Development View and Evolution Perspective of PMD

Aprajita, Inderjot, Senjuti

Development View

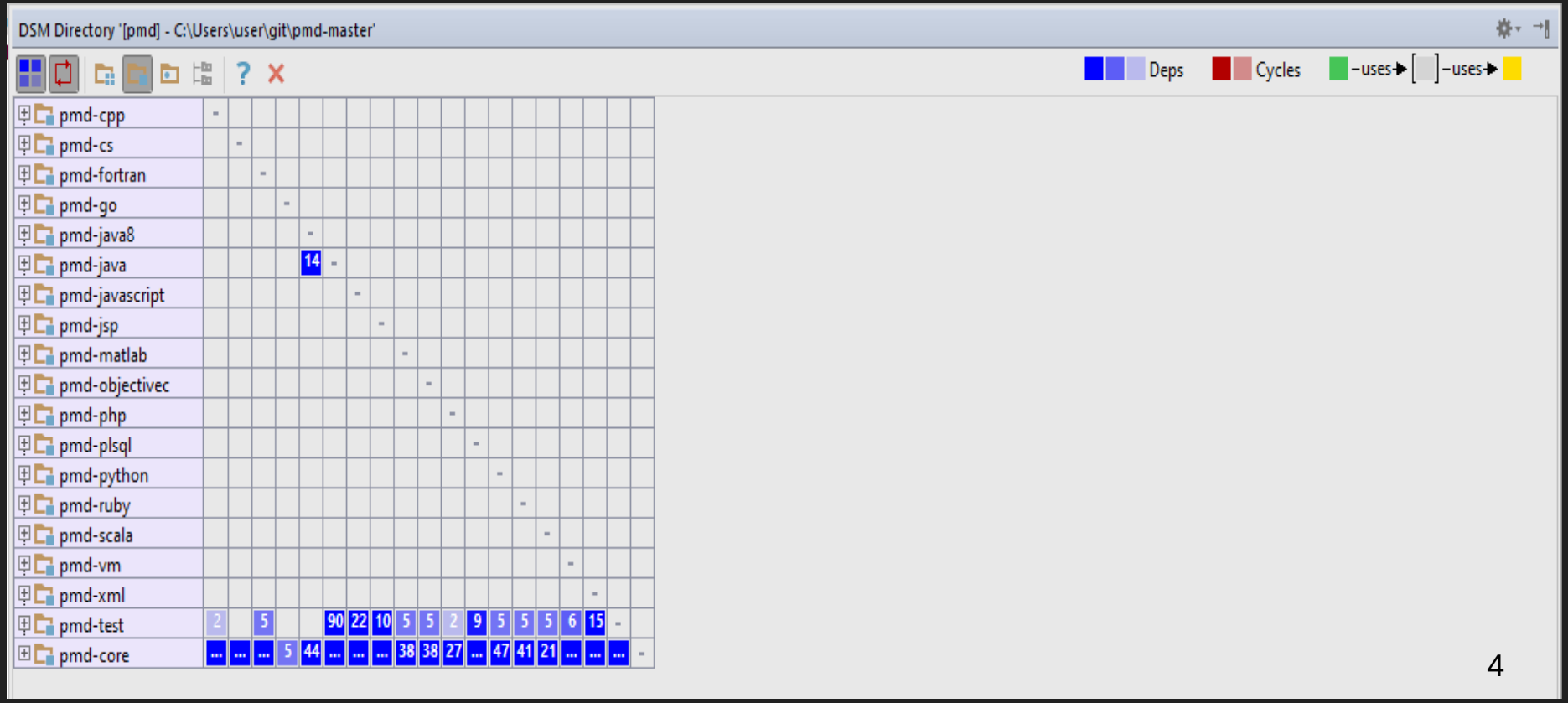
Concerns addressed:

- Module Dependencies and organization
- Commonalities in the design module
- Codeline organization

Module Dependencies and Organization

- All language modules are dependent on PMD-Core.
- Language modules are independent of each other.
- CPD module is loosely coupled and independent of other sub-modules present in the PMD-Core module.
- Language specific CPD modules depend on the CPD present in the PMD-core only.

