FASTTypeScript

FASTTypeScript metamodel generation using **FAST** traits and **TreeSitter** project

Aless HOSRY, Benoit VERHAEGHE

Berger Levrault

Who are we?

badetitou.github.io

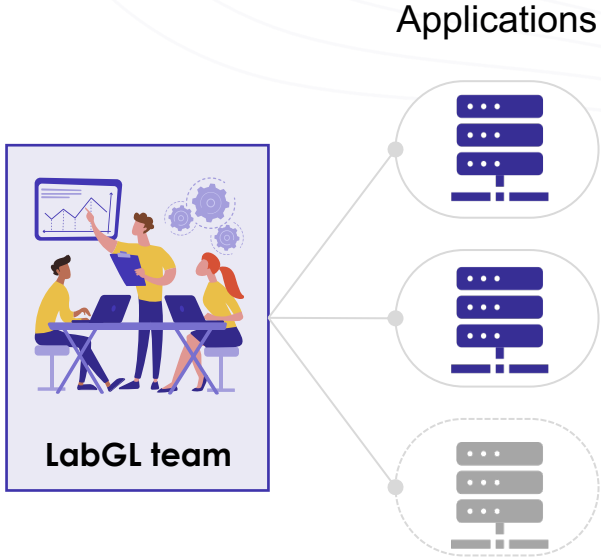
- Scientific project manager at Berger-Levrault
- Lyon, France, 2021 – Present
- R&D in software analysis, team supervision

Alesshosry.github.io

- Research Engineer at Berger-Levrault
- Lyon, France, 2025 – Present
- R&D in software analysis

Motivation

- **DRIT**: a department specialized in research and composed of researchers, PhD students, engineers and trainees.
- **LabGL**: a research team specialized in Software Analysis, by submitting scientific publications and creating prototypes and innovations.



Motivation

We use different materials, but we rely on: 

- **Moose**: Platform for software and data analysis in Pharo. Allows importing, parsing data, modelling, measuring, querying …
- **Famix**: Abstract representation of source code
- **FAST**: Famix Abstract Syntax Tree
- **MoTion**: Object pattern matching in Pharo
- …

Motivation

- At Berger-Levrault we use a variety of languages to develop our applications, for example:
- **Java** is used to develop back-end applications → LabGL team can develop tools using FASTJava and FamixJava.
- **TypeScript** is used to develop front-End applications → FASTTypeScript does not exist, LabGL team is blocked ☹

Problem

FASTTypeScript is missing :(

- Build a new metamodel
- Develop a new TypeScript parser

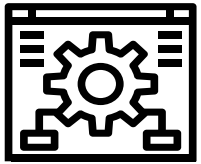
Methodology

Typescript code

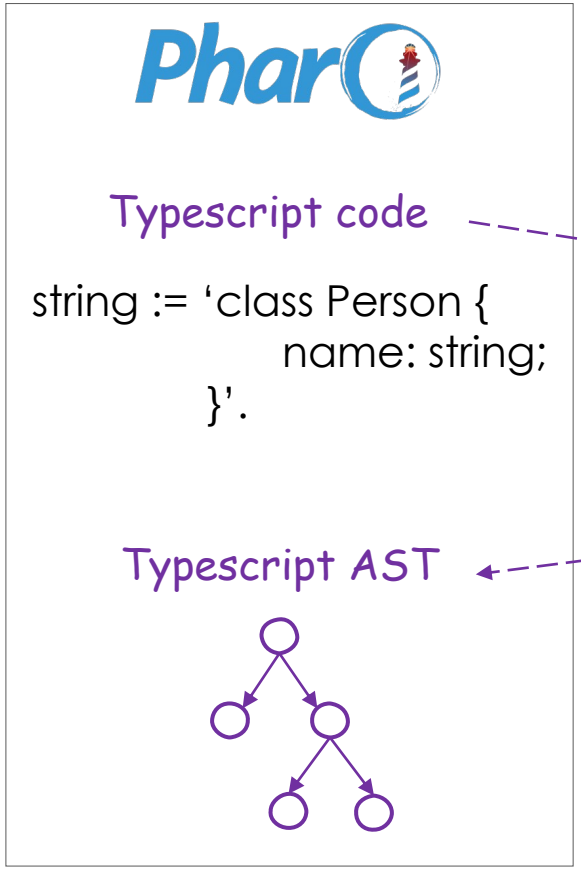
```
string := 'class Person {
            name: string;
          }'.
```

FFI protocol

External parser

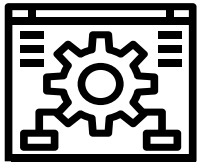


Methodology



FFI protocol

External parser



Solution

For parsing we chose **Tree-Sitter**:

