Characterizing Verification Tools through Coding Error Candidates Reported in Space Flight Software

C.R. Prause[1], R. Gerlich[2], R.Gerlich[2], A. Fischer[3]

Data Systems in Aerospace DASIA 2015

May 20th, 2015, Barcelona, Spain

[1]Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Bonn, Germany
E-Mail: Christian.Prause@dlr.de

[2]Dr. Rainer Gerlich System and Software Engineering BSSE
Immenstaad, Germany
E-Mail: Rainer.Gerlich@bsse.biz
Ralf.Gerlich@bsse.biz

[3]etamax space GmbH
Braunschweig, Germany
E-Mail: A.Fischer@etamax.de

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Contents

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Introduction

- Verification tools are in widespread use
- Their actual capabilities have not been systematically assessed yet
- To improve the situation, DLR has initiated an evaluation of 5 widely-used tools

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Error, Fault, Failure, Defect

- Error: Bad or undesired state
- Fault: Cause of an error (“coding mistake”)
- Failure: Externally visible non-compliance as result of an error
- Defect: Any trouble with a software product, its external behaviour or its internal features, including maintainability.
- Error may be abstract (“virtual machine”) or concrete (“on target hardware”)
- Every fault is a defect, but not vice versa.

Fault

⇩ leads to

Error

⇩ leads to

Failure

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

False/True, Positive/Negative

BSSE System and Software Engineering

	Defect present	Defect NOT present
Report present	True Positive	False Positive
Report NOT present	False Negative	True Negative

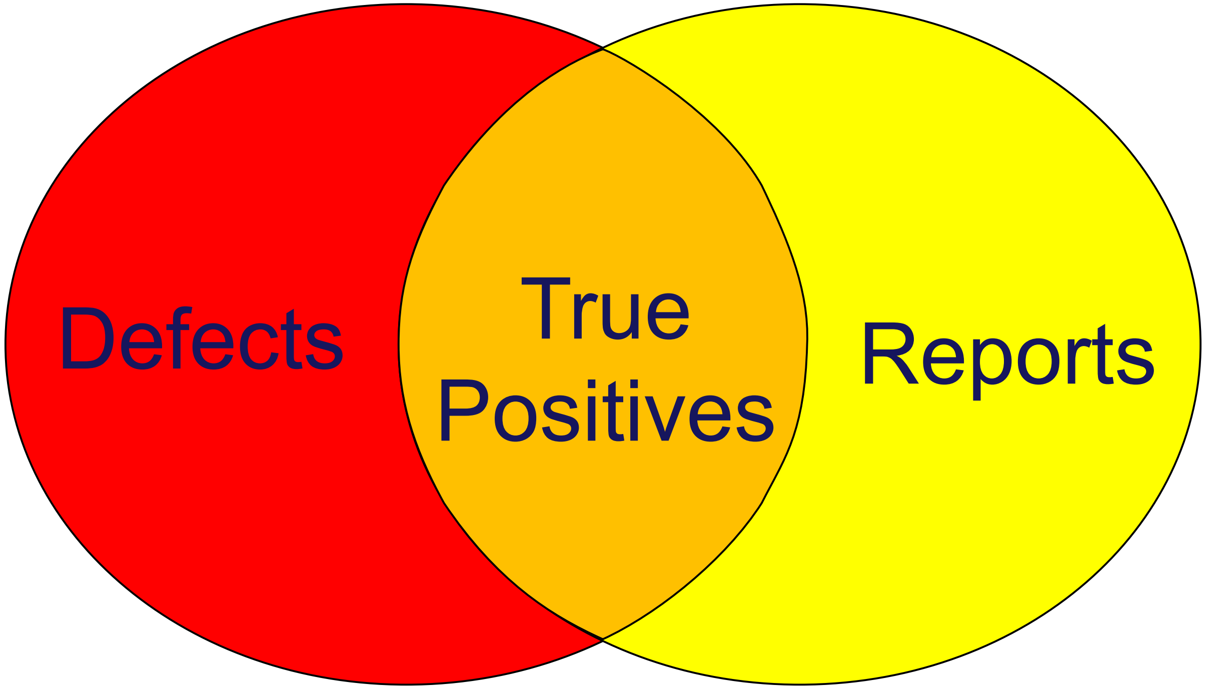
- False Negative: (Possibly critical) defect remains undetected
- False Positive: Added effort without added value
- High number of false positives may mean that not all reports can be analysed

