s = ''' this a
13    looooooo...
14    ooooooooo...
15    ooooooong
16    way to say $hello''';
17  }
```

```
void main(List<String> args) {
  var hello = 'hello';
  var s  = ''+' this a '+'\n'+'  looooooo...'+'\n'+'  ooooooooo...'+'\n'+'  ooooooong'+'\n'+'  way to say $hello'+'\n';
}
```

limitations of SmaCCDart

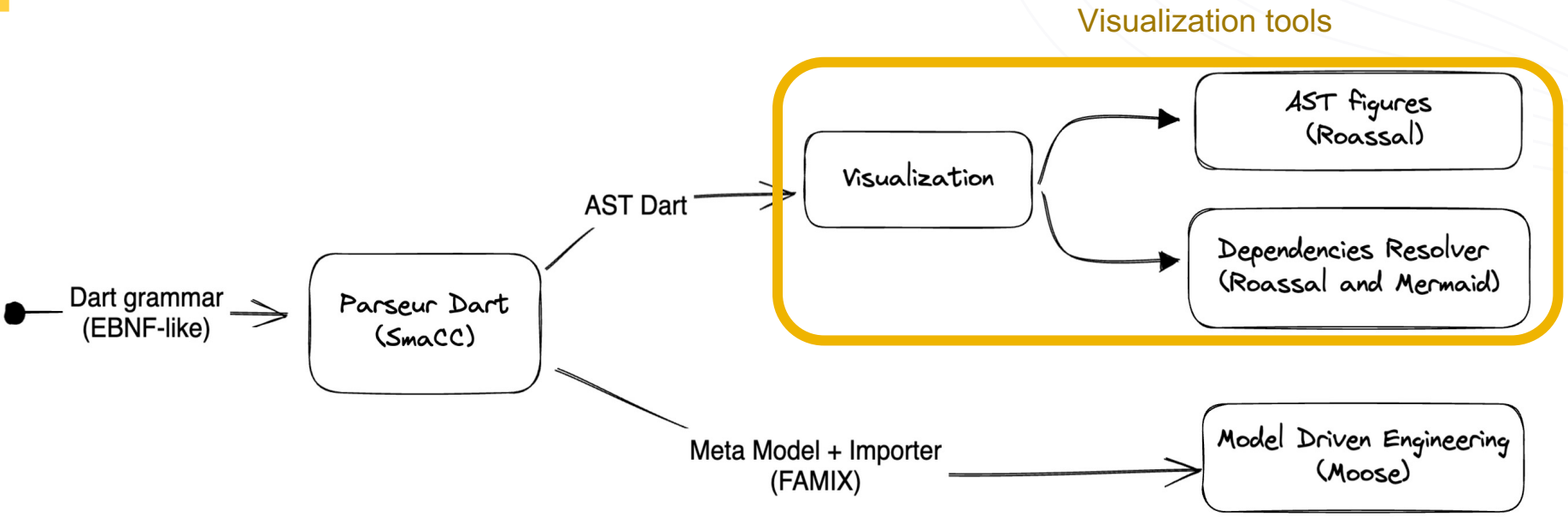
1. Limited to the Dart2 grammar (by ANTLR)
2. No official EBNF grammar by Google a its specification (with ambiguities reported)
3. We use code-refactoring to handle *multiline string interpolations* and ***performance issues***

```
@override
Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
   title: Text('Addition App’),
  ),
  body: Center(
   child:
    TextField(labelText: 'Number 1'),
    TextField(labelText: 'Number 2’),
  ),
  ElevatedButton(