- Provides **incremental parsing** based on Wagner et al. work (*)
- Able to parse many languages: Java, TypeScript, Python, C, C# …
- Open source: >20k stars (Tree-sitter) and >400 stars (Tree-sitter TypeScript)

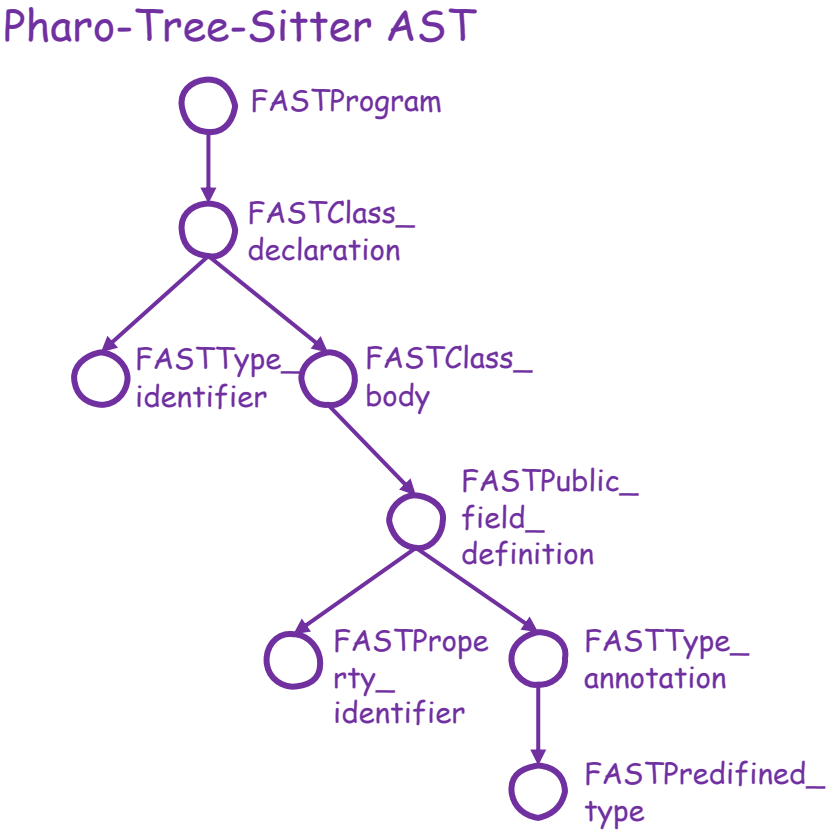
** Efficient and flexible incremental parsing Wagner et al.,1998*

Solution

- Integration in Pharo under **Pharo-Tree-Sitter**:

- Languages supported: TypeScript, Python, C, Mermaid, Groovy
- Cross-platform: Mac, Ubuntu, Windows
- Automated library generation for easy setup and CI/CD integration

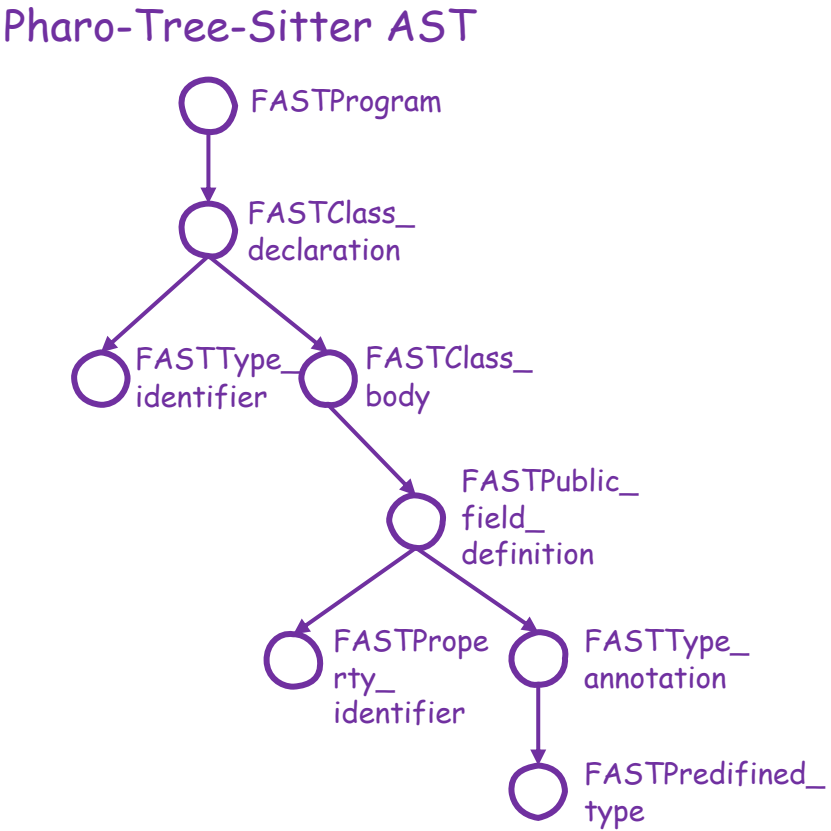
Solution




```
class Person {
    name: string;
}
```

