PMD:
Functional & Information View

Presented by: Aprajita, Inderjot, Senjuti

To Recap...

What is PMD?

PMD is a static source code analyzer. It is shipped with a copy paste detector.

A static source code analyzer finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation etc.

A copy paste detector detects duplicate code.

PMD supports many languages like Java, JavaScript, PLSQL, Apache Velocity, XML etc.

PMD allows users to write custom rules (for the static source code analyzer) and add more languages

Who is our view aimed towards?

Technical Stakeholders:

Developers who can understand and potentially contribute to the system

Functional View

Concerns:

1. Functional Capabilities
2. Internal structures
3. External Interfaces
4. Functional design philosophy

Functional Capabilities

1. Static Source Code Analyzer
2. Copy Paste Detector
3. Support for multiple languages

Internal Structures

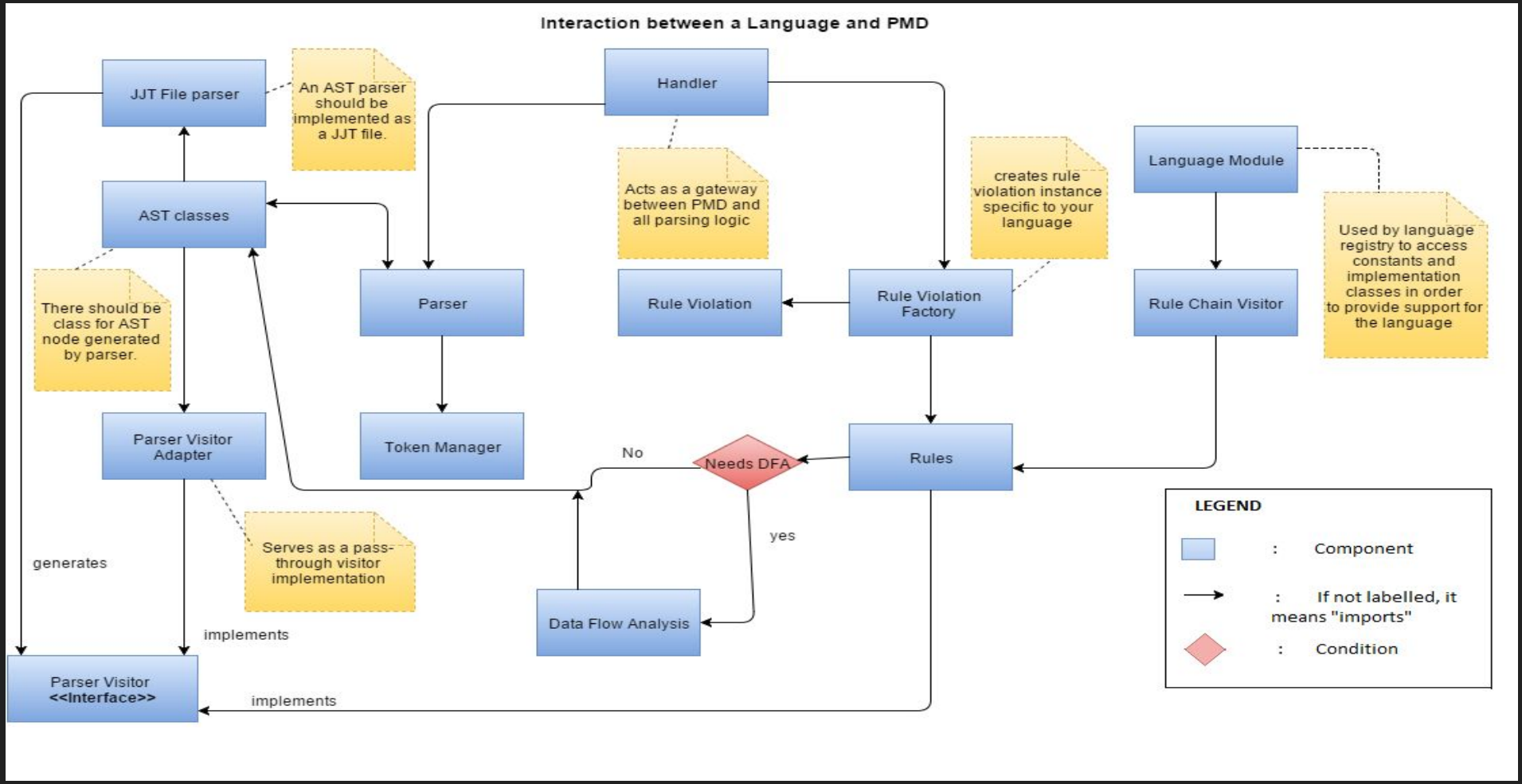
As a Static Source Code Analyzer:

1. Uses a JavaCC generated parser to parse source code

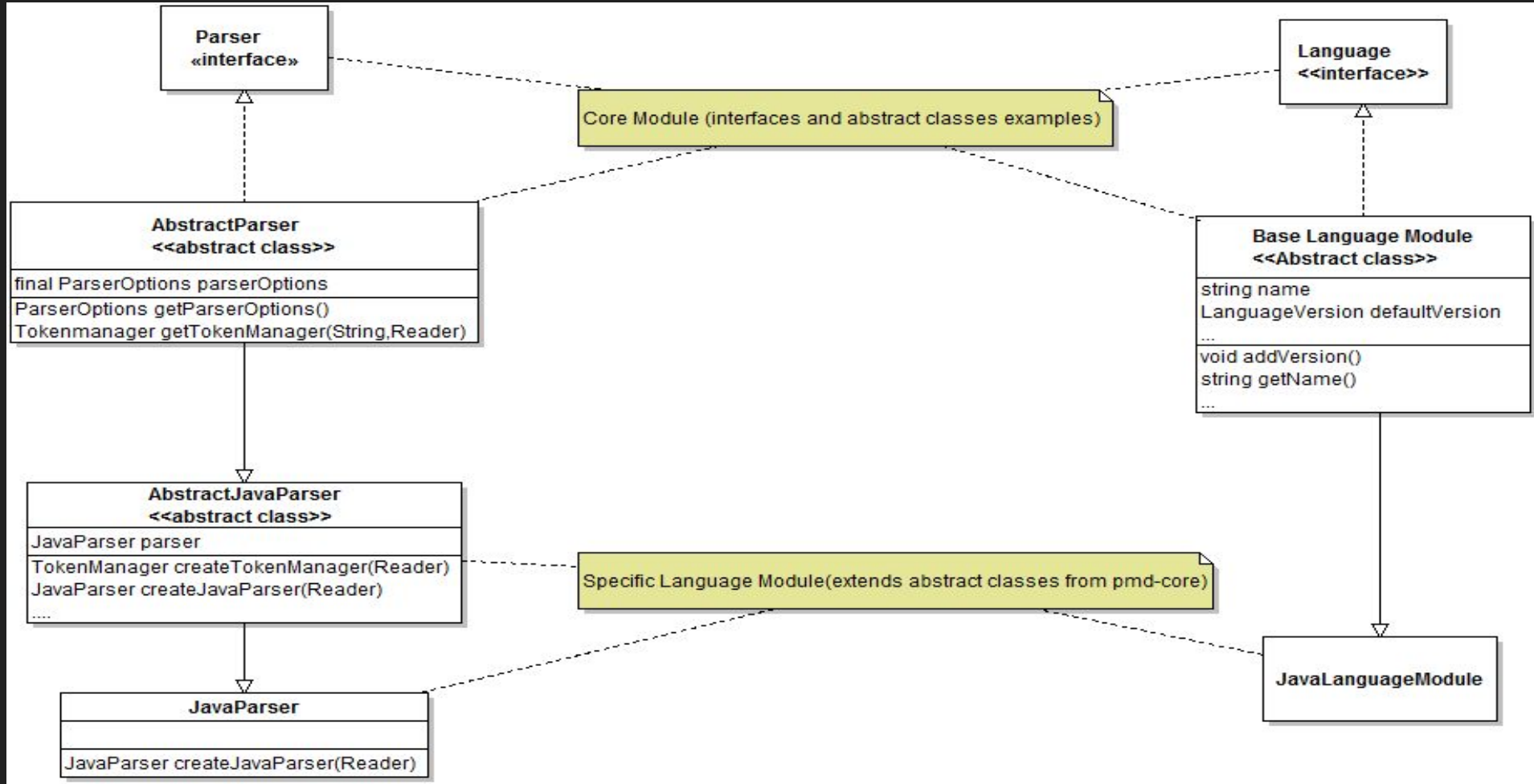
2. Generates Abstract Syntax Tree (AST)

