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
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 entirely
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

- 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.

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

We do not compare tools or measure coverage against CWE
Collect Observational Data from Cases

Three Organizations,

<table>
<thead>
<tr>
<th>org</th>
<th>Proj</th>
<th>Domain</th>
<th>Total Hours</th>
<th>Tools</th>
<th>When used</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>Embedded</td>
<td>71749</td>
<td>SCA</td>
<td>Personal Code Review</td>
</tr>
<tr>
<td>B</td>
<td>16</td>
<td>Business Intelligence</td>
<td>23170</td>
<td>SCA/SBA</td>
<td>Pre Code Inspection</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>Engineering CAD</td>
<td>26759</td>
<td>SCA/SBA</td>
<td>Integration</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td></td>
<td>121708</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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?

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)…

<table>
<thead>
<tr>
<th>PSP0.1</th>
<th>Project Plan Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Settings</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Program Size</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Time In Phase</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Defects Injected</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Defects Removed</strong></td>
<td></td>
</tr>
<tr>
<td><strong>No-Frames version</strong></td>
<td>&quot;Best for printing&quot;</td>
</tr>
</tbody>
</table>

---

### /Non Project/PSP for Engineers/Program 2B

#### PSP0.1 Project Plan Summary

- You have not entered the planned size for this project.

#### Program Size

<table>
<thead>
<tr>
<th>Category</th>
<th>Plan</th>
<th>Actual</th>
<th>To Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (B)</td>
<td>81</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Deleted (D)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Modified (M)</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Added (A)</td>
<td>84</td>
<td>249</td>
<td>165</td>
</tr>
<tr>
<td>Reused (R)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Added and Modified (A+M)</td>
<td>150</td>
<td>114</td>
<td>165</td>
</tr>
<tr>
<td>Total Size (T)</td>
<td>165</td>
<td>249</td>
<td></td>
</tr>
<tr>
<td>Total New Reusable</td>
<td>51</td>
<td>51</td>
<td></td>
</tr>
</tbody>
</table>


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

\[
\text{Product} = \text{ProdRate} \times \text{Time}
\]
\[
\text{Defects} = \text{DefRate} \times \text{Time}
\]

Similar to Jones “Tank and Filter”,
Simplifies assumptions found in Boehm/Chulani COQUALMO
Integrates defect and cost projections

Does not consider organizational bookkeeping costs

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.

If Static Analysis finds the defect, it is less expensive to locate and fix than failures in Int/Sys/Accept Test

Tools reduce source search cost
Org C, **Static Analysis** fixes defects with less effort than **Test**

**defect_fix_time_minutes for Acpt Test, Code/Design Injections**

- **AT Average**: Savings from avoiding expensive defects
- **S(C/B)A Average**: Cheaper to find and fix defects with static analysis than with test

---

**Anderson-Darling Normality Test**

- A-Squared: 7.67
- P-Value: <0.005

**Summary Statistics**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>13.057</td>
</tr>
<tr>
<td>StDev</td>
<td>24.532</td>
</tr>
<tr>
<td>Variance</td>
<td>601.800</td>
</tr>
<tr>
<td>Skewness</td>
<td>3.1601</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>11.7424</td>
</tr>
<tr>
<td>N</td>
<td>51</td>
</tr>
</tbody>
</table>

**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

Applied SCA/SBA in Integration

33% reduction escaped defects

Defect Density

Development Process
### Org C at Integration, Activity Effort With/Without

#### Cost and Benefits of Integrating Software Assurance Tools

A Quantitative Look © 2018 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

<table>
<thead>
<tr>
<th>Org C at Integration, Activity Effort With/Without</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="graph.png" alt="Graph showing activity effort with and without integration" /></td>
</tr>
</tbody>
</table>

**Small increase in effort**

Because more defects found and fixed

- With: 0.01680.08340.0.0.0.0.0.0.0.065384.22.1710.0.0.1.816320740335330979548.6232420.0.0.05772555571580.0.97.4670.0
- Without: 0.01680.08340.0.0.0.0.0.0.0.065384.22.1710.0.0.1.816320740335330979548.6232420.0.0.05770.05641590.0.97.4760.0
Org B at Compile, Defect Density With and Without SC/BA at Compile Bench Integration

30% fewer escapes

Code Inspection

Defect Density [Defects/KLOC]

Development Process

With

Without
Org B at Integration, Effort With and Without

Less Test Time
Fewer Defects to fix

Activity Effort

No change

Development Process
Cost/Benefits of Software Assurance Tools

Static Code Analysis Only

Results, SCA
Org A, Acceptance Test Defect Fix Times

Test defects are much more expensive for Org A

Results include rows where 'removed_phase_ordinal' ≥ 26.
Org A at Review, Defect Density With and Without

SCA “replaces” Personal Review

No Automation Run Very Early

Code Inspection Highest yield

18% reduction escapes
Org A at Review, Effort With and Without

Cumulative Effort

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

Removal Yields, Code Peer Inspection vs Personal Review

Inspection_Yield = 0.1146 + 2.183 Personal_Review_Yield

S = 0.132265
R-Sq = 77.5%
R-Sq(adj) = 75.0%
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 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”

Cost by Method (comparable projects)

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)