More About Code Inspection

Lecture 4b

Agenda

- Key Characteristics of Code Inspection Tools
- Categories of Code Inspection Tools

Key Characteristics of Code Inspection Tools

Key Characteristics of Code Inspection Tools

- **Be designed for security**
  - Tools that focus purely on software quality is good to some extent when it comes to robustness, but tools with a security module have more critical security knowledge built into them and the bigger the knowledge base a tool have the better.
- **Support multiple tiers**
  - Today not many programs are written solely in one language or targeted to a single platform. More often the application is written in a number of different languages and runs on many different platforms.

Key Characteristics of Code Inspection Tools

- **Be extensible**
  - A good tool needs a modular architecture that supports several different analysis techniques. In that way when a new attack is discovered the tool can be expanded to find them as well. Furthermore, the tool should have the ability to let the users add their own rules.
- **Be useful for security analysts, QA teams and developers alike**
  - The tools should make it possible for the analyst to focus their attention directly on the most important issues. Furthermore, it should support not only analysts but also the developer who need to fix the problems discovered by the tool.

Key Characteristics of Code Inspection Tools

- **Support existing development processes**
  - It should be easy to integrate the tool with existing build processes and Integrated Development Environments (IDEs).
- **Make sense to multiple stakeholders**
  - The tool needs to support the business. Different views for release managers, development managers and executives can support for example release decisions and help control rework costs.

Categories of Code Inspection Tools

Categories of Code Inspection Tools

- **Static checking**: The aim here is to find common coding errors.
- **Error detection/bug finding**: The aim of the tools in this category is to find bugs and report them.
- **Software verification**: The tools in this category guarantee the absence of errors by providing proofs.
- **Type qualifier inference**: The tools in this category specify as well as check program properties.