Module Dependencies and Organization

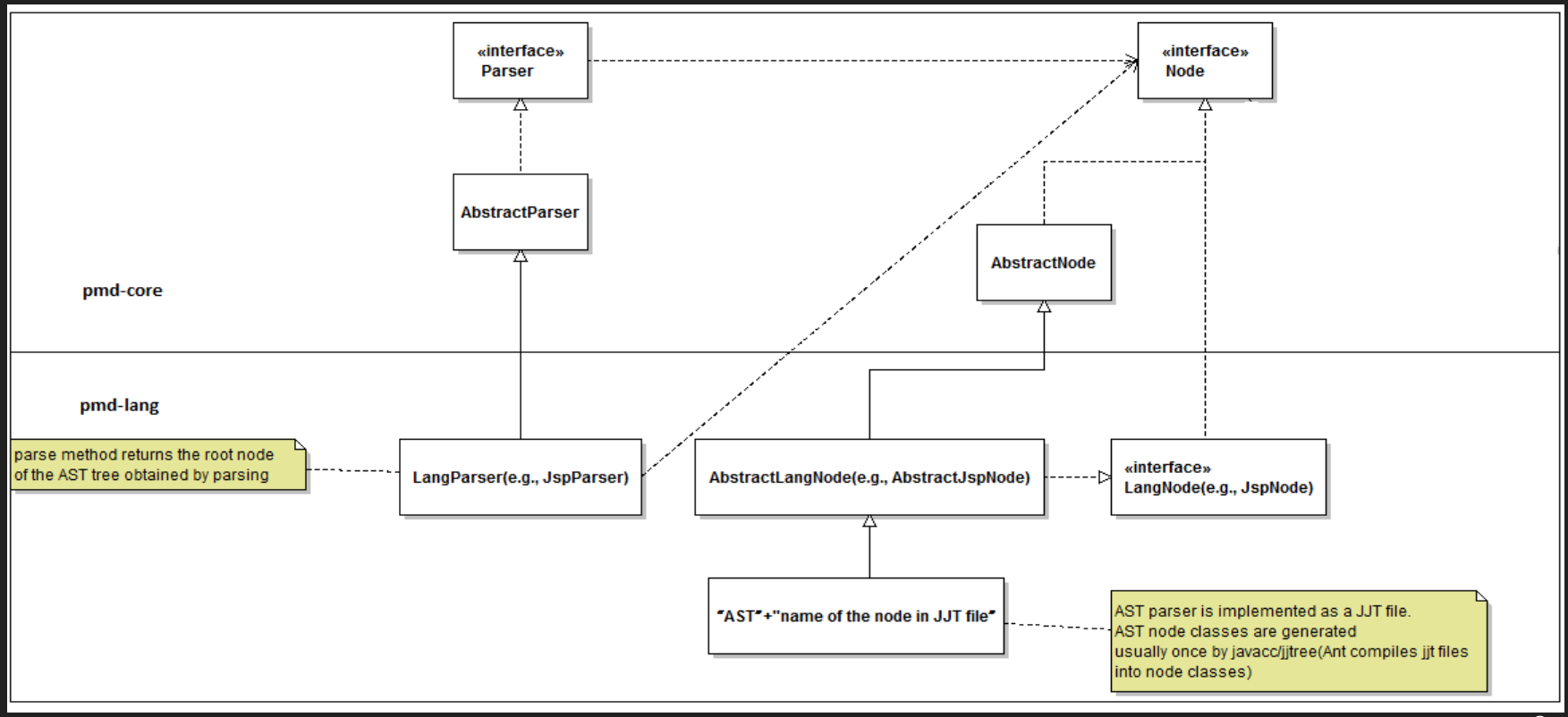


Common Design Model

Parsing of source files:

- Standard software components - jjt file.
- Standard design - shown in the next diagram

Common Design Model - Standard Parser Design



Common Design Model

Definition of Rules:

