Evaluation of Verification Results Continued: More Tools, More Software, More Aspects

R. Gerlich[1], R. Gerlich[1],
Sergio Montenegro[2], Frank Flederer[2], Jens Gerlach[3], Jochen Burghardt[3],
C.R. Prause[4]

Eurospace Symposium „Data Systems in Aerospace“ DASIA‘2017
May 30 – June 1, 2017, Gothenburg, Sweden

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

[3] Fraunhofer-Fokus, System Quality Center
Berlin, Germany
E-Mail: jens.gerlach@fokus.fraunhofer.de,
jochen.burghardt@fokus.fraunhofer.de

[2] Julius-Maximilians-University, Informatik VII
Wuerzburg, Germany
E-Mail: sergio.montenegro@uni-wuerzburg.de
frank.flederer@uni-wuerzburg.de

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

Contents

- Characterization of Tools and Applications
- Evaluation Process
- Analysis Results
- Unit Testing vs. Analyses
- Lessons Learned and Conclusions

Introduction

Note

The results presented here **strongly depend** on

- putting focus on safety critical issues

  tool messages must address faults which result in a failure or in a violation of good engineering practices

- the chosen application software and its fault profile

  some fault types may not be present

- the selected subset of functions subject of evaluation

  some fault types may not be present in this subset

- the observed number of defects per defect type

  the number of defects acts as a weight when deriving figures over all defect types

Characterization of Tools and Applications

Analysis Approaches and Tools

BSSE System and Software Engineering

Analysis Approaches	
static	abstract interpretation
	dataflow
	symbolic execution
	analysis based on dedicated checking and tracking
dynamic	auto-stimulation / automated testing

Tool		Type	Analysis Approach	Appl.
1	xxx	static	abstract interpretation	1
	Frama-C			2
2	yyy			1,2
3	DCRTT	dynamic	auto-stimulation	1,2
4	zzz	static	symbolic execution, dataflow analysis	1
	PC-lint		Analysis based on dedicated checking and value tracking	1,2
5	QA/C		Symbolic execution, dataflow analysis	1
	www			2
6	gcc	compiler	syntax, semantic, type checking	1

Tools vs. Application and Study

BSSE System and Software Engineering

Study	Tool					
	1	2	3	4	5	6
ESVW	xxx	yyy	DCRTT	zzz	QA/C	gcc
FSVW	Frama-C	yyy	DCRTT	PC-lint	www	

Appl.	Tool								
	1		2	3	4		5		6
1 / C	xxx	Frama-C	yyy	DCRTT	zzz	PC-lint	QA/C		gcc
2 / C++			yyy	DCRTT		PC-lint		www	

Application Characterization

BSSE System and Software Engineering

Property	Application 1 C	Application 2 C++
Size / KLOC, total h+c	42	20
Functions, total	610	611
c-Files, total	49	55
with functions	39	
without functions	49	
h-files	96	104
Functions, manually evaluated	60	60

Evaluation Process

Evaluation Criteria

Classification Category	Criterion	Applied Condition	Applied to Application
validity	tool	Is the tool message formally correct?	1,2
	state	Can an undesired state be reached?	2
context	with context	Input domain may be constrained by callers	1,2
	without context	Maximum input domain can be used	1,2

