

- A Study of Static Analysis Tool -

2009.

4

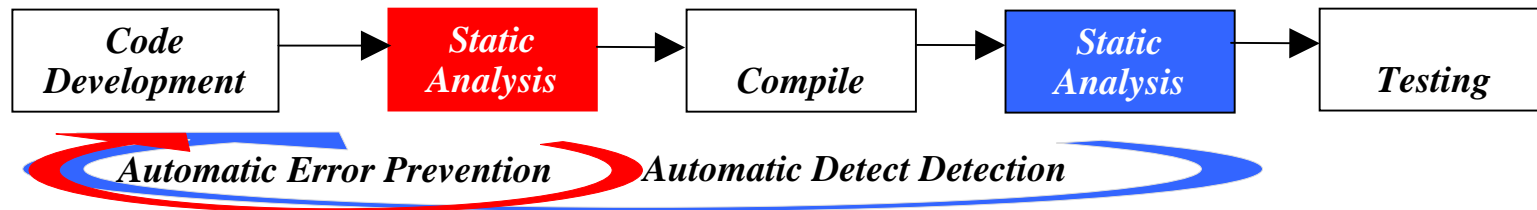
Soft 4 **Soft**



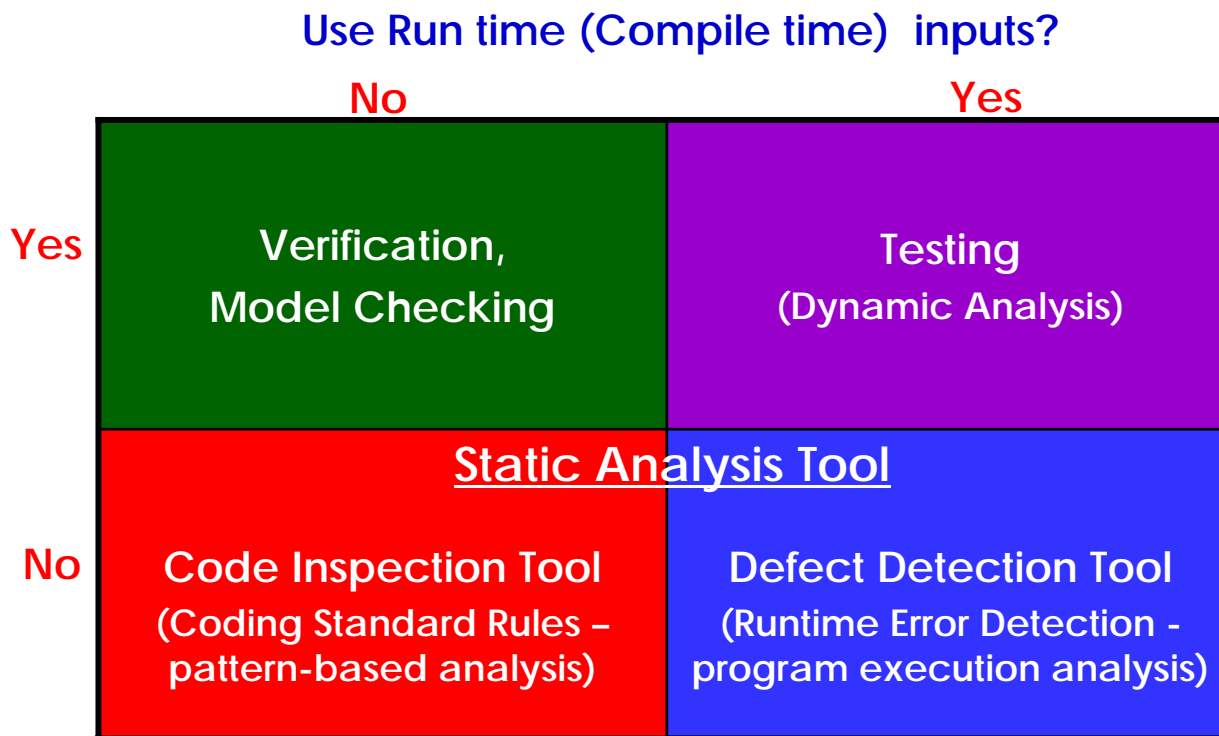
A Study of Static Analysis Tool

1

◆ The different categories of Static Analysis Tools



Use Behavior (Requirements) Specs?