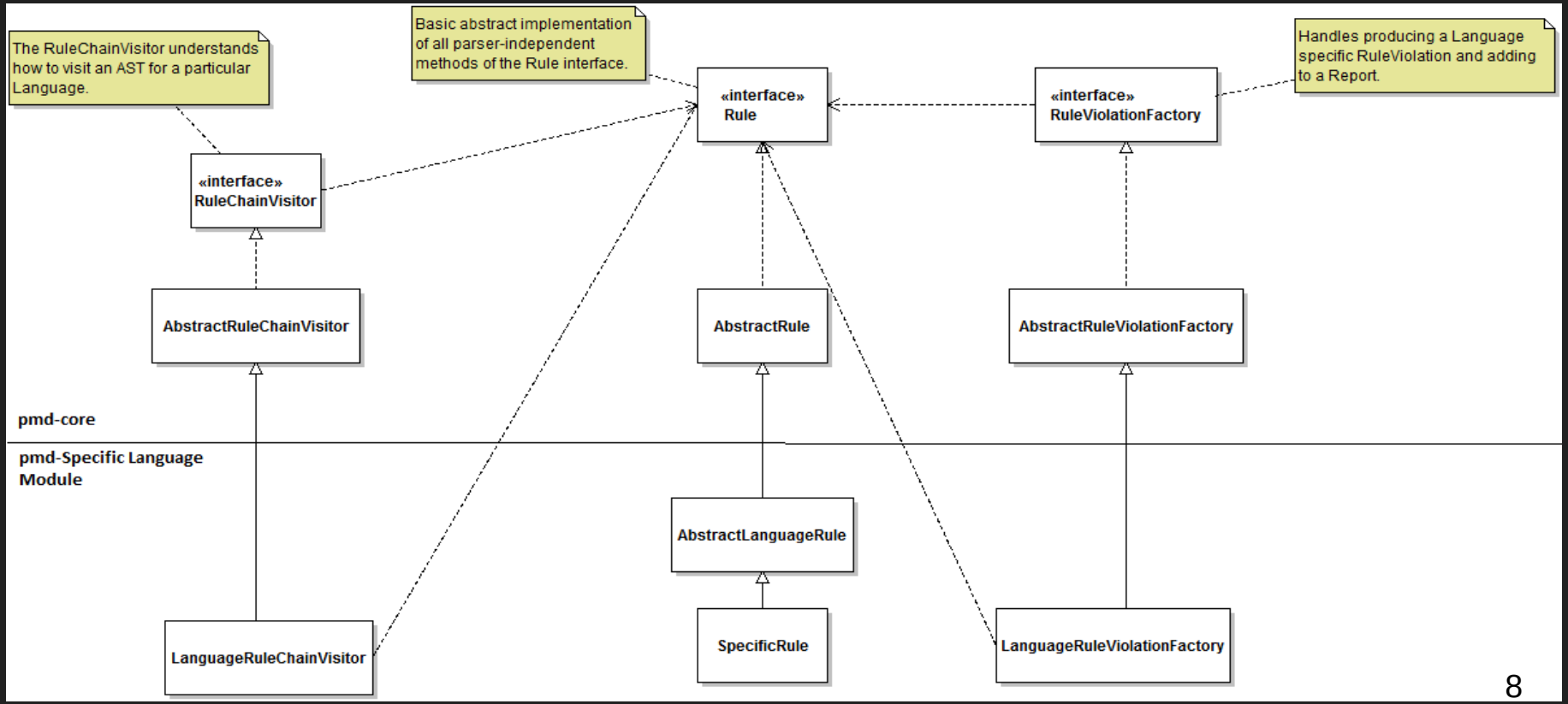
- Standard software components:

  For adding rules - .java class or xpath expression.

  Rules should be added to existing or new ruleset XML file.

