The Cost of SoC Bugs

Ken Albin
Oracle Labs
It began with a question…

• “How much will verification cost for project X?”
• Project X was a large, multi-site SoC integrating IP from several sources, some of unknown quality.
• Quality on previous projects had been “OK”, management was just concerned about increasing costs.
• Two observations from initial analysis…
... two observations:

1) The verification team controlled the basic infrastructure development and task execution - they could estimate this part by using complexity to scale previous similar efforts.

2) On the other hand, the verification team had very little control over the number of design bugs which would require debugging and rework, especially in new IP coming with unknown quality levels. Was there any industry data we could use to estimate the incremental cost of bugs?
My ulterior motive

• In this talk I use the term “bug” and “defect” interchangeably, but really it should be “change”.

• A small design change, and especially a requirements change, can turn into a large amount of work downstream in the development process.

• More than once I heard that a change was “just a couple lines of RTL” and I wanted data to show what the real impact might be.
Verification is critical to success

The later the bug is found, the more expensive to fix.

- Before RTL hand off: $10K for re-design
- Before Tape out: $100K for re-layout
- After Sample out: $1000K for re-spin
- After Mass production: >$10000K for re-call
Silicon Debug, Doug Josephson and Bob Gottlieb, (Paul Ryan)


Three parts to this talk

1) Where did these numbers come from anyway?
2) What are the real cost components of bugs?
3) What can you do with this data?
Part 1:
Where did these numbers come from anyway?
Sources of Bug Costs

Four major cost categories associated with design defects are:

• Missed market windows.
• Liability for safety or security defects.
• Damage to a company’s reputation.
• Engineering costs of finding and repairing defects during development.
Where did 10x come from?

• Charts similar to the table and chart above appeared in many places, but generally without attribution.

• Hardware projects are looking more and more like software development with reference models, object-oriented testbenches, etc. – maybe we can find justification for the curve in the software world.
http://www.ibeta.com/qa-on-demand/risks-of-not-testing-properly/
Catch Bugs Early!

Source: “Software Internationalisation Tools and Solutions” - Xerox

Cost to fix

Development Phase when an i18N bug is detected

Requirements
Architecture and Design
Coding
Testing
Acceptance
Localization
Maintenance

It’s far more efficient to find and fix i18n issues at the source level, rather than depending upon testing and localization iterations.

http://lingoport.com/internationalization-roi/
IEEE Computer: $14,102!
Costs of Correcting Defects


http://www.slideshare.net/mlevendusky/Cost-of-Correcting-Defects
Costs of Correcting Defects (Example)

Source: IEEE Computer Society

https://f14testing.wordpress.com/2009/12/
http://www.xqual.com/documentation/tutorial_test_metrics.html
Or maybe not so much

“Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase.”

• The charts do show roughly 100x from requirements to maintenance, but …

• No charts. No numbers. Not in the copies of the article I was able to obtain anyway.

• The search continues…

Relative Cost of Fixing a Defect

- Req’s: 1x
- Design: 5x
- Code: 10x
- Unit Test: 20x
- System Test: 40x
- UAT: 50x
- Post-Release: >150x

Cost of fixing bugs at various stages of software delivery

- Design: 1x
- Development: 10x
- Testing: 25x
- Staging: 50x
- Production: 150x

Cost of fixing a bug at different phases of the SDLC

- Requirements: 1
- Design: 5x
- Code: 10x
- Dev Testing: 20x
- Acceptance: 50x
- Production: >150x

http://codedx.com/ide-integration-helps-developers-adopt-application-security-testing-tools/
Advanced OOP and Design Patterns


Minutes to fix

https://www.scrumalliance.org/community/articles/2013/january/quality-is-free
http://www.jucs.org/jucs_13_5/realising_the_benefits_of/jucs_13_5_0669_0678_hall.html
The Relative Cost of Fixing Defects

THE RELATIVE COST OF FIXING DEFECTS

ILLUSTRATION BY SEQUE TECHNOLOGIES

http://www.sequetech.com/blog/2014/09/05/rising-costs-defects-infographic
Figure 1: Relative Costs to Fix Software Defects (Source: IBM Systems Sciences Institute)