3. Rule is called whenever required type is encountered in the AST by PMD

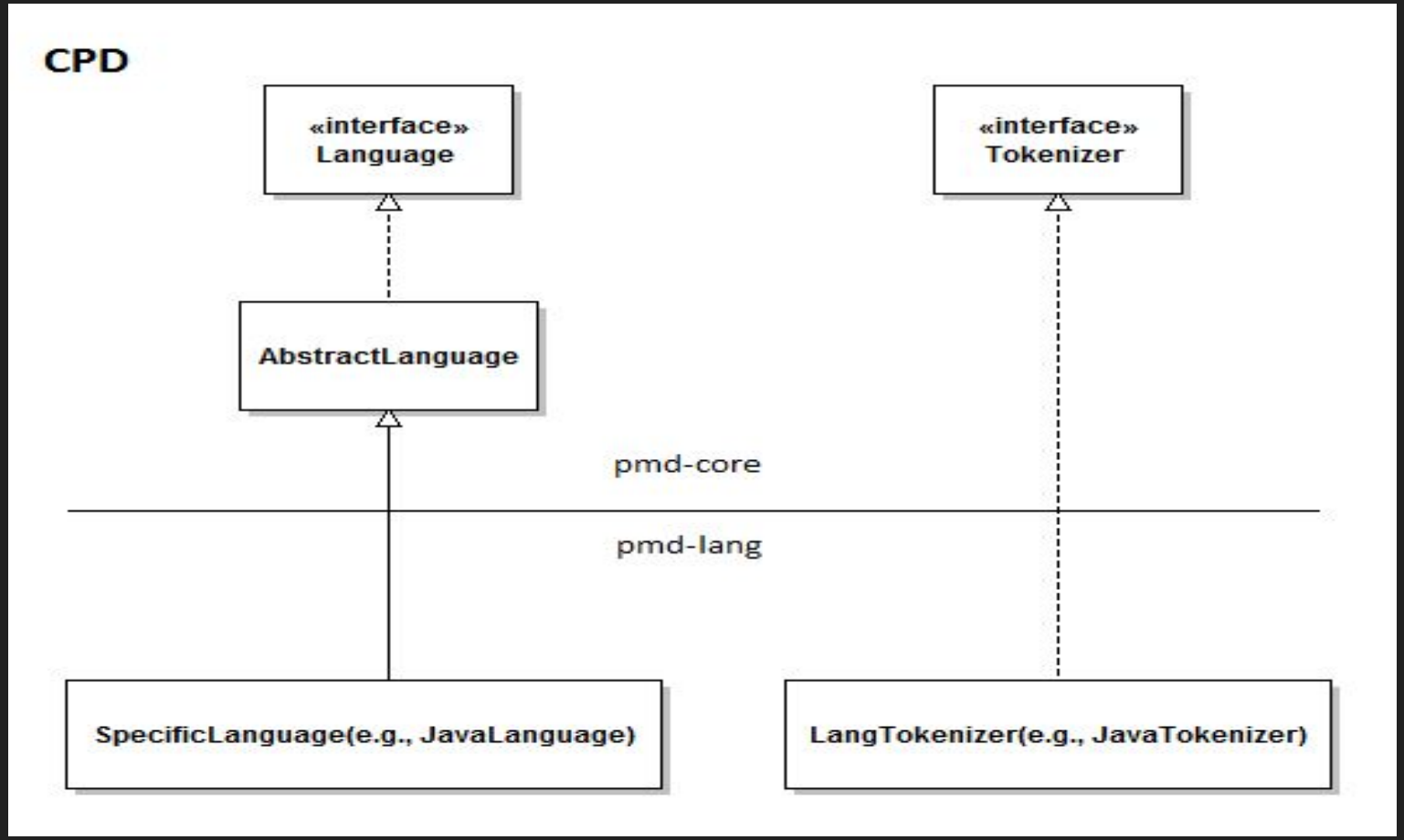
Interactions between PMD's Internal Interfaces:



What does each component represent?



