

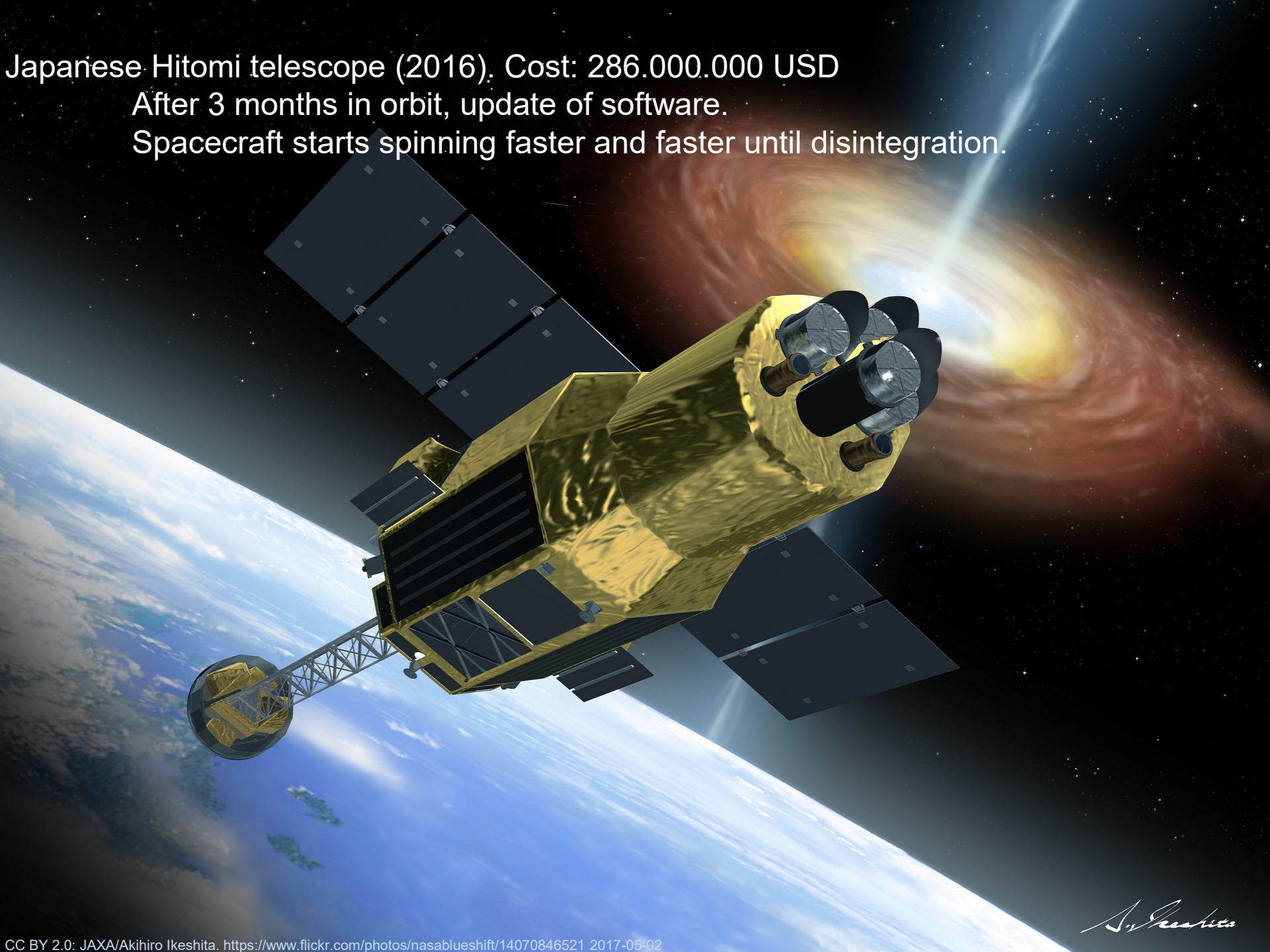
Evaluating Automated Software Verification Tools

Christian R. Prause


Rainer Gerlich

Ralf Gerlich

Japanese Hitomi telescope (2016). Cost: 286.000.000 USD
After 3 months in orbit, update of software.
Spacecraft starts spinning faster and faster until disintegration.