✓ AST generated

Solution → Limited ☹

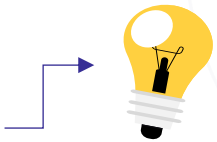



```
class Person {
    name: string;
}
```

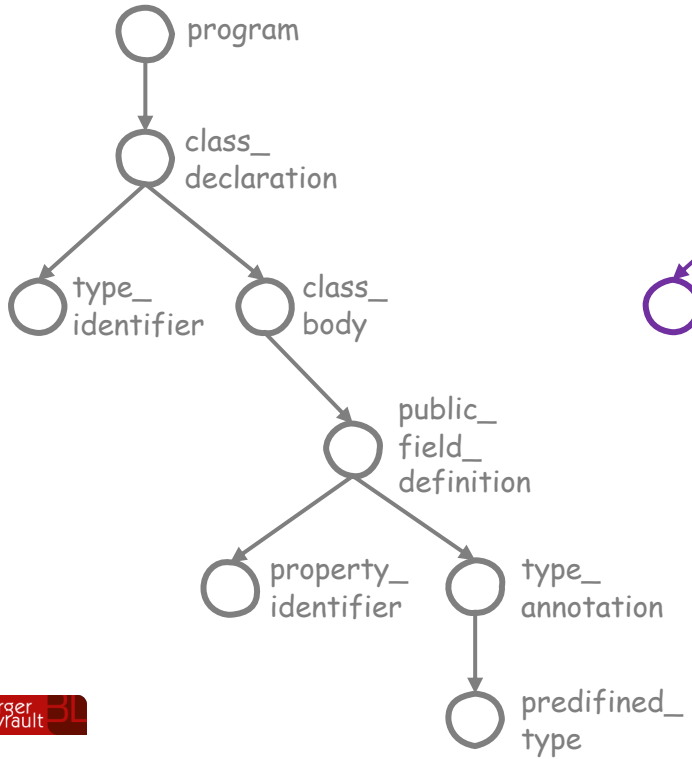
- ✓ AST generated

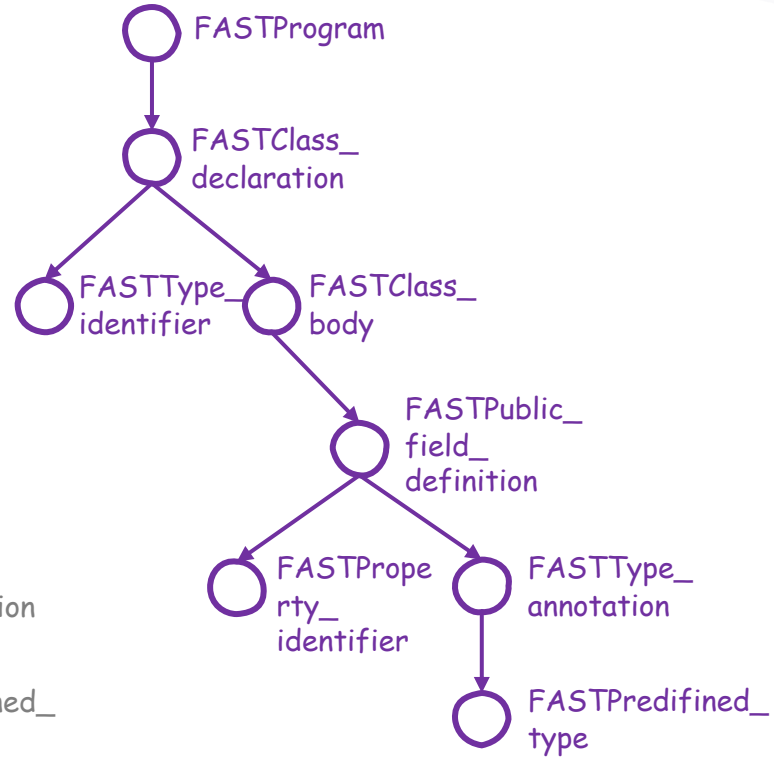
  Limited info per each node

Solution



Pharo-Tree-Sitter AST ⟶ FASTTypeScript



