

“

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

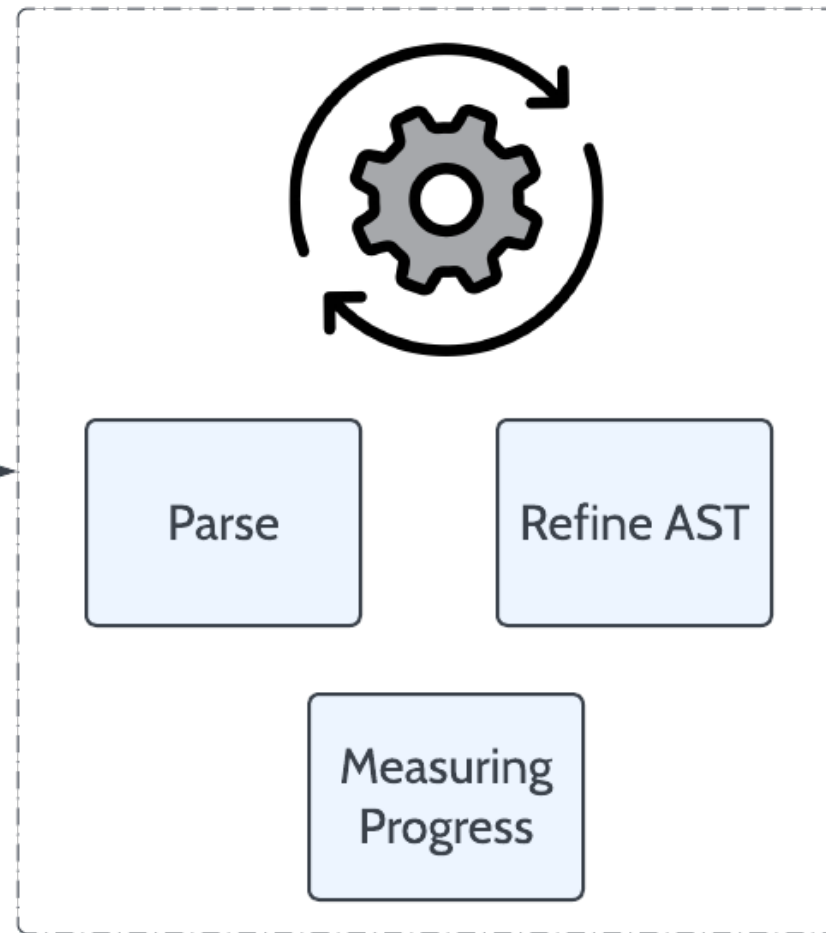


What can we do?

