“

The Chisel Method for building parsers

Let’s start with an experiment

I guess you are thinking

- Parsers? What a boring topic. Parsing is a solved problem!
- I learned what I needed to learn about parsing 10 or 20 or 30 years ago at the university

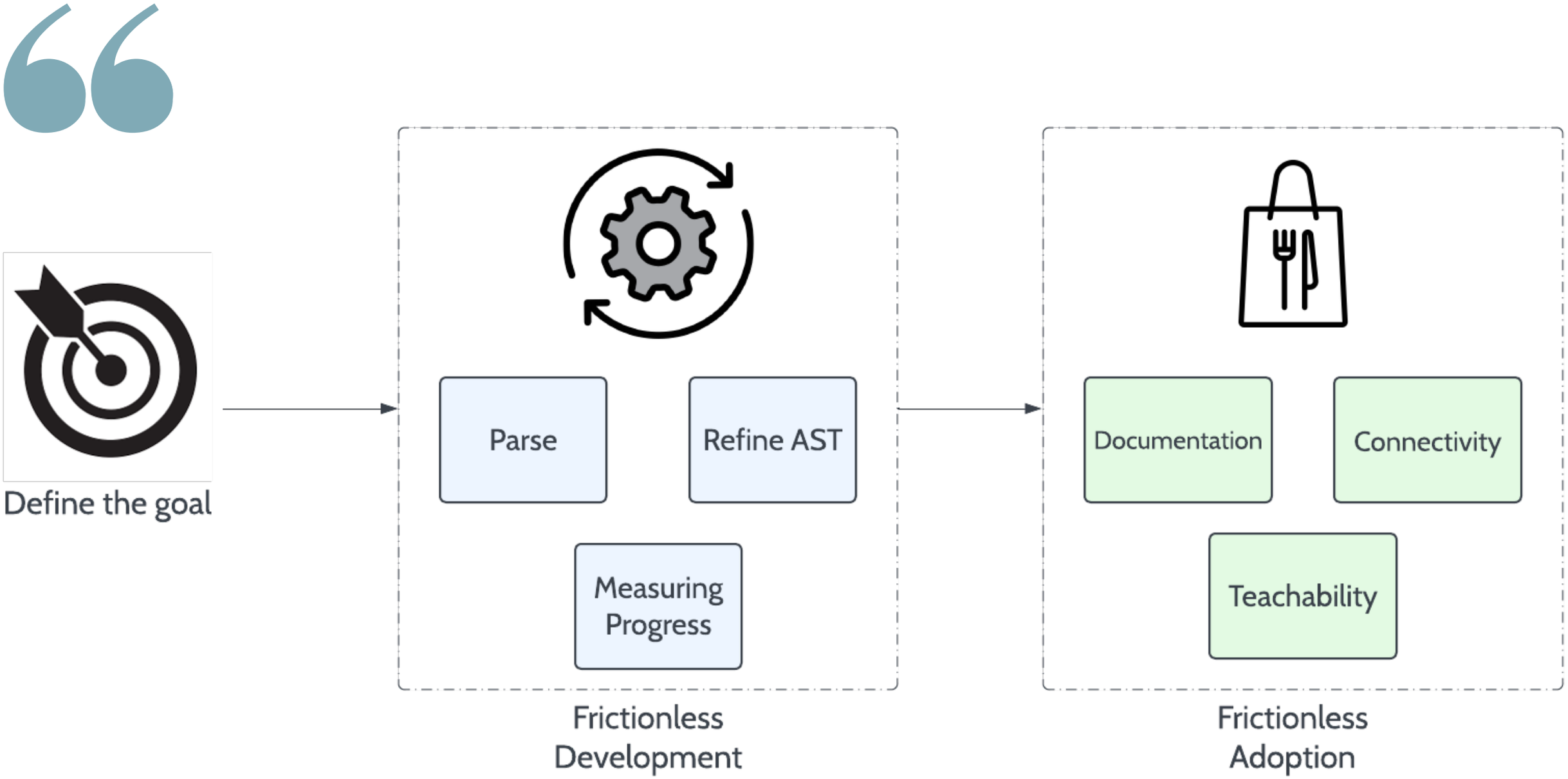
Like flying!

Flying: same but different

What is wrong with building parsers?

- It is perceived by many as a black magic art
- Clients have not a clear idea of what a parser does
- It is a trial and error process
- It takes a lot of time to train people
- There are repetitive tasks involved
- Parsers are not easy to integrate