Doubts on tool criterion

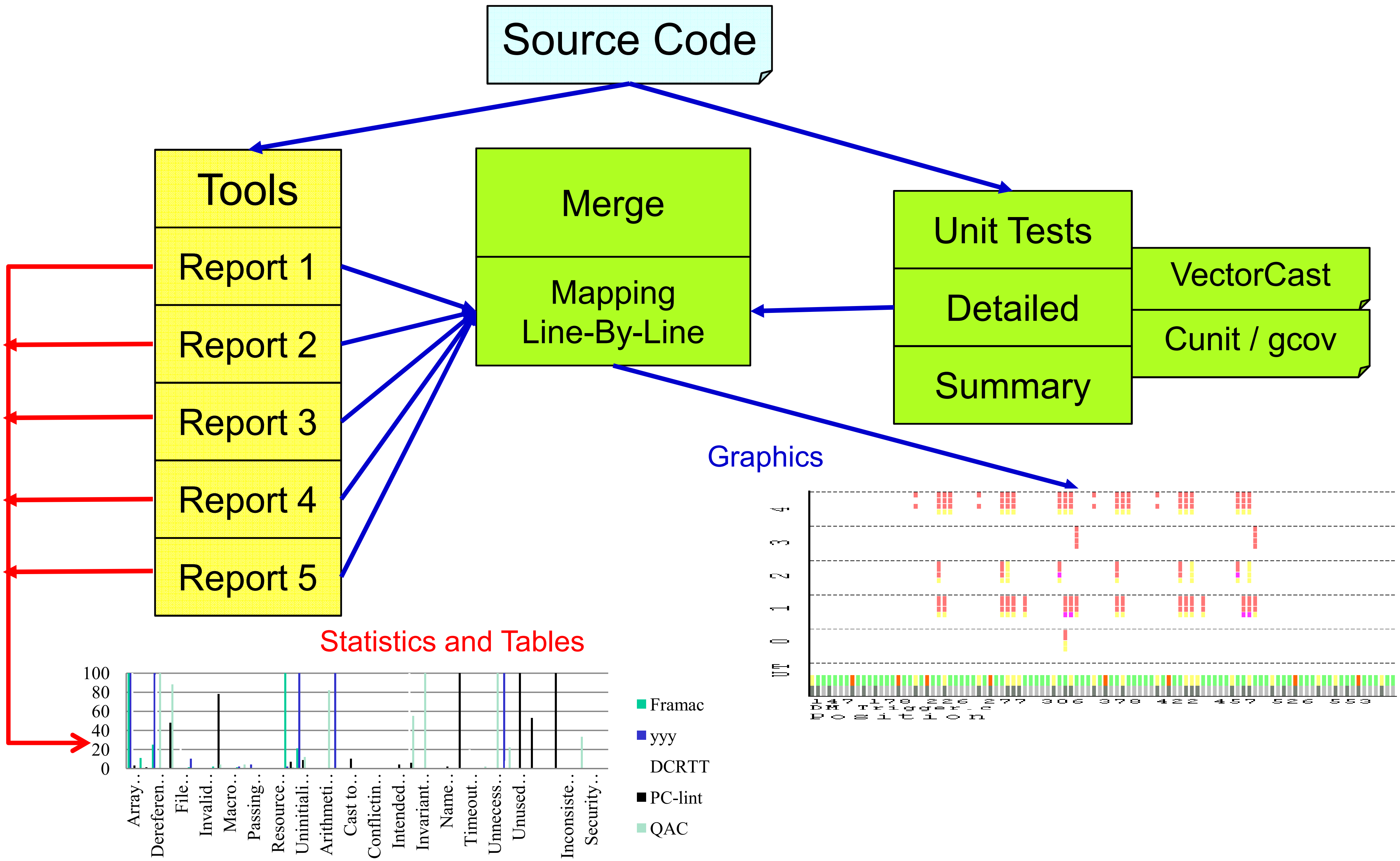
```
while (1) { }
```

Doubts on state criterion

```
unsigned int exp,s;
unsigned int c;
exp=…;
s  =…;
if (s==0)
  c=exp;
else
  c=-exp;
return(int)c;//defect?
```

```
unsigned int exp,s;
unsigned int c;
a=…;
s=…;
if (s==0)
  c=exp;
else
  c=-exp;
if (c<3)// defect!
```

Logic Flow: Tools and Unit Tests



Analysis Results

Reported Defects (not TP)

BSSE System and Software Engineering

Function Set	Tool Reports Application 1				
	Frama-C	yyy	DCRTT	PC-lint	QA/C
all, raw	10124				
all	1913	948	1480	5245	4976
selected	107	165	187	43	232
ignored, all	39	0	5	3100	2870
ignored, selected	0	0	0	0	0
critical, all	1874	616	942	146	393
critical, selected	107	137	102	6	93

Function Set	Tool Reports Application 2						
	Frama-C	yyy	DCRTT			PC-lint	www
			1	2	3		
all, raw							
all		2132	365	366	370	11999	798
selected		508	73	78	80	107	39
ignored, all		182	0	0	0	8614	510
ignored, selected		41	0	0	0	0	0
critical, all		1155	193			737	141
critical, selected		376	14			55	10

