Cost and Benefits of Integrating Software Assurance Tools
A Quantitative Look

William Nichols

Software Engineering Institute

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-1362

Bottom Line Up Front

Static Code Analysis (SCA) and Static Binary Analysis (SBA) **improve quality** and **reduce pre-release development effort**.

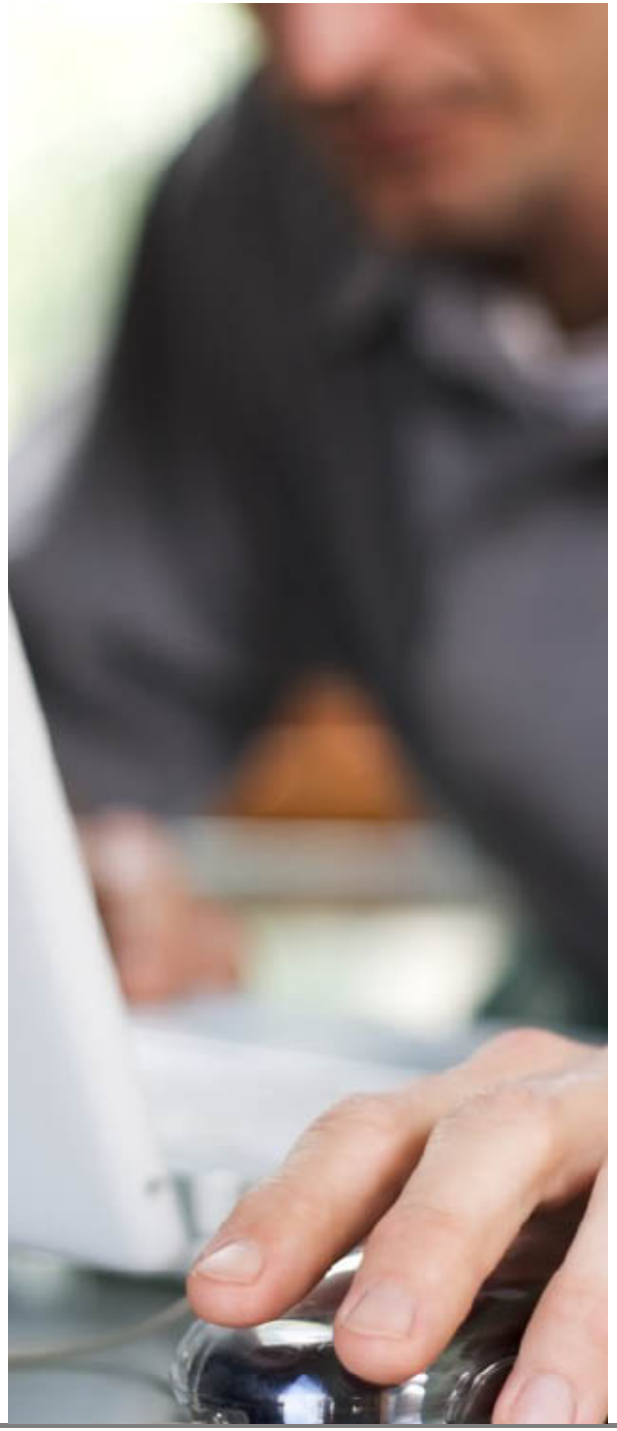
- Using **SCA** will reduce released defects, and will modestly shorten development effort and time by reducing test time.
- Static Binary Analysis (**SBA**) reduces released defects, but has less effect on reducing total development time.
- Integrate/Automate S(C/B)A into your development and builds
  - Use SCA **before** unit test.
  - Use SBA (or SCA/SBA) before or during integration test.

Operationally, no worse than effort neutral **during** development prior to release.

Cost/Benefits of Software Assurance Tools

The Question

Are Static Analysis tools cost effective before release?



Problem

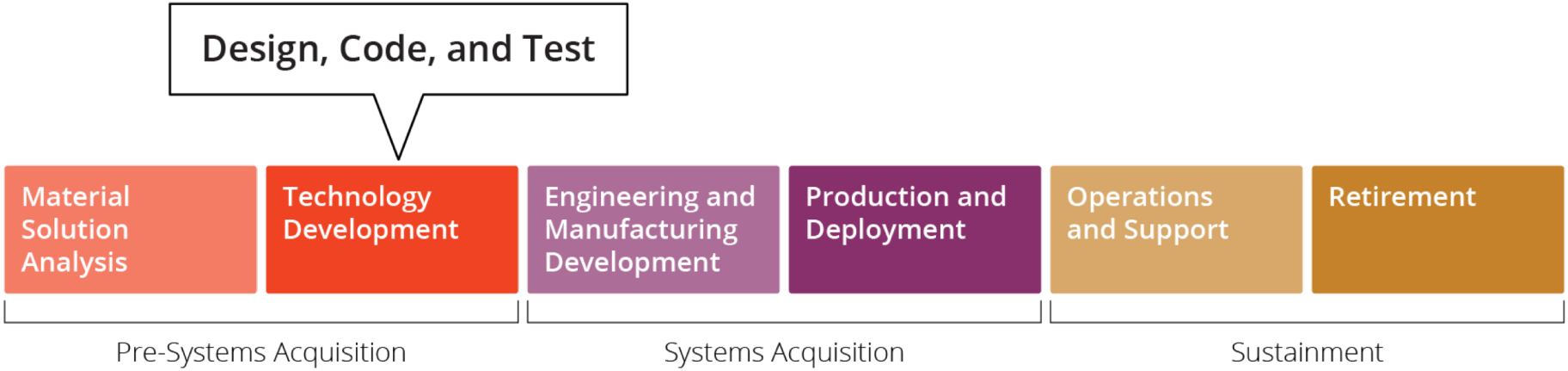
Enable transition of security techniques and tools into the developmental lifecycle

- DoD developers are policy **mandated** simultaneously to
  - contain costs (e.g. Better Buying Power) while
  - improve security, Many checks have been mandated by DoD policy
  - transition to the Risk Management Framework [NIST 2010] approach [Marzigliano 2014]) , [DoD 8510.01]

Who pays for this benefit? How long must we wait for ROI?

- Will Static Analysis pay off long term, but cost in the short run?
- There is **no hard data** to drive when, where, and how to implement Secure Development Techniques (SDT) into the SDLC
- Without hard data, developers often avoid the cost/benefit trade-off entierly

Examine the Development only



Other evidence that improved defect removal **reduces** total cost of ownership **TCO**, while making the software more reliable and secure.

We are interested in net cost during the development. Does it cost more to use static analysis tools? If so, how much?

Questions

- What do the tools cost to use during the development cycle?
- What are the quantifiable results in quality, effort, and security?
- Where in the development cycle should different types of tools be applied?

This study does not ask about

- Tool acquisition cost
- Training costs

Solution: SEI Research Project CESAW

Composing Effective Software Security Assurance Workflows

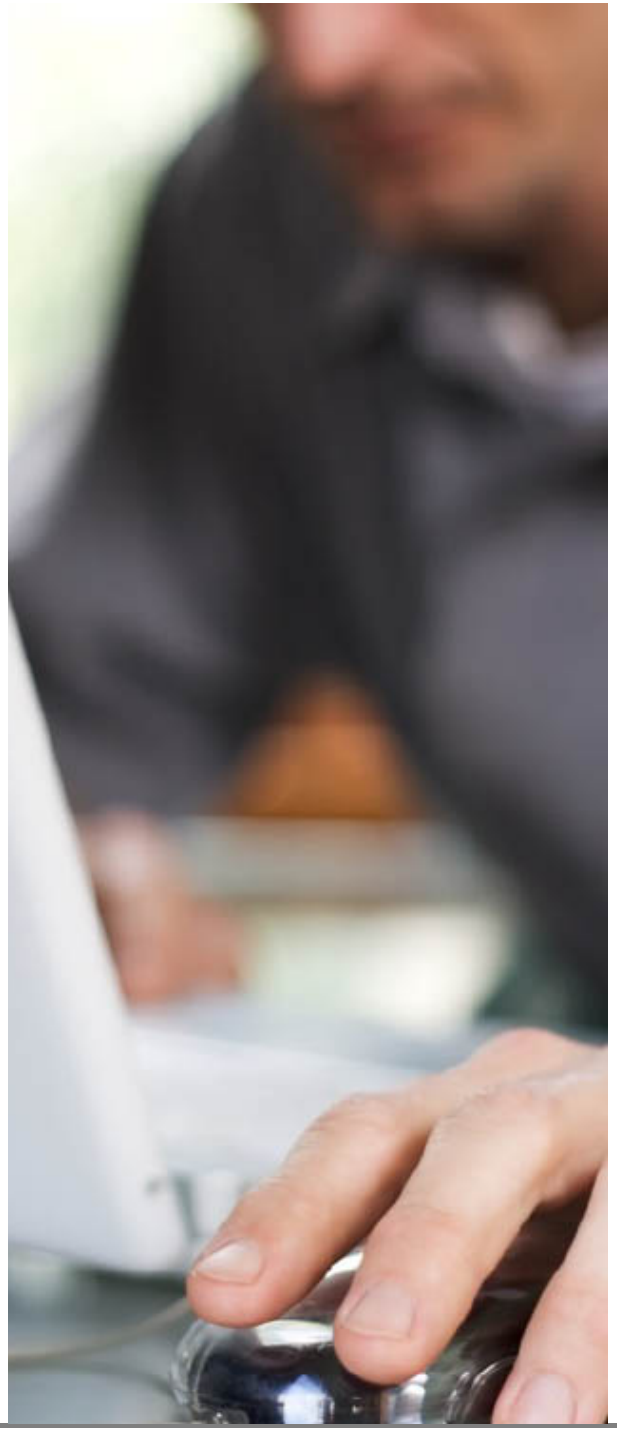
CMU/SEI-2018-TR-004

https://resources.sei.cmu.edu/asset_files/TechnicalReport/2018_005_001_528494.pdf

- Observe teams using software assurance tools in real environments.
- Instrument and measure the development process activities and products.
- Compare project outcomes with and without the tools.

Cost/Benefits of Software Assurance Tools

Approach



Effectiveness vs. Efficacy

Efficacy means how well the tools **can** work (e.g. remove CWW) under specific conditions. Other studies address this.

Effectiveness means how the tool affects **real world outcomes**, cost, schedule and quality.

Effectiveness	Efficacy
Observational	Experimental
Works under real world conditions	Works under ideal and controlled conditions
Focus on product/project outcomes	Tool and technique outcomes
Extended duration	Limited duration
Large(er) samples	Small studies
External validity	Internal validity

We do not compare tools or measure coverage against CWE

Collect Observational Data from Cases

Three Organizations,

org	Proj	Domain	Total Hours	Tools	When used
A	5	Embedded	71749	SCA	Personal Code Review
B	16	Business Intelligence	23170	SCA/SBA	Pre Code Inspection
C	14	Engineering CAD	26759	SCA/SBA	Integration
	35		**121708**		

Organization A used a single team over a period of multiple years on sequential projects.

Orgs B and C included separate teams with overlapping team members.

What Did They Do?