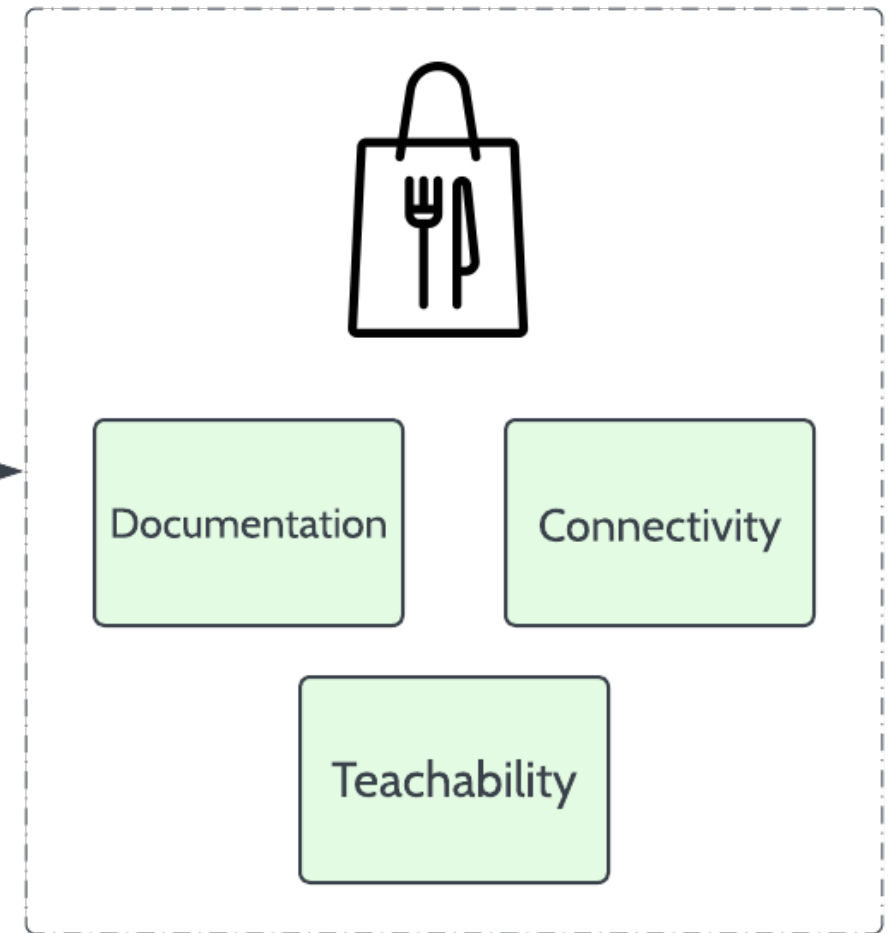




Define the goal



Frictionless
Development



Frictionless
Adoption

The Chisel Method



“



Define the goal

The Chisel Method - Goal setting



“ Start with the end in mind

language:Java stars:>1000 forks:>500

3.3k results (360 ms)

Sort by: Most stars Save ...

- TheAlgorithms/Java**
All Algorithms implemented in Java
search java algorithm algorithms sort
Java · 54.1k stars · Updated 1 hour ago
- spring-projects/spring-framework**
Spring Framework
framework spring spring-framework
Java · 53.3k stars · Updated 14 hours ago
- google/guava**
Google core libraries for Java
java guava
Java · 48.4k stars · Updated 10 hours ago
- ReactiveX/RxJava**
RxJava – Reactive Extensions for the JVM – a library for composing asynchronous and event-based programs using observable sequences for t...
java flow rxjava reactive-streams
Java · 47.3k stars · Updated yesterday
- NationalSecurityAgency/ghidra**
Ghidra is a software reverse engineering (SRE) framework
reverse-engineering disassembler software-analysis
Java · 43.3k stars · Updated 1 hour ago

Left Sidebar:

- Wikis: 0
- Topics: 0
- Marketplace: 21k
- Languages**
 - JavaScript
 - Python
 - Java
 - TypeScript
 - Go
 - C++
 - C
 - PHP
 - C#
 - HTML
 - More languages...
- Advanced**
 - Owner
 - Size
 - Number of followers
 - Number of forks
 - Number of stars
 - Date created
 - Date pushed
 - Topic
 - License
 - Archived

“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

The screenshot displays the StarLasu IDE interface. On the left, a sidebar contains navigation items: 'Validation' (selected), 'Blessed ASTs', 'AST Coverage', 'Test bench', and 'Documentation'. The main area shows the 'Validation' tab for the 'PSSKolasuParser' project. It reports 'Parsed with exceptions: 4, with errors 2, successfully: 6 of 12 (completed)'. A progress bar is shown below this text. Below the progress bar, there are four radio buttons: 'Unsuccessful' (selected), 'Exceptions', 'Errors', and 'All'. A list of files with their validation status is displayed:

- example21.pss (Unsuccessful)
- example19.pss (Unsuccessful)
- example_page66_1.pss (Exception)
- example14.pss (Unsuccessful)
- example1.pss (Exception)
- example11.pss (Unsuccessful)