  ⇒ True positives may effectively become false negatives

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Sensitivity, Precision

Sensitivity=
TP/Defects



Precision=
TP/Reports

- Higher sensitivity ⇒ Less false negatives
- Higher precision ⇒ Less false positives
- Ideally, both should be as high as possible.

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Verification Issues

- Verification tools shall aid in the detection of non-compliances
- But: Actual capabilities of tools (in contrast to advertised capabilities) are not known

⇒ Use of a tool provides nothing more than a good feeling

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

The Position of Standards

- ECSS (space domain)
  - Verification: Confirmation that product is built right
  - Recognises varying degrees of verification effort
  - Software Verification Plan subject to negotiations
  - Recommends use of static analysis tools in general
- DO178C/ED-12C (aviation)
  - Verification: detect and report faults (“unintended functionality”)
  - Detailed process definitions in the standard
  - Verification tools subject to qualification

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Standards vs. Tools

- Neither DO178 nor ECSS address tool characteristics
- DO178 requires tool qualification
  - Show that tool performs correctly in scenario agreed upon
- ECSS does not address tool characteristics
  - Tool selection must be agreed upon by customer and supplier
- Tool diversification is not a topic in either standard

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Fault Coverage vs. Code Coverage

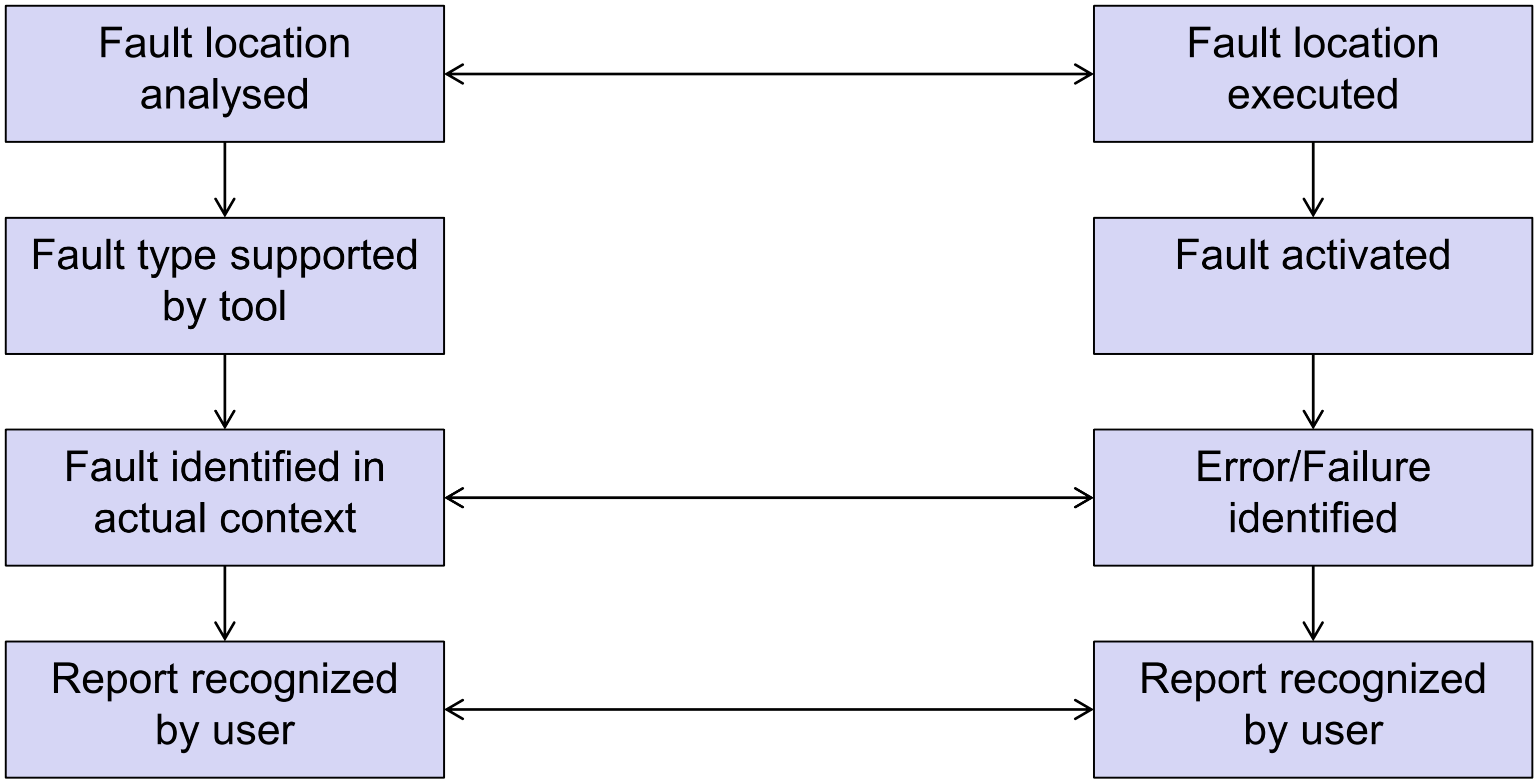
- Standards focus on code/requirements coverage
  - Define concrete coverage figures
  - ECSS: coverage fixed only for highest safety category
- Fault coverage is not addressed
  - Cannot be derived, as number of faults not known
  - But defect type coverage can be addressed