Phase/Stage of the S/W Development in Which the Defect is Found

- Design: 1x
- Implementation: 6.5x
- Testing: 15x
- Maintenance: 100x
Relative Cost of Fixing Defects

http://www.slideshare.net/drdawson/secure-software-development-life-cycle
A breakthrough!
Figure 1: Cost of Bug Elimination in the Software Development Lifecycle [NIST 2002]
http://blog.pdark.de/2012/07/21/software-development-costs-bugfixing/
Relative Cost to Correct a Defect

Development Phase

Requirements  Design  Code  Test  Operation

https://fcbqacorner.wordpress.com/
Relative Cost of Fixing Errors

Operation and Maintenance
System Test
Unit Test and Integration
Code and Debug
Design
Requirements

SDLC Phase
Another breakthrough!
The Cost of Defects

Fix Earlier, reduce cost

Implementation Unit Testing Integration Testing System Testing In-service

http://www.embeddedinsights.com/channels/2012/03/19/unit-test-tools-and-automatic-test-generation/
FIXING SECURITY EARLY IN THE SDLC SAVES UP TO 90% OF THE REMEDIATION COSTS

“Cost to find/fix a defect during integration/system test is 15-90 times higher than at design/coding”

[Escalating cost to find and fix a defect or design flaw as it is discovered late in the Software Development Life Cycle (IDC, 2005)]

http://www.cert2connect.com/
Cost of finding a bug at different testing Stages

Cost of Fixing a bug

Testing Type  | Unit Testing | Integration Testing | System Testing | Acceptance Testing | Regression Testing
---|---|---|---|---|---

http://www.theautomatedtester.co.uk/blog/2008.htm
The graph shows the cost of fixing bugs at different stages of the software development lifecycle, from Requirements Gathering to Live. The x-axis represents the point at which a bug is discovered, while the y-axis represents the cost of fixing the bug. The graph illustrates that the cost of fixing a bug increases significantly as the bug is discovered later in the development process. This emphasizes the importance of effective testing and quality assurance practices to minimize costs and improve software reliability.
THE COST OF FIXING A BUG

Applied Software Measurement, Capers Jones 1996

Coding  Unit Test  QA Testing  Field Test  Post release

25 $  1,000 $  16,000 $
The Cost of Bugs

Cost to Fix vs. Lifecycle Stage

- Specification
- Design
- Code
- Unit Test
- System Test
- UAT
- Release

% Defects introduced in this phase
% Defects found in this phase
$ Cost to repair defect in this phase

Source: Applied Software Measurement, Capers Jones, 1996

https://utbrudd.bouvet.no/2012/03/09/the-vicious-release-circle/
Cost to find/fix a defect during integration/system test is 15-90x higher than at design/coding.

Static analysis tools find defects and design flaws “in phase”

System testing
Integration testing
Unit testing
Code inspection

Cost to find and fix a defect

Design  Coding  QA  Production
Cost of an error

- Strategic planning
- Requirements
- Design
- Construction
- Transition
- Production

https://enectoux.wordpress.com/tag/business-architecture/
Cost of finding and fixing defects increases over time.

FIGURE 1.2

http://istqbexamcertification.com/what-is-the-cost-of-defects-in-software-testing/
Cost of Change

Requirements | Analysis and Design | Coding | Testing in the Large | Production

Time

Copyright 2002 Scott W. Ambler

http://www.agilemodeling.com/essays/costOfChange.htm
Cost of Bugs

- Cost Multiplier

Lifecycle Phase:
- Requirements
- Design
- Coding
- Unit Testing
- Integration Testing
- System Testing
- Installation Testing
- Acceptance Testing
- Post-live
- Post-live Change

http://www.mediacurrent.com/blog/why-qa-your-website
Relative cost-to-fix a bug in various project phases

bug fix costs

http://watirmelon.com/2013/05/
my personal favorite

outliers
http://arashiqe.blogspot.com/2012/07/software-cost-of-defects.html
The graph illustrates the cost of reparation in dollars, depending on where the fault was detected. The cost increases significantly as the fault is detected at higher levels: chip, board, system, field, and space. The cost is measured on a logarithmic scale, with the y-axis ranging from 1 to 1,000,000,000.