Evolution of Evaluation Results

BSSE System and Software Engineering

- **Previous Study ESVW / Tool Set**

  results strongly depend on

  - application
    - complexity
    - defect profile
    - number of defects
  - Tool
    - defect types supported

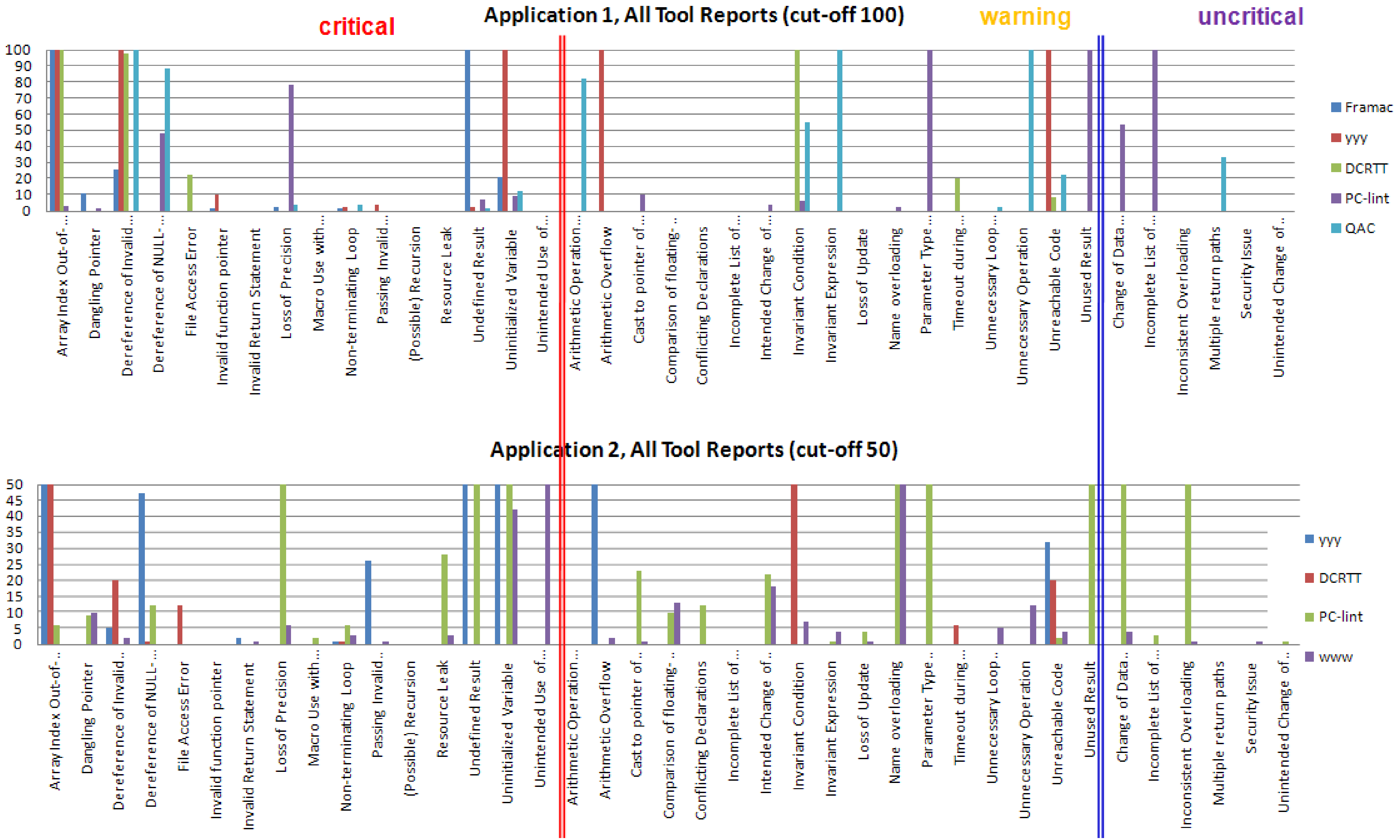
Appl.	Cnt	%	Unique Contributions and Tool Combinations
1	4	0.79	FramaC yyy
	2	0.40	FramaC yyy QAC
	29	5.74	FramaC
	4	0.79	FramaC yyy DCRTT QAC
	7	1.39	FramaC yyy DCRTT
	1	0.20	FramaC DCRTT QAC
	4	0.79	FramaC DCRTT
	23	4.55	yyy DCRTT
	23	4.55	yyy DCRTT QAC
	79	15.64	yyy
	18	3.56	yyy QAC
	92	18.22	DCRTT
	12	2.38	DCRTT QAC
	43	8.51	PC-lint
	164	32.48	QAC
	505	**100.00**	**Total**
2	9	1.61	yyy DCRTT
	362	64.76	yyy
	56	10.02	DCRTT
	1	0.18	DCRTT www
	98	17.53	PC-lint
	1	0.18	PC-lint www
	32	5.72	www
	559	**100.00**	**Total**