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Fault Detection

Static Analysis

Test



© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Complementary Analysis Methods

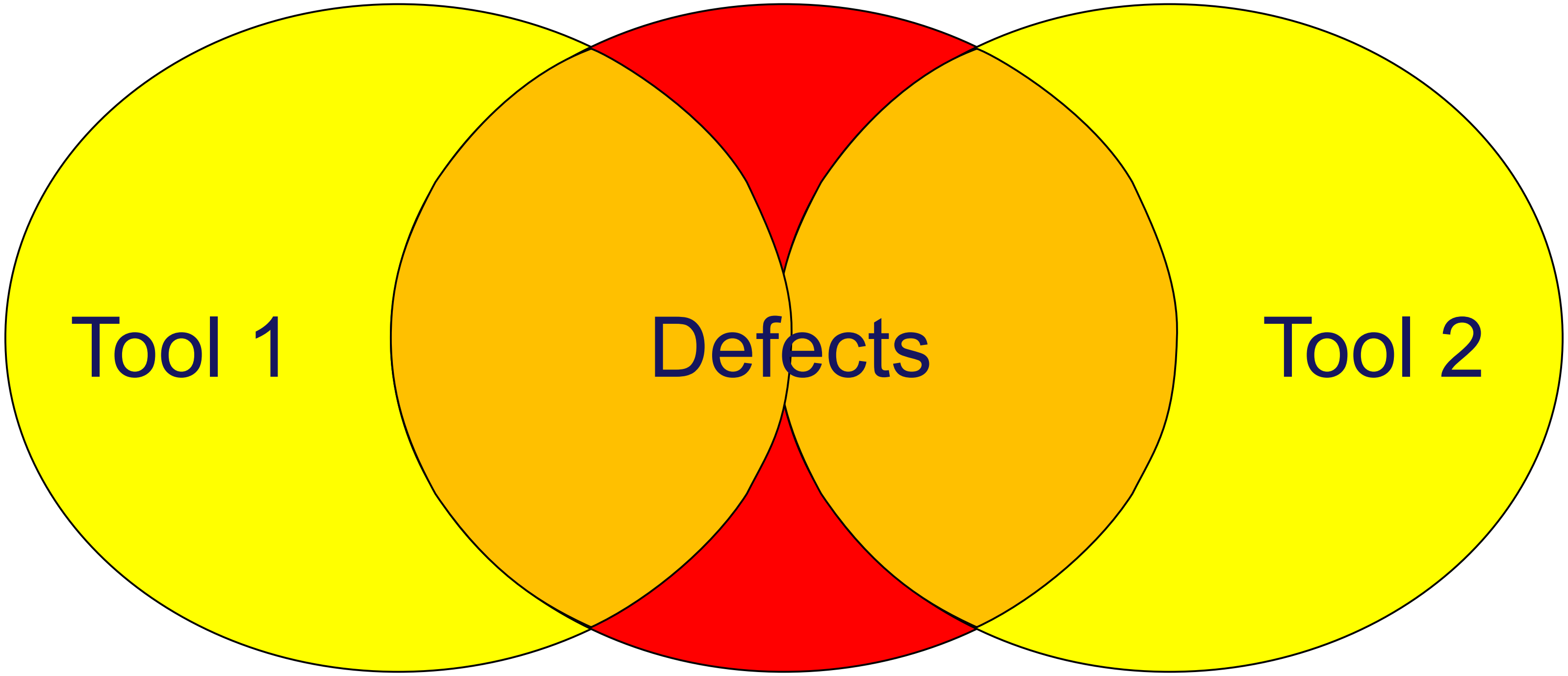
BSSE System and Software Engineering

- Dynamic Analysis
  - Limitations: Only sample scenarios
  - Representative Environment possible (“test on target”)
- Static Analysis
  - Limitations: Only specific defect types
  - Conservative guarantee possible, at the cost of limited precision
- Model-based Analysis may suffer from lack of model accuracy
  - E.g. Symbolic execution, abstract analysis

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Mastering Risks of False Negatives

- Risk: Tool is expected to cover defect type, but does not
- Mitigation
  - Knowledge about actual tool capabilities
  - Use of multiple, complementary tools



© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Objectives of the activity

- Estimate sensitivity and precision of verification tools by defect type, based on the analysis of a piece of space software.
- Establish a reproducible process for such an estimation.

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

General Process Considerations

BSSE System and Software Engineering

- Tools should be evaluated independently of each other
- Worst- and Best-Case
  - Initial Configuration without information
  - Optimized Configuration with feedback from tool vendors
- TP/FP status must be established
  - Manual analysis
  - Number of reports may be too high to analyse all
  - Random subset of reports is analysed
- Original and conditioned S/W version
  - Induce defects known from other space S/W

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Fault Activation

```
float sin(float x) { return 0.0f; }
void someSystemFunction(short n) {
  float y = sin((float)n*M_PI);
  /* ... */
}
```

- `sin()` is faulty
- But: no failure at system level!
- Fault is temporarily disabled

⇩ Maintenance

```
float sin(float x) { return 0.0f; }
void someSystemFunction(short n) {
  float y,z;
  y = sin((float)n*M_PI);
  z = sin((float)(n+1)*M_PI/2);
  /* ... */
}
```

- `sin()` is <u>still</u> faulty
- Temporarily disabled fault has been activated!

Is a report about the faulty sin() a false positive or not?