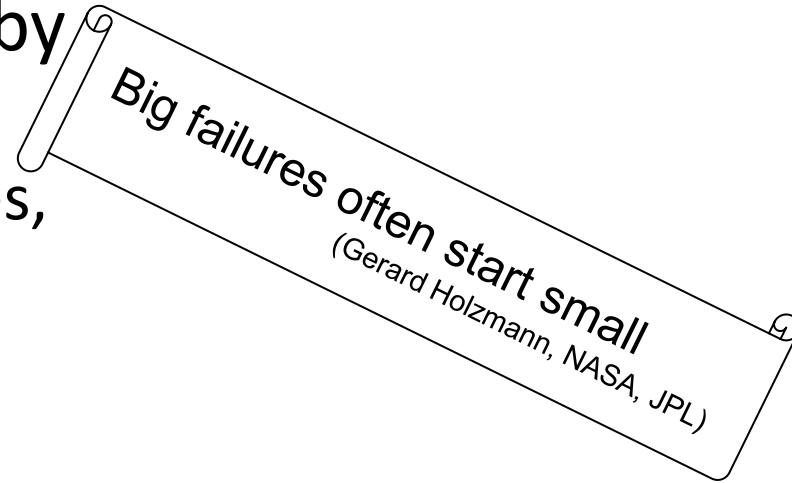
A. Ikeshita



Ariane 5.01, 1996
Cost: 1 billion EUR
Software glitch: on-board computers crash one after another.
Leads to development of Polyspace tool

Spacecraft Software

- Spacecraft = one-of-a-kind device
- Software assumes critical functions
- RAMS plays critical role, e.g. by
 - testing and validation,
 - safety & dependability analyses,
 - standardization,
 - process control,
 - process improvement
- Important: verification of code using **automated software verification (ASV)** tools



Automated software verification

- ECSS demand: “verify source code robustness”
- Examples
 - resource sharing, pointers, division by zero, control and data flow, internal consistency, non-deterministic behavior, data corruption, security breaches, square roots of negative numbers, overflows, underflows, out-of-bounds array, illegal type conversions, non-initialized data

ASV Tools

- even when tools seemingly have same functionality
 - underlying technology not comparable
 - each tool finds defects not found by others
- General motivation:
What can practitioners expect from using different tools?