   onPressed: performAddition,
   child: Text('Add’),
  )
 );
}
```

```
@override
Widget build(BuildContext context) {
 return new Scaffold(
  appBar: new AppBar(
   title: new Text('Addition App’),
  ),
  body: new Center(
   child:
   new TextField(labelText: 'Number 1'),
   new TextField(labelText: 'Number 2’),
  ),
  new ElevatedButton(
   onPressed: performAddition,
   child: new Text('Add’),
  )
 );
}
```

~5 times faster on a class 50 lines

Plan


Visualization tools
Dart grammar
(EBNF-like)
Parseur Dart
(SmaCC)
AST Dart
Visualization
AST figures
(Roassal)
Dependencies Resolver
(Roassal and Mermaid)
Meta Model + Importer
(FAMIX)
Model Driven Engineering
(Moose)

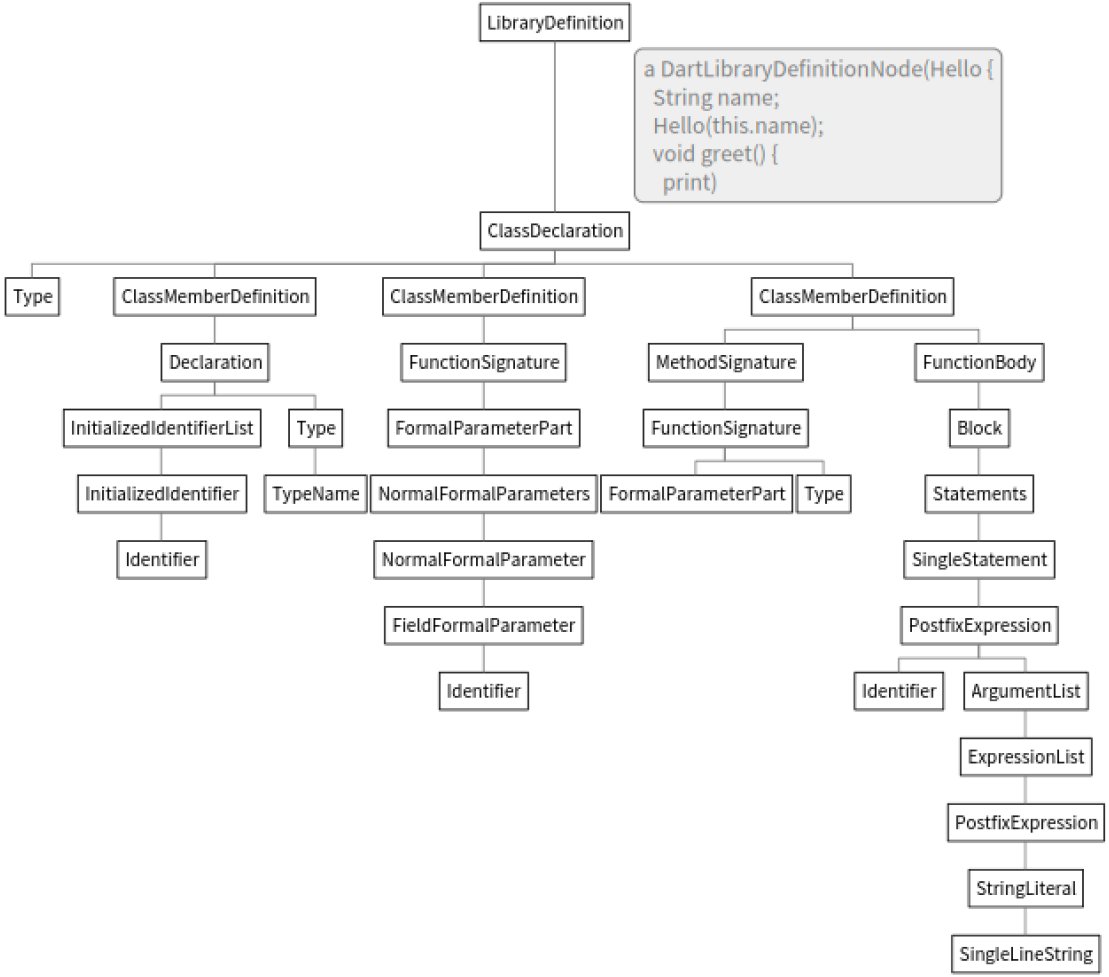


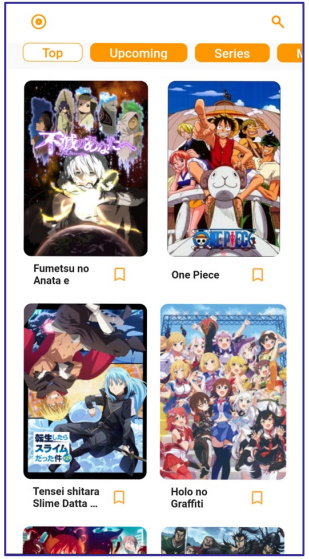
Figure – AST visualization made with Roassal3

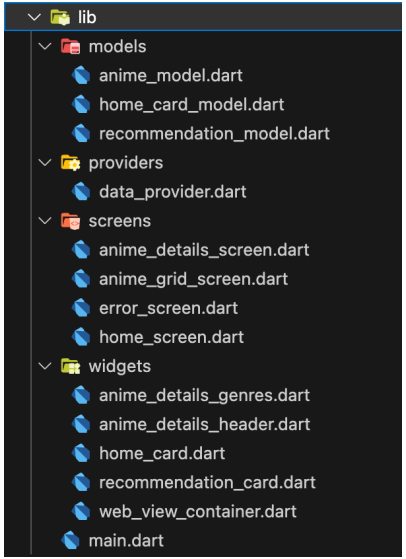