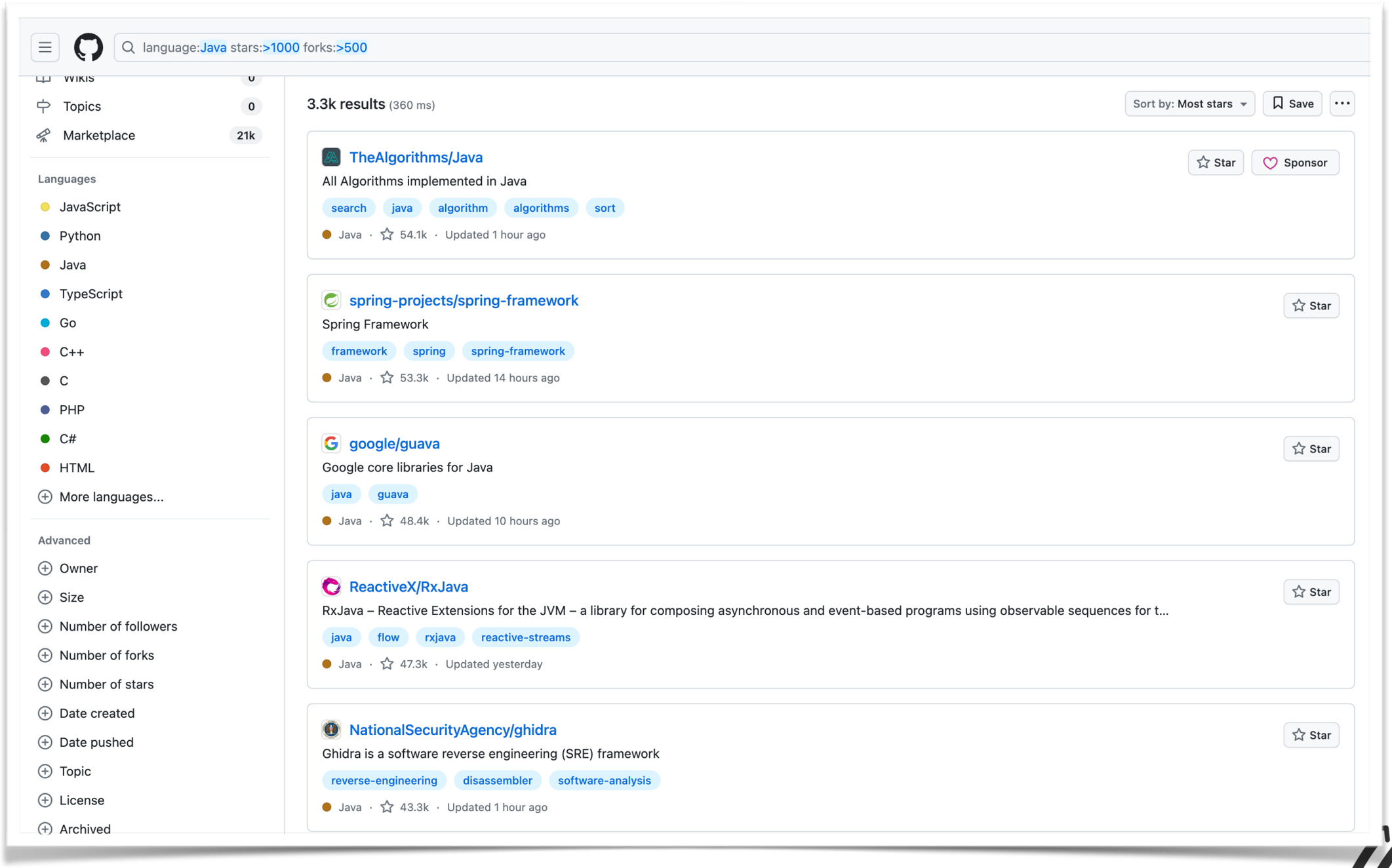
What can we do?



The Chisel Method

The Chisel Method - Goal setting

Start with the end in mind



Quality checks

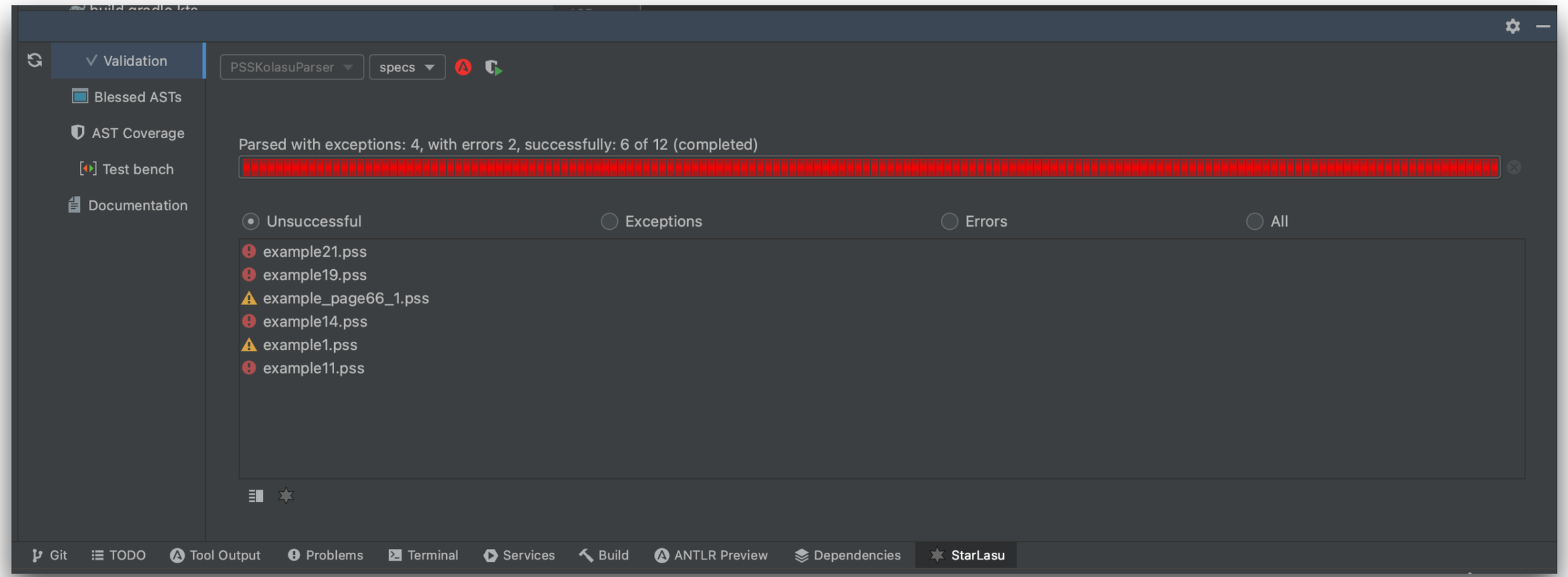
1. We can produce an AST for each single valid example.