Code Inspection Tools per Categories

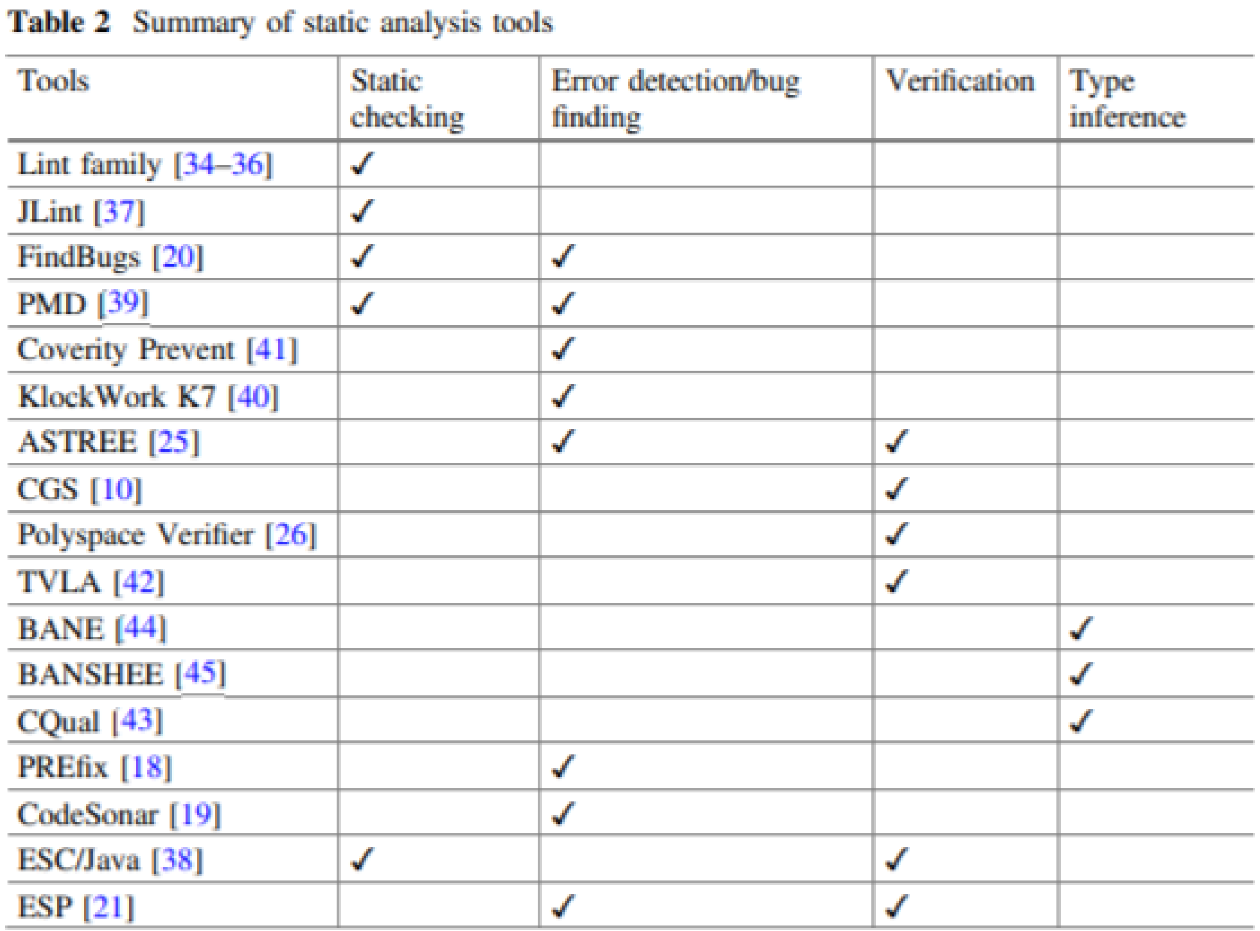

Table 2 Summary of static analysis tools

Tools	Static checking	Error detection/bug finding	Verification	Type inference
Lint family [34–36]	✓			
JLint [37]	✓			
FindBugs [20]	✓	✓		
PMD [39]	✓	✓		
Coverity Prevent [41]		✓		
KlockWork K7 [40]		✓		
ASTREE [25]		✓	✓	
CGS [10]			✓	
Polyspace Verifier [26]			✓	
TVLA [42]			✓	
BANE [44]				✓
BANSHEE [45]				✓
CQual [43]				✓
PREfix [18]		✓		
CodeSonar [19]		✓		
ESC/Java [38]	✓		✓	
ESP [21]		✓	✓	

Some Code Inspection Tools

- **PMD** is a static code analysis tool, capable of automatically detecting a wide range of potential defects and unsafe or nonoptimized code.
  - It focuses on pre-emptive defect detection. It comes with a rich and highly configurable set of rules, and you can easily configure which particular rules should be used for a given project.
  - It integrates well with IDEs such as Eclipse, and it also fits well into the build process thanks to its smooth integration with Maven.

Some Code Inspection Tools

- FindBugs is another static analysis tool for Java, similar in some ways to PMD, but with a quite different focus.
  - It is not concerned by formatting or coding standards and only marginally interested in best practices. In fact, it concentrates on detecting potential bugs and performance issues. It does a very good job of finding these, and can detect many types of common, hard-to-find bugs.
  - Indeed, it is capable of detecting quite a different set of issues than PMD with a relatively high degree of precision. As such, it can be a useful addition to your static analysis toolbox.

Next On

Take Away Points

- Code Inspection Tools
  - Which are the key characteristics of a tool to be considered?
  - Which are the categories of code inspection tools?
  - The names of some tools.

Further Reading

- Patrik Hellström, “Tools for static code analysis: A survey”
  - Chapter 2: Static Analysis
- Anjana Gosain and Ganga Sharma, “Static Analysis: A Survey of Techniques and Tools”
- Ian Sommerville, “Software Engineering”
  - Chapters 8, 12 and 24
- John Ferguson Smart, “Java Power Tools”
  - Chapter 22: Pre-emptive Error Detection with PMD
  - Chapter 23: Pre-emptive Error Detection with FindBugs

Next Lecture

- Functional Testing