Experimenting visualisations

Testing SmaCCDart on a Flutter App

Use case

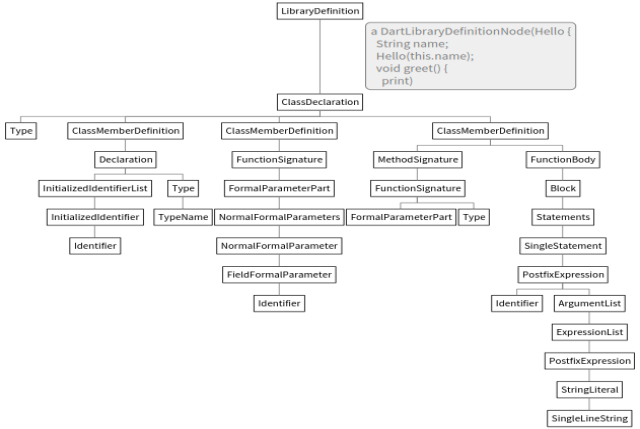
Using the parser to extract the file dependencies of an opensource flutter app





AnimSearch: 14 dart files & 2178 LoC

for each file



Extract its corresponding an AST

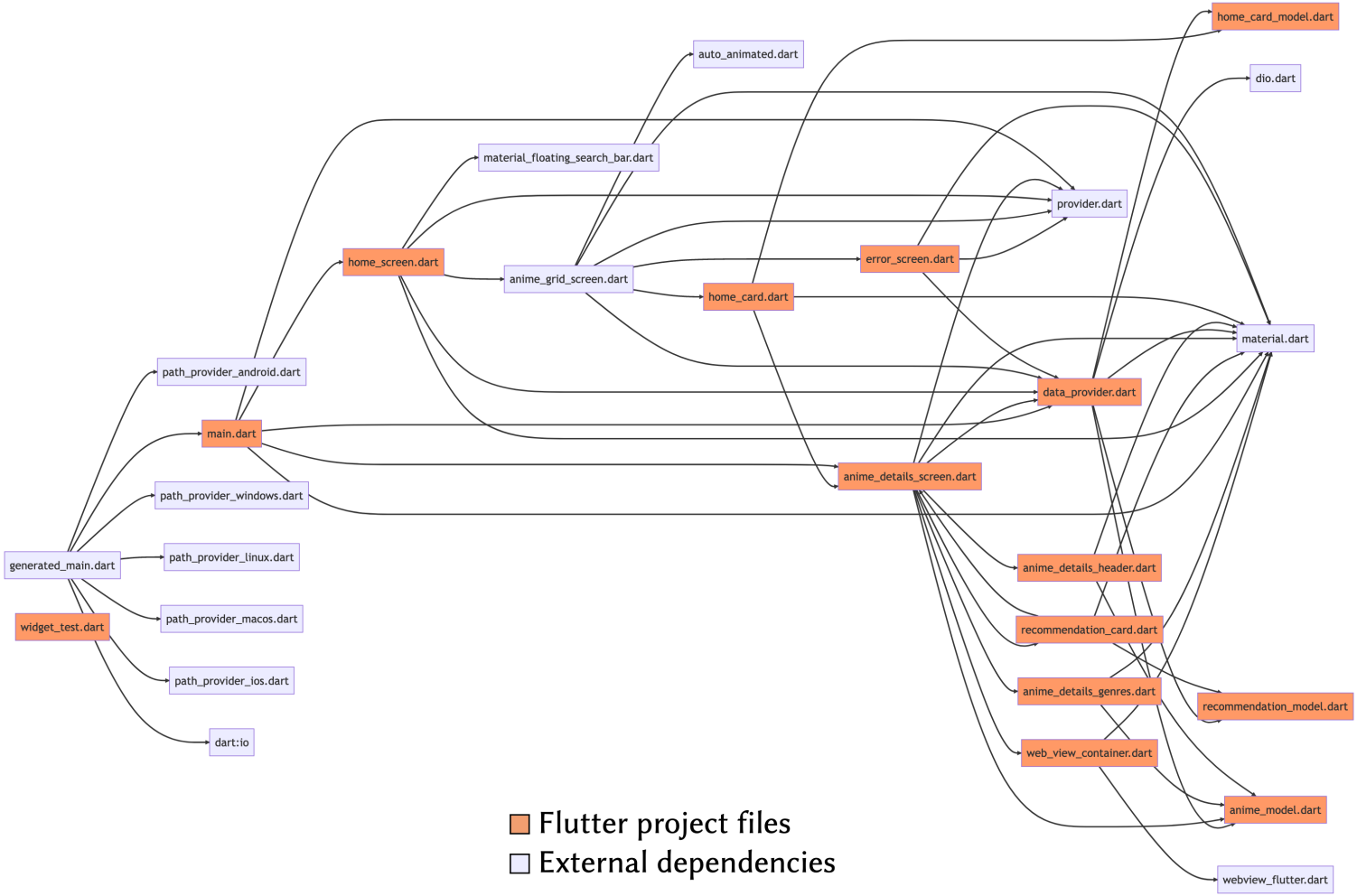


Figure – a mermaid visualization of the file dependencies of AnimSearch

limitations

With an increasing number of entities to display, visualization becomes hard to understand.
Thus, we rely on model driven engineering to continue the analysis.

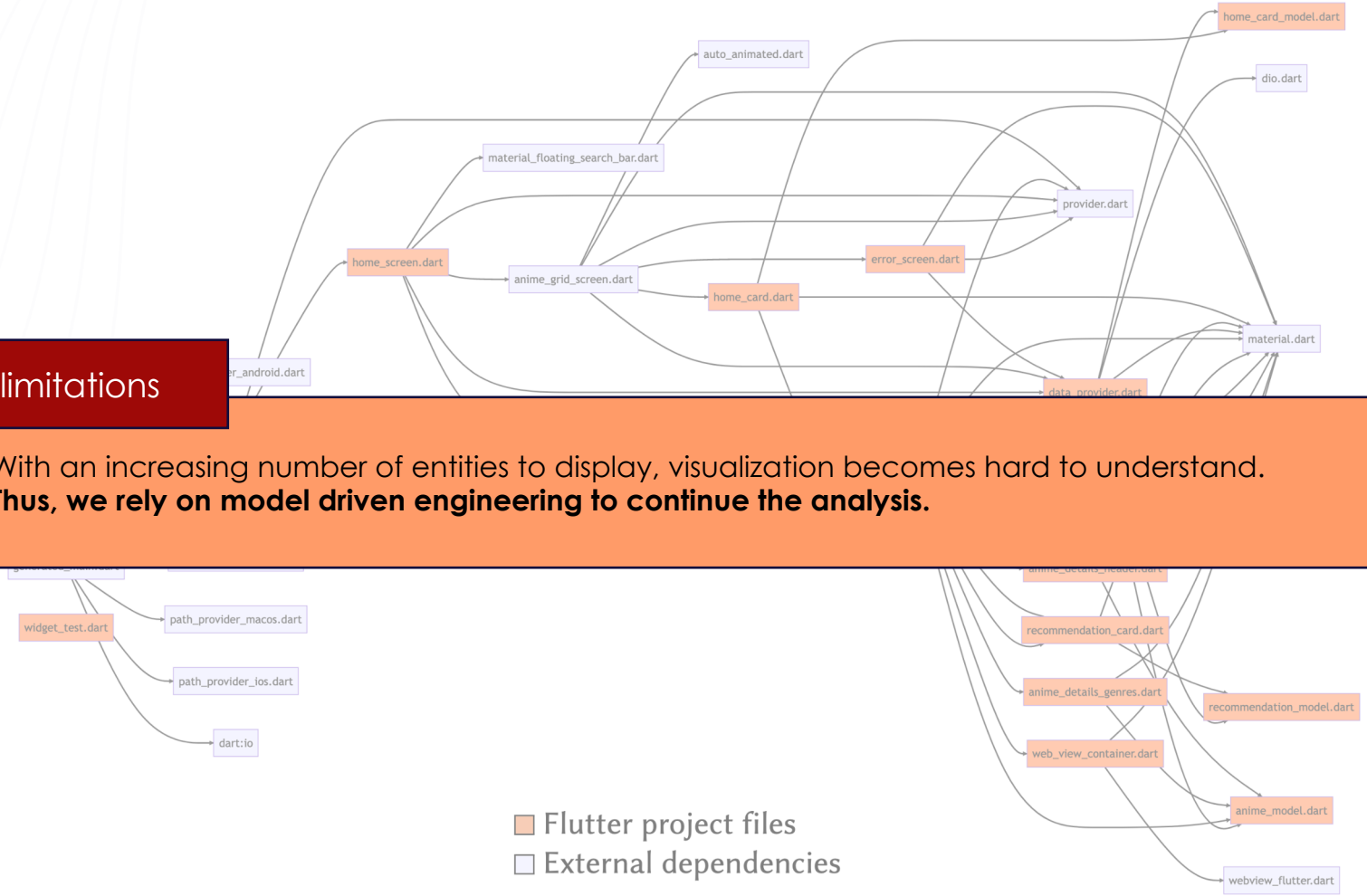
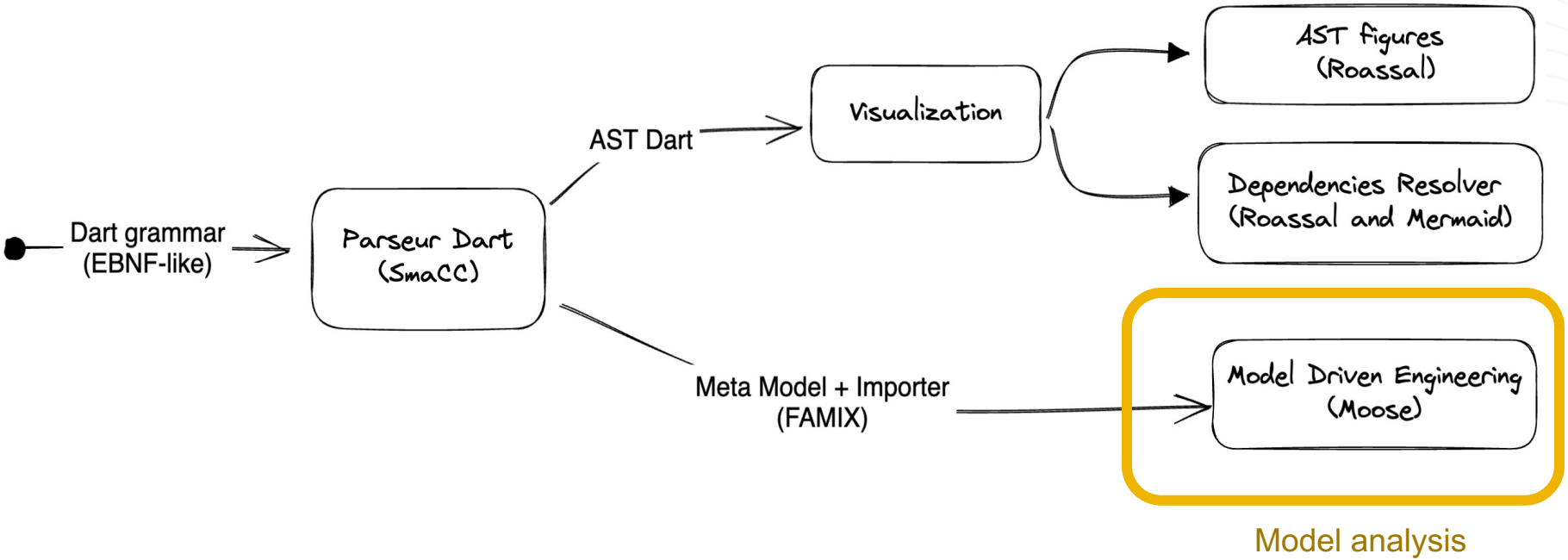


Figure – a mermaid visualization of the file dependencies of AnimSearch

Plan

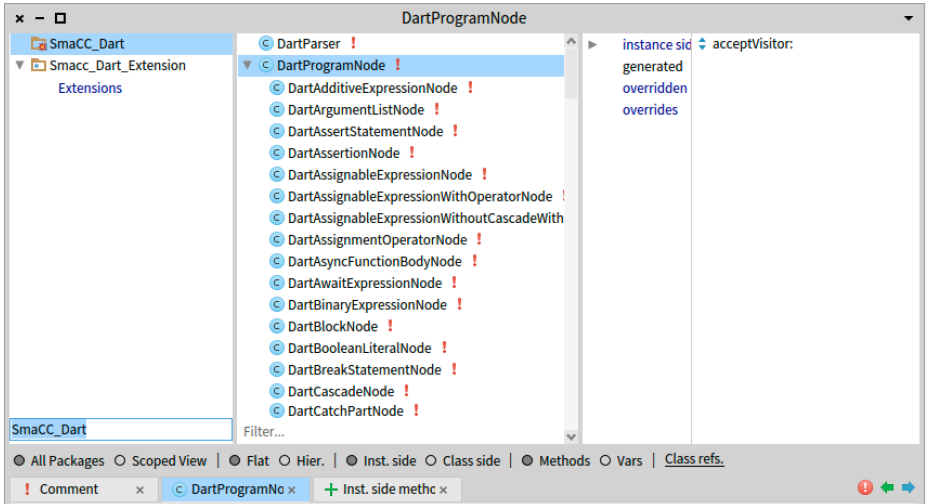


Model analysis