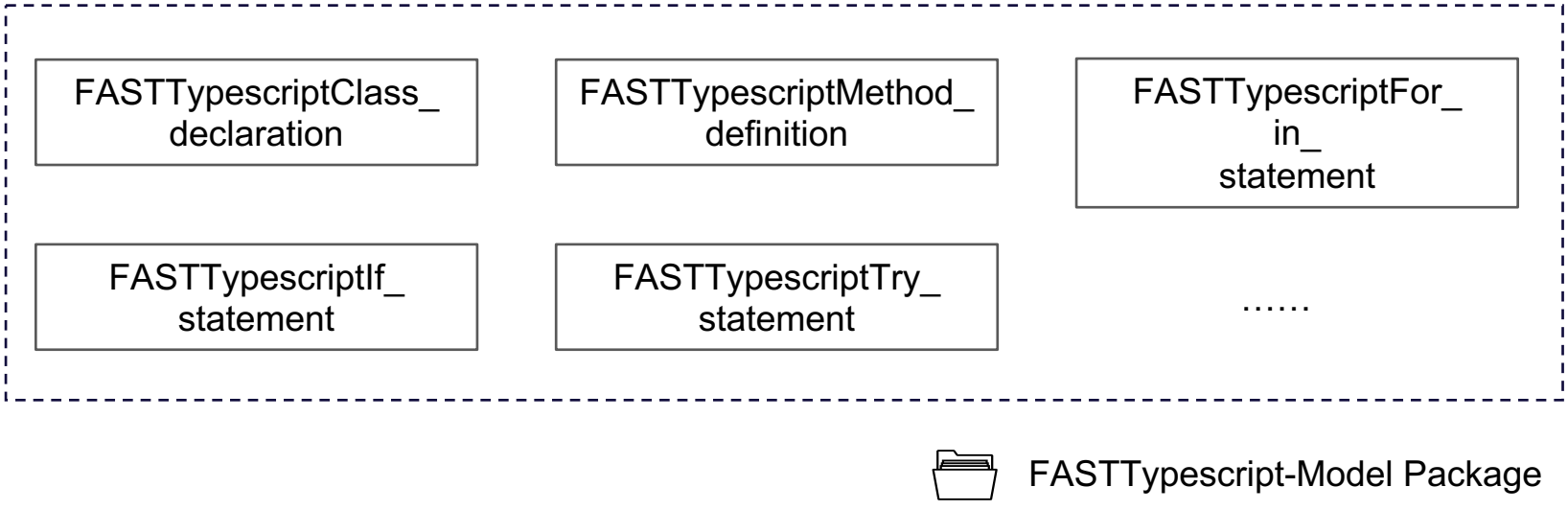



```
class Person {
   name: string;
}
```

- ✓ AST generated
- ✓ Info provided per each node

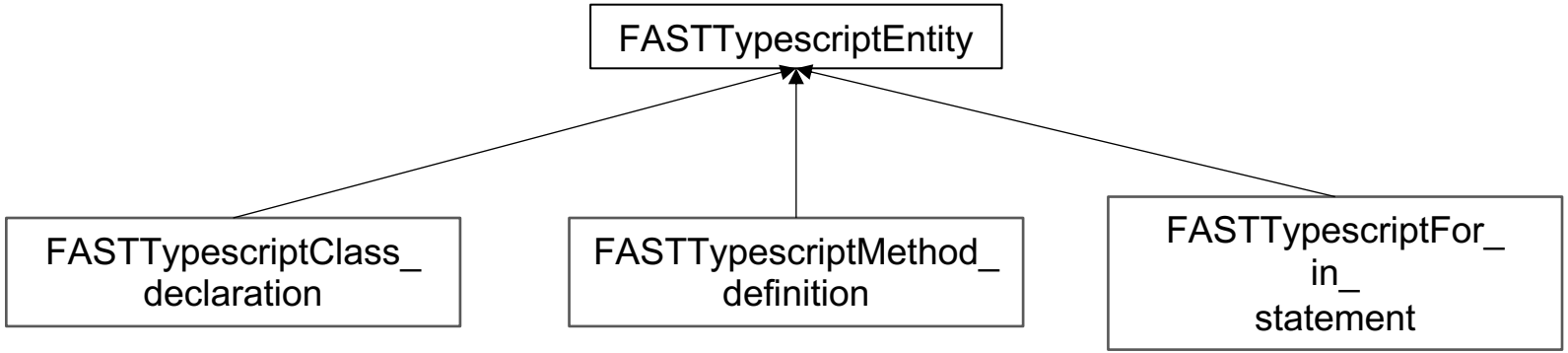
Solution

- Create first version of FASTTypeScript metamodel automatically based on the grammar of Tree-Sitter-TypeScript