Tasks hierarchy

Task progress information

Task and Schedule

Project/Task	PT	Time	PV	Plan Date	Replan	Forecast	Date	Dep	%C	%S	EV
My Project, Release 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]			[illegible]	[illegible]	[illegible]
Non Project/Training	8:00	0:00	3.4%	2/4/06	2/26/06	[illegible]/11/06					
Project/My Project											
Planning	12:00	8:12	5.1%	1/7/06	1/5/06	1/5/06	1/5/06		100%	68%	5.1%
Requirements	30:00	43:14	12.7%	1/21/06	1/30/06	1/30/06	1/30/06		100%	144%	12.7%
Architectural Design	15:00	8:12	6.4%	2/11/06	3/3/06	3/8/06			33%	55%	2.1%
Prototype	31:00	25:58	13.1%	2/18/06	3/8/06	3/16/06		✓	65%	84%	8.5%
Release 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]				[illegible]	
Component A	35:00	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]				[illegible]	
Planning	6:28	2:00	2.7%	2/18/06	3/10/06	3/20/06				31%	
Design	2:26	0:00	1%	2/18/06	3/11/06	3/22/06					
Design Review	0:20	0:00	0.1%	2/25/06	3/11/06	3/22/06					
Code	13:46	6:00	5.8%	2/25/06	3/15/06	3/29/06				44%	
Code Review	1:22	0:00	0.6%	3/4/06	3/15/06	3/30/06					
Compile	2:17	0:00	1%	3/4/06	3/16/06	3/31/06					
Test	6:02	0:00	2.6%	3/4/06	3/19/06	4/5/06					
Postmortem	2:18	0:00	1%	3/4/06	3/20/06	4/6/06					
Component B	18:00	0:00	7.6%	3/11/06	3/29/06	4/20/06		●			
Component C	19:00	0:00	8.1%	3/25/06	4/7/06	5/3/06					
Component D	20:00	0:00	8.5%	4/1/06	4/16/06	5/18/06					
Testing	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		●			
Integration 1	8:00	0:00	3.4%	4/8/06	4/20/06	5/24/06					

Add Task... | Remove Task | Move Task Up | Move Task Down | Flat View

From	To	PT	CPT	CPV	Time	CT	EV
1/1/06	1/7/06	15:00	15:00	5.1%	13:25	13:25	5.1%
1/8/06	1/14/06	15:00	30:00	11.9%	10:01	23:26	8.9%
1/15/06	1/21/06	15:00	45:00	17.8%	11:00	34:26	11.9%
1/22/06	1/28/06	15:00	60:00	22.5%	16:00	50:26	15.3%
1/29/06	2/4/06	15:00	75:00	31.8%	13:24	63:50	19.9%
2/5/06	2/11/06	15:00	90:00	36%	13:22	77:12	22.5%
2/12/06	2/18/06	15:00	105:00	44.5%	8:24	85:36	28.4%
2/19/06	2/25/06	15:00	120:00	50.4%	0:00	85:36	28.4%
2/26/06	3/4/06	15:00	135:00	56.9%	0:00	85:36	28.4%
3/5/06	3/11/06	15:00	150:00	63.1%	0:00	85:36	28.4%
3/12/06	3/18/06	15:00	165:00	69.3%	0:00	85:36	28.4%

Add Row | Insert Row | Delete Row

Collaborate | Errors | Filtered Chart | Chart | Report | Close | Save

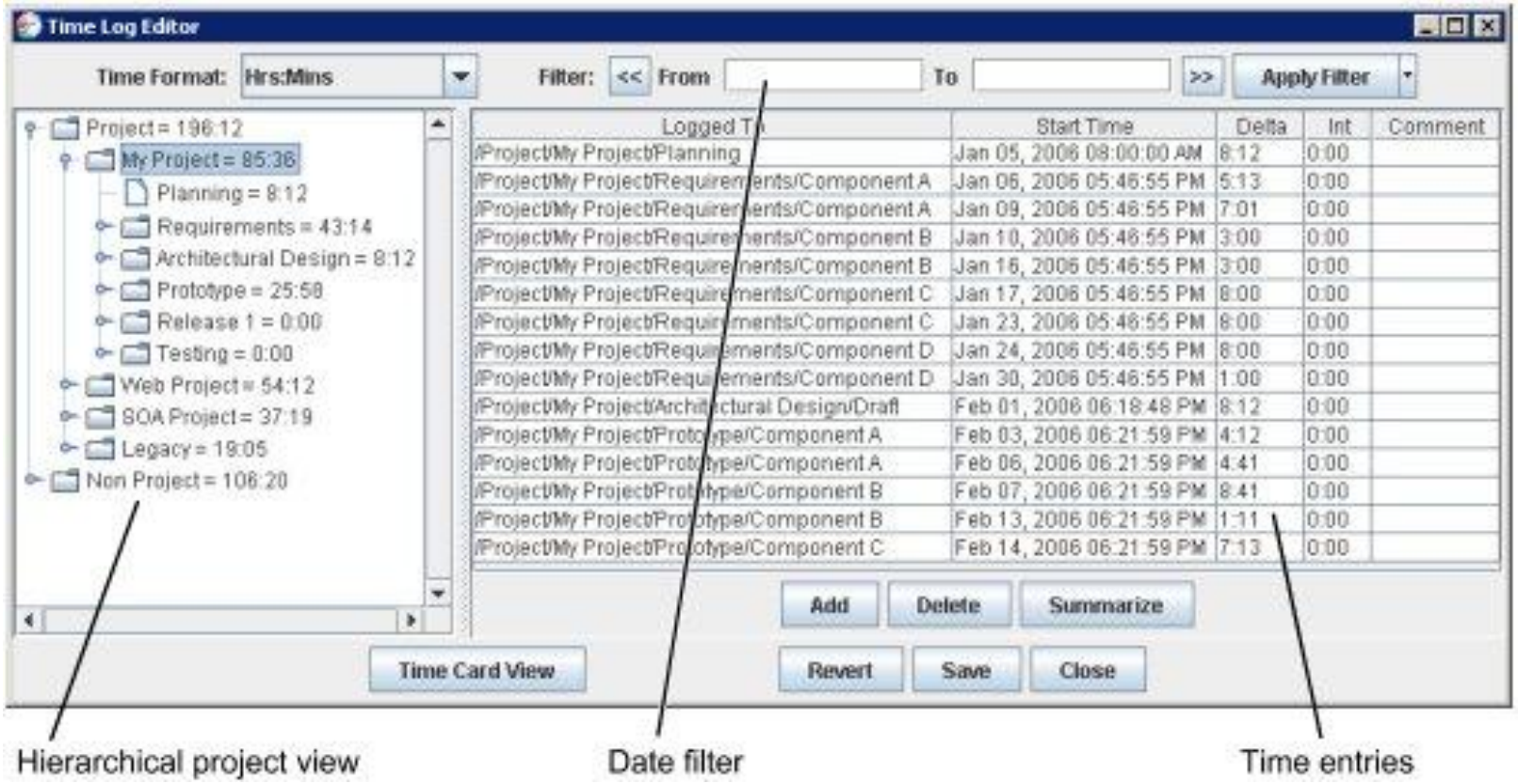
Task Schedule

Structured the WBS
Defined process flow,
List work tasks (hour to few days)

How long did it take?

Track the effort (in minutes) for every project task

Precise direct time (stop-watch) rather than calendar duration.



What did they Build?

Track Product Size (pages, Lines of Code)…

PSP0.1 Project Plan Summary

Project Settings

Program Size

Time In Phase

Defects Injected

Defects Removed

No-Frames version
(Best for printing)

/Non Project/PSP for Engineers/Program 2B

PSP0.1 Project Plan Summary

You have not entered the planned size for this project.

Program Size

	Plan	Actual	To Date
Base (B)		81	0
Deleted (D)		0	0
Modified (M)		30	0
Added (A)		84	249
Reused (R)		0	0
Added and Modified (A+M)	150	114	165
Total Size (T)		165	249
Total New Reusable		51	51

Adapted from "PSP Materials," copyright © 2006 Carnegie Mellon University. Used by permission.

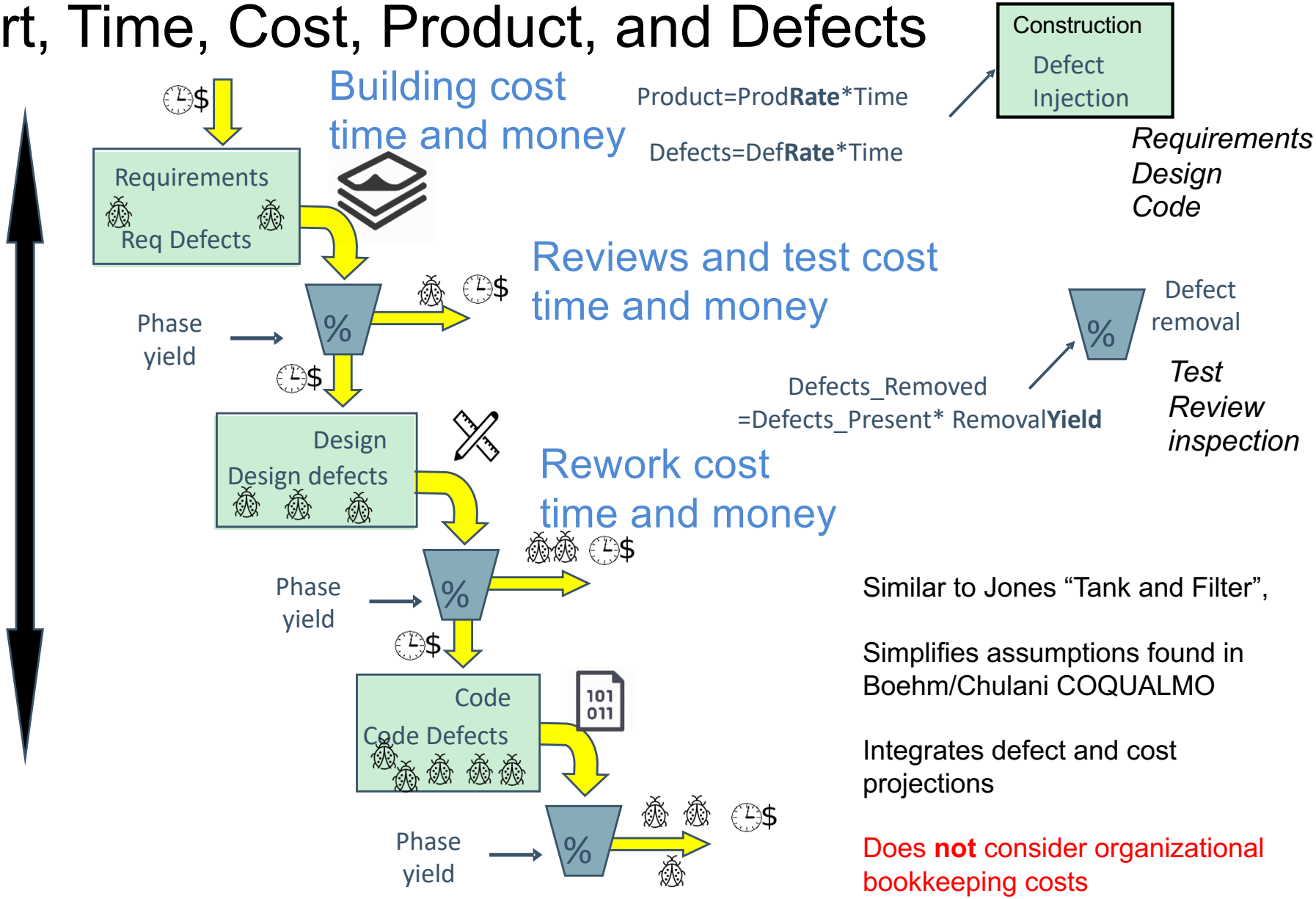
Unlock read-only data Export to: HTML Excel *Help...*

What was the rework?