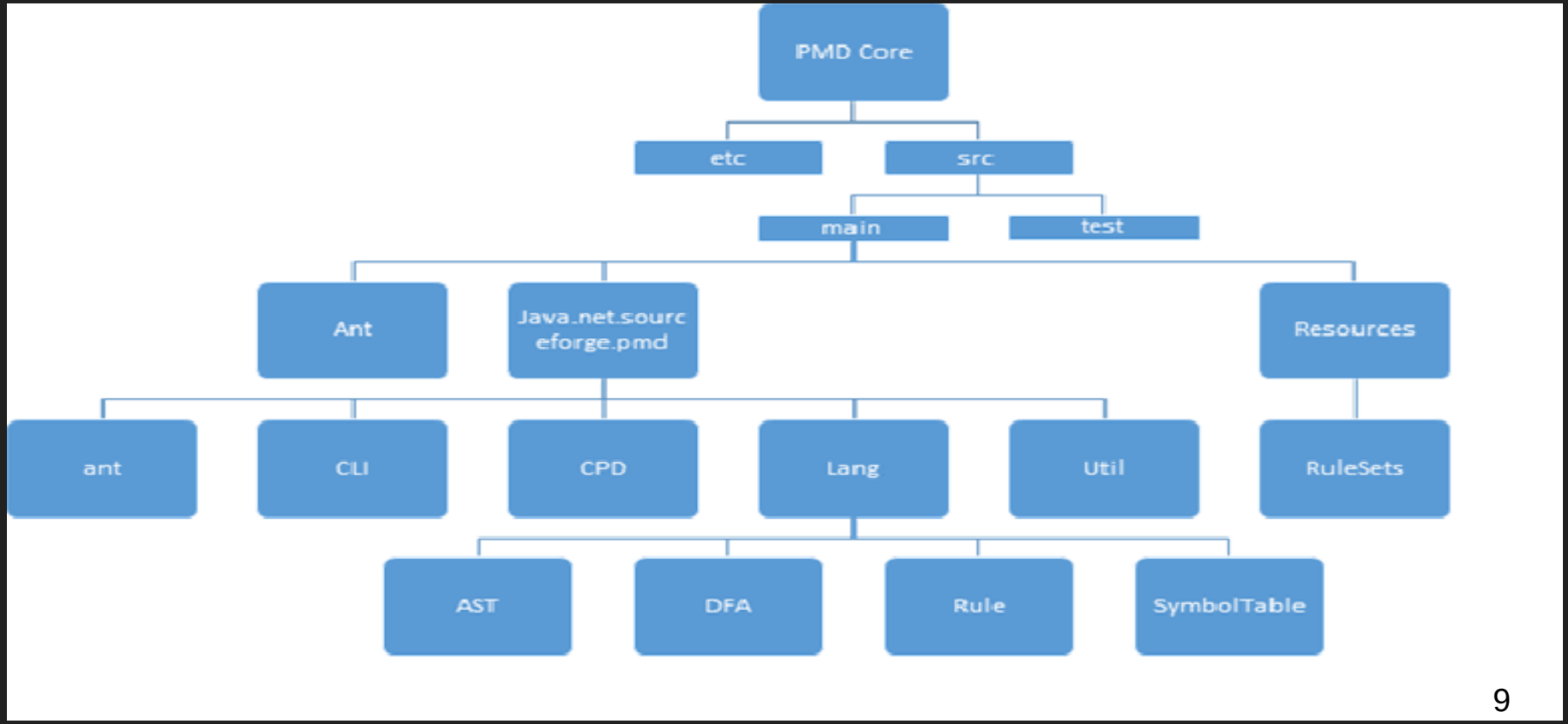
- Standard design - shown in the diagram below.