   This ensures we can parse.

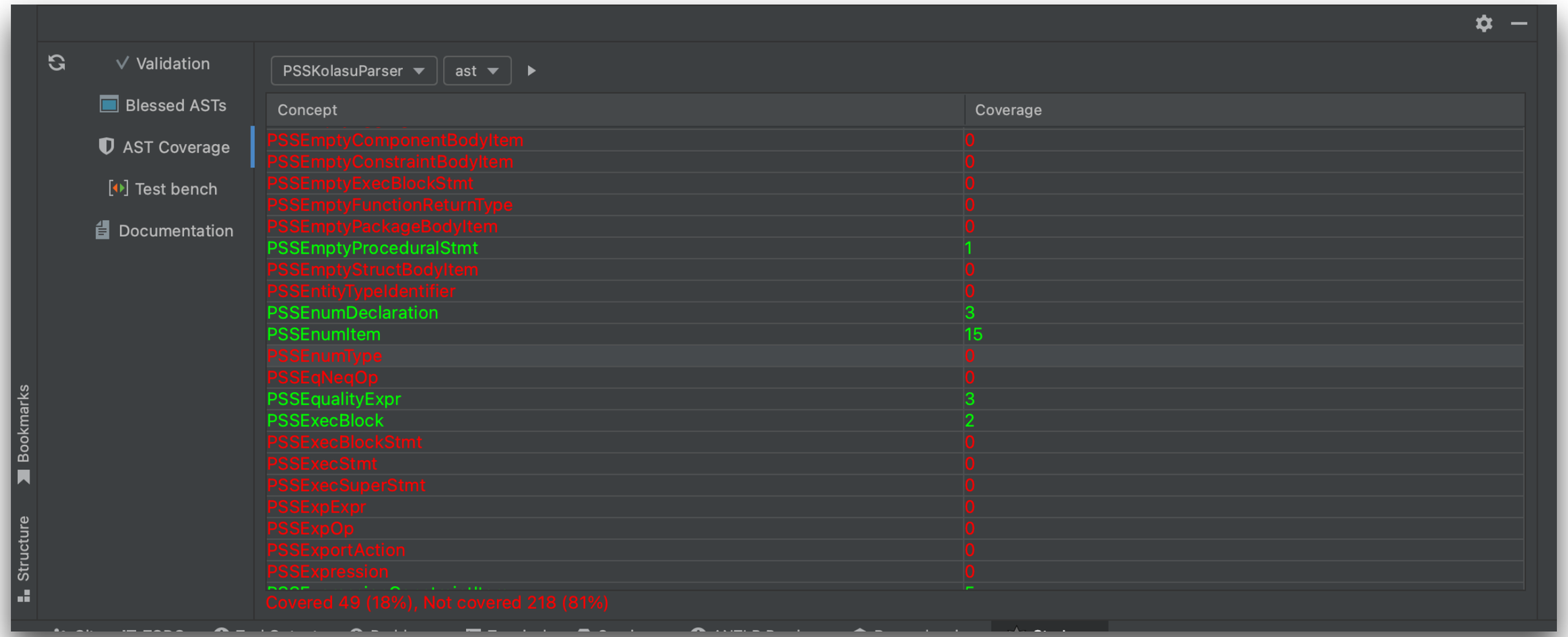
2. Each Concept is validated by at least one blessed example.