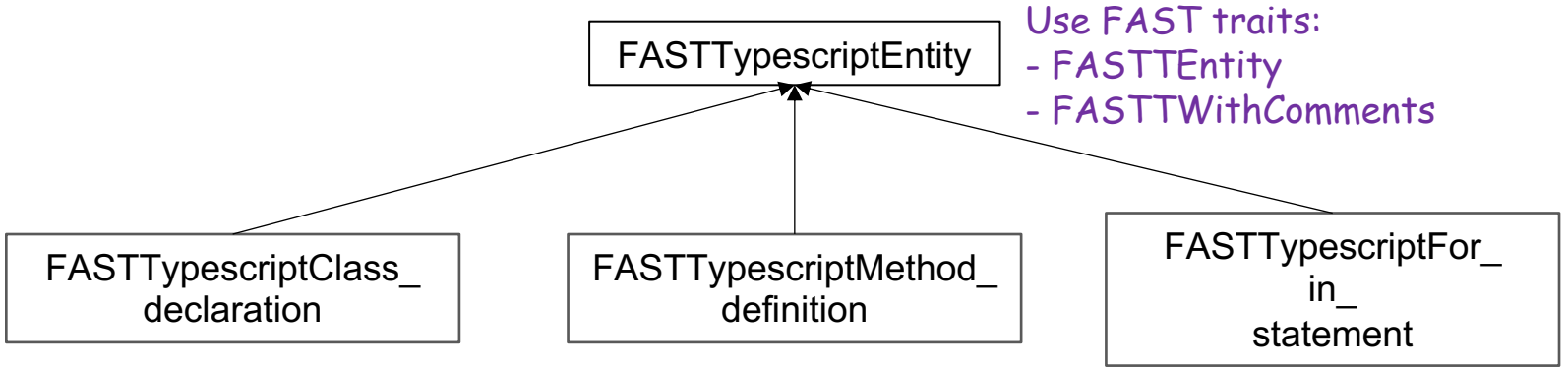
Solution

- Create first version of FASTTypeScript metamodel automatically based on the grammar of Tree-Sitter-TypeScript
- Add properties for each class to enrich the metamodel and use FAST traits



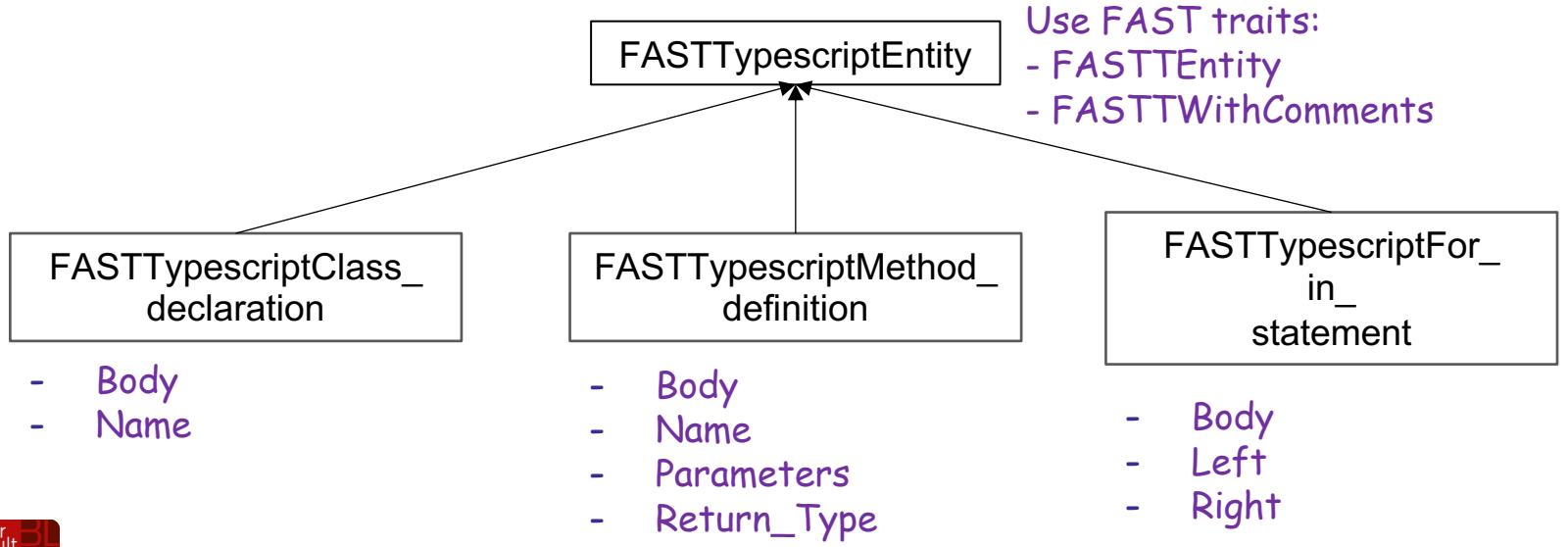
Solution

- Create first version of FASTTypeScript metamodel automatically based on the grammar of Tree-Sitter-TypeScript
- Add properties for each class to enrich the metamodel and use FAST traits