The URL provided is: http://www.pldworld.com/ hd/1/www.ireste.fr/fd/vcl/tools/vmethods.htm
Apparently this cost curve…

- Applies equally to hardware and software
- Works for any number of phases
- Applies to any kind of development phase

At some point I became skeptical…

After the IEEE Computer reference we looked at above, the NIST 2002 citation sounded the most authoritative. Let’s take a look:
### Table 5-1. Relative Cost to Repair Defects When Found at Different Stages of Software Development *(Example Only)*

X is a normalized unit of cost and can be expressed terms of person-hours, dollars, etc.

<table>
<thead>
<tr>
<th>Requirements Gathering and Analysis/Architectural Design</th>
<th>Coding/Unit Test</th>
<th>Integration and Component/RAISE System Test</th>
<th>Early Customer Feedback/Beta Test Programs</th>
<th>Post-product Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>1X</td>
<td>5X</td>
<td>10X</td>
<td>15X</td>
<td>30X</td>
</tr>
</tbody>
</table>
Software archaeology

• There are some in the software world that have traced through the tangles of references (e.g., Graham Lee in the blog cited below).
• The result was quite surprising to me…

source of all cost-to-fix charts
In Barry Boehm’s *Software Engineering Economics* (1981), he identifies the sources of data in this chart:

- IBM-SSD [Fagan, 1976]
- GTE [Daly, 1977]
- TRW [“several TRW projects”]
- SAFEGUARD [Stephenson, 1976]

- Later updated to include “two smaller, less formal software projects analyzed in [Boehm, 1980]”
Quick summary

• Software studies, not hardware
  – The last stage spike is maintenance/operational
• What is counted in the cost is generally not defined
• Commonly used data is old or perhaps made up
• Frequently used to advocate new methodologies

BUT...

• Bug fixes certainly seem more expensive late in the project
Late vs. latency vs. phase

• One of the original interpretations was that bugs are simply more expensive to fix when discovered late in the development lifecycle.

• A later interpretation attributed increased cost to the time the defect was latent in the design.

• More recently some have focused on the cost of rework and noticed that rework costs less in the phase where the defect was created. This led to the development of phase-containment approaches and metrics.
### Table 5-2. Preliminary Estimates of Relative Cost Factors of Correcting Errors as a Function of Where Errors Are Introduced and Found (Example Only)

<table>
<thead>
<tr>
<th>Where Errors are Introduced</th>
<th>Requirements Gathering and Analysis/Architectural Design</th>
<th>Coding/Unit Test</th>
<th>Integration and Component/RAISE System Test</th>
<th>Early Customer Feedback/Beta Test Programs</th>
<th>Post-product Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Gathering and Analysis/Architectural Design</td>
<td>1.0</td>
<td>5.0</td>
<td>10.0</td>
<td>15.0</td>
<td>30.0</td>
</tr>
<tr>
<td>Coding/Unit Test</td>
<td></td>
<td>1.0</td>
<td>10.0</td>
<td>20.0</td>
<td>30.0</td>
</tr>
<tr>
<td>Integration and Component/RAISE System Test</td>
<td></td>
<td></td>
<td>1.0</td>
<td>10.0</td>
<td>20.0</td>
</tr>
</tbody>
</table>

Planning Report 02-3 “The Economic Impacts of Inadequate Infrastructure for Software Testing”
Prepared by: RTI for National Institute of Standards & Technology Program Office, Strategic Planning and Economic Analysis Group May 2002
Importance of Early Verification, Validation, & Test

Relative Cost to Fix Defects per Phase Found


http://ww2.distek.com/casestudy/modeling-simulation-paper/
Cost of fixing a Bug

(source: Barry Boehm "Equity Keynote Address" March 19, 2007)

http://www.slideshare.net/BosniaAgile/empiricism-with-scrum-by-ralph-jocham
Phase in Which a Defect Is Introduced

- Requirements
- Architecture
- Construction

Phase in Which a Defect Is Detected

- Requirements
- Architecture
- Construction
- System Test
- Post-Release

Cost
SoC waterfall development

• While out of favor in software development, SoC hardware refinement and physical implementation is necessarily mostly waterfall development.
• Moreover, the development phases generally involve different teams at different sites performing quite different tasks. The cost of deliveries across phase boundaries can be significant.
• As such, the phase-containment ideas developed in the software world may apply.
Part 2:
What are the real cost components of bugs?
requirements

specification

IP_002 RTL

specification

IP_150 RTL

specification

IP_299 RTL

specification

IP_001 RTL

SoC RTL

netlist

opt. netlist

layout

Post-Si
IP RTL
SoC RTL
IP RTL
Post-Si
layout
optimized netlist
netlist
requirements specification
Stage vs. entire design flow