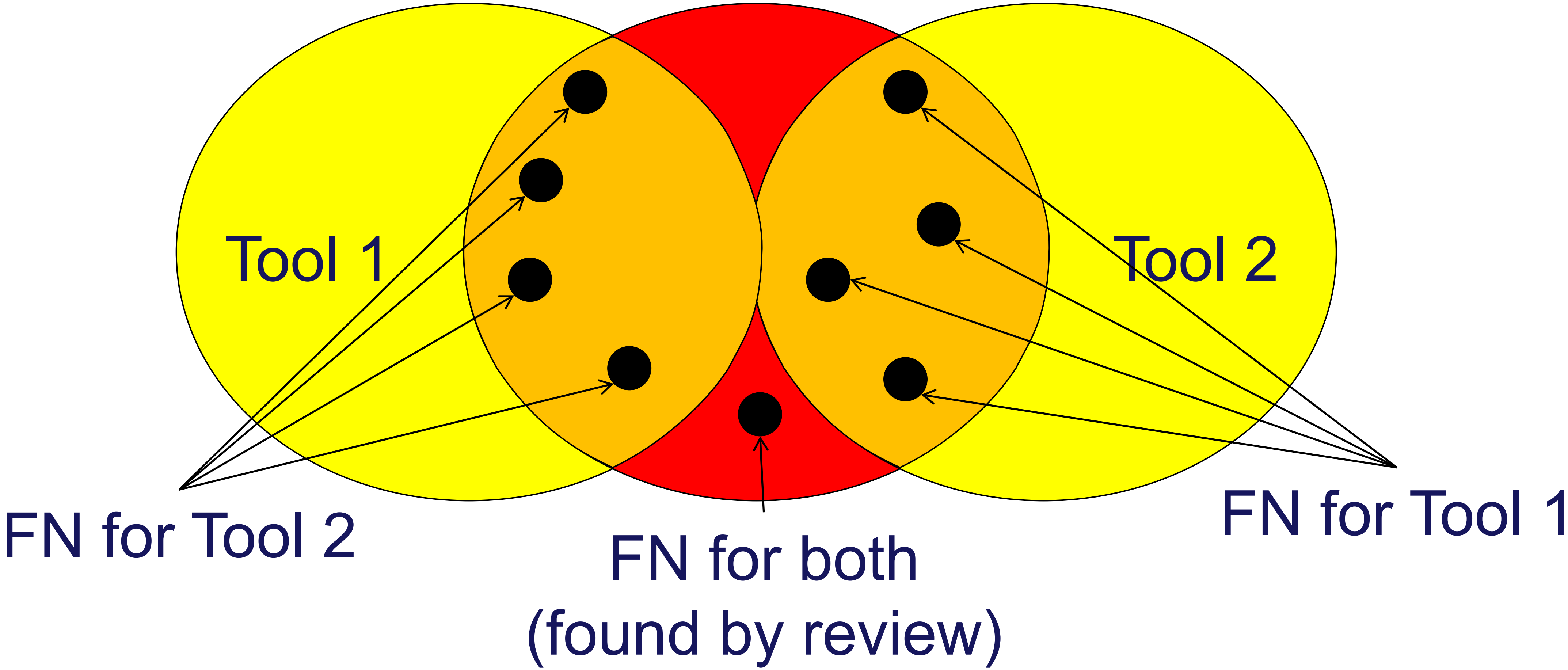
Conflicting Issues

- As a unit, the `sin()`-example is faulty ⇒ true positive
- At system level:
  - First version is correct ⇒ false positive
  - Second version is faulty ⇒ true positive
- More generic: “Design by contract”
  - Caller ensures pre-conditions
- But: additional effort to
  - document contract
  - prove adherence to contract at every call...
  - ...and every change!
- Increased risk for reuse