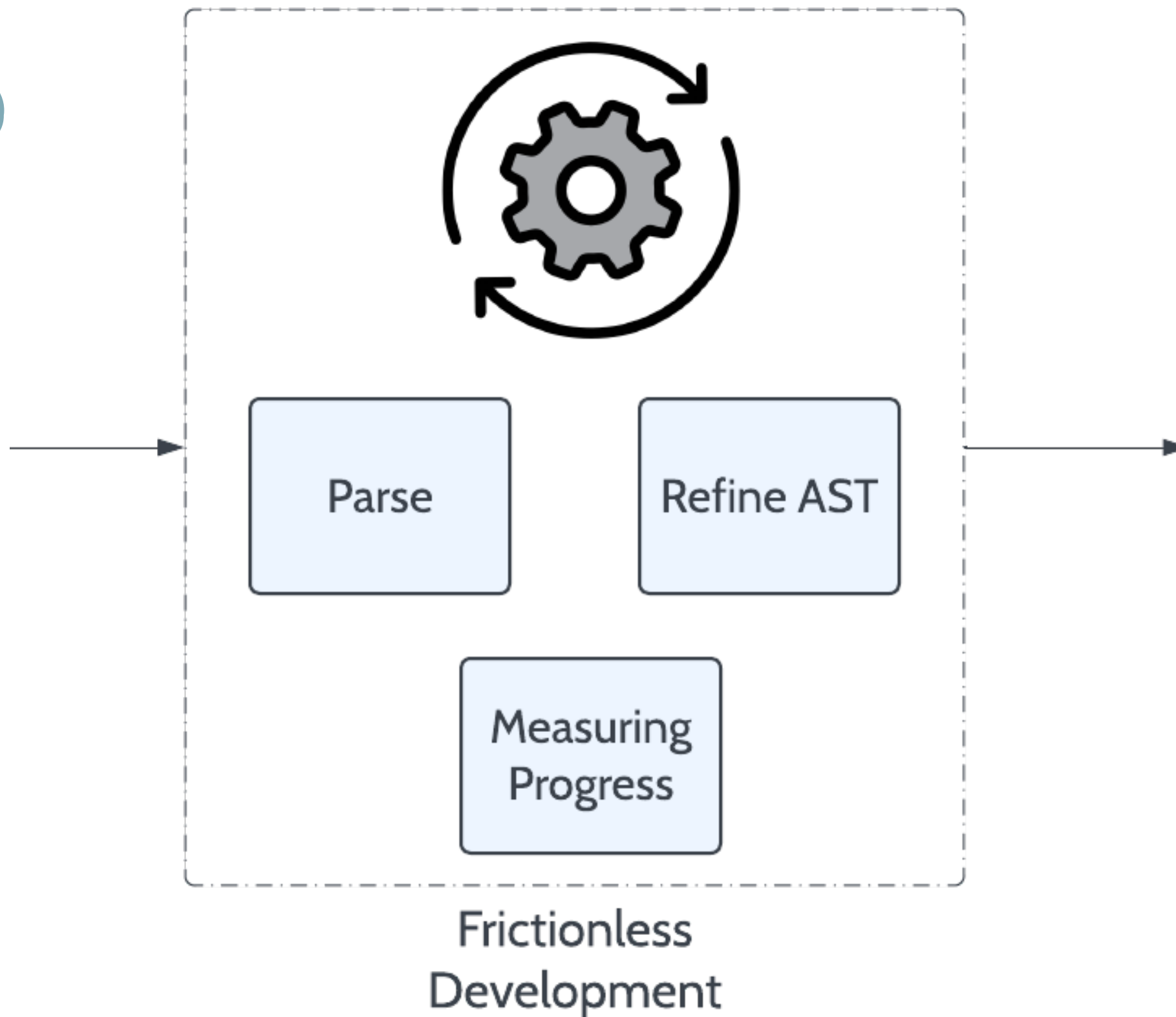
The bottom status bar includes icons for Git, TODO, Tool Output, Problems, Terminal, Services, Build, ANTLR Preview, Dependencies, and StarLasu.