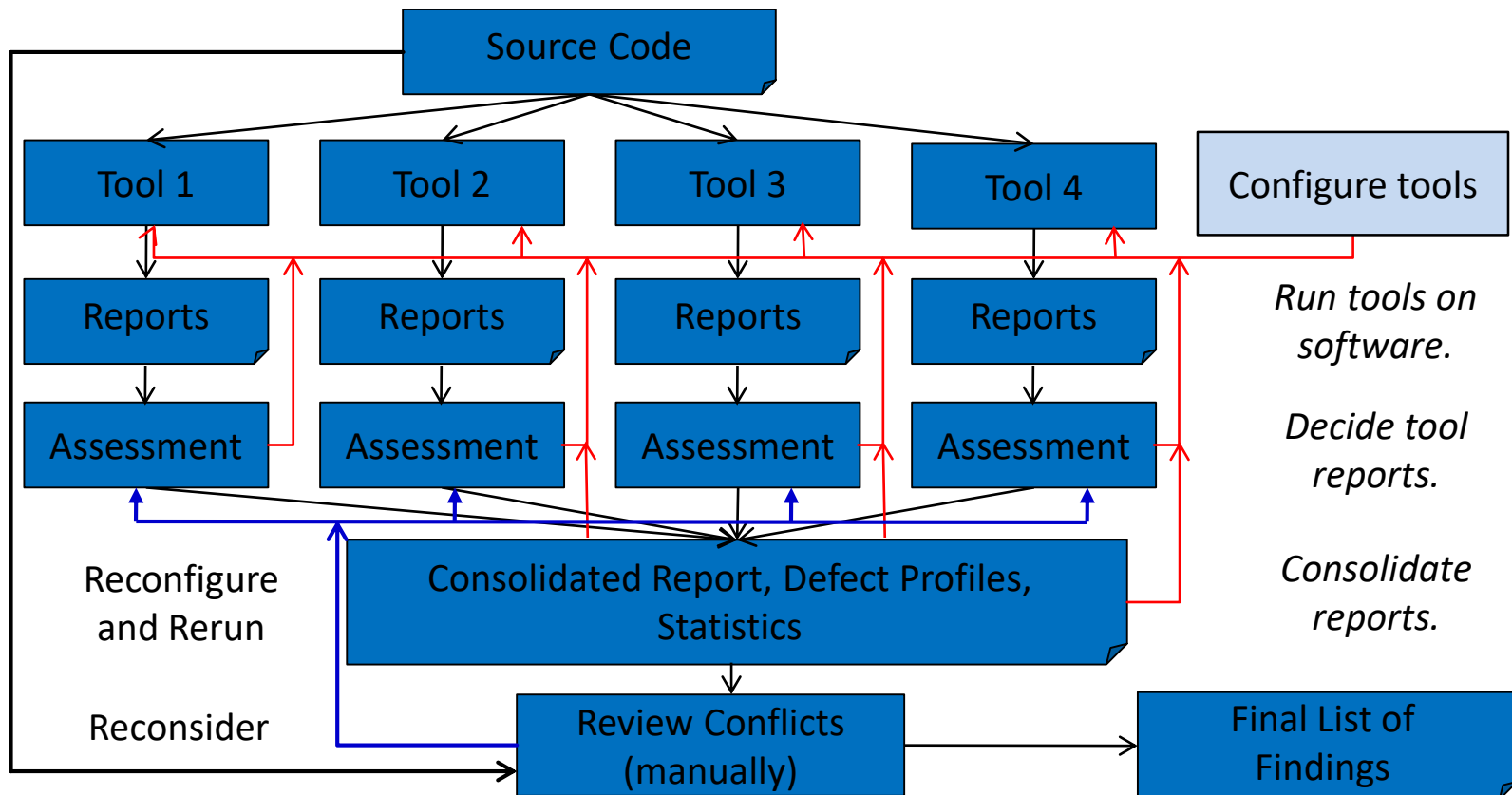
Research questions

1. Is it justified to apply ASV to already qualified software?
2. What is the best ASV tool available?
3. Are there significant differences between tools' capabilities?
4. Does longer analysis runtime mean less reports or better results?
5. Are tools that issue more reports less cost-efficient than tools that report fewer ones?
6. Is it effective and efficient to apply more than one tool?
7. Would a simpler evaluation (e.g., counting reports) lead to comparable results?

Materials / Method

- Material
 - real-world, qualified flight software
 - ASV: Polyspace BF+CP, QA C, Klocwork, DCRTT, gcc
- Method:
 1. Apply all six tools to software; collect the reports
 2. Consolidate reports into single set
 3. Validate each report by reviewing code
 4. Data analysis and evaluation

Overview Data Gathering Process



Report-by-report analysis

<i>Code</i>	<i>reported by tool</i>			<i>A</i>	<i>B</i>	<i>C</i>	<i>check</i>
<pre>int main(int argc, char* argv[]) { for(int i = 0; i <= argc + 1; i++) { printf("Arg %d: %s", i, argv[i]); } printf("Hallo Welt!"); }</pre>				X			✓
					X	X	✓ ✓
					X		✗
						X	✓

Output data

Contingency Table: Fault Reports

Classifying Reports

Signal Detection Theory:

$$\text{Sensitivity} = TP / (TP + FN)$$

$$\text{Precision} = TP / (TP + FP)$$



Defect present?

Defect present

Defect NOT present

Does tool issue a bug report here?

Tool issued report

Tool issued NO report

true positive

false positive

false negative

true negative

		Defect present?	
		<i>Defect present</i>	<i>Defect NOT present</i>
Does tool issue a bug report here?	<i>Tool issued report</i>	<i>true positive</i>	<i>false positive</i>
	<i>Tool issued NO report</i>	<i>false negative</i>	<i>true negative</i>

In Practice: Challenges

- Difficulties exporting data from tools
- Aligning fault types/reports (fault catalog)
- Critical vs. warning reports
- With- and without-context views
- Consecutive faults
- DeWitt-Clauses