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

False Negative Detection

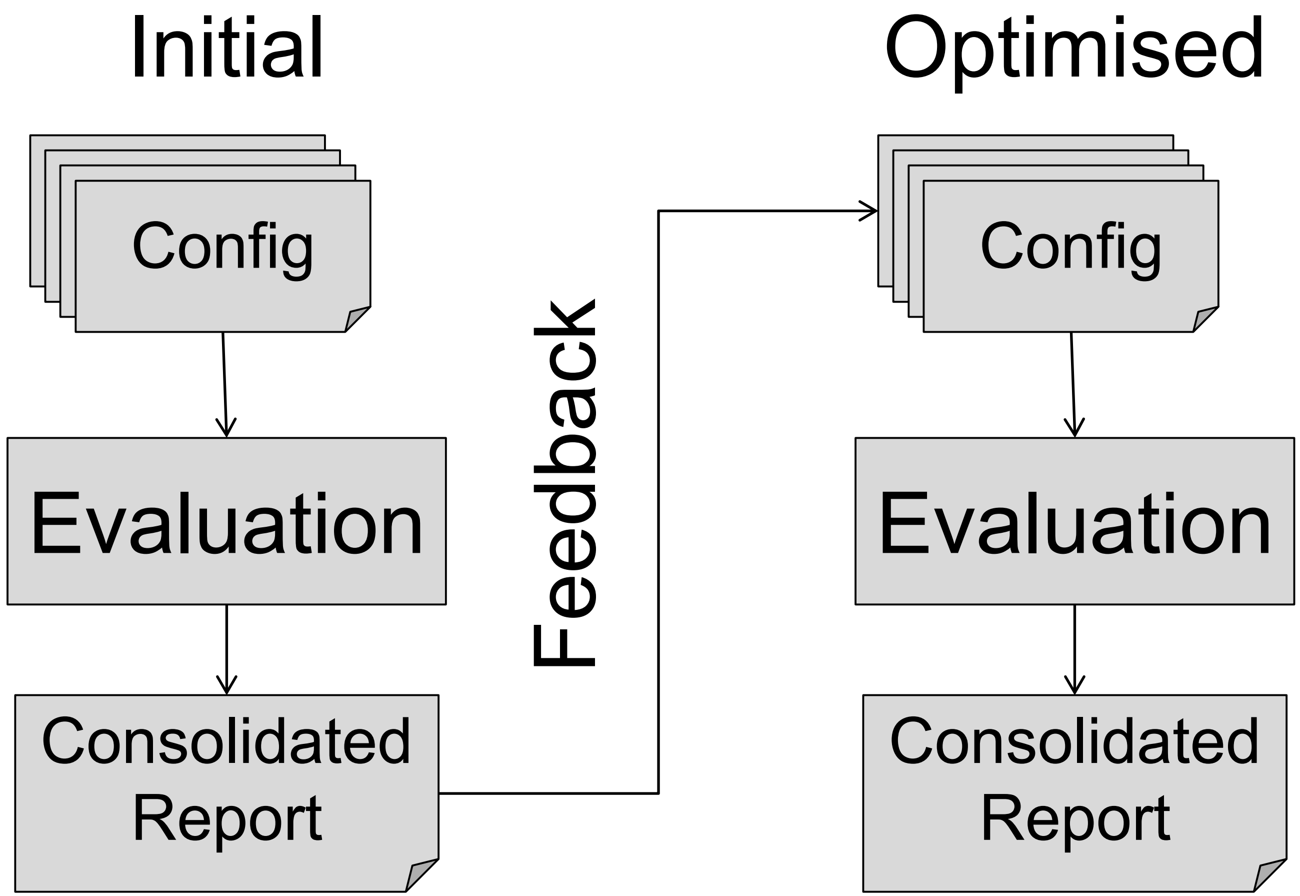
BSSE System and Software Engineering



- True positives for one tool may be false negatives for other tool
- Additional findings possible during report review