- **Current Study FSVW / Tool Set**
  - *dependencies confirmed: quite different results*
    - many trivial reports
    - many unstructured reports
    - many duplicated reports
    - different reports on same issue
  - in addition
    - impact by language (C $\Rightarrow$ C++): may drive flood of reports

Appl.	Number of Tools	Coincidences
1	0	0
	1	407
	2	61
	3	33
	4	4
2	0	0
	1	548
	2	11
	3	0
	4	0

Comparison of Profiles Appl. 1 vs. Appl. 2

BSSE System and Software Engineering



Defect Profiles vs. Criticality

BSSE System and Software Engineering

deeper analysis required

Application 1	Set	Criticality	Tool				
			Frama-C	yyy	DCRTT	PC-lint	QA/C
	full set	critical	*1874*	616	942	146	393
		warning	0	332	533	631	1680
		uncritical	0	0	0	1368	33
		ignored	39	0	5	3100	2870
		total	1913	948	1480	*5245*	*4976*
	Subset	critical	107	137	102	6	93
		warning	0	28	85	37	127
		uncritical	0	0	0	0	12
		ignored	0	0	0	0	0
		total	107	165	187	43	232

Application 2	Set	Criticality	Tool				
			Frama-C	yyy	DCRTT	PC-lint	www
	full set	critical		*1155*	193	*737*	141
		warning		795	172	2181	141
		uncritical		0	0	467	6
		ignored		182	0	*8614*	510
		total		2132	365	*11999*	798
	Subset	critical		*376*	14	55	10
		warning		133	59	49	27
		uncritical		0	0	4	3
		ignored		41	0	0	0
		total		550	73	108	40

Transition Rates Tool-TP ⇒ State-FP

BSSE System and Software Engineering

Figures do heavily depend on evaluator's interpretation of state criterion !

Frama-C, PC-lint only

most interesting transition from developer's point of view

Application 1				
Criterion Transition	**With**	**With %**	**Without**	**Without %**
Tool TP / State TP	81	34.32	83	39.34
Tool FP / State TP (impossible)	0	0.00	0	0.00
Tool TP / State FP	7	2.97	7	3.32
Tool FP / State FP	148	62.71	121	57.35
Total	236	100.00	211	100.00

yyy, DCRTT, PC-lint, www

Application 2				
Criterion Transition	**With**	**With %**	**Without**	**Without %**
Tool TP / State TP	112	39.30	123	43.16
Tool FP / State TP (impossible)	0	0.00	0	0.00
Tool TP / State FP	137	48.07	102	35.79
Tool FP / State FP	36	12.63	60	21.05
Total	285	100.00	285	100.00

Unit Testing vs. Analyses

Unit Tests vs. Analyses

BSSE System and Software Engineering

- **Application 1**
  - already subject of unit testing
  - defects found were fixed
  - analyses applied to final version
- **Application 2**
  - subject of verified-by-use, DCRTT already applied to platform-independent part
  - 4 defects found during unit testing (NULL for fd, file not opened, w/o ctxt), not fixed
  - analyses applied to same version

Overview on Number of Unit Tests and Functions

Test Mode	Application	Functions under Test	Performed Unit Tests	Test / Function	Average Coverage	
					stmt	cond
manuallly	1	368	954	2,59	94,14	89,63
	2	60	164	2,73	85,03	60,41
DCRTT	1	610	1042420	1708,89	87,35	95,78
	2	466	216348	464,27	76,89	76,08