Solution

- Create first version of FASTTypeScript metamodel automatically based on the grammar of Tree-Sitter-TypeScript
- Add properties for each class to enrich the metamodel and use FAST traits



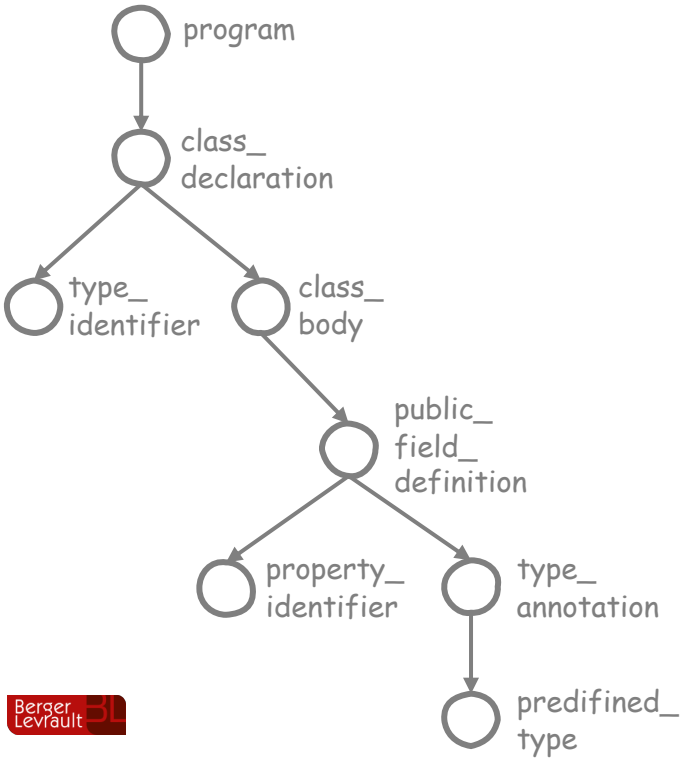
Solution

- Create first version of FASTTypeScript metamodel automatically based on the grammar of Tree-Sitter-TypeScript
- Add properties for each class to enrich the metamodel and use FAST traits
- Create an importer from Pharo-Tree-Sitter to FASTTypeScript

Solution

Pharo-Tree-Sitter AST

FASTTypeScript

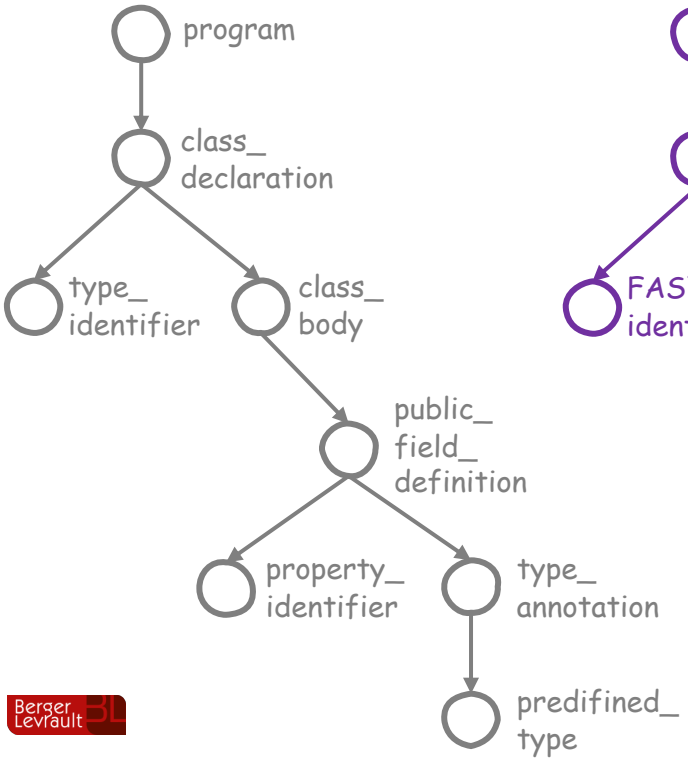


Importer main tasks:

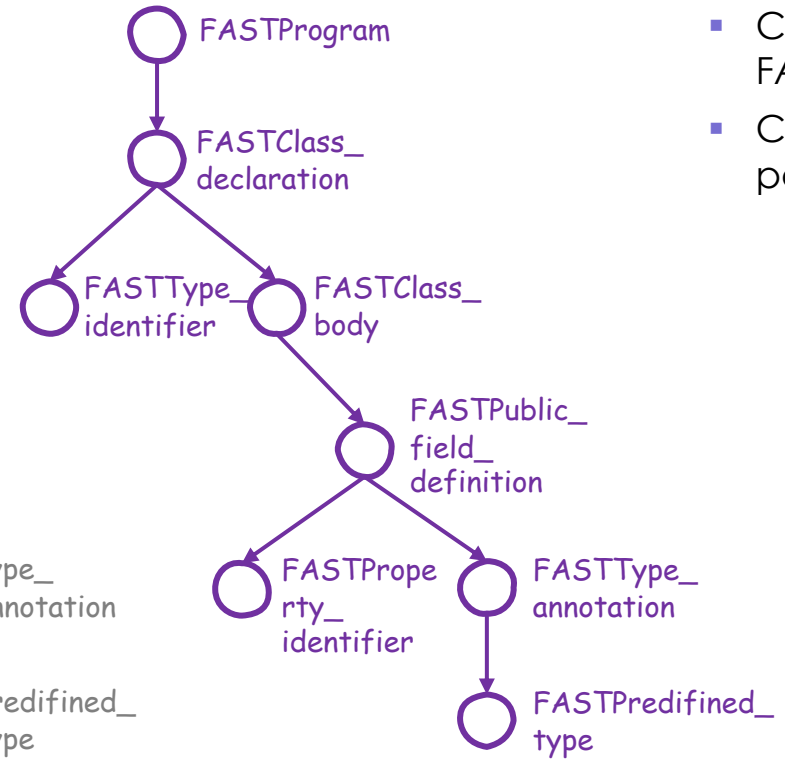
- Creates a new instance of FASTTypeScript