Process Overview

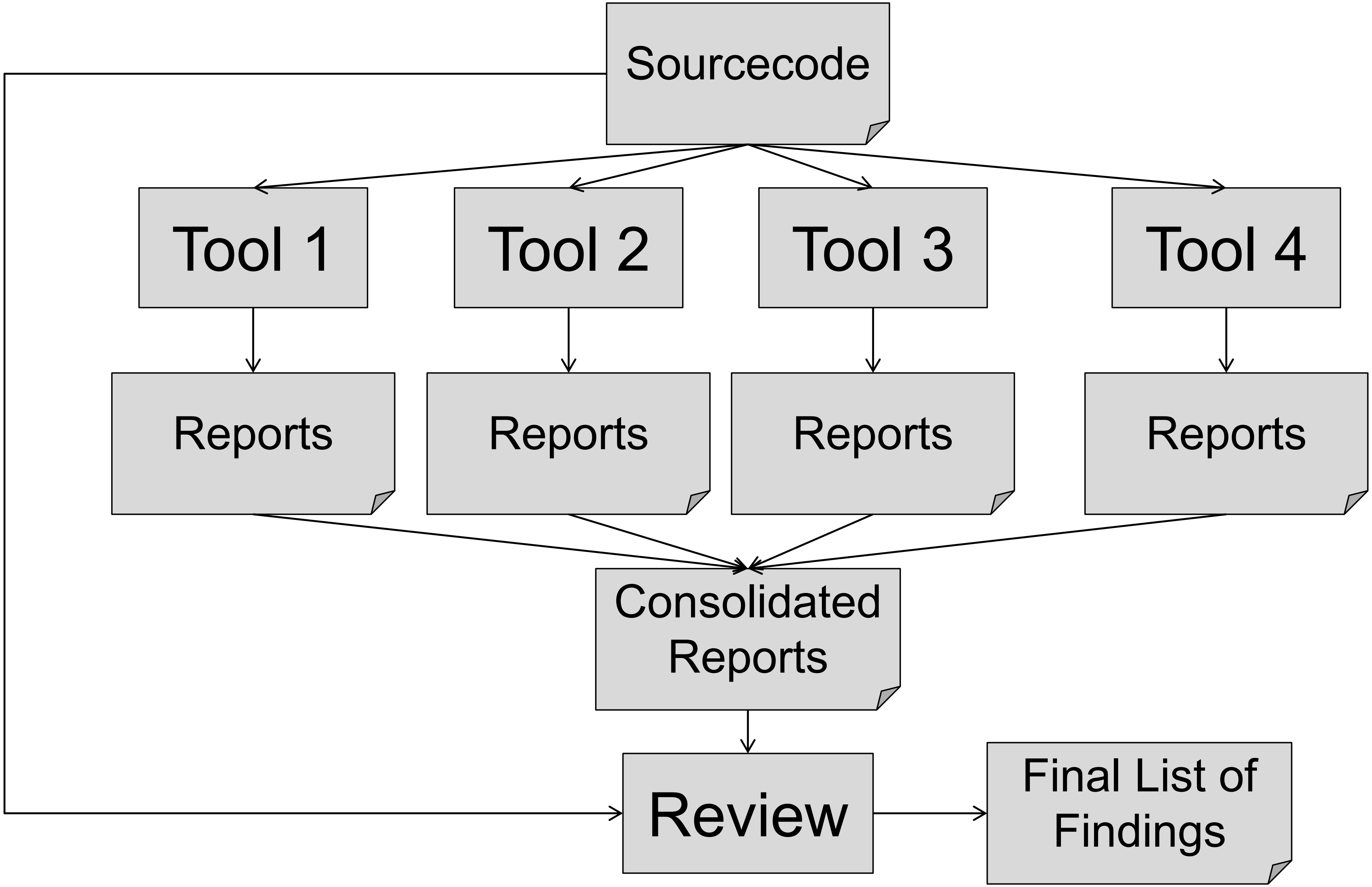
BSSE System and Software Engineering



Repeated for conditioned S/W version

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

The Evaluation Process



© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

The Tool Candidates

BSSE System and Software Engineering

- The candidate tools cover the methods
  - Abstract interpretation
  - Symbolic execution
  - Automated Test/Stimulation with heuristic oracles
- Tools are widely known and/or have been used in space projects already
- No manual intervention required, except...
  - configuration
  - analysis of results

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

The Selected Code

BSSE System and Software Engineering

- Developed for space use
- Size data:
  - 85 c-Files
  - 119 h-Files
  - 825 Functions
  - 45kLOC (w/o comments, empty lines)

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Conclusions

BSSE System and Software Engineering

- Study shall estimate sensitivity, precision of tools by defect type
- Major step forward regarding defect coverage expected
  - Tools can be selected matching verification strategy
  - Feedback to tool vendors
- May increase effectiveness and efficiency in S/W V&V
- Community is invited to contribute...
  - Known defect types
  - Tool suggestions for future investigations

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015

Thank you for your attention!

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2015