“Quality checks



“



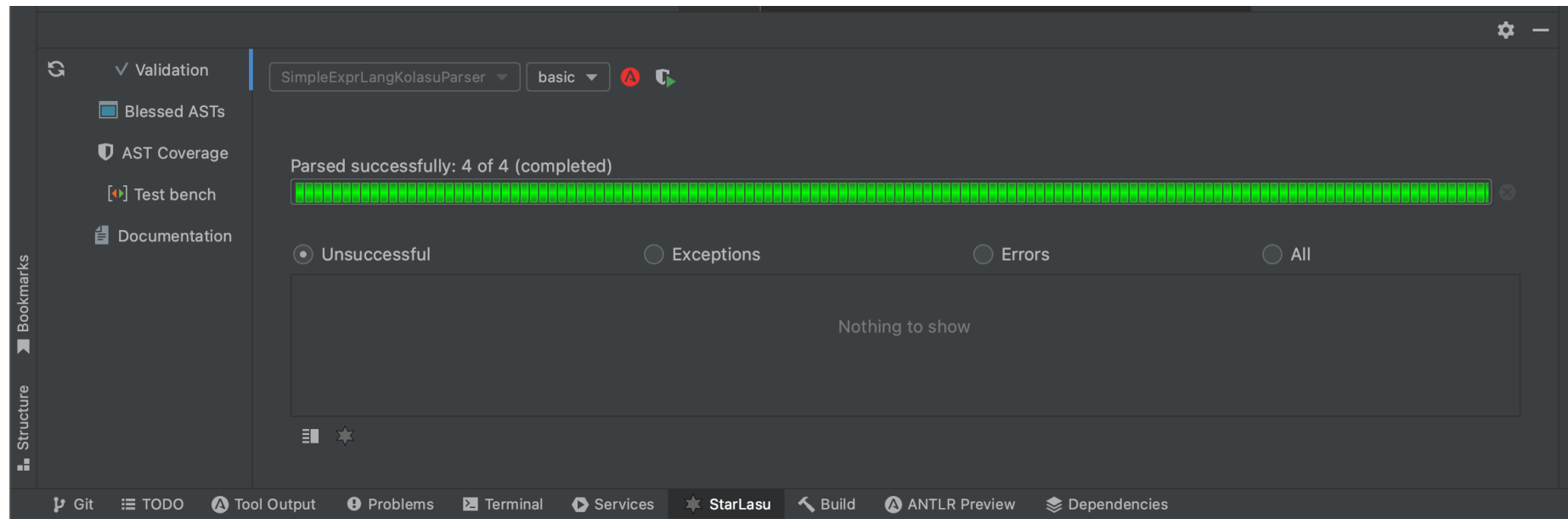
The Chisel Method - Workflow



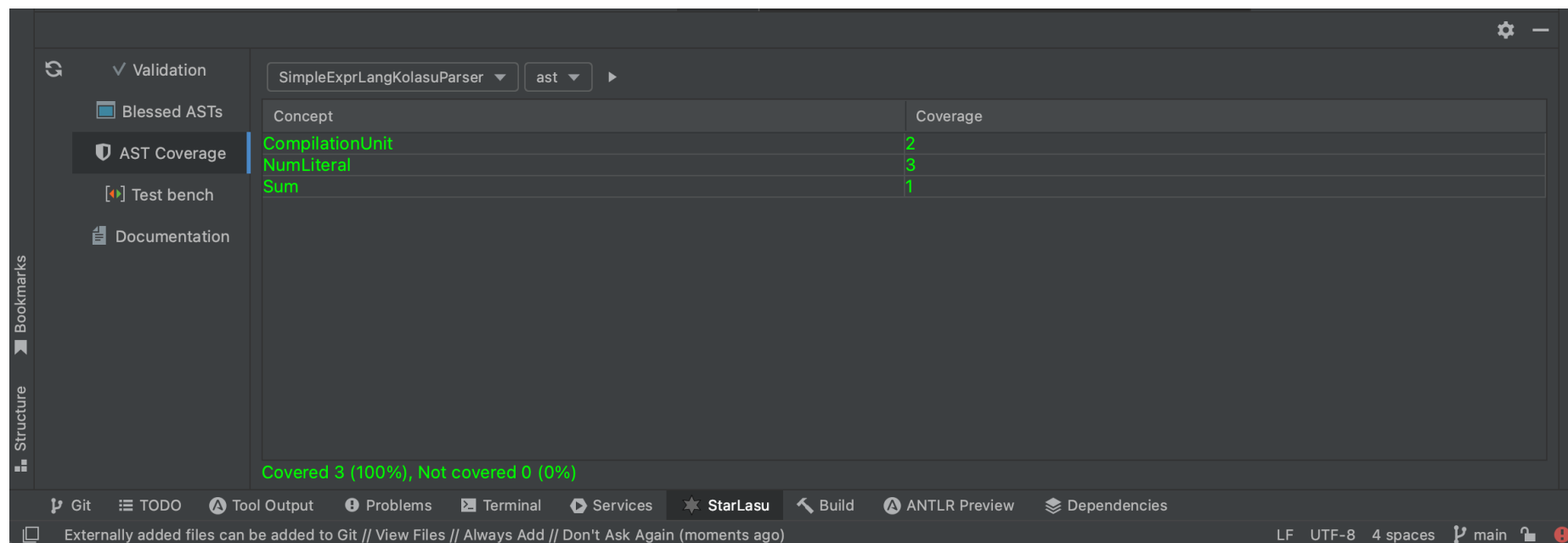
Workflow Demo



“ Show all checks are green



The screenshot shows the 'Validation' tool interface. The left sidebar contains a 'Bookmarks' section with 'Structure' and 'Bookmarks' icons. The main panel displays the 'SimpleExprLangKolasuParser' with a 'basic' configuration. A green progress bar indicates 'Parsed successfully: 4 of 4 (completed)'. Below the progress bar, there are four radio buttons: 'Unsuccessful' (selected), 'Exceptions', 'Errors', and 'All'. The main content area is empty, displaying 'Nothing to show'. The bottom status bar includes icons for Git, TODO, Tool Output, Problems, Terminal, Services, StarLasu, Build, ANTLR Preview, and Dependencies.



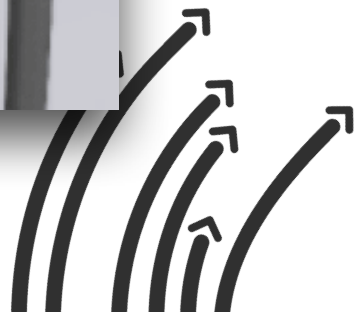
The screenshot shows the 'AST Coverage' tool interface. The left sidebar contains a 'Bookmarks' section with 'Structure' and 'Bookmarks' icons. The main panel displays the 'SimpleExprLangKolasuParser' with an 'ast' configuration. A table shows the coverage for three concepts:

| Concept | Coverage |
|-----------------|----------|
| CompilationUnit | 2 |
| NumLiteral | 3 |
| Sum | 1 |

At the bottom of the table, it says 'Covered 3 (100%), Not covered 0 (0%)'. The bottom status bar includes icons for Git, TODO, Tool Output, Problems, Terminal, Services, StarLasu, Build, ANTLR Preview, and Dependencies. The status bar also shows 'Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (moments ago)' and 'LF UTF-8 4 spaces main'.



“Where are we?”



“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



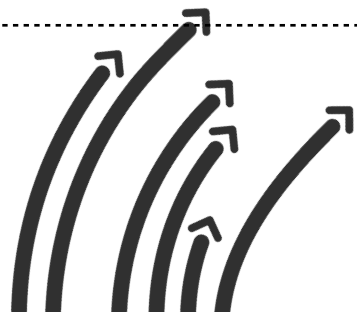
“Where are we?”