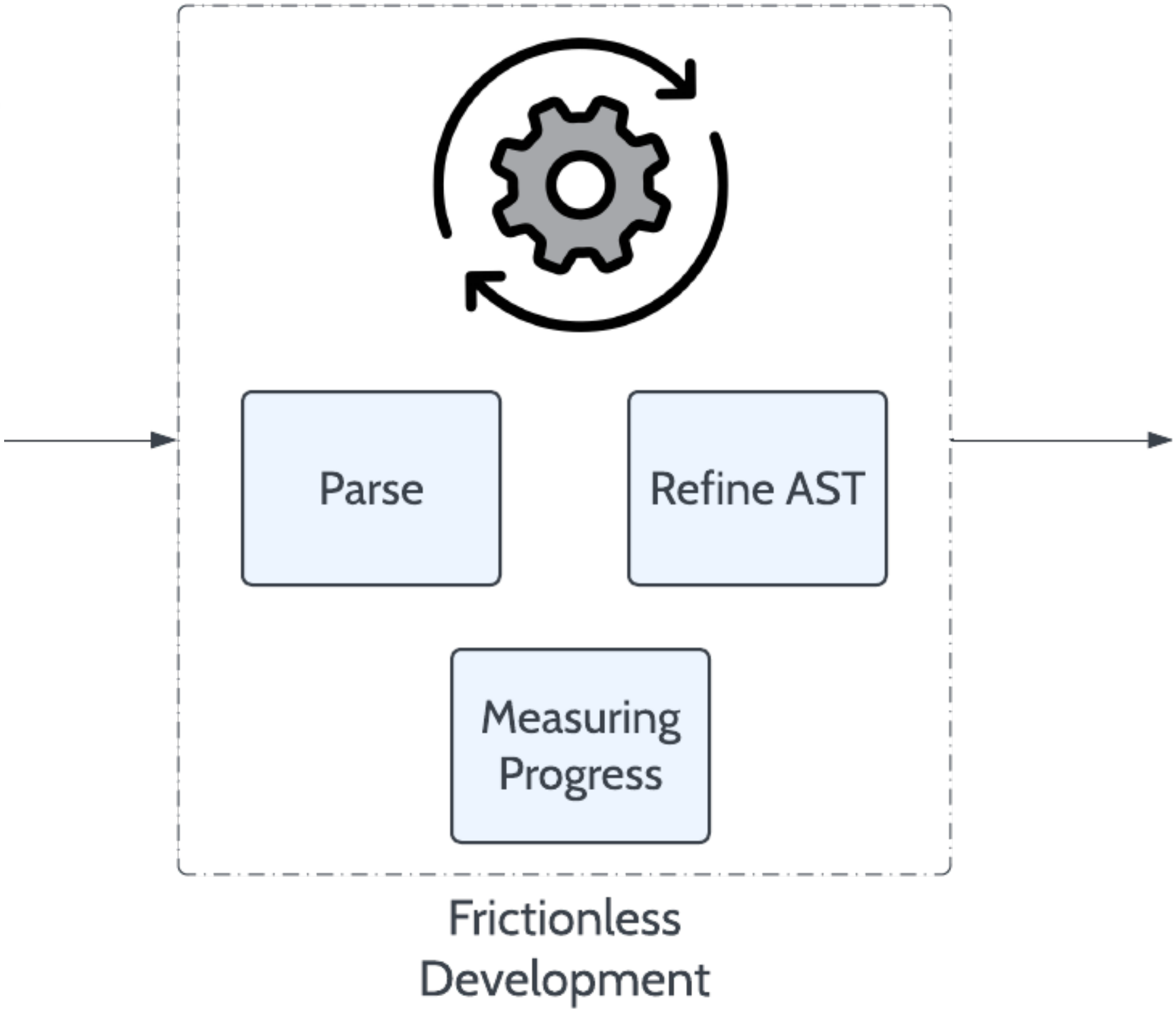
   This ensures that the model of the code produced is valuable.