Track each defect found

- The effort needed to find and fix the defect
- A description of the defect
- The activity in which defect was found and removed
- The activity in which the defect was injected
- Identify the SCA/SBA specific issues

Model the Workflow Central Tendencies Effort, Time, Cost, Product, and Defects



Similar to Jones "Tank and Filter",

Simplifies assumptions found in Boehm/Chulani COQUALMO

Integrates defect and cost projections

Does **not** consider organizational bookkeeping costs

Nichols, W. R. Plan for Success , Model the Cost of Quality. (2012). *Software Quality Professional*, *14*(2), 4–11.

Analysis

Check data quality

Summarize descriptive statistics

Identify data distributions

Parameterize defect fix costs in each phase

Parameterize defect removal yields in each phase, including tool use.

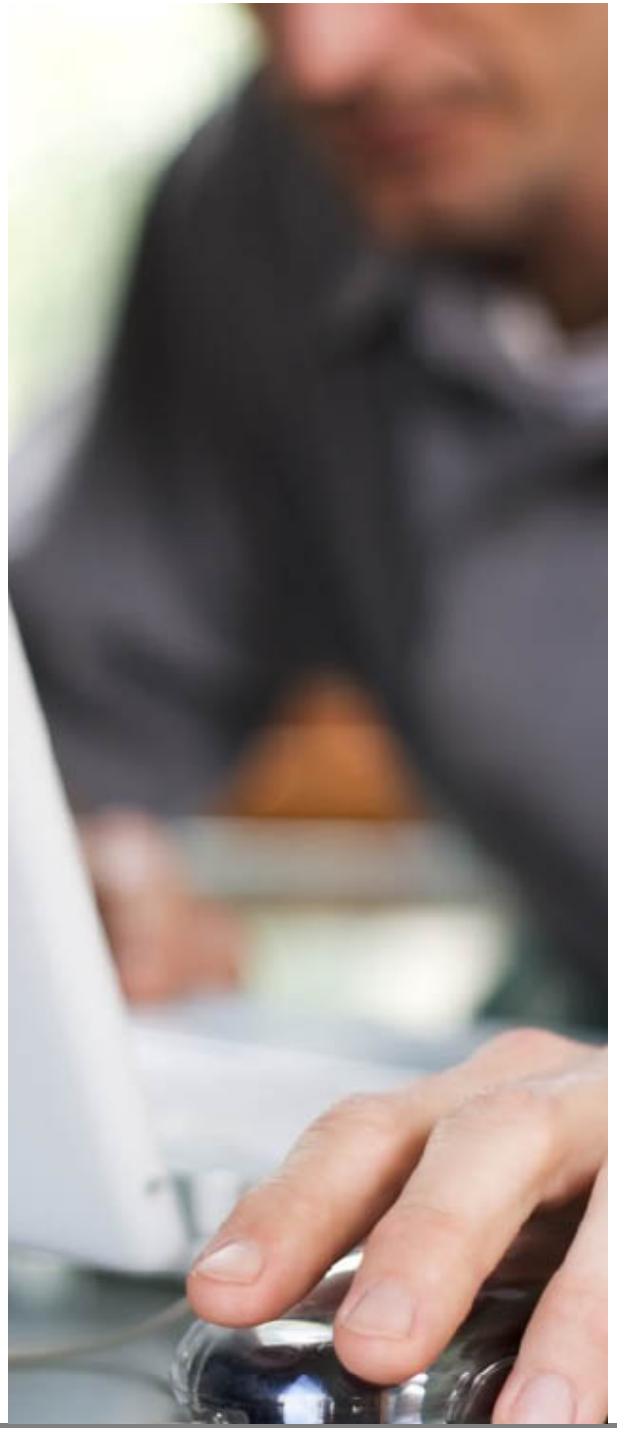
Model aggregate behavior

Simulate efforts and defects without the tools in the workflow

Composing Effective Software Assurance Workflows

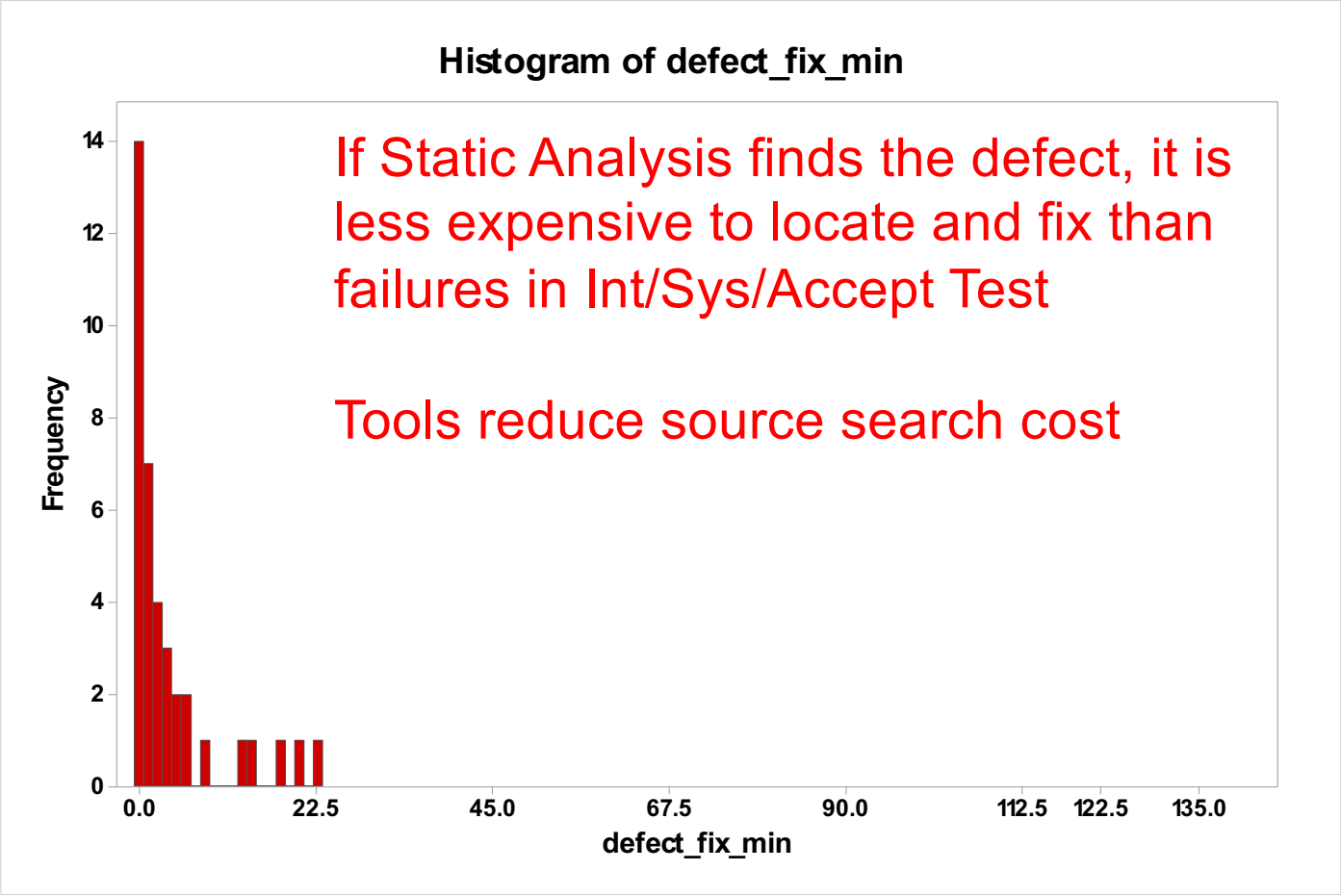
Compare all defect with SCA and SBA

Results SCA with SBA



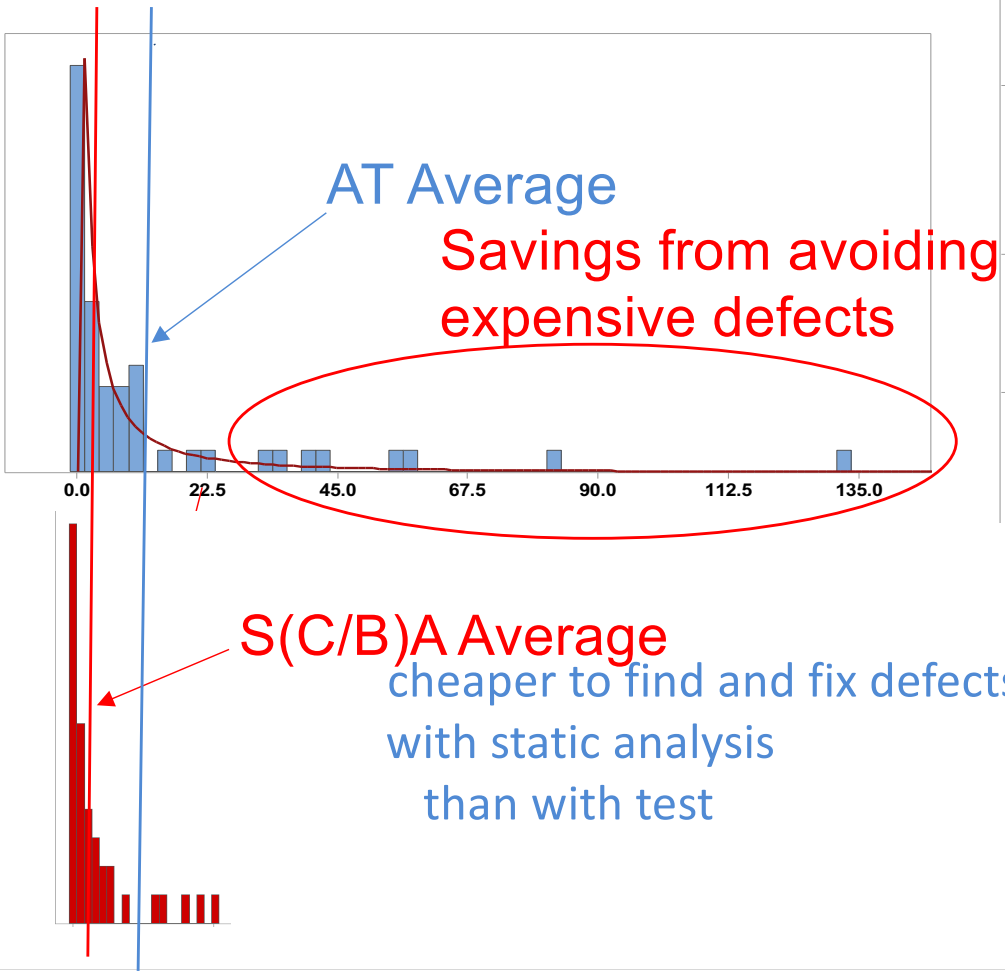
Org C, Static (C/B) Analysis Defect Fix Time

Automated into the integration, issues reported to developers.