Result data

	Overall	ToolA	ToolB	ToolC	ToolD	ToolE	ToolF
① Tool runtime (minutes)	-	10	300	600	15	3	5
② True positive $\xrightarrow{\text{with context}}$ false positive	11%	0%	13%	20%	0%	0%	10%
③ Total number of critical reports	86	8	73	48	1	11	58
④ Total number of warnings reported	184	13	14	71	1	17	96
④ Ratio of critical reports out of all reports	32%	38%	72%	29%	50%	39%	31%
⑤ Unique contribution (critical)	39	1	23	10	0	2	3
⑥ Consecutive fault ratio	$\approx 50\%$	32%	48%	38%	33%	72%	37%
⑦ Sensitivity / precision (total) in %	100 / 78	10 / 90	37 / 77	38 / 59	1 / 100	13 / 89	69 / 87
⑦ Sensitivity / precision (critical) in %	100 / 83	8 / 75	72 / 81	48 / 97	1 / 100	14 / 91	55 / 83
⑦ Sensitivity / precision (warning) in %	100 / 76	10 / 100	18 / 71	32 / 45	1 / 100	12 / 88	77 / 88
⑧ Time wasted on false positives (minutes)	373	20	54	269	0	25	88
⑨ Analysis time for all reports (minutes)	1083	80	372	735	11	97	510
⑨ Analysis time per source line of code (min.)	0.45	0.03	0.16	0.31	0.00	0.04	0.21
⑨ Min/max analysis time per report (minutes)	0 / 61	0 / 25	0 / 25	0 / 61	5 / 6	0 / 18	0 / 25
Ⓐ Avg. time to find a warning true positive	5.13	4.21	4.96	9.42	5.50	3.88	3.57
Ⓐ Avg. time to find a critical true positive	15.25	13.33	7.29	21.62	11.00	9.70	13.08
Ⓑ Similarity (Jaccard) to optimal profile	1.00	0.09	0.34	0.31	0.01	0.12	0.64
Ⓑ Similarity (Jaccard) to opt. critical profile	1.00	0.08	0.61	0.47	0.01	0.14	0.49
Ⓒ Avg. critical true positive when run as 2 nd	16.0	3.2	38.4	23.2	0.6	5.4	25.4
Ⓒ ... when run as 3 rd	11.2	1.9	30.0	16.1	0.3	3.1	15.9
Ⓒ ... when run as 4 th	8.2	1.4	24.7	11.7	0.1	2.2	9.4
Ⓓ Avg. additional total effort when run as 2 nd	226.1	42.8	245.6	614.8	8.6	59.0	385.8
Ⓓ ... when run as 3 rd	177.5	23.9	161.8	534.1	6.8	37.9	300.6
Ⓓ ... when run as 4 th	147.9	15.7	110.9	483.6	5.6	26.7	245.2
Ⓔ Avg. add. effort per true positive when 2 nd	14.5	13.37	6.40	26.50	14.33	10.93	15.19
Ⓔ ... when run as 3 rd	16.5	12.58	5.39	33.17	22.67	12.23	18.91
Ⓔ ... when run as 4 th	19.1	11.21	4.49	41.33	56.00	12.14	26.09
Ⓕ Predict functions with many warnings, R^2	0.43	0.02	0.00	0.12	0.01	0.21	0.24
Ⓕ Predict functions with many criticals, R^2	0.40	0.00	0.19	0.12	0.00	0.01	0.08
Ⓖ Perceived usability	-	++	0	0	+	+++	+

RQ Answers (short) (1/4)

- Is it justified to apply ASV to already qualified software?
 - Quality improvement: **Yes**.
 - Economic perspective: **maybe**.
- Are there significant differences between different ASV tools' capabilities?
 - **Yes**. Using different tools always adds something.

RQ Answers (short)

(2/4)