Application	Files		Functions	
	total	affected by test	total	affected by test
1	39	25	610	368
2	40	24	557	60

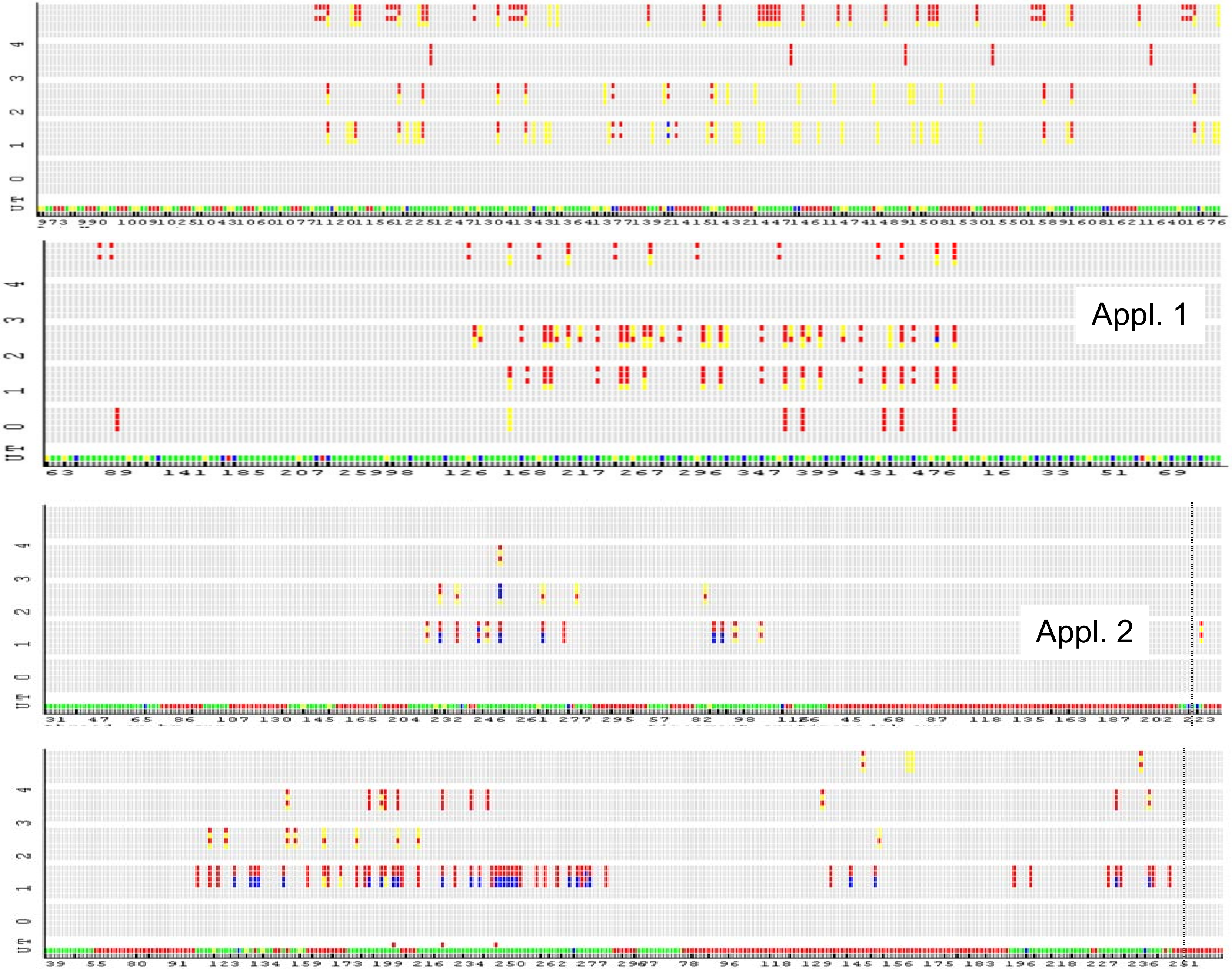
True Positives vs. UT-Coverage

BSSE System and Software Engineering

Appl.	Description	TP in non-covered lines	TP in covered lines	Total TP	% TP in non-covered / total TP	% TP in covered / total TP	TP per non-covered line	TP per covered line
1	tool / with ctxt	25	302	327	7.65	92.35	0.1656	0.3471
	tool / without ctxt	23	312	335	6.87	93.13	0.1523	0.3586
	state / with ctxt	7	104	111	6.31	93.69	0.0464	0.1195
	state / without ctxt	7	105	112	6.25	93.75	0.0464	0.1207
2	tool / with ctxt	36	233	269	13.38	86.62	0.1593	0.2852
	tool / without ctxt	33	215	248	13.31	86.69	0.1460	0.2632
	state / with ctxt	14	131	145	9.66	90.34	0.0619	0.1603
	state / without ctxt	15	145	160	9.38	90.63	0.0664	0.1775

- **TP distribution**
  - TP in covered line ~2x as TP in non-covered line
  - matter of complexity?
- **To Do**
  - distribution per critical TP etc.

Merge of Analyses and UT-Results




```
tool  criterion / w/o  ctxt
state criterion / w/o  ctxt
tool  criterion / with ctxt
state criterion / with ctxt
```

False Positive
True Positive
TP and FP reported for the same line

```
QA/C
PC-lint
DCRTT
yyy
FramaC
```

```
www
PC-lint
DCRTT
yyy
----
```

True Covered
Both Covered
Exception
Fault
Not Covered
False Covered
Condition Line
Stmt Line

Complementarity

- **Unit testing and analyses are complementary to a major degree**

  Surprised?

- Unit testing
  - demonstration of compliance with requirements
  - focus on functionality

- Analyses (static, dynamic)
  - aiming to demonstrate presence or absence of faults
  - considers large set of conditions
  - increased capability to detect defects, but still not perfect

Lessons Learned and Conclusions

About Reporting

BSSE System and Software Engineering

- **Number of reports**
  - some tools seem to maximize the number of reports
    - “*the more, the better*”
    - however: too many reports (related to False Positives) limit visibility on True Positives
  - “*the minimum possible is the better choice*”
  - *“the more comprehensive, the better”*
- **Relevance**
  - False Positives are more likely for certain defect types than for others
  - Classification into “for sure” and “may be” True Positives not really helpful

    if /because True Positives need to be fixed

    “definite, must be“

    “apparent, did not expect”

    “suspicious, possible, may be, possibly, may not, could be”
  - for “really” critical applications impossible to neglect “may be” reports