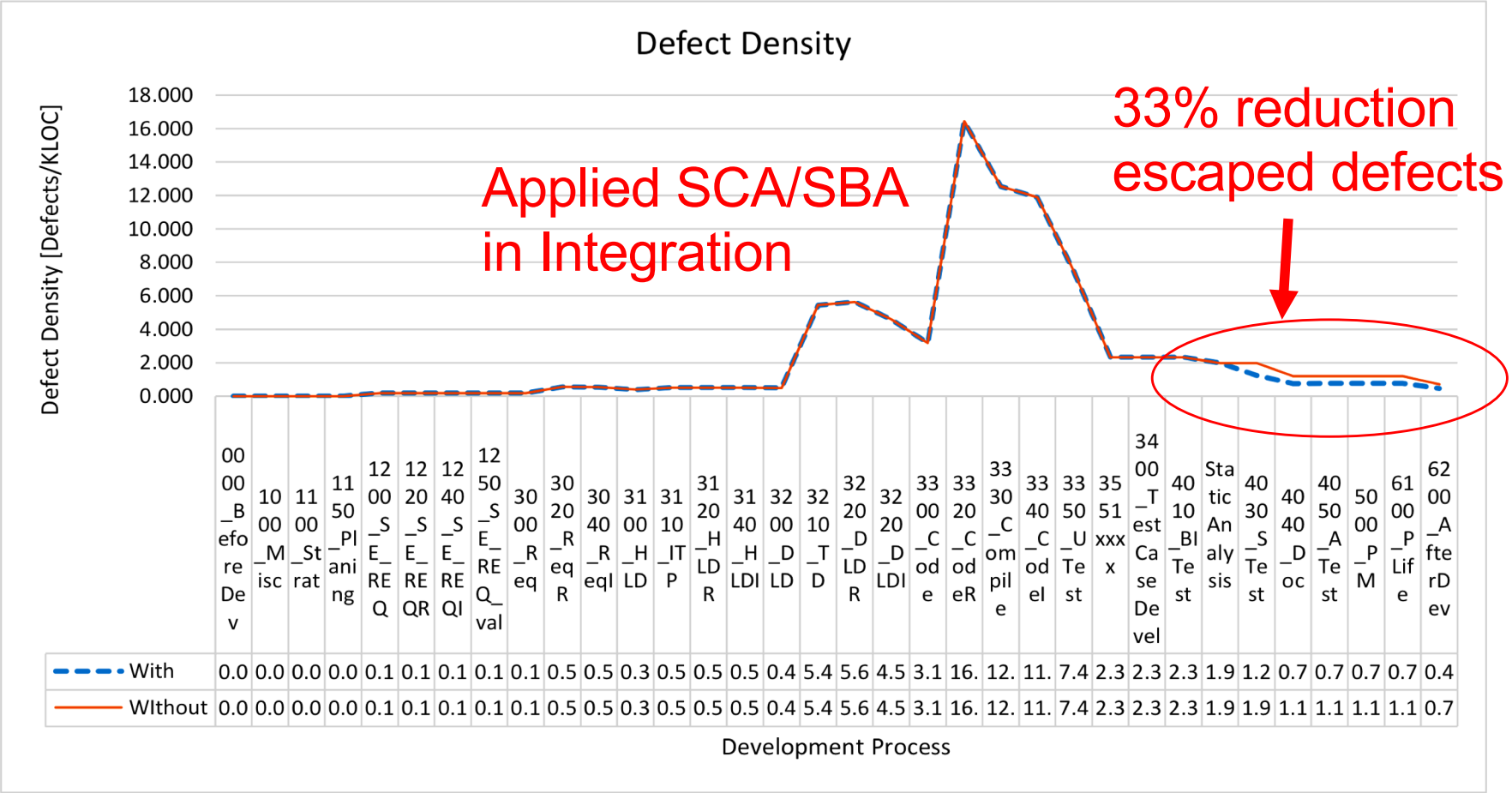
Org C, **Static Analysis fixes defects with less effort than Test**

defect_fix_time_minutes for Accpt Test, Code/ Design Injections



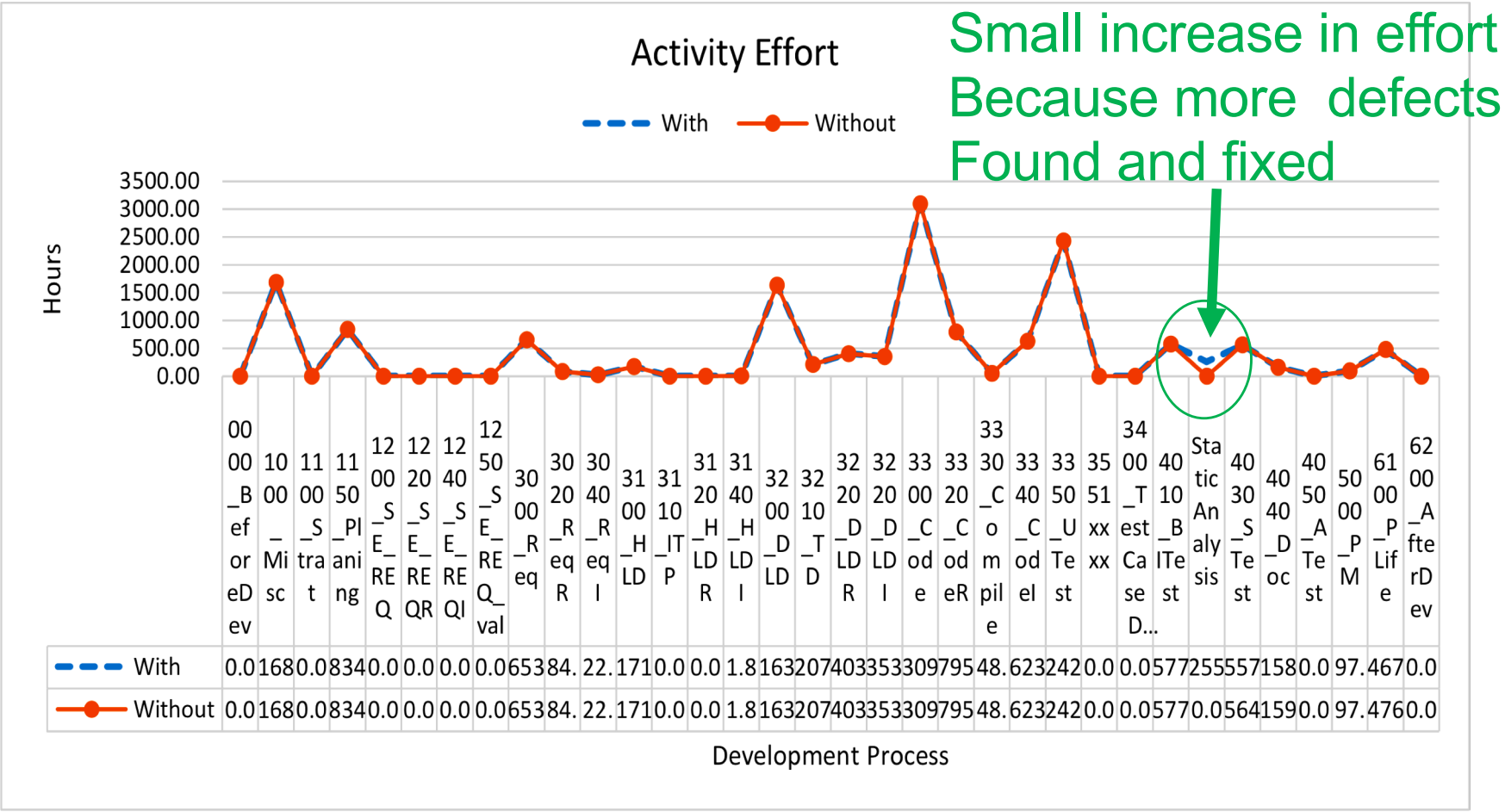
Anderson-Darling Normality Test	
A-Squared	7.67
P-Value	< 0.005
Mean	13.057
StDev	24.532
Variance	601.800
Skewness	3.1601
Kurtosis	11.7424
N	51
Minimum	0.200
1st Quartile	0.400
Median	2.800
3rd Quartile	10.600
Maximum	133.000
95% Confidence Interval for Mean	
6.157	19.956
95% Confidence Interval for Median	
1.101	7.800
Interval for StDev	
30.495	