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

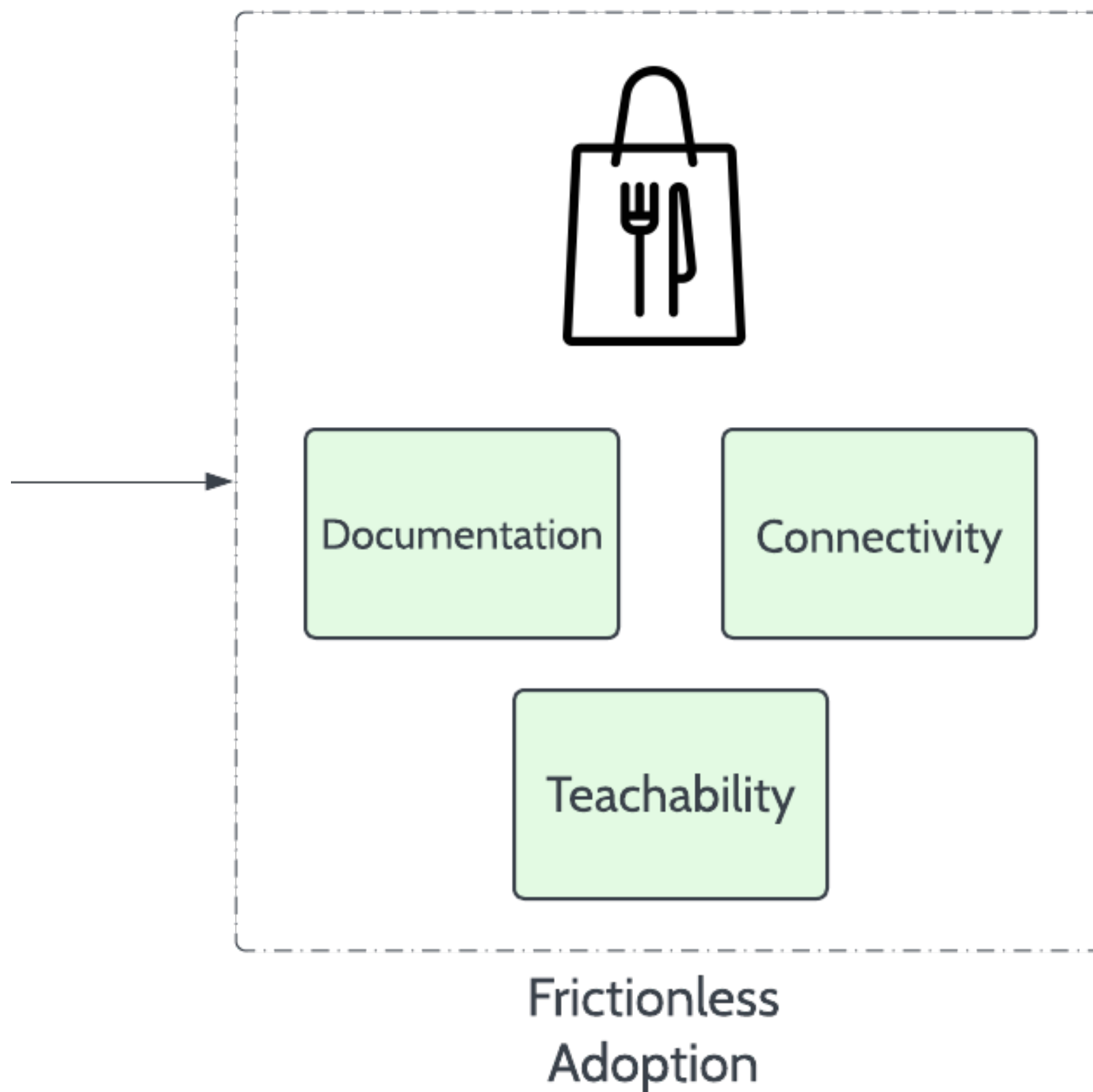
| Examples covered | Progress |
|------------------|----------|
| 250 | 9 % |
| 500 | 25 % |
| 750 | 45 % |
| 1,000 | 70 % |