• Modeling cost can be done at the project-level or within a single stage:
  – Decisions to increase quality upstream (e.g., shifting resources into IP verification) must typically be done by management at the project level.
  – Within a stage there are many opportunities to improve efficiency, and the stage team may be empowered to make those changes.

• Let’s look at a single stage…
Accepts and releases

Accept
Release
Base Line
Bug Reports
Debug loop
Detect
Fix/Regress
Isolate
Integrate/Commit
Baseline work

- Accept
- Release
- Debug loop
- Detect
- Isolate

- Base Line
- Bug Reports
- Fix/Regress
- Integrate/Commit

3/2/2016
Ken Albin, Oracle Labs
Debug loop:
Detect

Base Line

Accept
Release
Bug Reports
Debug loop
Detect
Fix/Regress
Isolate
Integrate/Commit

3/2/2016
Ken Albin, Oracle Labs
Debug loop:

**Isolate**

- Accept
- Base Line
- Release
- Bug Reports

- ISOLATE
- BASE LINE
- INTEGRATE/COMMIT
- ACCEPT
- RELEASE
- BUG REPORTS

- DEBUG LOOP
- DETECT
- FIX/REGRESS

3/2/2016

Ken Albin, Oracle Labs
Debug loop: Fix/Regress
Debug loop:
Integrate/Commit

I/C  Integrate/Commit

I  Integrate

D  Detect

F/R  Fix/Regress

A  Accept

R  Release

I  Isolate

I/C  Integrate/Commit

Base Line

Accept

Release

Bug Reports

Debug loop

DVCON
2016
Design and Verification
Conference and Exhibition
United States
Bugs and communication

- Accept
- Base Line
- Release
- Bug Reports
- Debug loop
- Detect
- Fix/Regress
- Isolate
- Integrate/Commit

Diagram:
- A: Accept
- R: Release
- I/C: Integrate/Commit
- F/R: Fix/Regress
- D: Debug loop
Collecting data

• Most of the data needed for the model already exists but may be in different systems or may need to be counted differently. For example:
  – Bugs counted by where created, where found, fixed or not, etc.
  – Compute resources may be tracked by IT, and their use may be associated with development phase (e.g., “debug saturated” vs. “bug hunting”)
  – Debug tool license usage can track debug activity.

Estimates can be used when necessary.
Part 3:
What can you do with this data?
Tuning the debug cycle

- Accept
- Base Line
- Release
- Bug Reports
- Debug loop
- Isolate
- Integrate/Commit
- Detect
- Fix/Regress
Uses of (partial) cost models

I have used partial models on real projects:

• What-if analysis of IP vs. SoC debugging
  – “if we had 10% fewer IP bugs, it would be $X cheaper”

• Compute utilization (“debug saturated” vs. “bug hunting”)

• Reversing a >$1M business decision on a tool
Used for good or evil?

- Caper Jones: better quality makes cost per bug higher (one reason for exponential chart).
- Collecting fine-grain data on debug and other activities could become Big Brother-ish if not used with care (and can lead to wrong conclusions!)
- Cut-throat managers may realize they can lower costs by pushing debug costs downstream.
Summary and conclusion

• The exponential charts are something like trends we have seen in real projects, but are not backed by relevant studies.

• A more accurate model can enable us to make business decisions about tools/methodology, and resource allocation.

• A model of the entire design flow is nice, but a single stage model can be very useful by itself.
Thank you.
What about agile?

“One insight shows the cost-escalation factor for small, noncritical software systems to be more like 5:1 than 100:1. This ratio reveals that we can develop such systems more efficiently in a less formal, continuous prototype mode that still emphasizes getting things right early rather than late.

Another insight reveals that good architectural practices can significantly reduce the cost-escalation factor even for large critical systems. Such practices reduce the cost of most fixes by confining them to small, well-encapsulated modules."