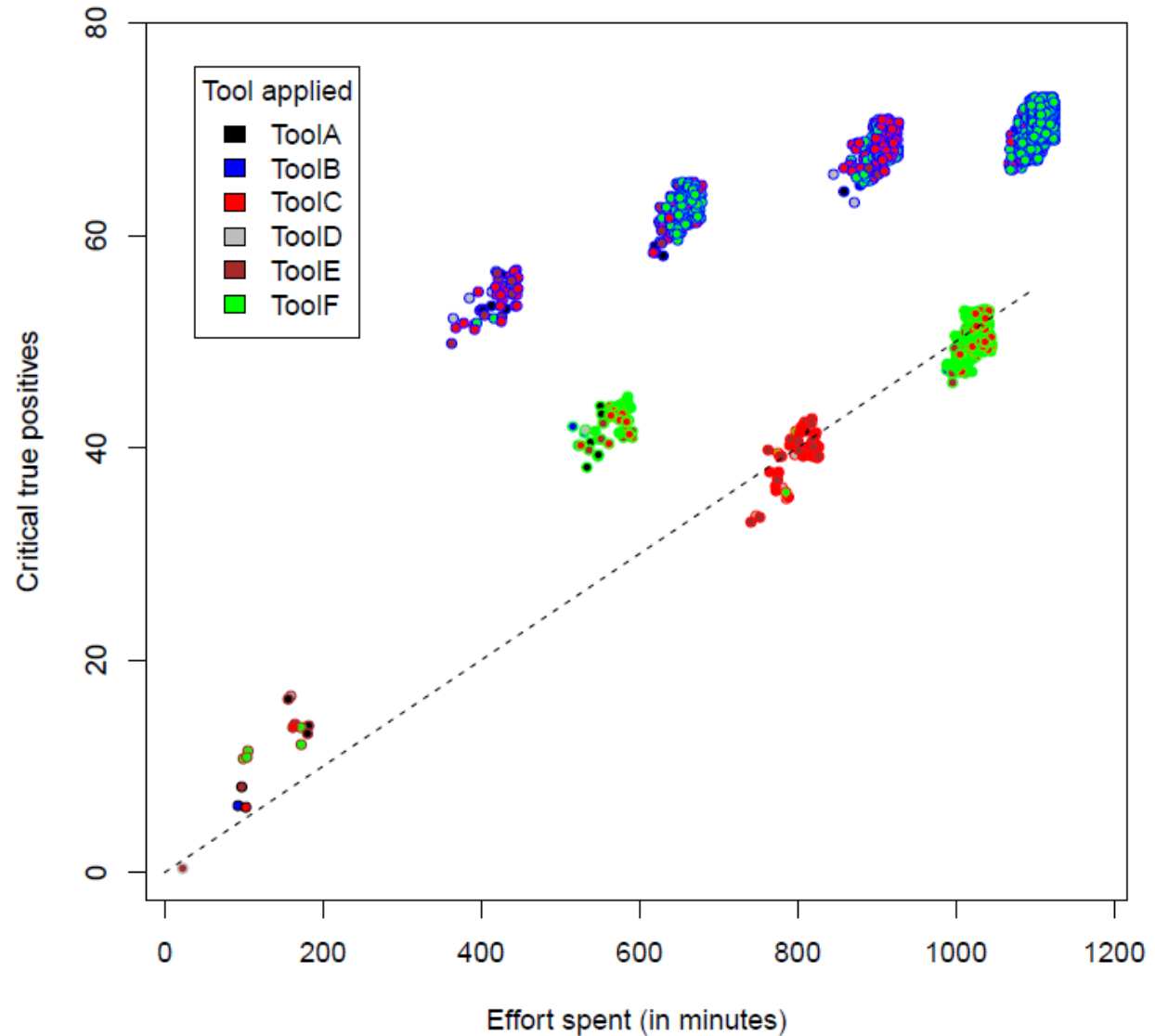
- What is the best ASV tool available?
 - **No** universal answer. It depends.
- Does longer analysis runtime mean more precise reports or better results?
 - **No.**
But longer runtime indicates focus on critical rather than warning messages.

RQ Answers (short) (3/4)

- Are tools that issue more reports less cost-efficient?
 - It depends. **Yes**, more reports. **No**, not more false positives.
- Is it effective and efficient to apply more than one tool?
 - Effective: Yes. More faults found.
 - Long-term efficiency:
 - Short-term efficiency: [next slide]

Combined efficiency vs. Review

- Two tools combined:
Similar to /
better than
reviewing



RQ Answers (short)

(4/4)

- Would a simpler evaluation method have led to comparable results?
 - Probably not.
- Often superficial analysis only (e.g., by interns)
 - Report counts alone are a bad predictor
 - Crafted test suites often neglect context
 - No clocking of analysis times
 - Consecutive faults
 - ...

Thank you for your attention!

- ASV tools important for avoiding costly failures
 - But tools are very different
- Present sophisticated method for characterizing tools
 - Analysis process
 - Data format for report database
 - Lessons learned
- Discussed several research questions

Data Format (columns per report)

Field	Description
ID	Unique identifier for the suspected fault; i.e., its ID in the consolidated list
file	Path and file name of the file where the report is found
function	Name of function containing the report
line	Source line number of report
type	Fault type according to our catalog
description	A human-readable description of report that justifies its type classification
implied by	If report is a consecutive fault, then the other one's ID (cf. grouping)
decision w/o context	TRUE if report is true positive without considering context, FALSE otherwise
analysis time	Minutes spent for analyzing report without considering context.
justification	Human-readable explanation for above decision
[decision with context]	Above three fields repeated for the "with-context" case
[found by ToolA...F]	For each ASV tool, whether or not it issued this report