Benefits of Static Analysis

- Early and cheaper detection of bugs (before test execution starts)

- Warnings about where bug clusters might exist, due to dangerous programming, high complexity, etc.

- Location of bugs dynamic testing might miss

- Detection of dependencies and inconsistencies in software models (e.g., such as link problems in Web pages)

- Improved maintainability of code and design

- Prevention of defects based on metrics gathered and lessons learned from analysis

Typical Static Analysis Bugs

- Referencing a variable with an undefined value
- Inconsistent interface between modules and components
- Variables that are never used
- Missing or wrong logic
- Excessive complexity
- Unreachable (dead) code
- Programming standards violations
- Security vulnerabilities
- Syntax violations of code and software models

Using Static Analysis Tools

- Typical users are…
  - Programmers, often during unit and integration testing
  - Designers and system architects during design
- During initial introduction against an existing system, static analysis tools may produce a large number of warning messages
- Compilers do some static analysis, but many sophisticated tools are available

Static Analysis vs. Dynamic Analysis

- We can test code by executing it (dynamic testing)
- There are other ways to find defects :
  - Reviews: people find the bugs
  - Static analysis: specific methods and tools are applied to the code, looking for a number of different defects and anomalies without actually executing the code, using a tool
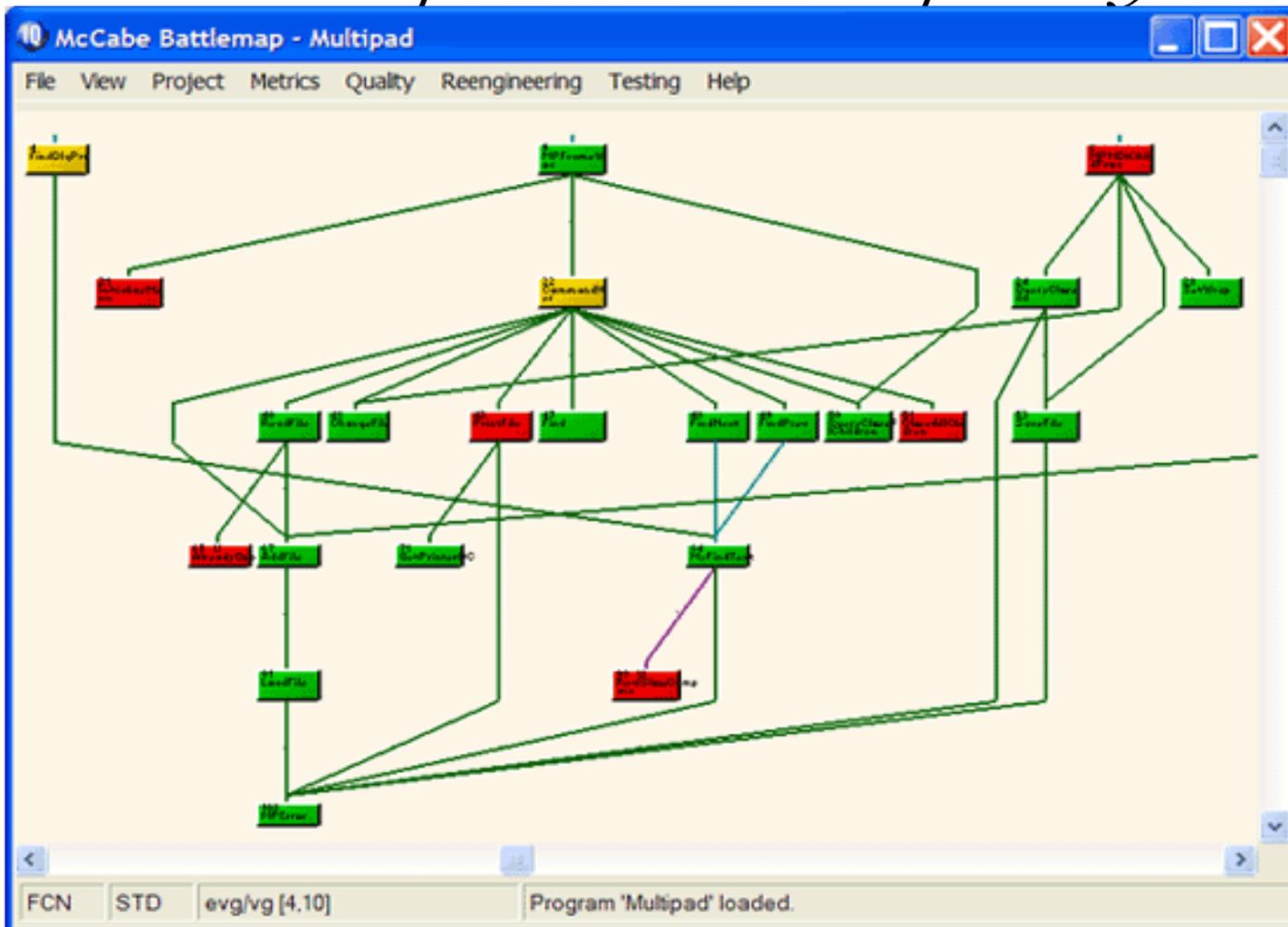 - Dynamic analysis: using tools to look for failures and anomalies in conjunction with executing the code