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

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



“



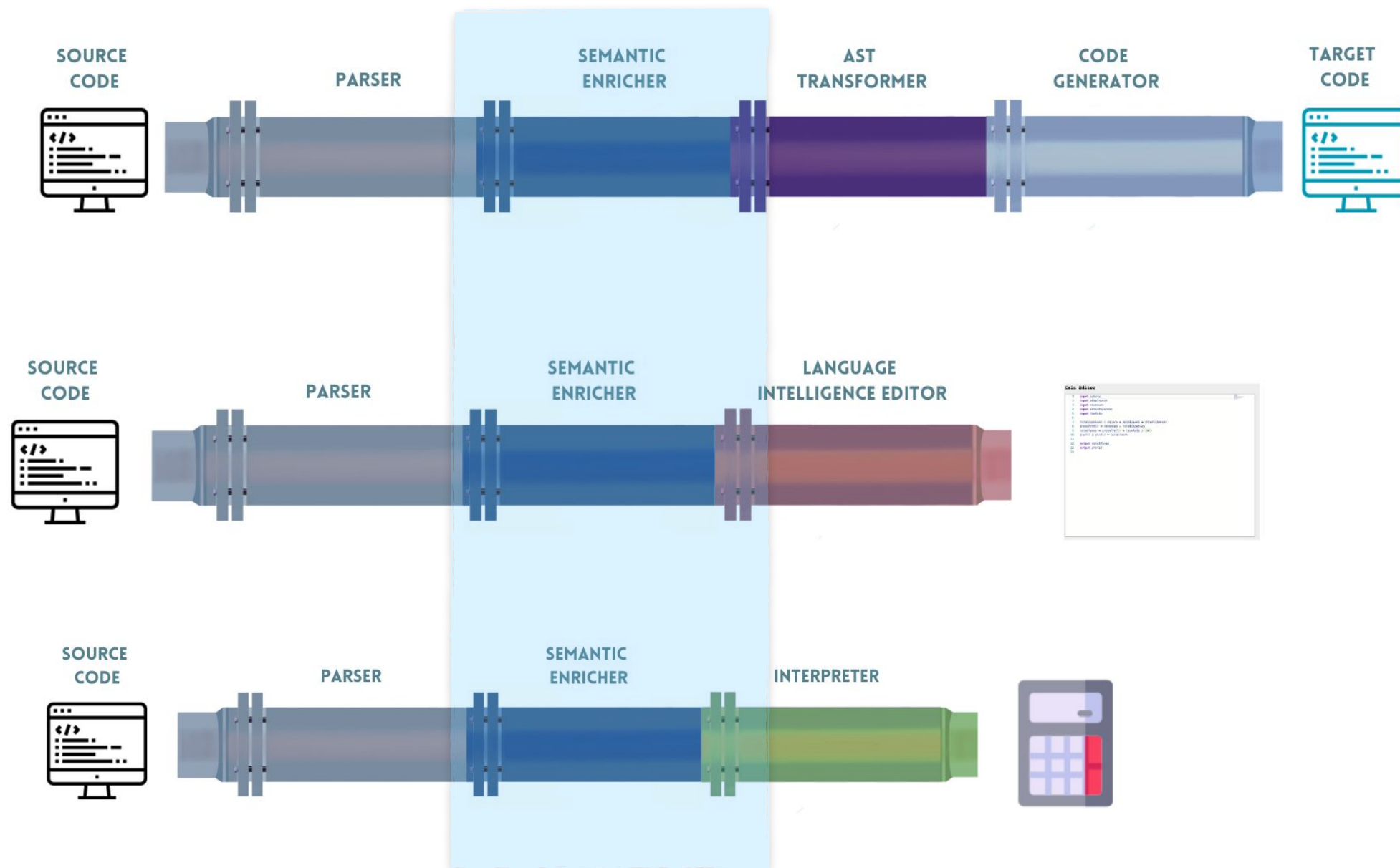
The Chisel Method - Adoption



“ Language Engineering Pipelines



“ Language Engineering Pipelines



“ Providing APIs

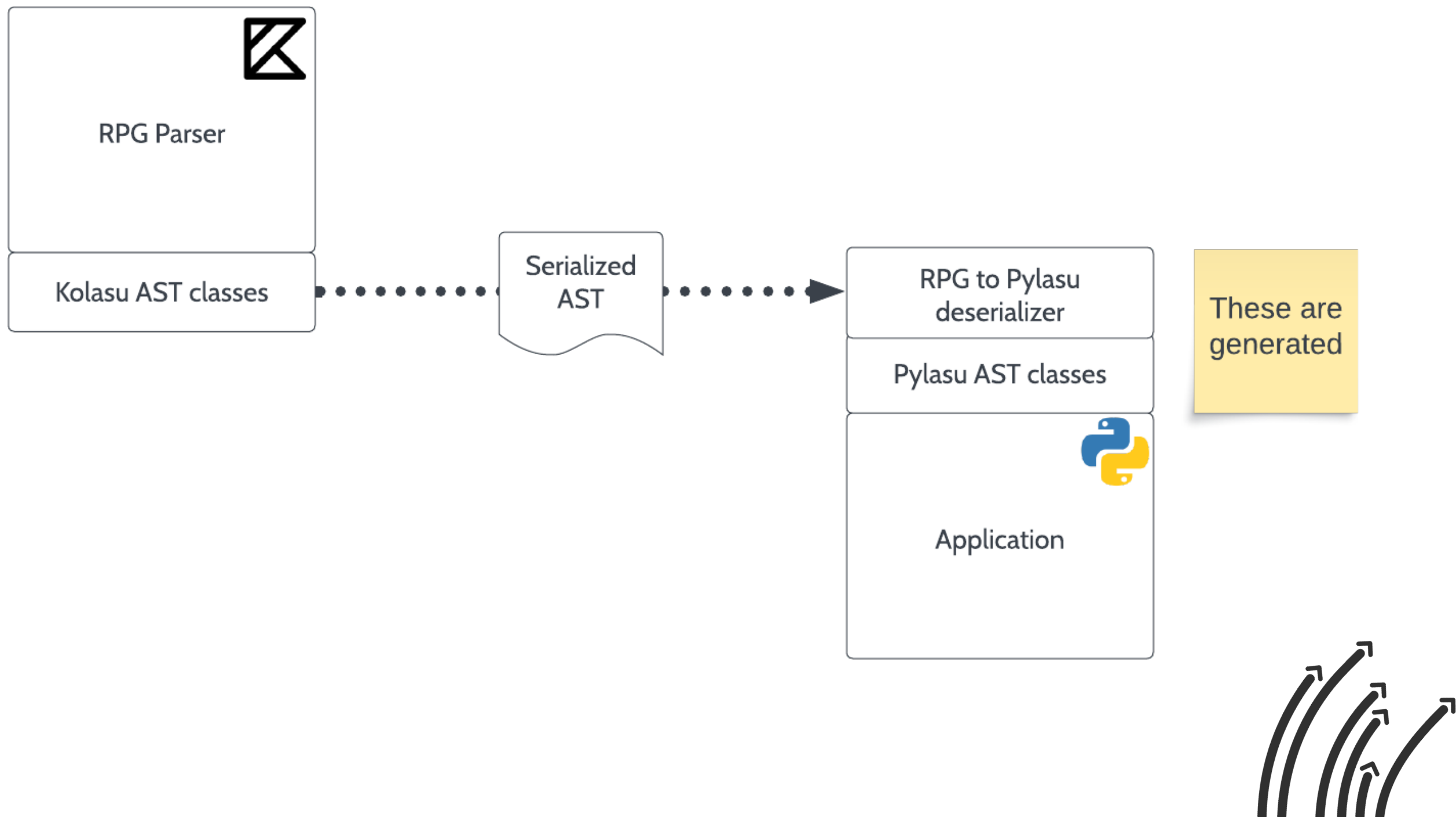
All of our parsers are based on the StarLasu libraries:

- 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

SimpleExprLangDocs

Search the docs...

Documentation

Introduction

com.strumenta.SimpleExprLang.ast

Abstract Classes and Interfaces

Expressions

Classes

Abstract Classes and Interfaces

Expression

Class (Abstract)

A

```
graph LR; E1[expression] --> P[PLUS]; P --> E2[expression]; P --> A[ASTERISK]; A --> E1; E1 --> N[NUM]; N --> E1
```

An Expression in SimpleExprLang is a very core element. I wish I had something smarter to write here.

Inherited Features

| Name | Type |
|-------------|--------------|
| position | Position? |
| origin | Origin? |
| destination | Destination? |

Subclasses

Multiplication NumLiteral Sum

Used In

Sum Multiplication Definition





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