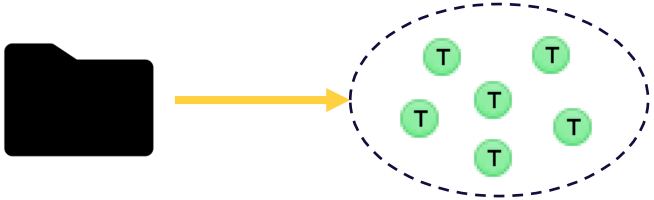
Defining a Famix Meta Model for Dart

Package Pharo
SmaccDart Parser

134 SmaCCDart nodes

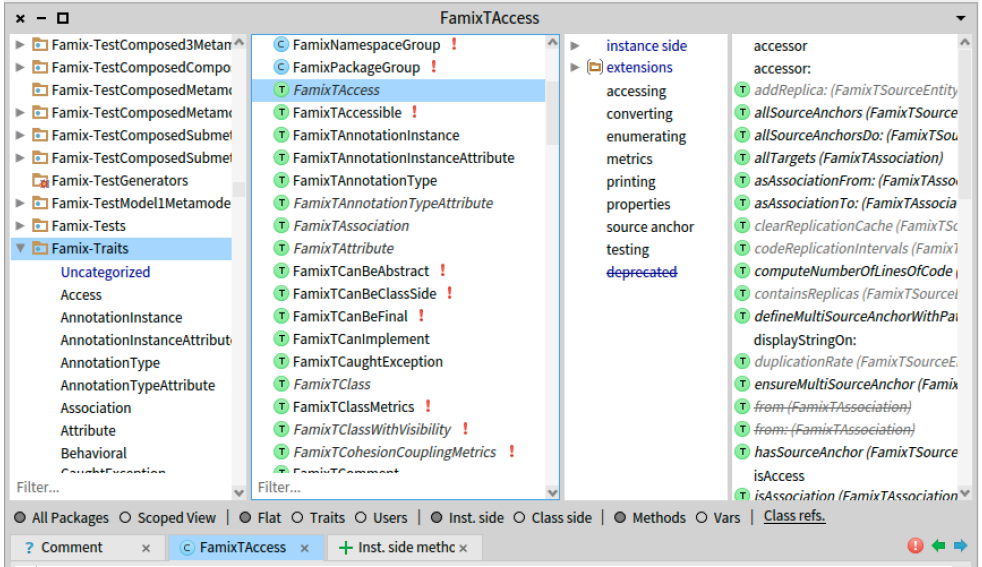


Defining a Famix Meta Model for Dart

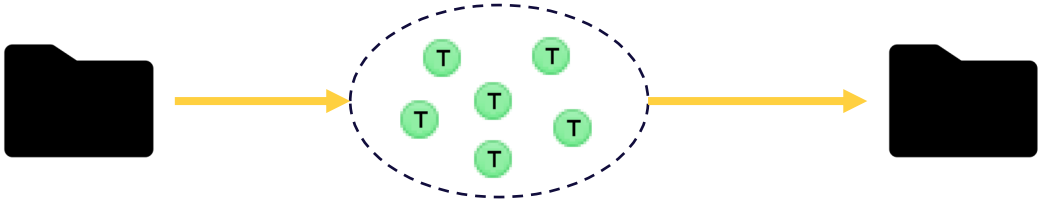


Package Pharo
SmaccDart Parser

Famix Traits



Defining a Famix Meta Model for Dart



Package Pharo SmaccDart Parser

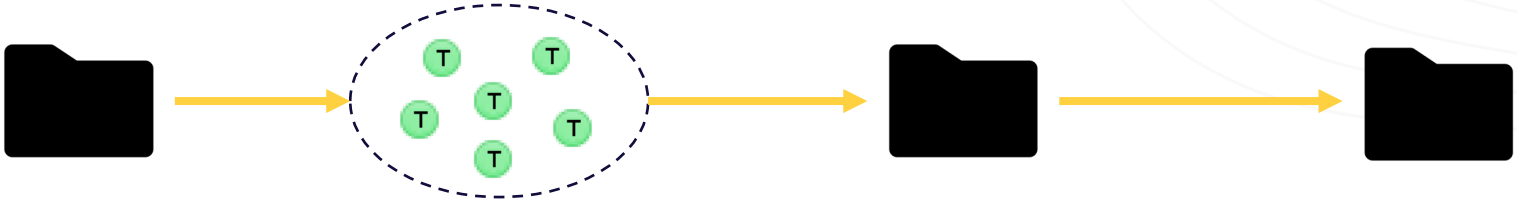
Famix Traits

Meta-model Generator

```
defineClasses

    super defineClasses.

    class := builder newClassNamed: #Class.
```

```
defineHierarchy

    super defineHierarchy.

    class --|> #TClass.
    class --|> #TType.
    class --|> #THasVisibility. "if begins with underscore _"
    class --|> #TCanImplement.
    class --|> #TWithMethods.
    class --|> #TWithAttributes.
```

Defining a Famix Meta Model for Dart