Quality checks



Quality checks

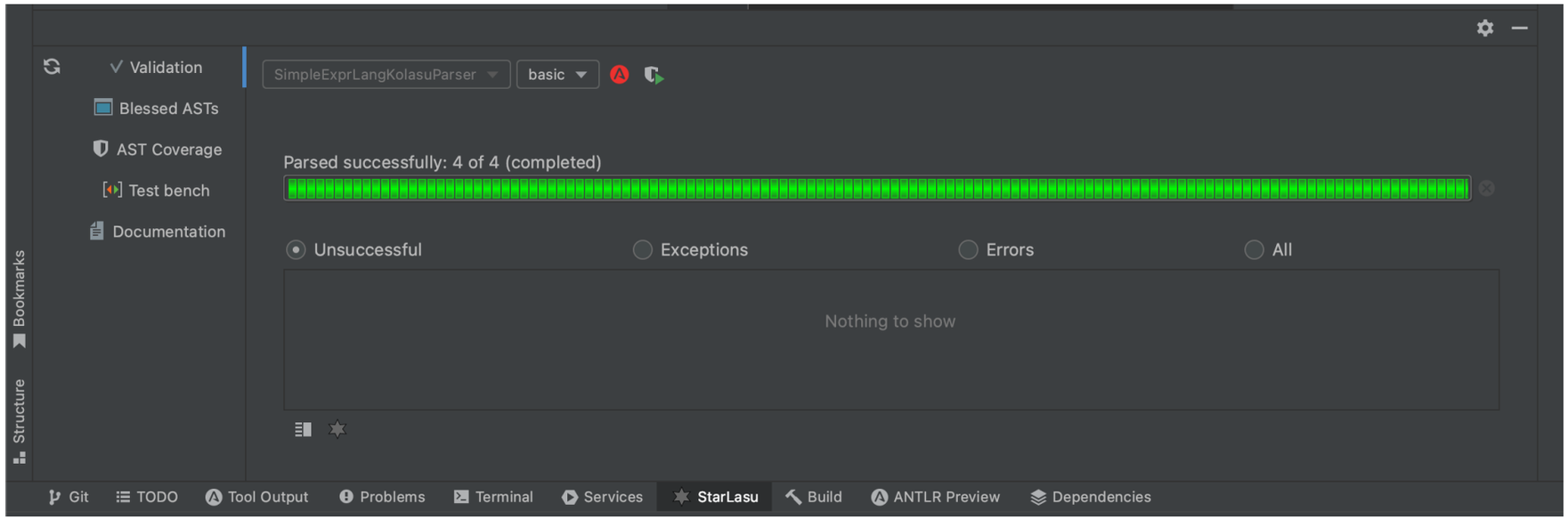


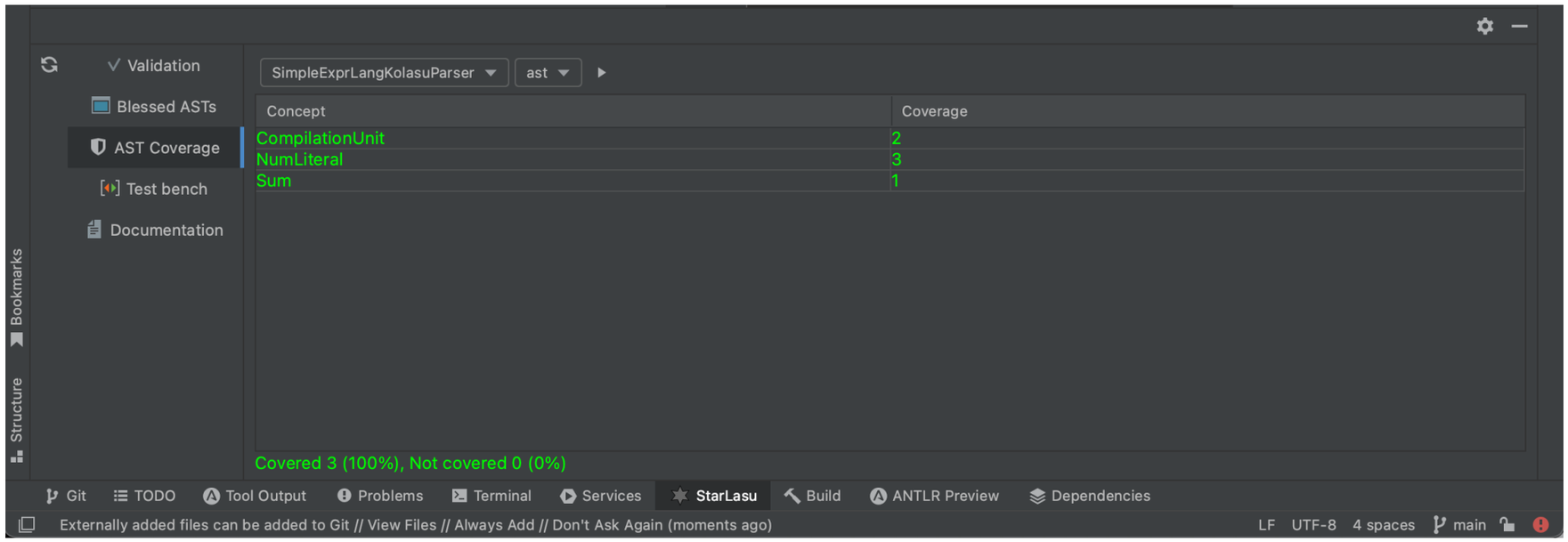


The Chisel Method - Workflow

Workflow Demo

Show all checks are green





Where are we?


Helsinki
Stockholm
Moscow
Ulaanbaatar
Warsaw
Kiev
Paris
Beijing
Vienna
Budapest
Madrid
Rome
Tashkent
Athens
Ankara
Kabul
Baku
Islamabad
Tehran
Baghdad
Overview
Above Cam
Chase Cam

Where are we?

- Setup is now automated, so it does not take time
- Time is spent in the Parse Phase and the AST Refinement Phase
- We know by experience that the Parse phase takes 70% of the time
- We know that the progress is not linear, i.e., the last bits are the most difficult

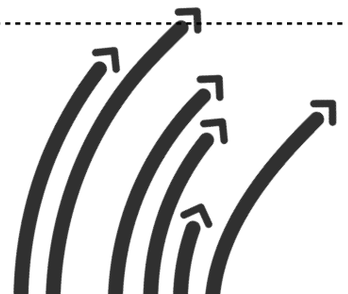
Where are we?

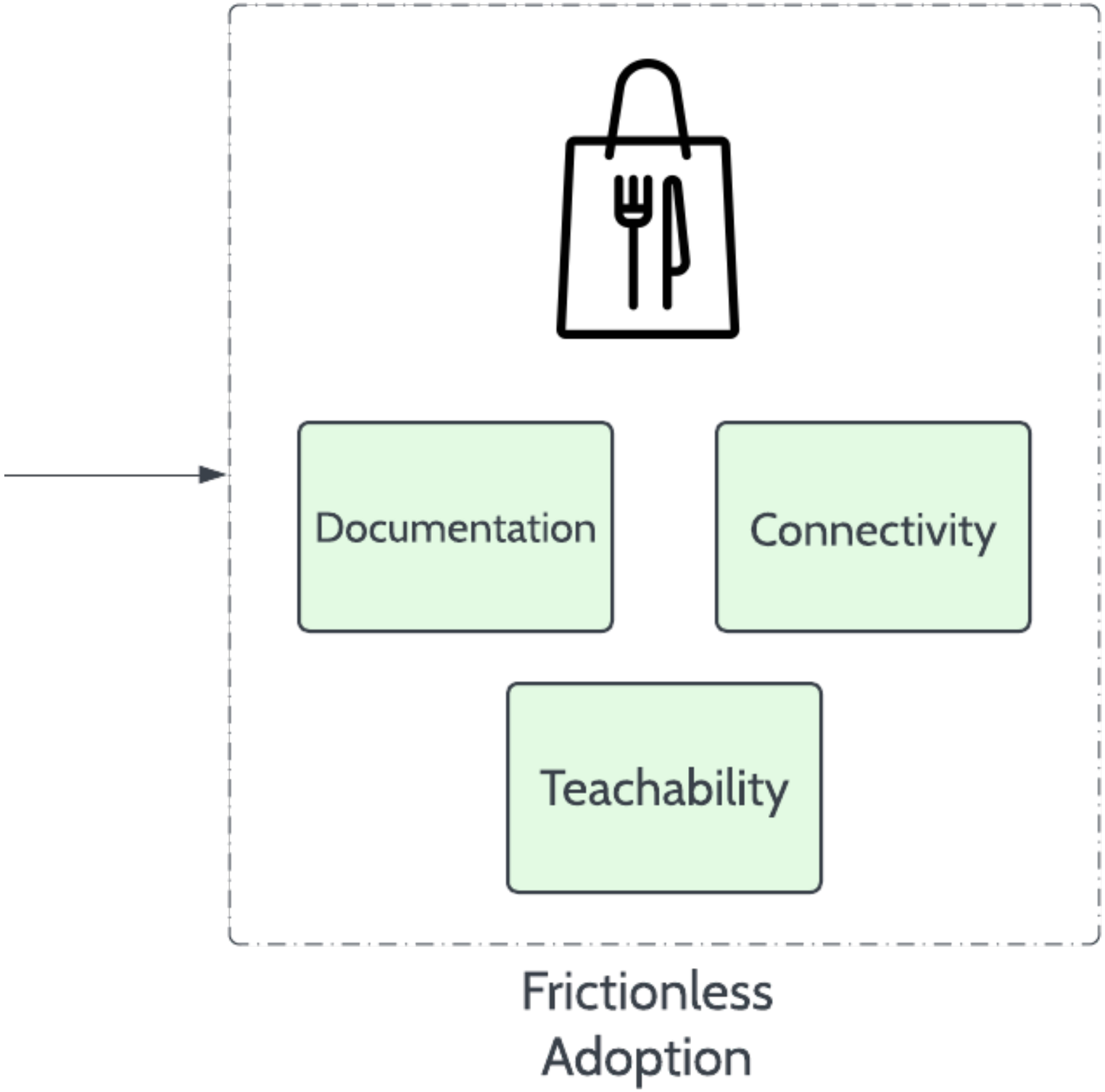
Parse Phase (0-70%) - 1,000 examples to cover