Example: Complexity Analysis
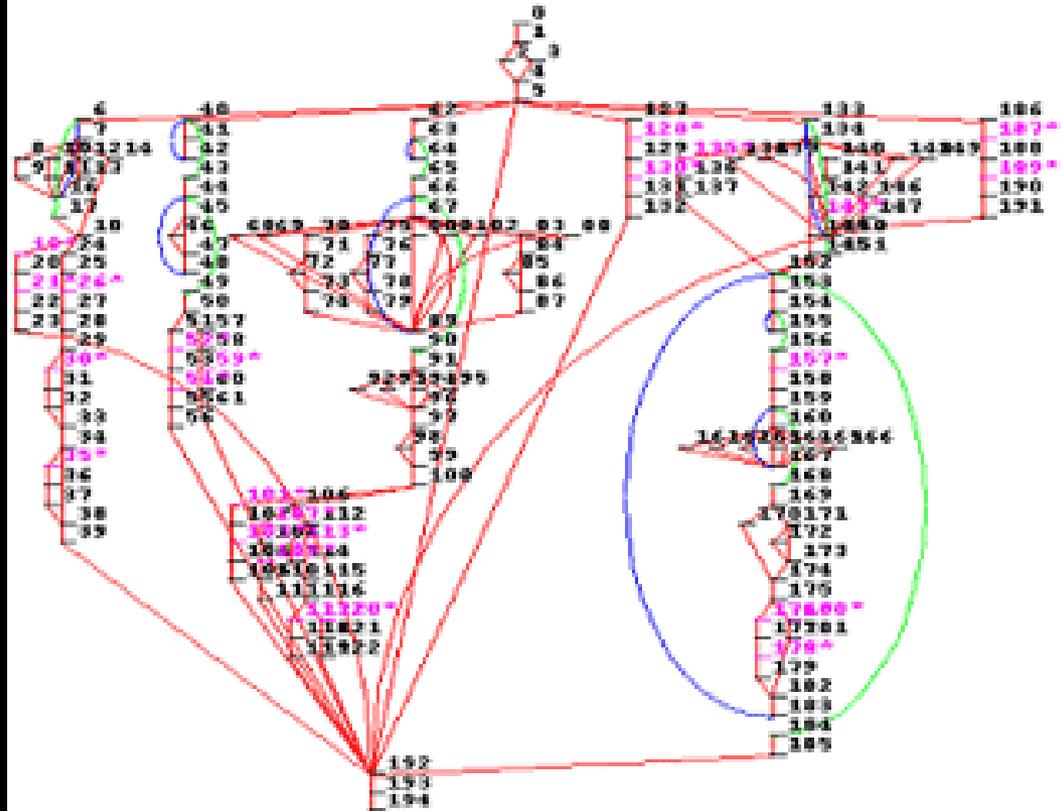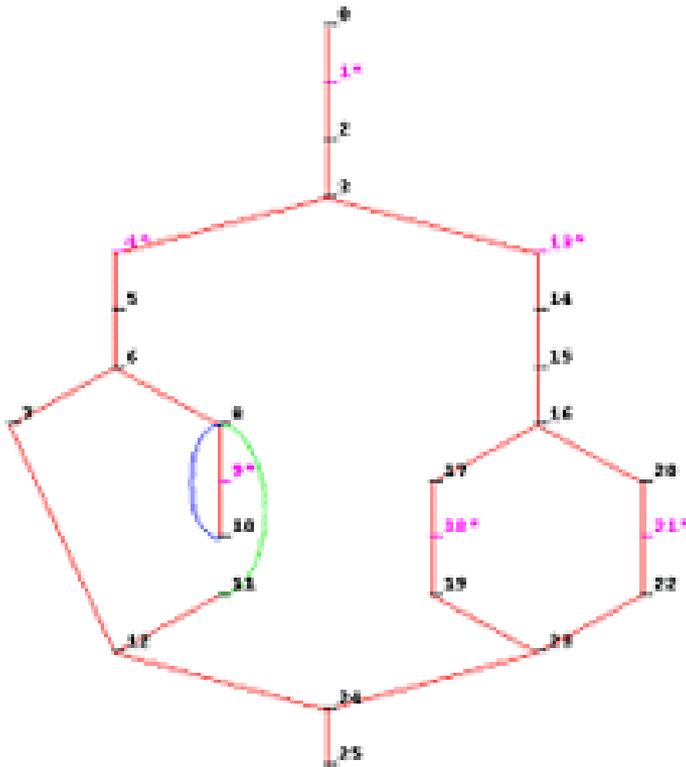
Simple Code vs. Complex Code