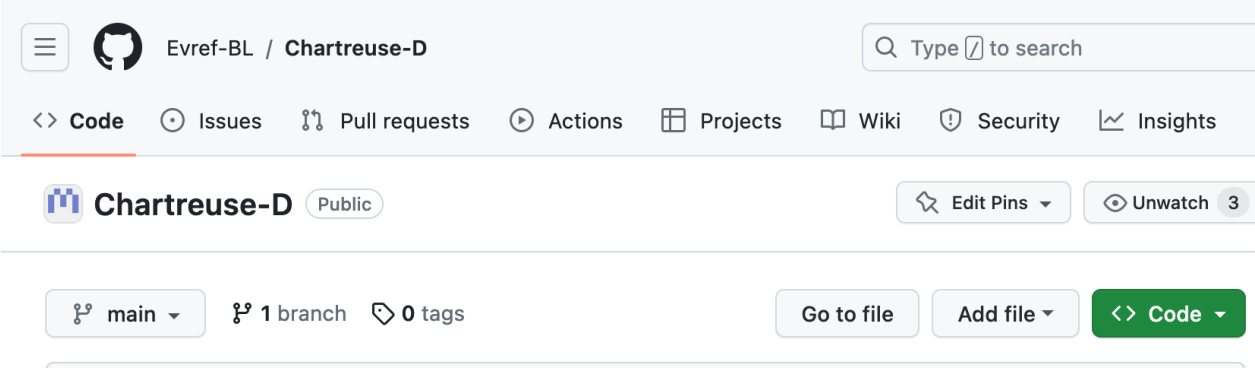
Package Pharo SmaccDart Parser

Famix Traits

Meta-model Generator

FamixDart meta-model Chartreuse-D

https://github.com/Evref-BL/Chartreuse-D
Meta-model Generator + model importer



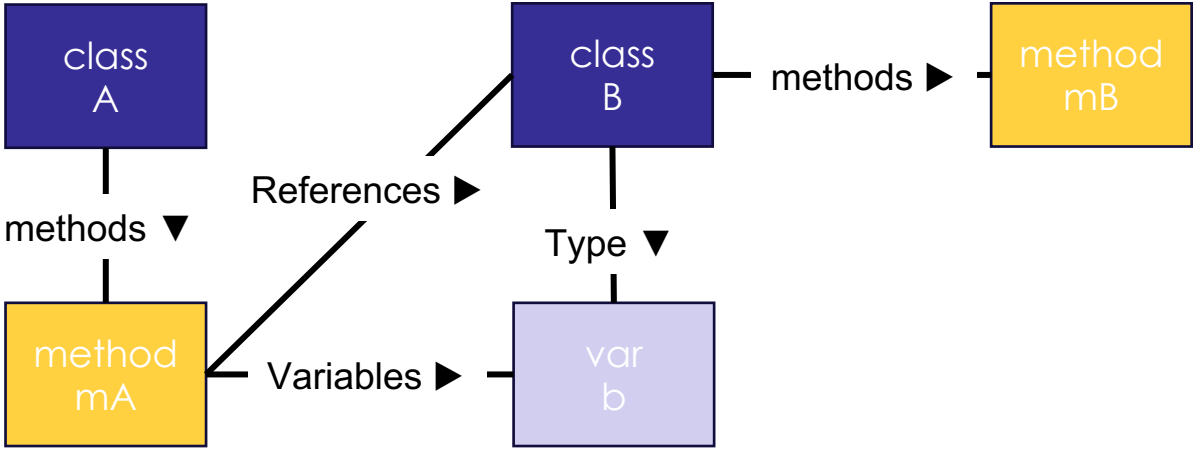
FamixDart Importer in an example

```
1   class A {
2     mA() {
3       var b = B();
4       b.mB();
5     }
6   }
7
8   class B {
9     mB() {
10      //...
11    }
12  }
```

1. Create instances (from its AST)

FamixDart model's entities

2. Symbolic Resolver from the AST



Entities with associations

Challenge to import FamixDart model

```
1   class A {
2     m({dynamic a, dynamic b}) {
3       print('from A');
4     }
5   }
6
7   class B {
8     m({dynamic a}) {
9       print('from B');
10    }
11  }
12
    Run | Debug
13  void main(List<String> args) {