Examples covered	Progress
250	9 %
500	25 %
750	45 %
1,000	70 %

AST Refinement Phase (70-100%) - 100 constructs to cover

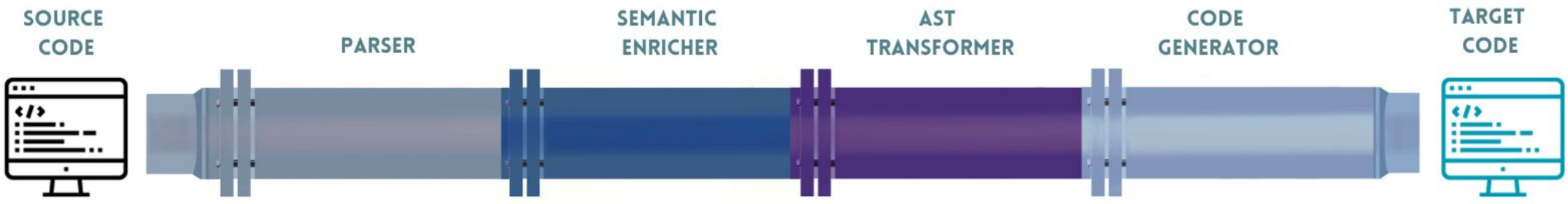
Constructs	Progress
25	74 %
50	81 %
75	89 %
100	100 %