- **Degree of detail**
  - provision of details may be required, but summary view is urgently needed, too

Issues on False Positives

BSSE System and Software Engineering

- **Principal origins of False Positives**
  - developer, e.g.
    - explicit/implicit casts: undefined result, overflow
    - unclear resource usage (memory, files, semaphores, …)
  - tool
    - e.g. unjustified report on name overloading
  - platform
    - language / compiler: missing constraints on range
    - hardware architecture: limited representation of numbers, “no group operations”

- **Principal measures to minimize False Positives**
  - developer
    - avoid ambiguous constructs provoking reports
  - tool
    - carefully choose tool(s)
    - filter reports, automate processing
  - platform
    - provide range constraints, if supported by a tool
    - insert checks if adequate and wherever/whenever jusitified

Epilogue

Platform generates hay
language, compiler, processor
limited representation
false positives

inherent

Developers generate needles
true positives

Developers generate hay
false positives

provoked or inherent

Tool finds needles
true positives

Tool generates hay
false positives

induced

Process removes hay
false positives
turn to
true negatives

Characterization of Verification Approaches

BSSE System and Software Engineering

- **Unit Tests**
  - demonstration of compliance with requirements, focus on functional aspects
  - limited subset of input domain sufficient, coverage-driven
  - verification goal is to pass tests
  - currentyl requires major effort at limited predictability on future defect rates
- **Verified-by-Use**
  - demonstration that software does properly work for a given scenario
  - implies that software was sufficiently exposed to set of relevant conditions
  - possibly enhanced compared to UT due to extended set of conditions
  - lean approach at limited predictability on future defect rates , focus on functional aspects
- **Static and dynamic analysis**
  - aiming to demonstrate presence or absence of faults
  - considers large set of conditions
  - increased capability to detect defects, but still not perfect
  - may imply overhead if improperly applied
  - capability to look beyond scenariosas used for UT and verified-by-use

Considerations on Verification Approaches

BSSE System and Software Engineering

Unit Testing and Verified-By-Use

If

you just want to know that you will get correct results under current conditions,

although these are only partially or fully unknown,

then

unit testing or verified-by-use should be sufficient.