Example: Call Complexity

Example: Code Parsing Tools

```c
#include <stdio.h>
int main ()  {
  char c;
  while (c != 'x'); {
    c = getchar();
    if (c = 'x') return 0;
    switch (c)  {
      case '\n':
      case '\r':
        printf("Newline\n");
      default:
        printf("%c",c);
    }
  }
  return 0; }
```

Output of **Splint**, an open source static analysis tool

1. Variable c used before definition

2. Suspected infinite loop. No value used in loop test (c) is

3. Assignment of int to char: c = getchar()

4. Test expression for if is assignment expression: c = 'x'

5. Test expression for if not Boolean, type char: c = 'x'

6. Fall through case (no preceding break)

Example: Checkstyle Capabilities

- Check for Javadoc comments for classes, attributes and methods
- Enforce naming conventions of attributes and methods
- Check cyclomatic complexity against a specified limit
- Check for the presence of mandatory headers
- Check for magic numbers
- Check whitespaces between specific characters
- Enforce good practices of class construction
- Check for duplicated code sections
- Check for visibility problems   *-- Wikipedia*

Conclusion

- In this webinar, we've seen how to apply static analysis to the find defects in code without running the system

- In our previous webinars, we looked at various behavior-based test techniques

- Static analysis allows us to use tools to evaluate the quality of the code directly

- Static analysis of code can find many defects cheaply and early in the lifecycle

…Contact RBCS

For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS conducts product testing, builds and improves testing groups and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address:	RBCS, Inc.
	31520 Beck Road
	Bulverde, TX 78163-3911
	USA
Phone:	+1 (830) 438-4830
Fax:	+1 (830) 438-4831
E-mail:	info@rbcs-us.com
Web:	www.rbcs-us.com