As Copy Paste Detector (CPD)



Addition of New Rules

1. Using Java

2. Using XPath Expression

External Interfaces

Data, events and control flows between the system and external systems can be categorized as external interfaces. PMD consists of the following external interfaces:

- Addition of new language (interaction between PMD system and the new language to be added)
- User interaction with PMD. There are 4 different ways in which user can interact with PMD.
  - Command line interaction.
  - Running PMD as an ANT task.

Functional Design Philosophy

1. Adheres to the **stable** functional design philosophy (partitioned into 2 modules: PMD core and PMD language module.
2. Exhibits **strong coherence**, **high cohesion** and **low coupling**.
3. Exhibits a **consistent design**.
4. Easily **extensible**.
5. No “God element” is present in its architecture which allows easy **separation of concerns**.
6. Overall very well-designed.

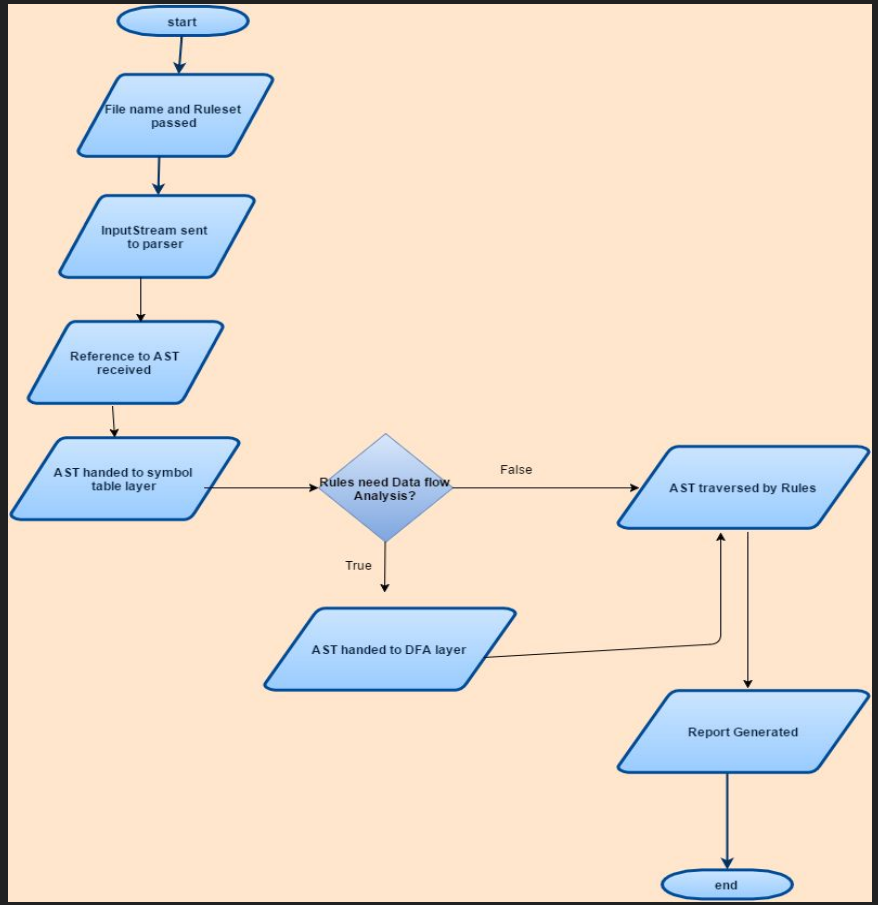
Information View

1. Data is never stored, retrieved or archived in PMD

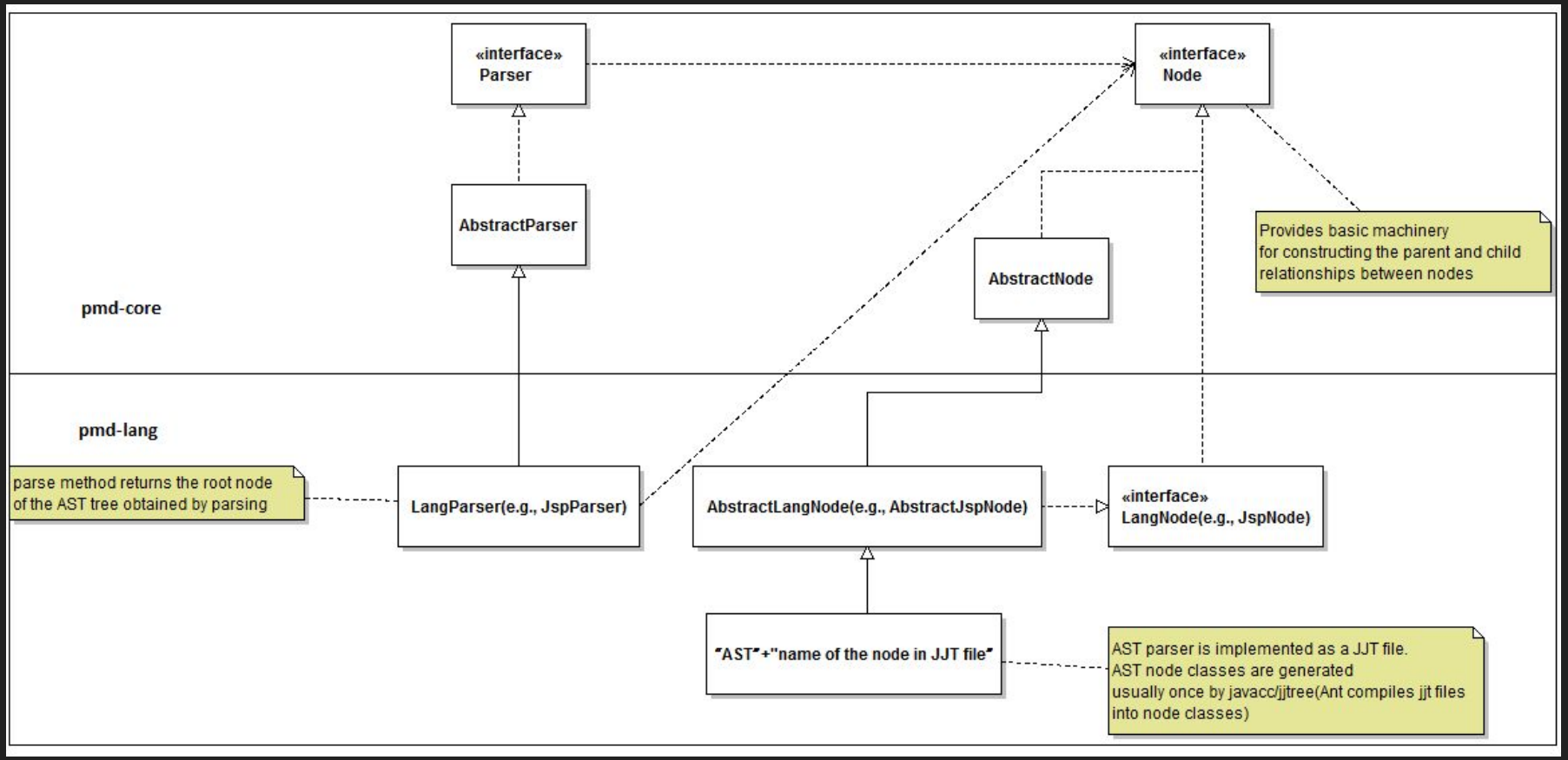
2. We focus on the Information Processing Life Cycle