◆ Static Code Analysis - Background

- The term *static code analysis* means different things to different people in the software industry. Two main static analysis approaches:

	Classification
	<ul style="list-style-type: none">• Pattern-based analysis• Program execution Analysis or• Data Flow-based Analysis (In the future, BugChecker Tool)

◆ Main Difference Analysis Technology

	Definitions
Pattern-based Static Code Analysis	<ul style="list-style-type: none">• An error prevention practice (The static analysis tool enforce proper input validation can prevent approximately 70% of the security problems cited by OWASP)• It means checking whether it has patterns known to cause defects or coding standard rules – (1)rules for preventing improper language usage, (2)satisfying industry standards (MISRA, JSF, Ellementel, etc.), and (3)enforcing internal coding guidelines
Program execution-based Static Code Analysis	<ul style="list-style-type: none">• An error-detection practice (It's not 100% accurate and you can't expect that it will uncover each and every bug lurking in your application)• Data flow static analysis statically simulates application execution paths, which may cross multiple units, components, and files (It's like testing without actually executing the code)• It can automatically detect potential runtime errors



◆ Static & Dynamic Tools - Pros and Cons

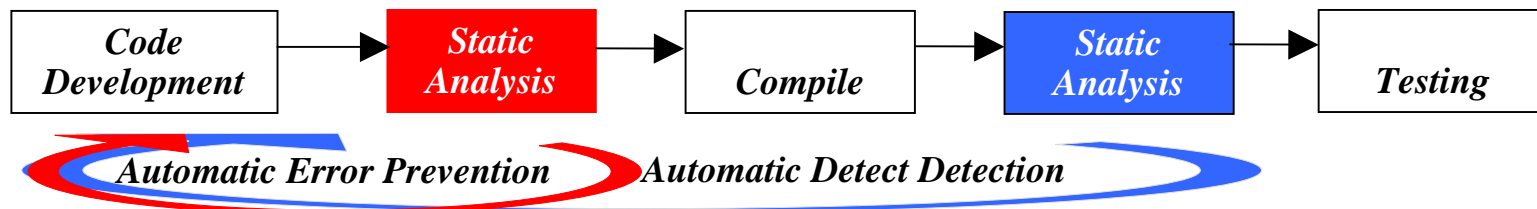
Tool	Pros	Cons
Static Tools	<ul style="list-style-type: none">• Can detect a wide range of coding flaws and find hard bugs• No test cases required to find problems• Faster and many more possible results (10x or more)• Bugs are uncovered very early in the SDLC	<ul style="list-style-type: none">• Watch out for the (large number of) false positive problems!• Does not relate to requirements (Incomplete information about the code)• Need programming competence to understand reports (Can be very superficial if it doesn't understand how your software is built from the source code)• Defects not (easily) found by static analysis (Functional incorrectness, Infinite loops, etc)
Dynamic Tools	<ul style="list-style-type: none">• Typically excel at detecting dynamic memory corruption and resource leak defects• Pinpoint precisely what went wrong in a given execution trace of the programs• Very low(no) false positive rate• Obviously relevant results	<ul style="list-style-type: none">• Only as good as your existing test suite• Potential performance implications that mask bugs (non-determinism)• Bugs are discovered later in the SDLC• False negatives



◆ Comparison of Static Code Analysis Tool - Summary

➤ Coding Rule Definition(Extensibility) - RESORT Tool

- **Error Prevention Approach** - Pattern-based analysis, Bugs are discovered early
- **Compiler Independent**
- **Bug Finding Time** - Any Time
- **No False Positive** - Detailed Rules Option
- **Works(analyzes) on Partial Code or Whole Code**



➤ Embedded Rule Engine(None Extensibility)

- **Error Detection Approach** - Program execution Analysis, Bugs are discovered later
- **Compiler Dependent**
- **Bug Finding(analysis) Time** - At Compile Time
- **Repeated Semi-manual Analysis to Eliminate False Positives of Warnings** - Need programming competence(expert) to understand reports, Need understand how SW is built from the source code
- **Works(analyzes) on Whole System** - If no, Possible Large False Positives.

If you cannot MEASURE it, you cannot IMPROVE it



4

103-6 KAIST

T215

Tel : 042 - 867 - 2277 ~ 8

Fax: 042 - 867 - 2279

: sales@soft4soft.com

: info@soft4soft.com

www.soft4soft.com