Org C at Integration, Defect Density With and Without



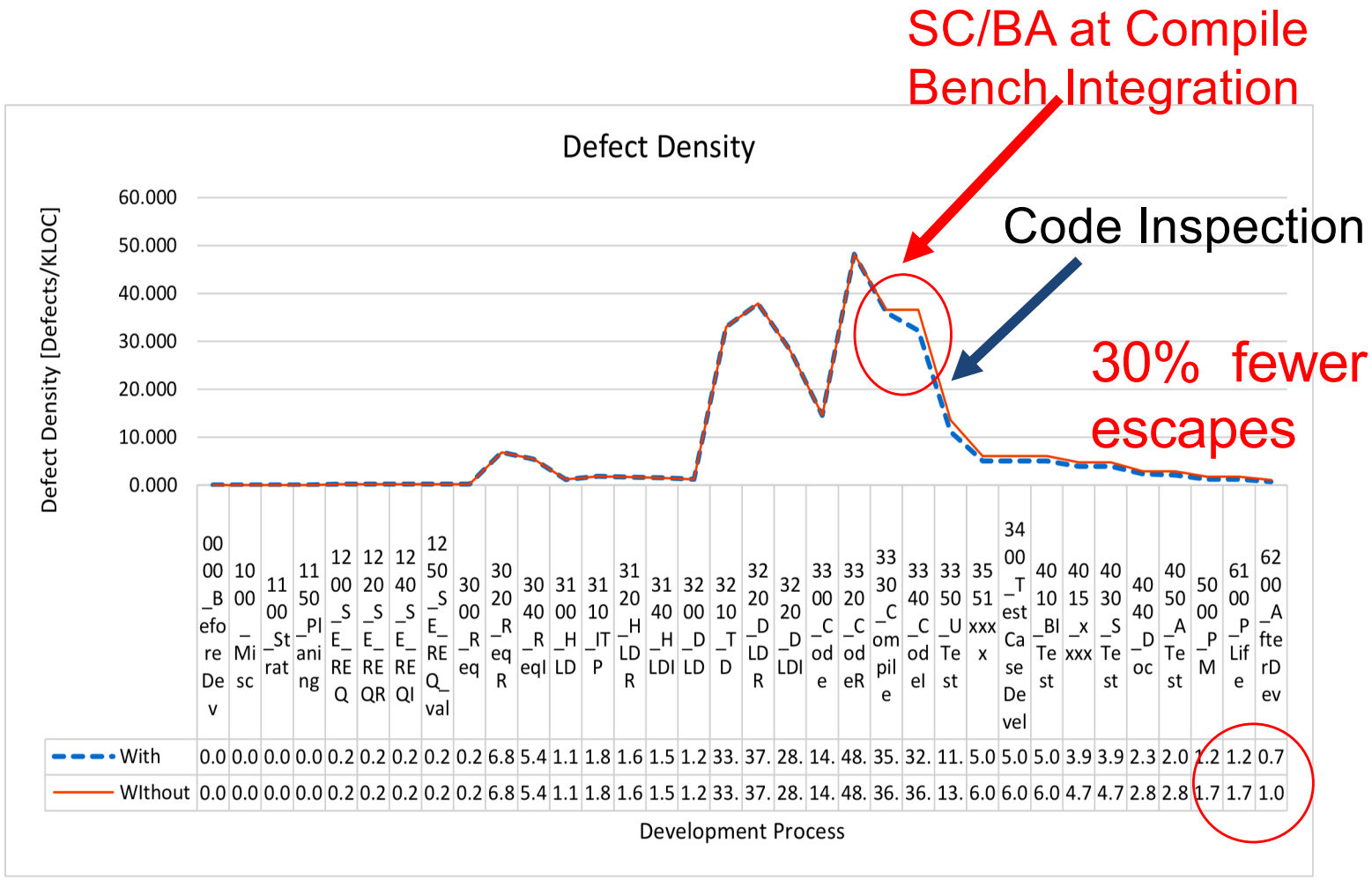
	0000_BeforeDev	1000_Misc	1100_Strat	1150_Planing	1200_SE_REQ	1220_SE_REQR	1240_SE_REQI	1250_SE_REQ_val	3000_Req	3020_ReqR	3040_ReqI	3100_HLD	3110_ITP	3120_HLDR	3140_HLDI	3200_DLD	3210_TD	3220_DLDR	3220_DLDI	3300_Code	3320_CodeR	3330_Compile	3340_CodeI	3350_UTest	3551xxxx	3400_TestCaseDevel	4010_BITest	Static Analysis	4030_STest	4040_Doc	4050_ATest	5000_PM	6100_PLife	6200_AfterDev
With	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.3	0.5	0.5	0.5	0.4	5.4	5.6	4.5	3.1	16.	12.	11.	7.4	2.3	2.3	2.3	1.9	1.2	0.7	0.7	0.7	0.7	0.4
WIthout	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.3	0.5	0.5	0.5	0.4	5.4	5.6	4.5	3.1	16.	12.	11.	7.4	2.3	2.3	2.3	1.9	1.9	1.1	1.1	1.1	1.1	0.7

Org C at Integration, Activity Effort With/Without



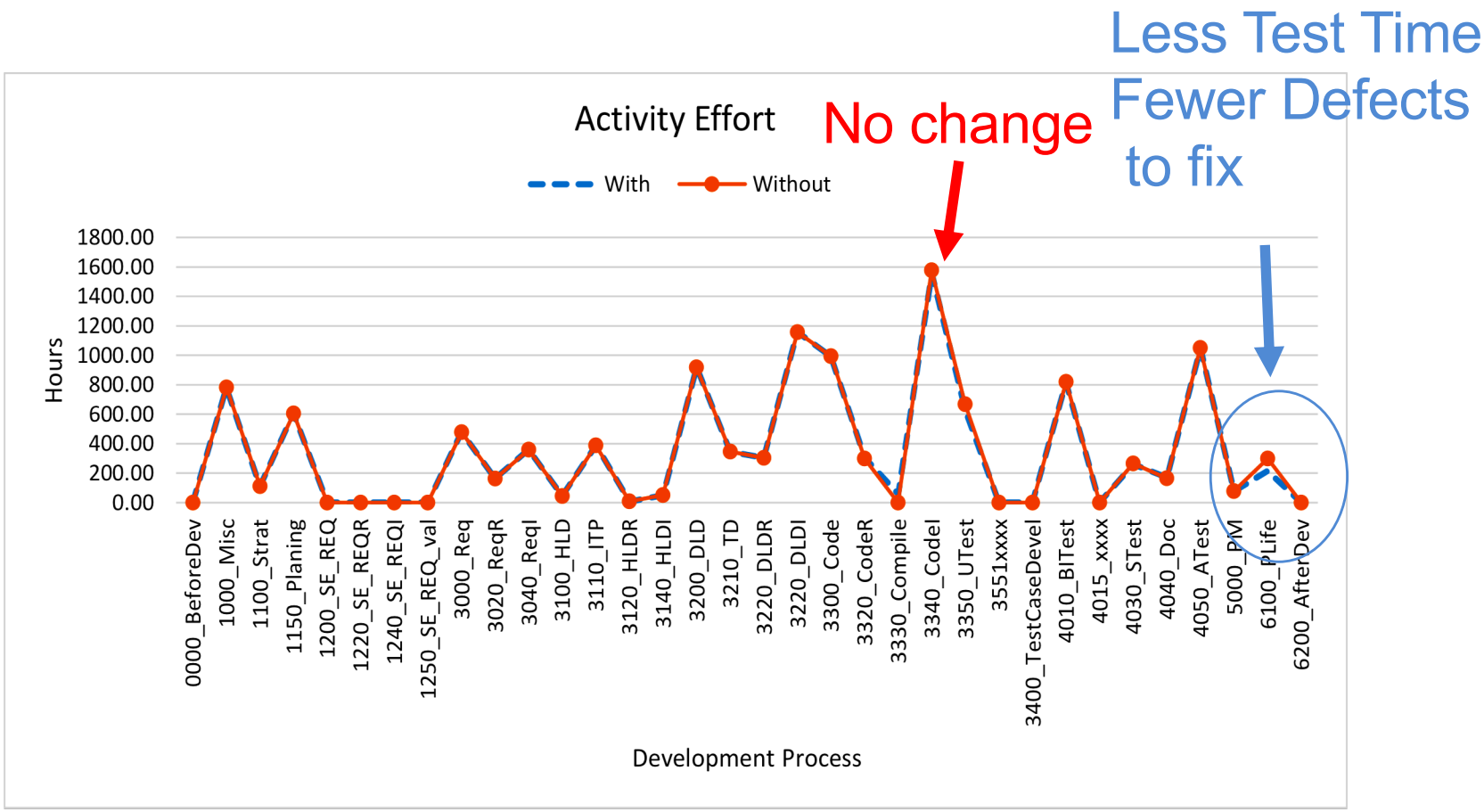
	0000_BeforeDev	1000_Misc	1100_Strat	1150_Planing	1200_SE_REQ	1220_SE_REQR	1240_SE_REQI	1250_SE_REQ_val	3000_Req	3020_ReqR	3040_ReqI	3100_HLD	3110_ITP	3120_HLDR	3140_HLDI	3200_DLD	3210_TD	3220_DLDR	3220_DLDI	3300_Code	3320_CodeR	3330_Compile	3340_Code	3350_UTest	3551xxxx	3400_TestCaseD...	4010_BITest	Static Analysis	4030_STest	4040_Doc	4050_ATest	5000_PM	6100_PLife	6200_AfterDev
With	0.0	1680	0.0	8340	0.0	0.0	0.0	0.0	6538	4.	22.	1710	0.0	0.0	1.8	1632	0740	3353	3097	95	48.	6232	42	0.0	0.0	5772	5555	7158	0.0	97.	4670	0.0
Without	0.0	1680	0.0	8340	0.0	0.0	0.0	0.0	6538	4.	22.	1710	0.0	0.0	1.8	1632	0740	3353	3097	95	48.	6232	42	0.0	0.0	5770	0.0	5641	590.0	97.	4760	0.0

Org B at Compile, Defect Density With and Without



	0000_Before Dev	1000_Misc	1100_Strat	1150_Planing	1200_SE_REQ	1220_SE_REQR	1240_SE_REQI	1250_SE_REQ_val	3000_Req	3020_ReqR	3040_ReqI	3100_HLD	3110_ITP	3120_HLDR	3140_HLDI	3200_DLD	3210_TD	3220_DLDR	3220_DLDI	3300_Code	3320_CodeR	3330_Compile	3340_Codel	3350_UTest	3551 xxxx	3400_TestCaseDevel	4010_BITest	4015_xxxx	4030_STest	4040_Doc	4050_ATest	5000_PM	6100_PLife	6200_AfterDev
With	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	6.8	5.4	1.1	1.8	1.6	1.5	1.2	33.	37.	28.	14.	48.	35.	32.	11.	5.0	5.0	5.0	3.9	3.9	2.3	2.0	1.2	1.2	0.7
WIthout	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	6.8	5.4	1.1	1.8	1.6	1.5	1.2	33.	37.	28.	14.	48.	36.	36.	13.	6.0	6.0	6.0	4.7	4.7	2.8	2.8	1.7	1.7	1.0

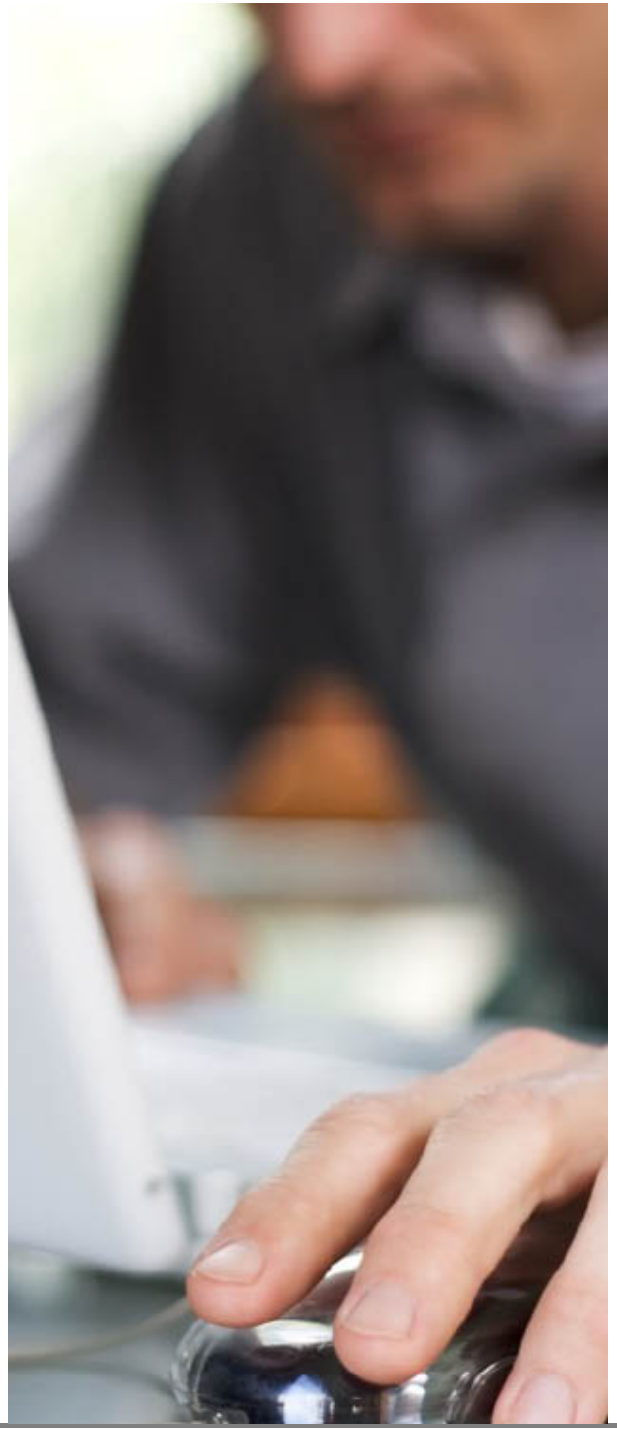
Org B at Integration, Effort With and Without