Common Design Model - Standard Rules Design



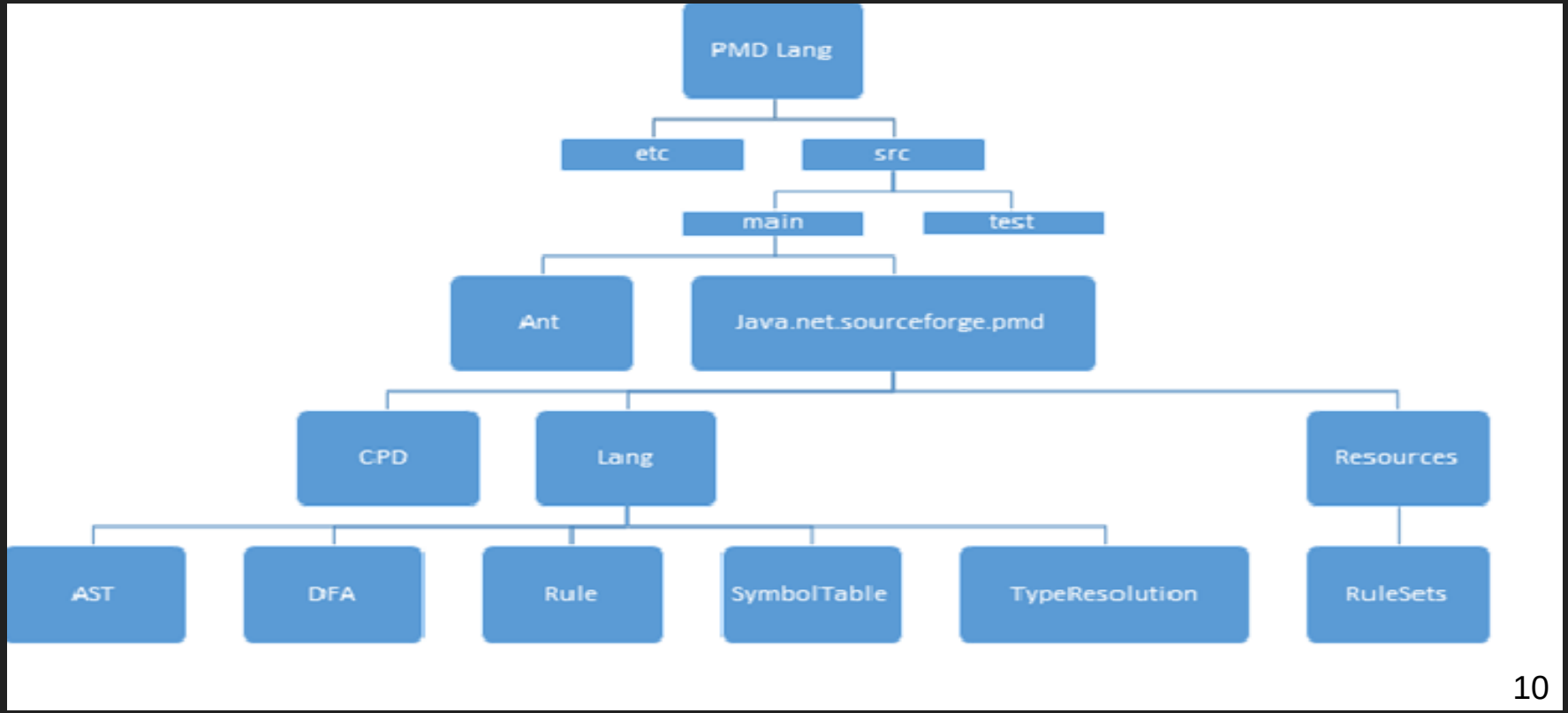
The Codeline Model

Directory structure for PMD-Core module:



The Codeline Model

Directory structure of PMD-SpecificLang module



Evolution Perspective

Concerns addressed:

- Dimensions of Change
- Evolution Trade-offs

Activities performed:

- Assess current ease of evolution
- How this perspective affected the architecture of PMD

Dimensions of Change

- Functional Evolution
- Platform Evolution
- Integration Evolution

Functional Evolution

- Addition of a new language

    * Magnitude: High/Medium

- Addition of new rules and/or rulesets

    * Magnitude: Low

Likelihood of functional changes: Total of 574 feature requests (226 open and 348 closed) and 275 patches (7 open and 268 closed)

Ease of evolution: Highly modularized, Easily extensible, Low inter-module dependencies

Integration Evolution

- IDEs widely used by developer community.
- PMD plugins can be created for IDE integration.
- Exist as different open source projects.
- Magnitude: High