Static and Dynamic Analysis

If

you want to know that the implementation is correct,

i.e. that you can(should) expect always correct results under arbitrary conditions,*

then

do apply a rigorous verification approach like static and dynamic analyses do support,

***and** support the actions required to achieve highest efficiency*

* tools are never perfect

Optimization of Verification

BSSE System and Software Engineering

- **Trade-off on verification approach**
  - trade-off required on evaluation criterion before use of a tool
  - What is required?
    - Is verification-by-use sufficient?
      $\Rightarrow$ no tool required at all !
    - More than verification-by-use required?
      $\Rightarrow$ one tool or more required
- **Consequences**
  - not sufficient just to apply a tool (*do not claim about high effort if not preparing for*)
  - minimize verification effort in advance by
    - choosing tool(s) with maximum coverage of defect type profile
    - considering reporting features / characteristics of tool(s)
    - (pre-)processing of tool output
  - sufficiently prepare for tool usage
    - consider impact on development and programming style
    - minimize False Positives in advance
  - continuous use of a tool
    - obtain early feedback
    - continuously obtain feedback

Quality of Reports

BSSE System and Software Engineering

Previous Study ESVW

- *comparable contributions from all tools, moderate number of reports*
- *a few trivial reports, only*

☺

- ***bad point:*** *none*
- ***good point:*** *sensitivity, precision, uniqueness, complementarity could be derived*

Current Study FSVW

- *heterogeneous contributions from tools, explosion of number of reports, in part*
- *many trivial reports*
- *many reports regarding O-O features, but major part negligible*

- ***bad point:***
  - *sensitivity sensitivity, precision, uniqueness, complementarity cannot be derived*
  - *as still significant effort required to make data comparable*
- ***good point:***
  - *situation suggested principal classification of reports apart from existing one for criticality regarding standard defect types*
  - *mapping: critical, less critical, trivial / negligible*
  - *(heuristic) rules: high probability for true positive, high probability for false positive*

Possible False Positives

BSSE System and Software Engineering

Category	Type	Example
Platform *inherent*	overflow	`int a,b,c;` **`c = a + b;`** `long long lli; double dbl;` **`lli= dbl;`** limited representation of numbers
Developer *provoked* *inherent*	precision	`int32_t a; uint32_t b;` **`b = a; a = b;`** loss of sign or MSB many explicit and implicit casts could be avoided some may not
provoked *inherent*	resource leak	`FILE *fd;` **`fd=fopen(„myFile“,“w“);`** release of resource not visible if open/close could be put in the same function if not (possible)
inherent	endless loop	**`while (1)`** non-terminating loop intended
provoked	out-of-bounds	`char a[UPLIM];for(i=0;i<UPLIM;i++)strcmp(a+ii,“myStr“);` access exceeds valid memory by 4 bytes unintended access of invalid memory, (possibly), no consequences !
provoked	Invalid use of minus ops	`unsigned int a,b; int c;` **`if (b!=0) a=-a; c=a;`** loss of MSB, replaced by sign bit conversion of a positive number into a negative, no consequences in this case!

Tool
induced
	overloading	`struct TyMyStruct {int elem}; struct TyMyStruct myData;`
		`void myFunc(int elem) {` **`myData.elem=elem`**`; return;}`
		no name conflict !

The Team

DLR	Sabine Philipp-May	Christian R. Prause
BSSE System and Software Engineering	Rainer Gerlich	Ralf Gerlich
AIRBUS DEFENCE & SPACE	Hans-Jürgen Herpel	Mladen Kerep
etamax (member of the ckc group)	Anton Fischer	Mario Pinto
Fraunhofer FOKUS	Jens Gerlach	Jochen Burghardt
Julius-Maximilians-Universität Würzburg	Sergio Montenegro	Frank Flederer

The project was funded by DLR Space Administration on behalf of the
German Ministry of Economics and Energy, BMWi under Contract No. 50 PS 1606

Thank you for your attention!

Questions?