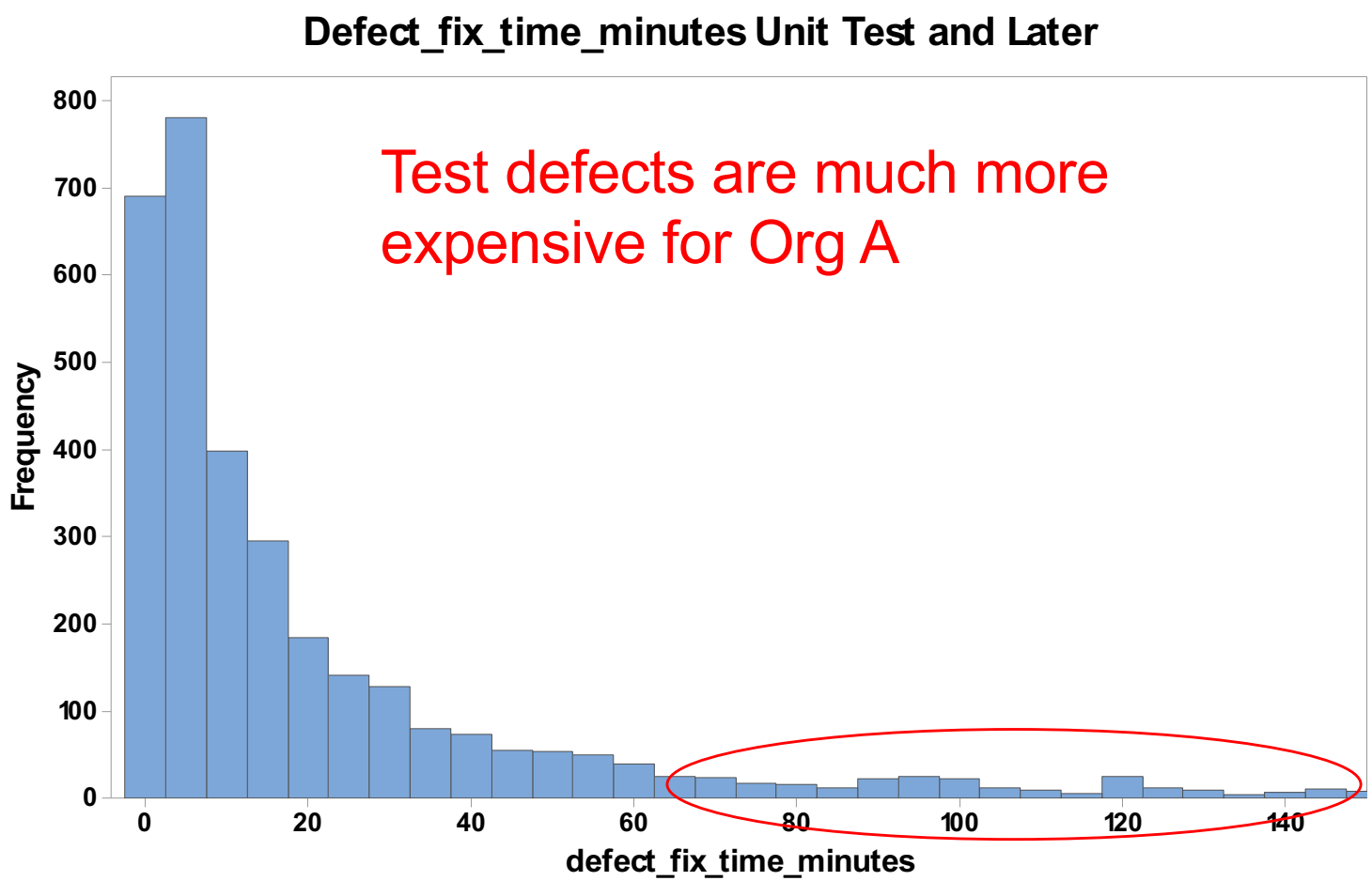
Cost/Benefits of Software Assurance Tools

Static Code Analysis Only

Results, SCA

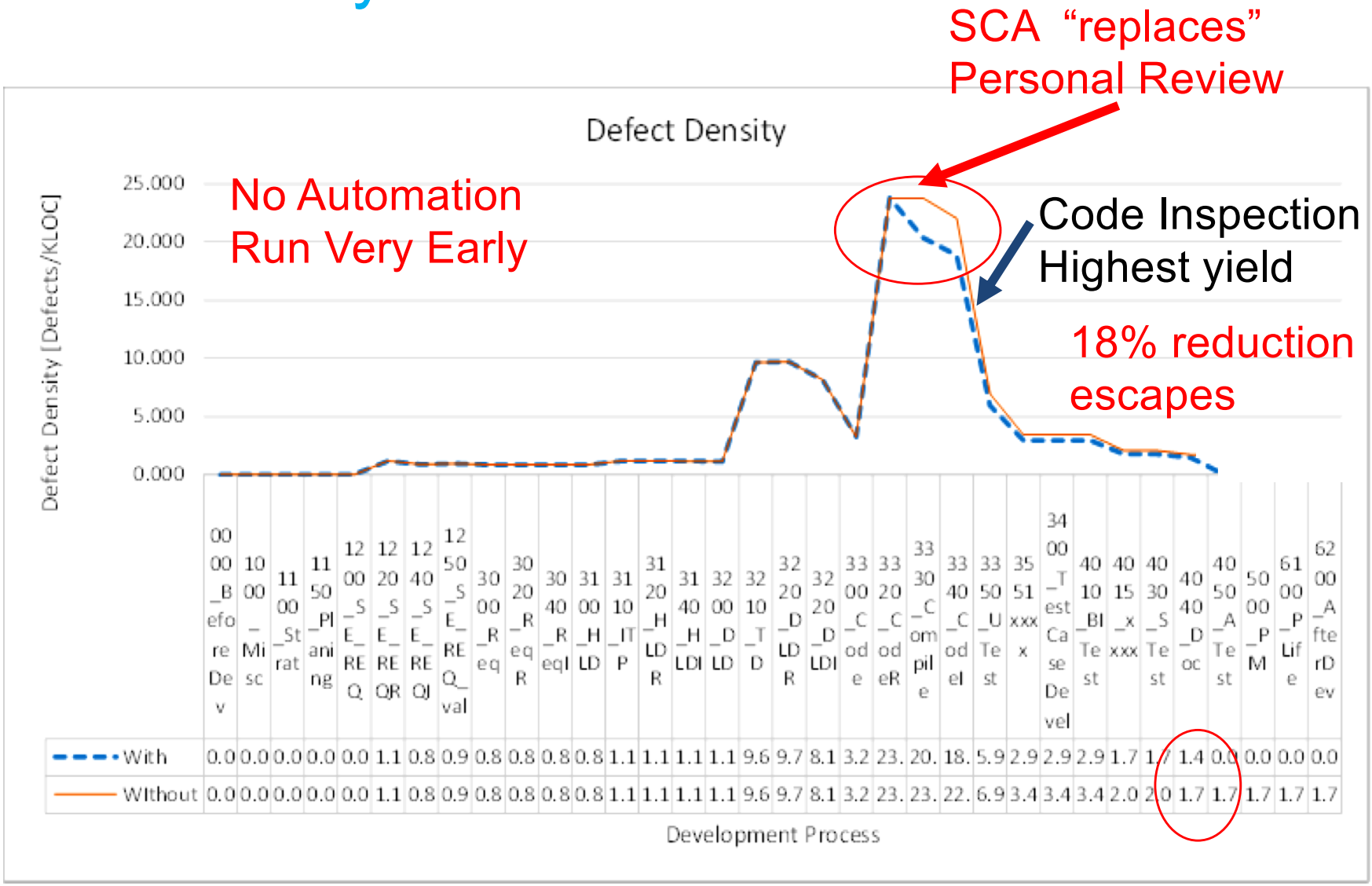


Org A, Acceptance Test Defect Fix Times



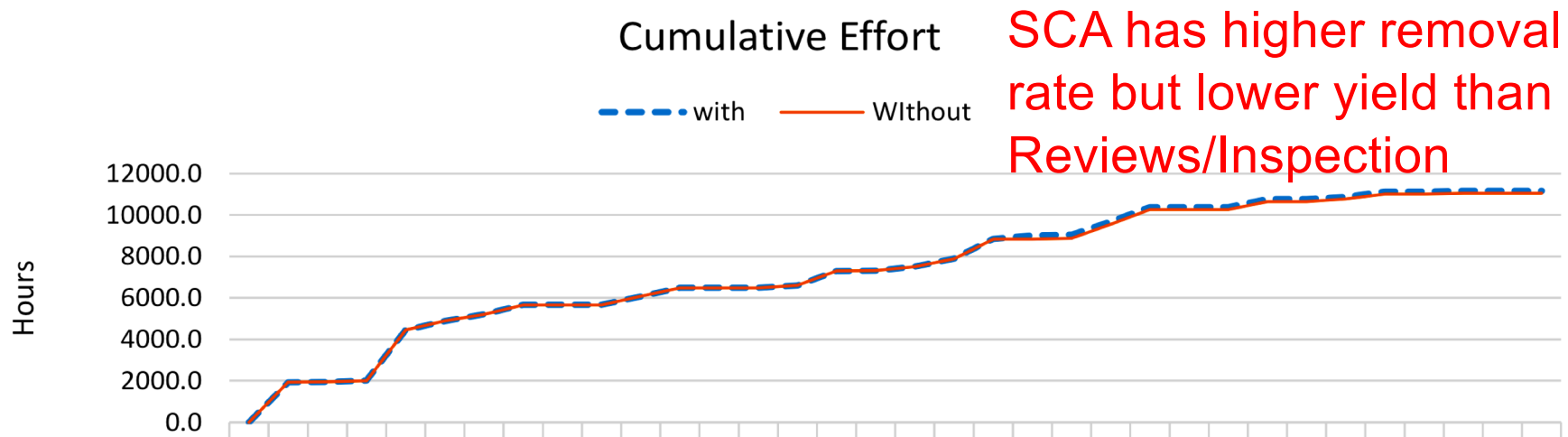
Results include rows where 'removed_phase_ordinal' >= 26.