Solution

Pharo-Tree-Sitter AST



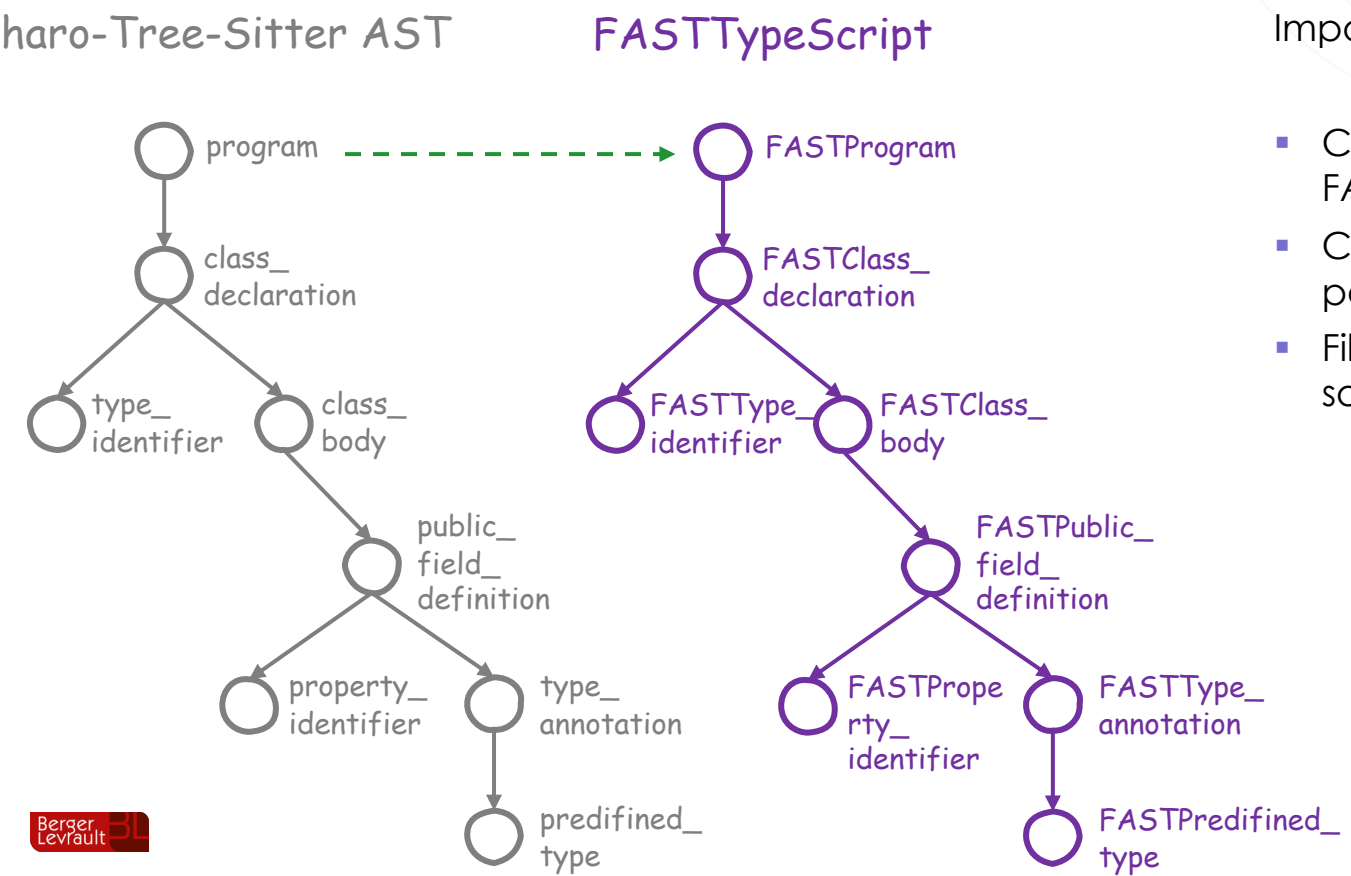
FASTTypeScript



Importer main tasks:

- Creates a new instance of FASTTypeScript
- Creates the nodes and set positions

Solution



Importer main tasks:

- Creates a new instance of FASTTypeScript
- Creates the nodes and set positions
- Fill the rootNode with the source code.

Demo

Use cases

- Methodology use case example:
  - Implementation of Famix metamodels for Perl C using Pharo-Tree-Sitter
- FASTTypeScript use case example:
  - 2 separate projects are using FASTTypeScript at BL
  - MoTion (pattern matching library) could be used with FASTTypeScript as a searching engine

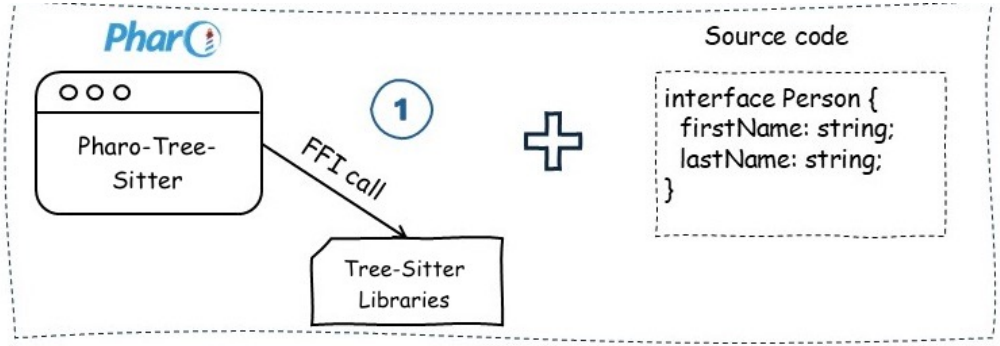
Use cases

MoTion pattern with FASTTypescript:

```
pattern := FASTTypescriptModel % {
   #entities <=>  FASTTypescriptMethod_definition % {
     #sourceCode <=>  'get\s+.*'.
     #children <=> FASTTypescriptStatement_block % {
         #children <=> { #'@child1'. #'*otherChilds'. }
     }.
   }
}.
```

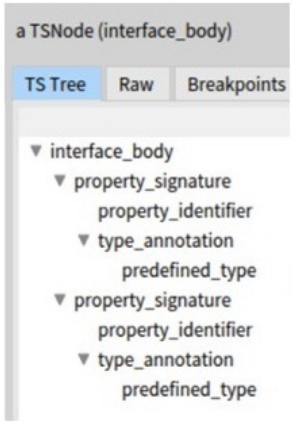
Future work

- Incremental parsing for FASTTypeScript
- Cover all properties: 30% already covered
- Carrefour FamixTypeScript FASTTypeScript
- Implementation of more BL projects that depends on FASTTypeScript
- Perhaps FASTNewLanguage ?



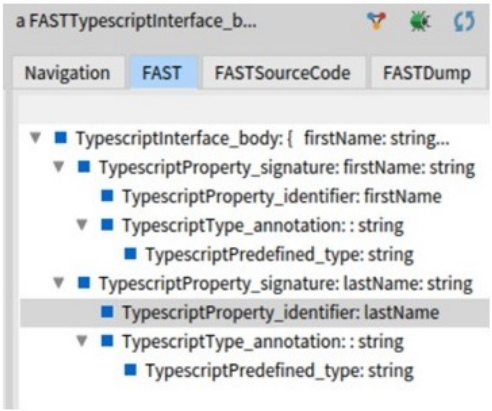
2

Parse and generate **TreeSitter** model



3

Import and generate a **FASTTypescript** model



Thank you for your attention