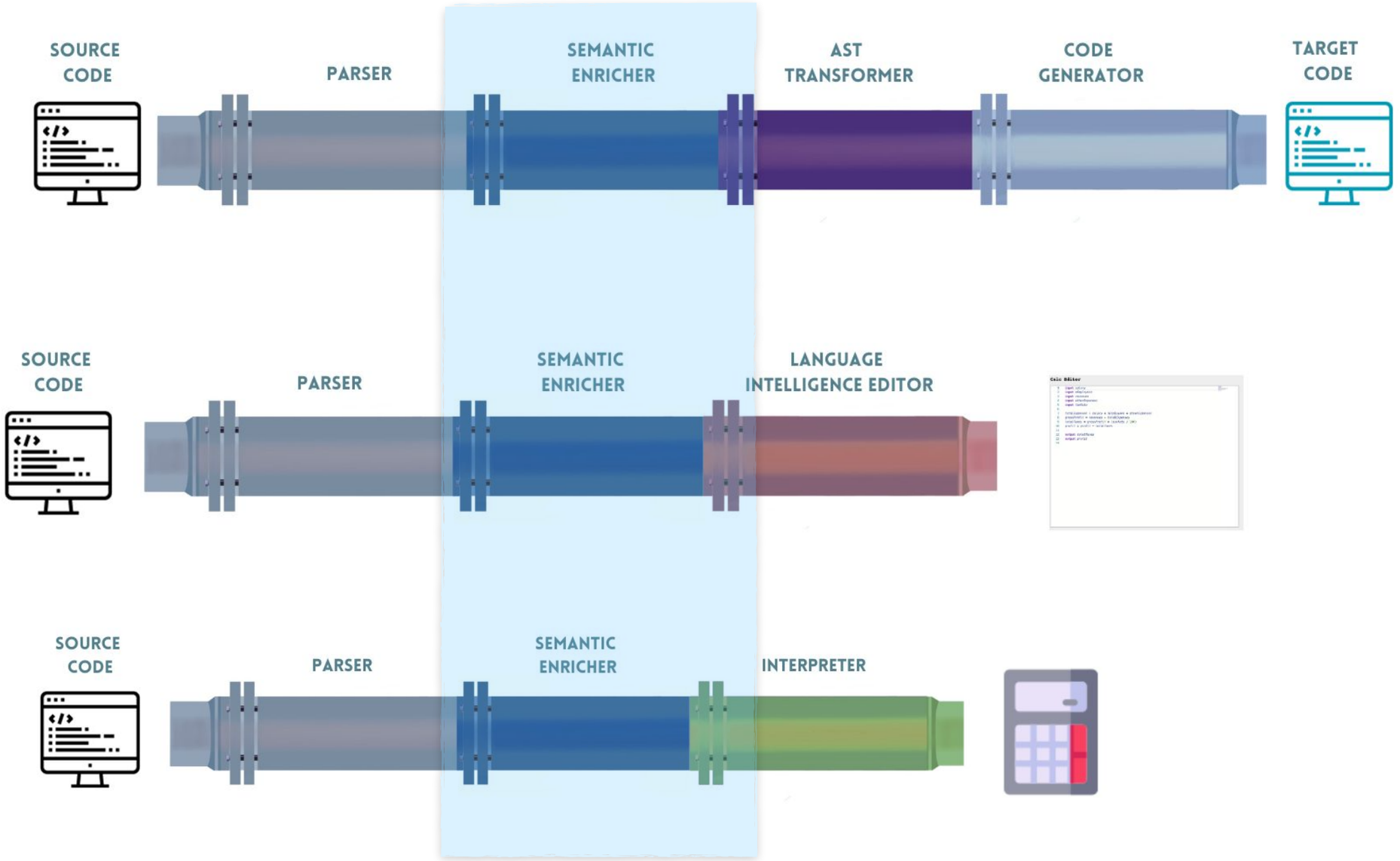


The Chisel Method - Adoption

Language Engineering Pipelines



Language Engineering Pipelines



9:30	Implementing Symbol Resolution in StarLasu	Lorenzo Addazi

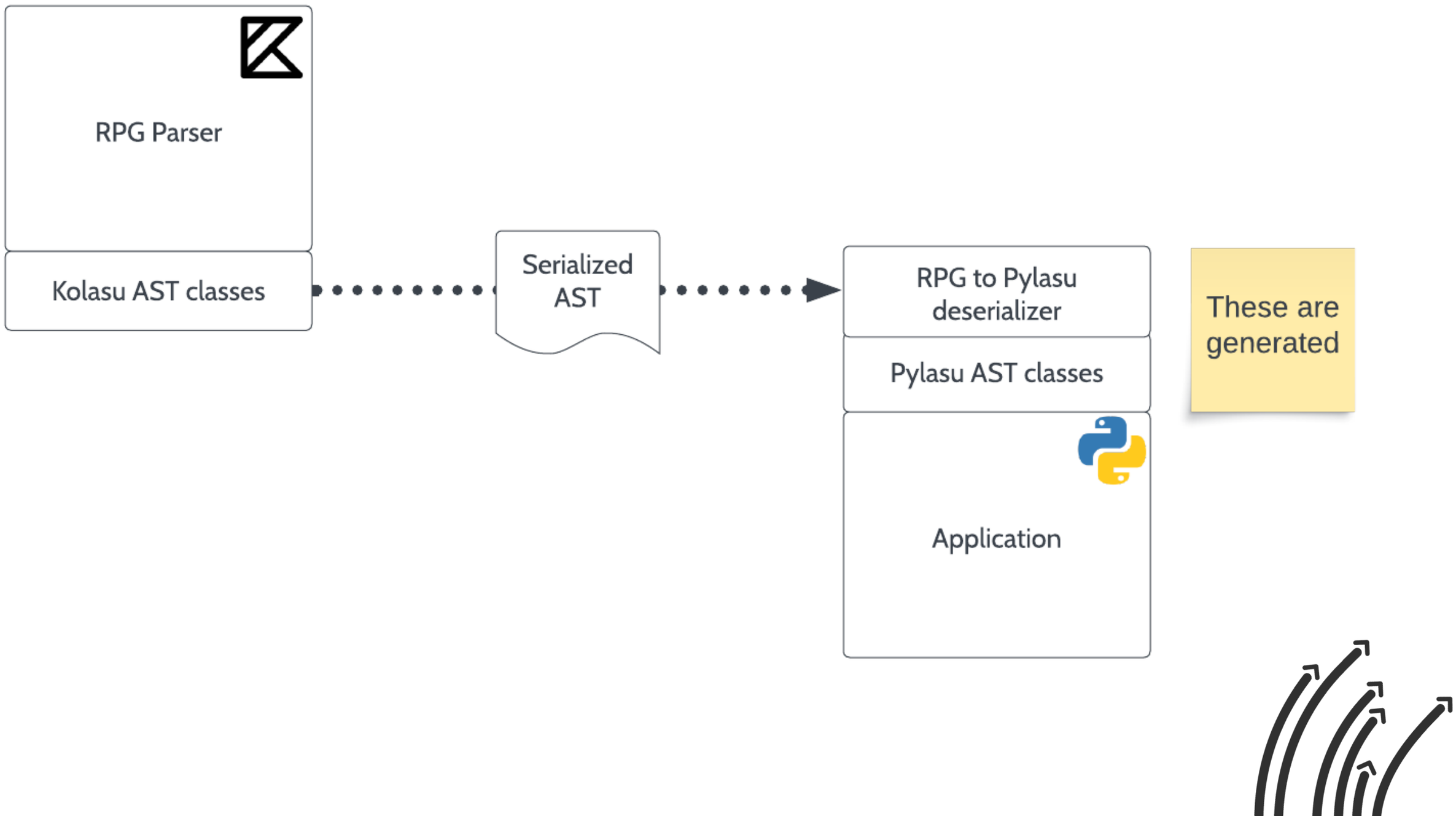
Providing APIs

All of our parsers are based on the StarLasu libraries:

- Kolasu for Kotlin
- Pylasu for Python
- Tylasu for Typescript
- SharpLasu for C#

All those libraries provide APIs to navigate and process the AST

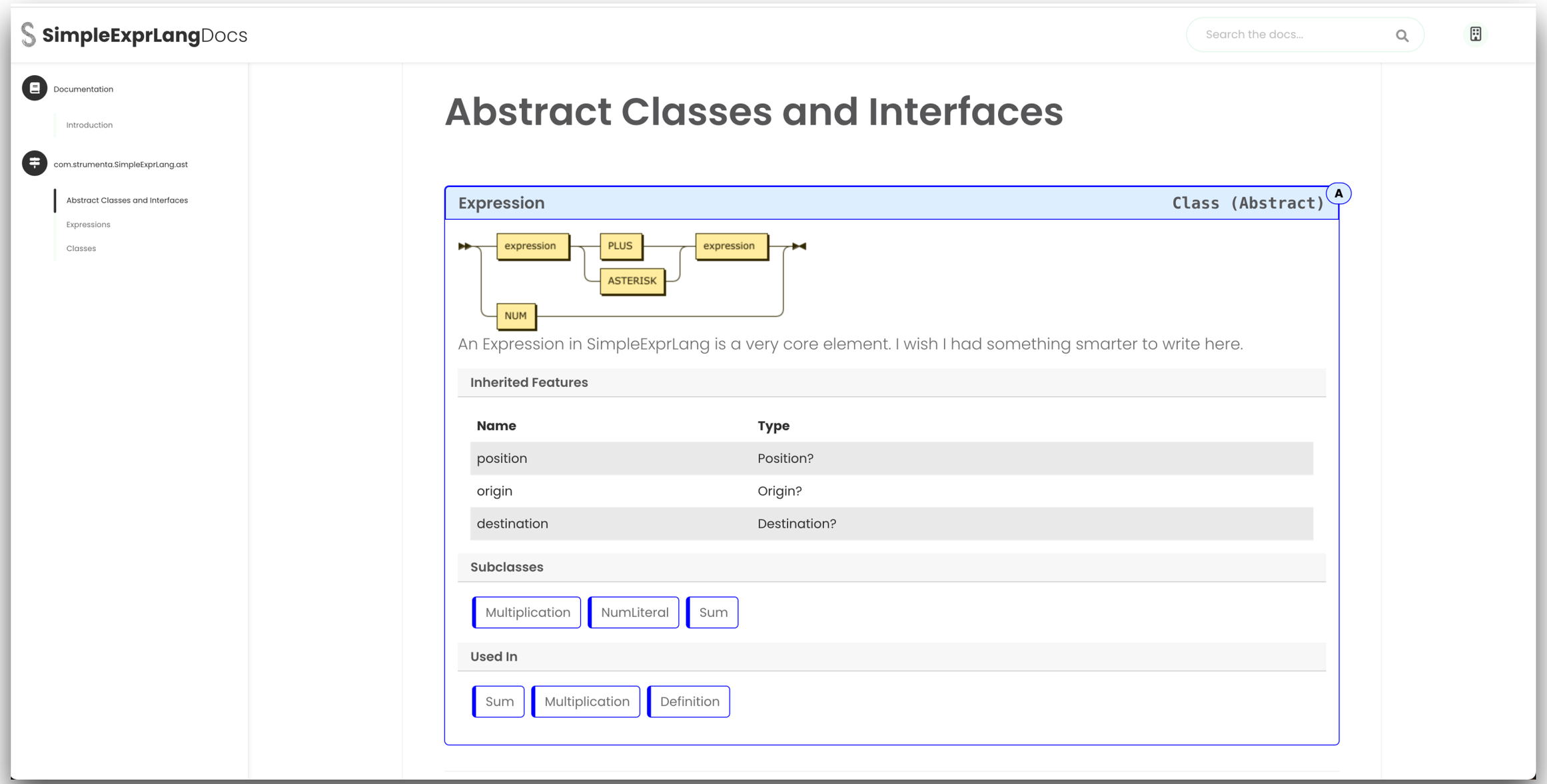
Generating adapters



Compatibility to the next level

11:00	LionWeb Initiative	Niko Stotz and Jos Warmer

Documentation



Teachability

Teachability reduce risks:

- the vendor providing the parser will be able to train new maintainers, if needed

- the client can take over maintenance of the parser, if needed

- complementary parsers, based on the same structure, can be developed (think of an HTML, JS, and CSS parsers that are compatible)

What are the deliverables

- A written method: currently described into 20 pages, later on it will become a video-course

- Supporting libraries: the ones necessarily at runtime are the StarLasu libraries, and they are open-source

- Tools: a gradle plugin and an IDEA plugin, that at the moment we use internally and we are refining. They just accelerate development

Why calling it the Chisel method?

Because it is about getting the information out of the code, as you use a chisel to take the statue out of the marble.

Also, Strumenta means tools in Latin, and Chisel is one tool.

What next?

- Refine this for the next 10 years
- ...while working on the method for building Transpilers (hopefully in time for next year LangDev!)

Thank you!

Feedback at:

federico@strumenta.com