Org A at Review, Defect Density With and Without



Org A at Review, Effort With and Without

Little net change in effort
SCA has higher removal rate but lower yield than Reviews/Inspection

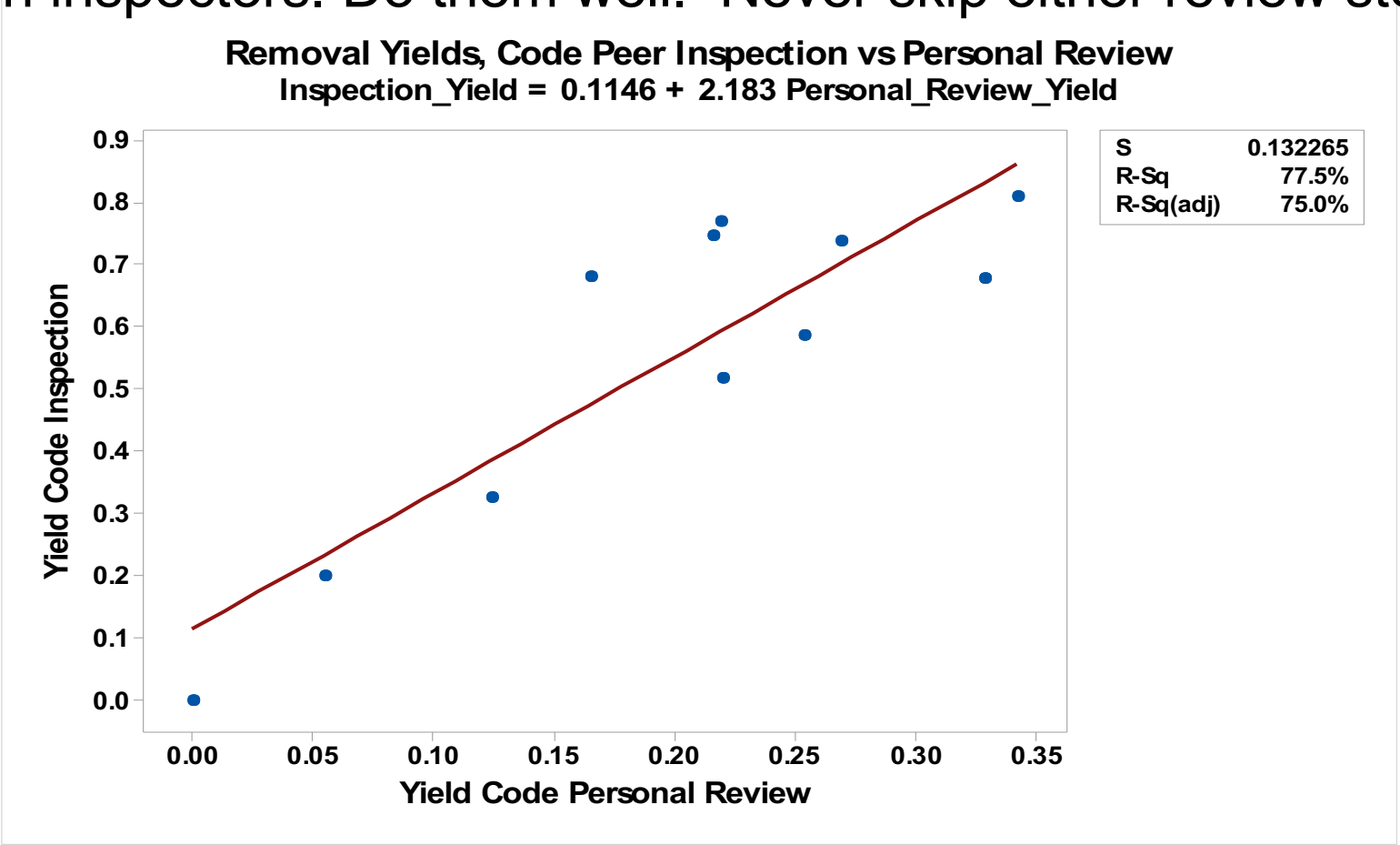


	0000_BeforeDev	1000_Misc	1100_Strat	1150_Planing	1200_SE_REQ	1220_SE_REQR	1240_SE_REQI	1250_SE_REQ_val	3000_Req	3020_ReqR	3040_ReqI	3100_HLD	3110_ITP	3120_HLDR	3140_HLDI	3200_DLD	3210_TD	3220_DLDR	3220_DLDI	3300_Code	3320_CodeR	3330_Compile	3340_CodeI	3350_UTest	3551xxxx	3400_TestCaseDe...	4010_BITest	4015_xxx	4030_STest	4040_Doc	4050_ATest	5000_PM	6100_PLife	6200_AfterDev
with	0.	19	19	20	44	48	52	56	56	56	60	64	64	64	66	72	73	75	78	88	90	90	96	10	10	10	10	10	10	11	11	11	11	11
WIthout	0.	19	19	20	44	48	52	56	56	56	60	64	64	64	66	72	73	75	78	88	88	88	95	10	10	10	10	10	10	11	11	11	11	11

Development Process

Personal Review Skills Transfer to Peer Inspection.

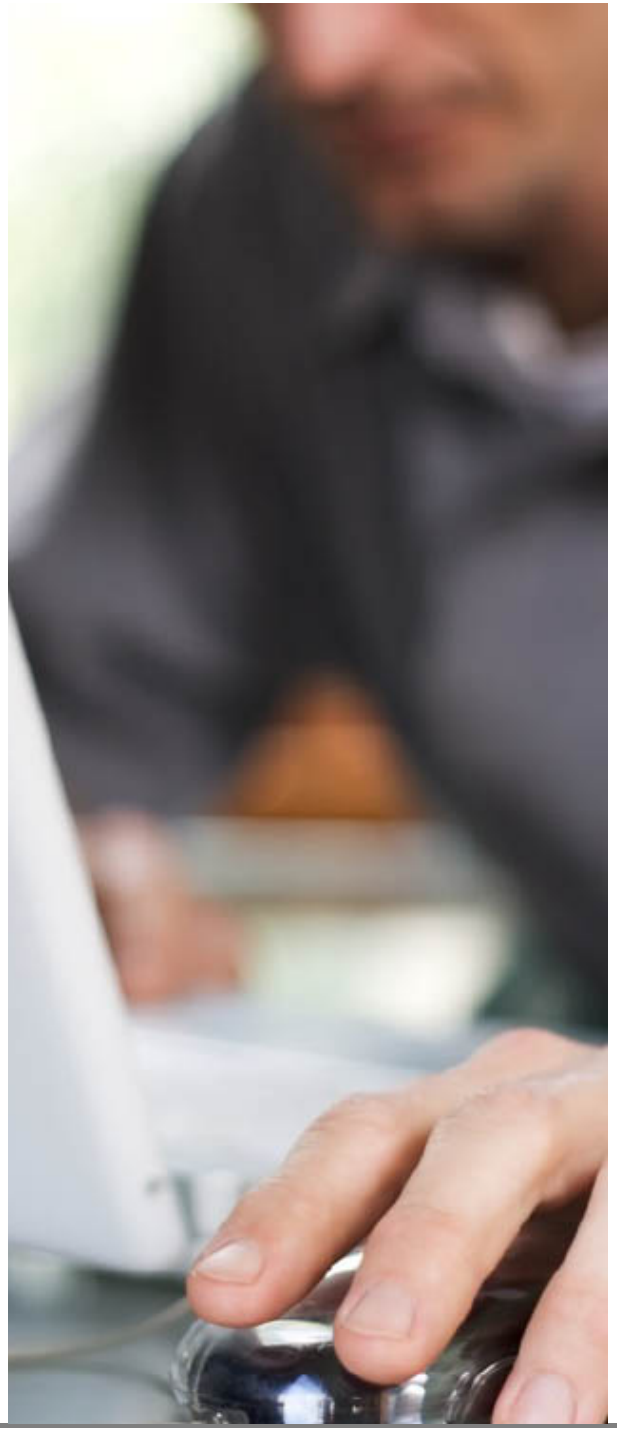
Personal (self) reviews and Peer Inspections are cost effective. Train inspectors. Do them well. Never skip either review step.



Composing Effective Software Assurance Workflows

Limitations

Discussion



Limitations

These teams already had high quality. They were disciplined enough to define a process and collect data and performed effective inspections. (SC/BA should have even more positive results if more issues escape into test. Even top performers benefit.)

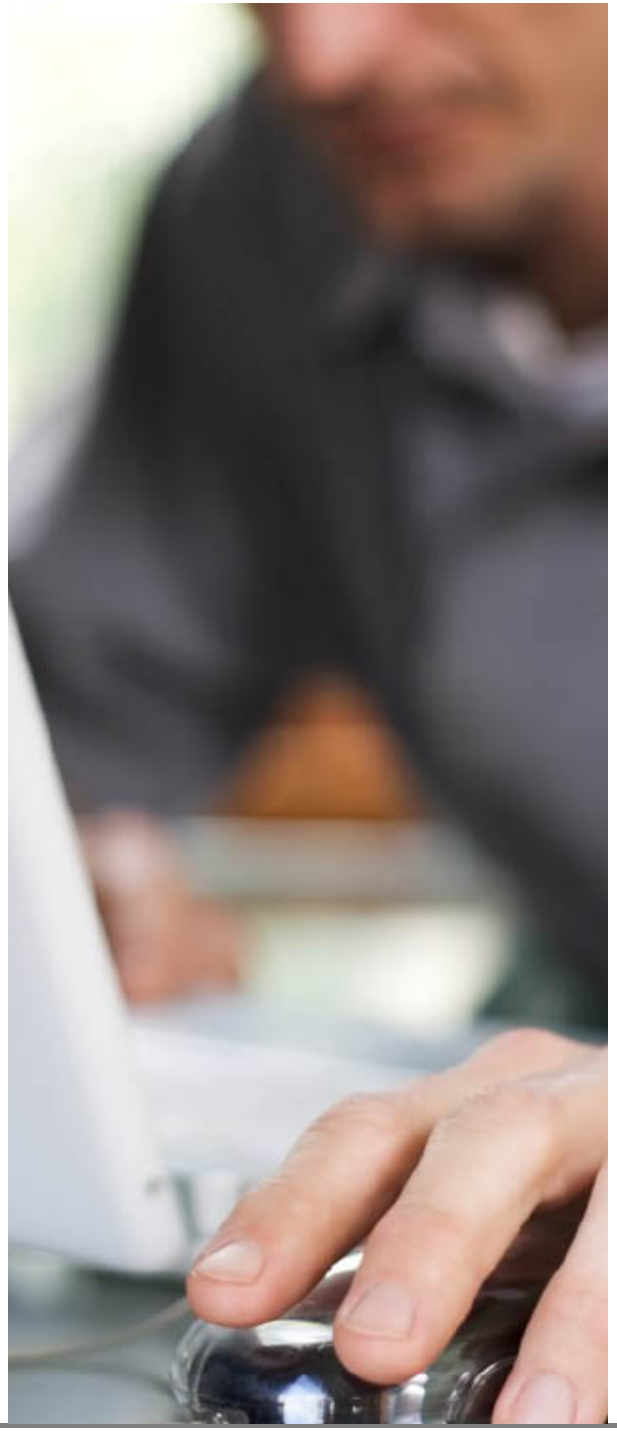
There are no specific measures of vulnerabilities in the final products. (Each tool has known efficacy for CWEs, at a minimum we find these. Other finds tend to reduce downstream costs.)