14    dynamic t = A();
15    t = B();
16    //how to staticly resolve this invocation ?
17    t.m(a: 1);
18  }
```

- *dynamic* introduces ambiguity for symbolic resolution.

- For **A**,**B** two classes, with each a method **m()**
- In **A**, **m()** as **two** optional parameters (**a**,**b**)
- In B, **m()** as **one** option parameter (**a**)

At line 17,
how to resolve which method is called between A::m() and B::m() ?

Conclusion

Take away points

- We explore how a new language like Dart can be analyzed by existing tools in Pharo, such as:
  1. **SmaCC** for parser
  2. **Roassal3** for visualizations
  3. And **Famix** for model driven engineering.

- We extend those tools to develop :
  1. A parser, **SmaCCDart**
  2. And a Famix Meta-model for Dart, **FamixDart**
  3. An famixDart import, **Chartreuse-D**

- Our tools are open sources with repository already available.

Future works (late 2023)

- On SmaCCDart
  - adding no regression tests
  - removing the source code refactoring 🤕
- On FamixDart
  - continuing the metamodel
  - handling dynamic types when importing associations
- … and in some years
  - having a Flutter app metamodel (i.e. handling platform specific code in Flutter)

From dart modeling to Flutter modeling

```
4   class AnimeDetailsGenres extends StatelessWidget {
5
6     Widget moveLabel(String text, dynamic pokeData) {
7       return Container(
8         decoration: BoxDecoration(
9           color: Colors.orange,
10          borderRadius: BorderRadius.circular(15),
11        ), // BoxDecoration
12        child: Center(
13          child: Text(
14            text,
15            style: TextStyle(
16              fontWeight: FontWeight.w600,
17              color: Colors.white,
18              shadows: <Shadow>[
19                Shadow(
20                  offset: Offset(2, 2),
21                  blurRadius: 7,
22                  color: Colors.grey,
23                ), // Shadow
24              ], // <Shadow>[]
25            ), // TextStyle
26          ), // Text
27        ), // Center
28      ); // Container
29    }
```

- Example of **AnimeDetailsGenres** widget
- Here we need to capture that :
  - A **Container** has a **Decoration** and a child
  - This child is **Center** widget
  - **Center** has a **Text** as child
  - Etc…
- None of these widgets are attributes of the class
- Yet, they define how **AnimeDetailsGenres** is composed.
- ***How to capture this information is our model ?***