3. This is supported by data flow through the system and implementation details such as data structures used

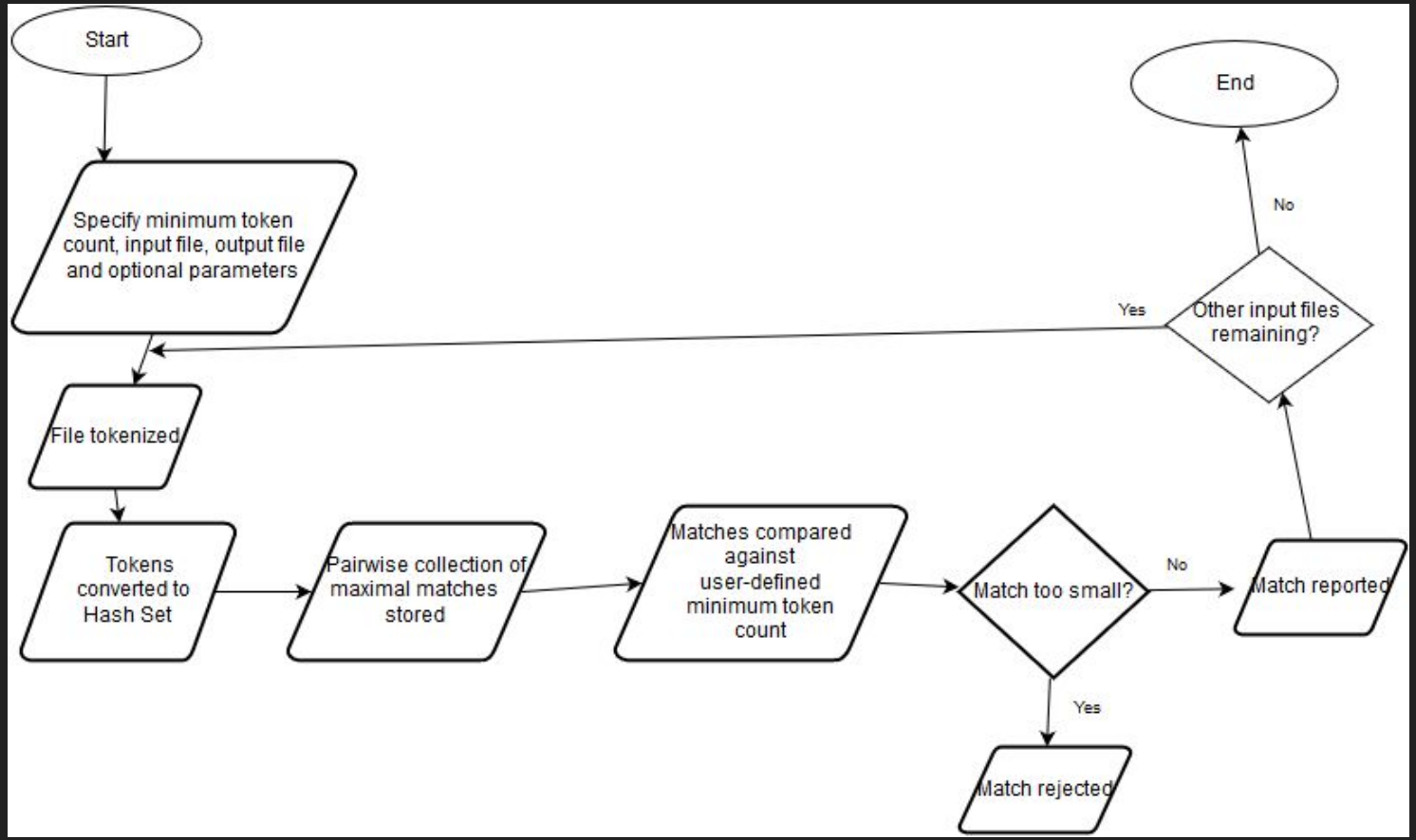
Data Flow in the Static Source Code Analyzer (between PMD and an external user)



Implementation Details Of AST Parser



Data Flow between CPD and an external user



Thank you!