Only static analysis tools use in this study. (these are commonly used, but no data on other tools or tool interactions)

Few companies and domains represented. (three diverse domains and multiple projects within each company demonstrate convergent results)

Cost/Benefits of Software Assurance Tools

Discussion and Summary



Observations

Three organizations each used a different tool at a different time in the development cycle.

Overall results (quality and effort) were similar.

Benefits of SCA/SBA apply broadly.

Observations,

Other empirical studies suggest the SCA tools have low overlap in finds,

Inspections remain key. Tools find defects faster, but

- Humans remain more effective (higher yield) overall,
- Humans must fix all defects identified,
- Fix time depends on how precise the defect is located,

Consider multiple static analysis tools

Tool aggregators should achieve much higher yield.

Three different tool usage approaches were effective

- Consider different tools before or after different process activities,
- Some tools may provide early direct feedback to developers,
- Inspections first may reduce false positives,
- SBA/SCA can substantially enhance integration test yield.

Observations

Other empirical studies suggest SCA are poor at predicting vulnerable components.

Vuls are sparse and the tools only find a fraction (20-25%) of them. SCA is a cost effective way to improve quality and security. Use the tools on **all** code to get the benefit.

The tools are far from perfect and can provide a false sense of security.

If the tool is imperfect, use it, do not rely on it.

We saw evidence of *risk compensation*, (Peltzman effect). **Supplement, Don't Substitute**. **High yield** manual reviews, tests and other activities are are still necessary. Do not skip manual reviews, *or perform superficially*.

Observations

SCA/SBA tools are imperfect, but prescriptive.

Human review has higher yields, but is stochastic.

Using SCA/SBA, developer effort is dominated by defect fix time.

Reviews and Inspections take about the same amount of time regardless of defects found. (time dominated by inspection not fix).

Counterintuitive,

For efficiency, apply SCA/SBA **after** manual inspections.

Other studies suggest tool false positive rates go down with cleaner code.

Bottom Line at Bottom

SCA and SBA **reduce development time** and improve quality

For SCA and SBA we have clear evidence to recommend

Build/automate S(C/B)A into workflows (faster and cheaper)

- SCA before unit test, SBA at integration before integration or system test.
- SCA has similar fix times to compilation warnings and inspections finds. Fast to find issues, but the same to fix. (for efficiency, run before integration or test)

Supplement, **don't Supplant** other effective defect controls,

- SCA still has a weaker yield than inspection, (which still have very high yields) (SCA is 3 times as fast but has 1/3 the removal yield)

SBA has similar fix times to integration defects, but much faster than test. Think of it as a way of Improving integration test yield

Call for Data

If you have data you can share,

If you would like the data used in this report,

If you would like a copy of the spreadsheet analyst tool, or

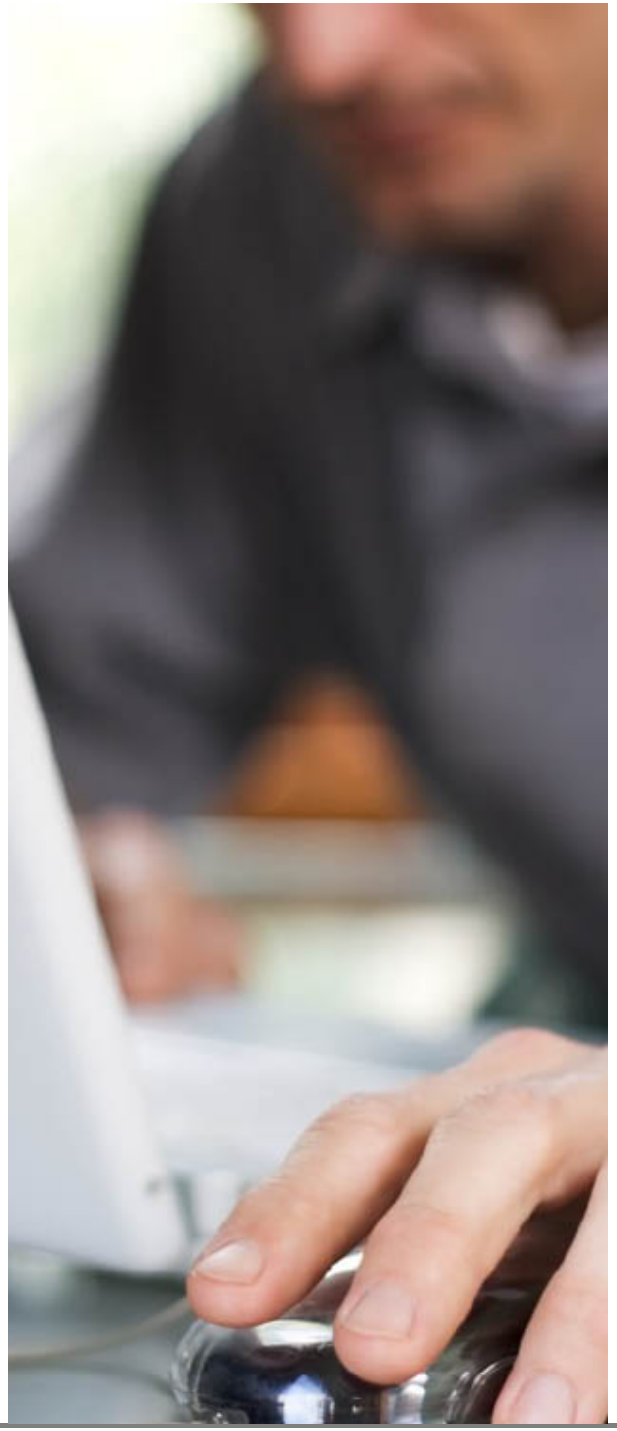
If you want to learn more about next steps including Causal Analysis and structural modeling

Contact

Bill Nichols wrn@sei.cmu.edu

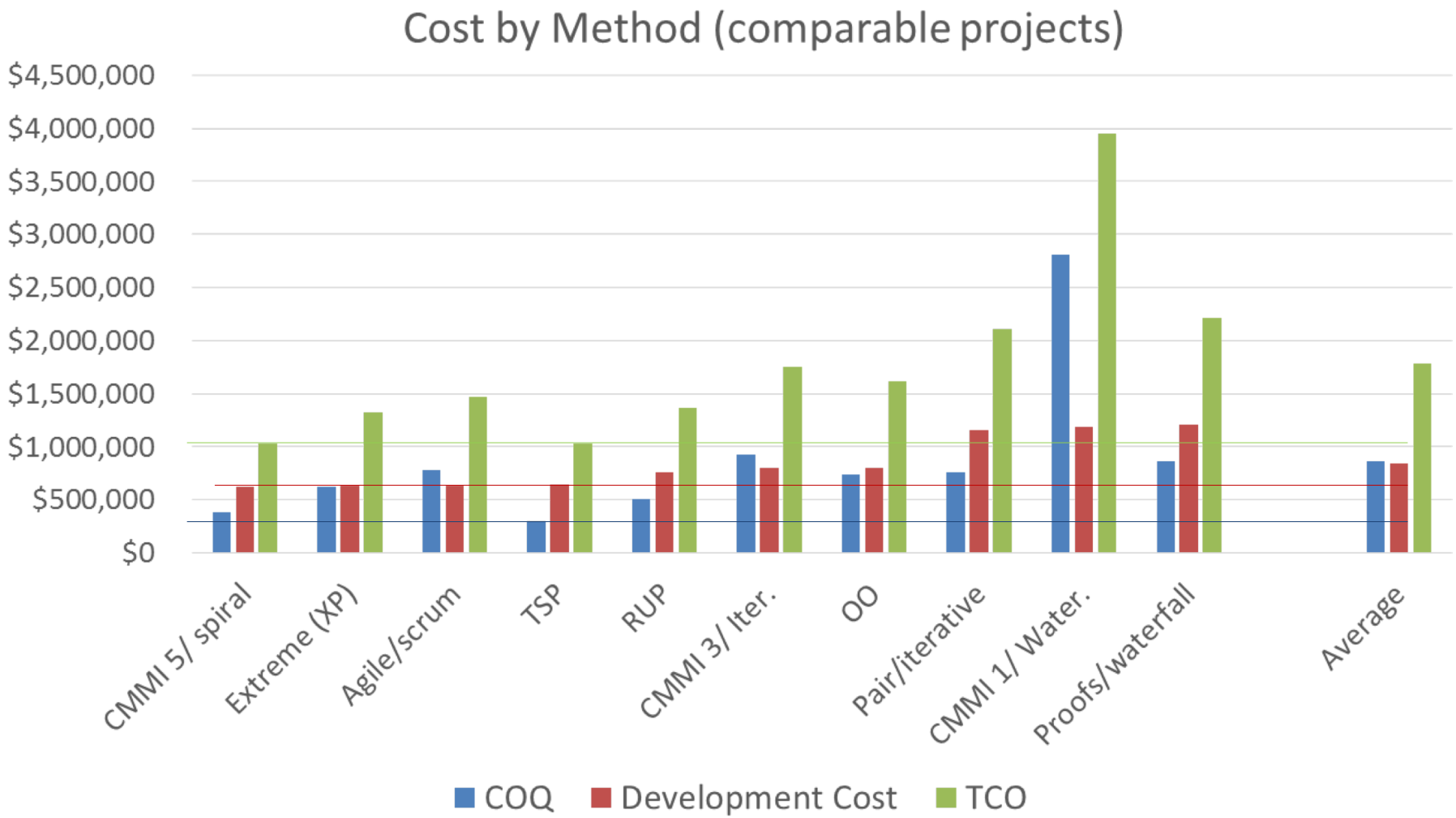
Cost/Benefits of Software Assurance Tools

Backup



Good methods show themselves, Cost

Comparable Size Projects, quality is not just free
"The Gift that Keeps on Giving"



Source: C. Jones http://www.infoq.com/articles/evaluating-agile-software-methodologies

TSP Defect Model

Is similar to COQUALMO in some key ways, but also has differences

1) Defect fix effort is explicitly quantified,
2) We calibrate with local TSP measurement (using TSP parameters)

As with COQUALMO, the assumption is that there is an underlying causal mechanism in software development. This leads us to the SCOPE project.

Metrics

Defect Density (throughout lifetime)

Vulnerability Density (found at each stage)

Phase Injection Rate [defects/hr] (derived)

Phase Effort Distribution (effort-hr)

Phase Removal Yield [% removed] (effectiveness)

Defect “Find and Fix” time [hr/defect] (what was found)

Defect Type (categorize what was wrong)

Defect Injection/Removal Phases

Zero Defect Test time [hr] (cost if no defects present)

Product Size [LOC] [FP] (for normalization and comparisons)

Development Rate (construction phase) [LOC/hr]

Review/Inspection Rate [LOC/hr] (cost of human appraisal)