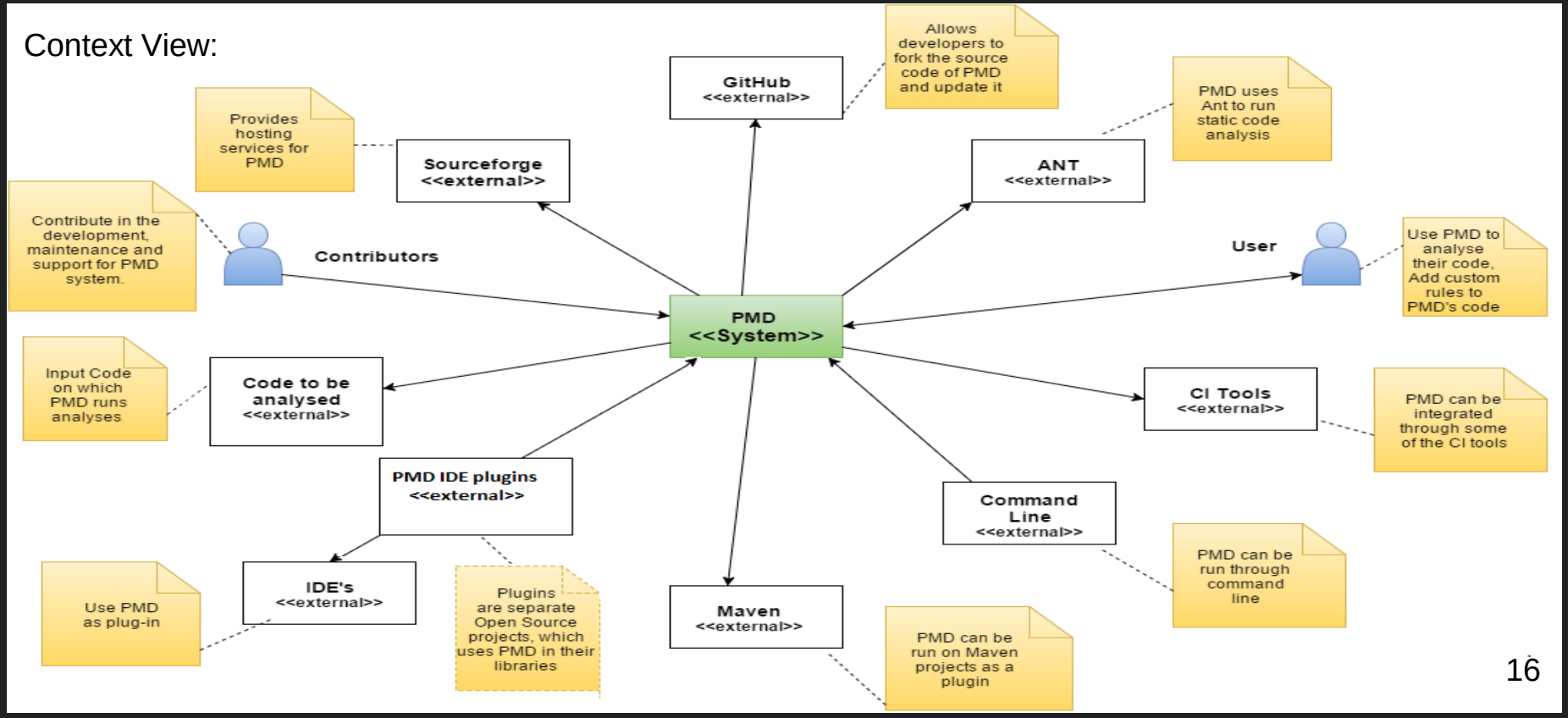
Evolution Trade-offs

Tactics identified:

- Contain Change
    * Separation of Concerns
    * Single Point of Definition
- Apply Design Techniques that facilitate Change
    * Generalization Pattern

Effect on PMD's Architecture

Context View:



Functional View

- “Handling of new languages” and “Addition of Rules and Ruleset” have been included as Internal Structures.
- Tactics Applied:

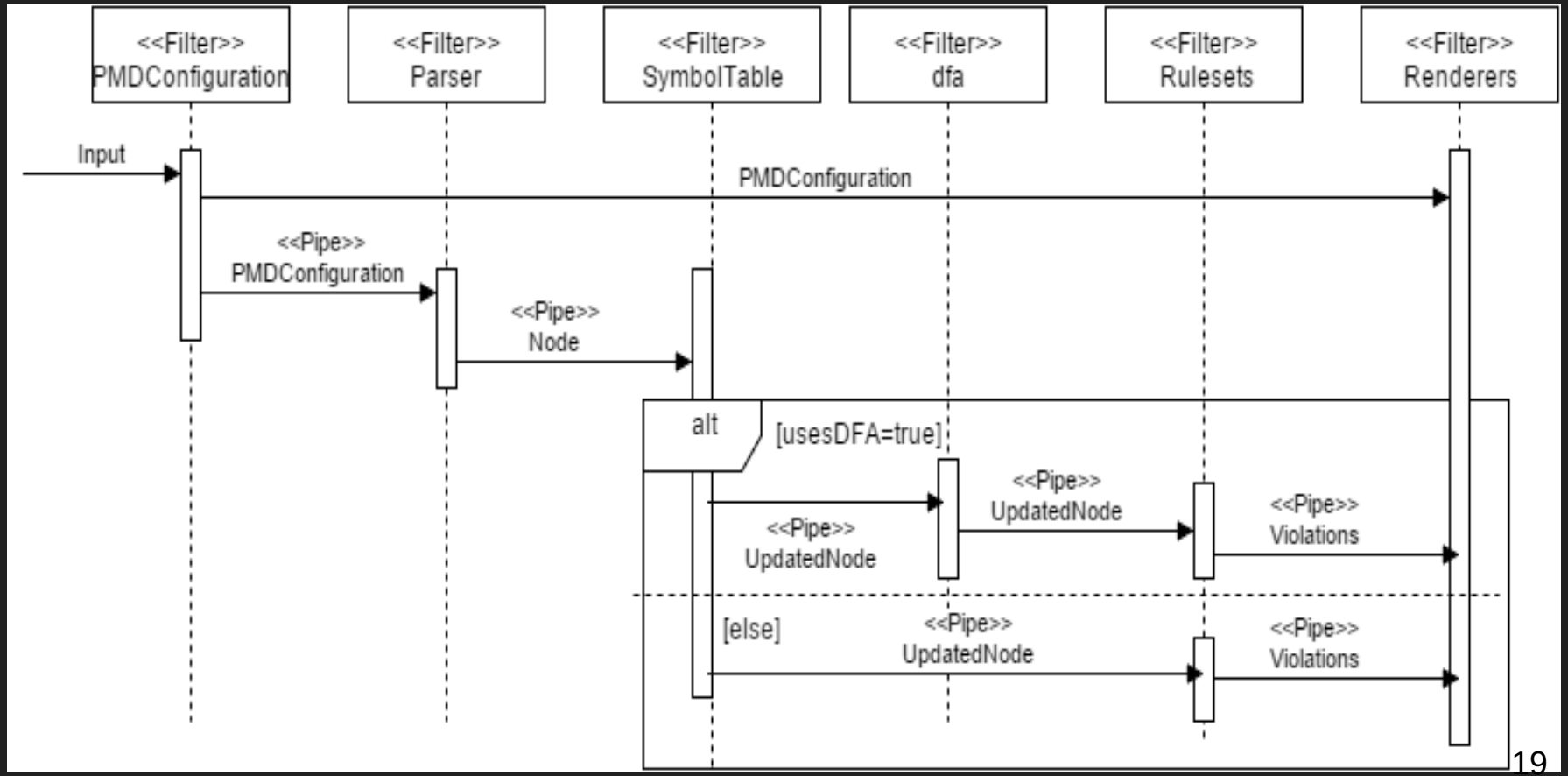
    * Separation of Concerns

    * Generalization Pattern

Information View

- Use of AST parser for every language
- Abstract Syntax Tree acts as common input to static source code analyzer
- Tactic applied: Single Point of Definition

Architectural Style- Pipes & Filters



Design Patterns

- Visitor Pattern - (AbstractRuleChainVisitor, ParserVisitor)
- Strategy Pattern - (MonoThreadprocessor, MutliThreadProcessor)
- Generalization Pattern