The Cost of SoC Bugs

Ken Albin

Oracle Labs

Hardware and Software

ORACLE

Engineered to Work Together

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

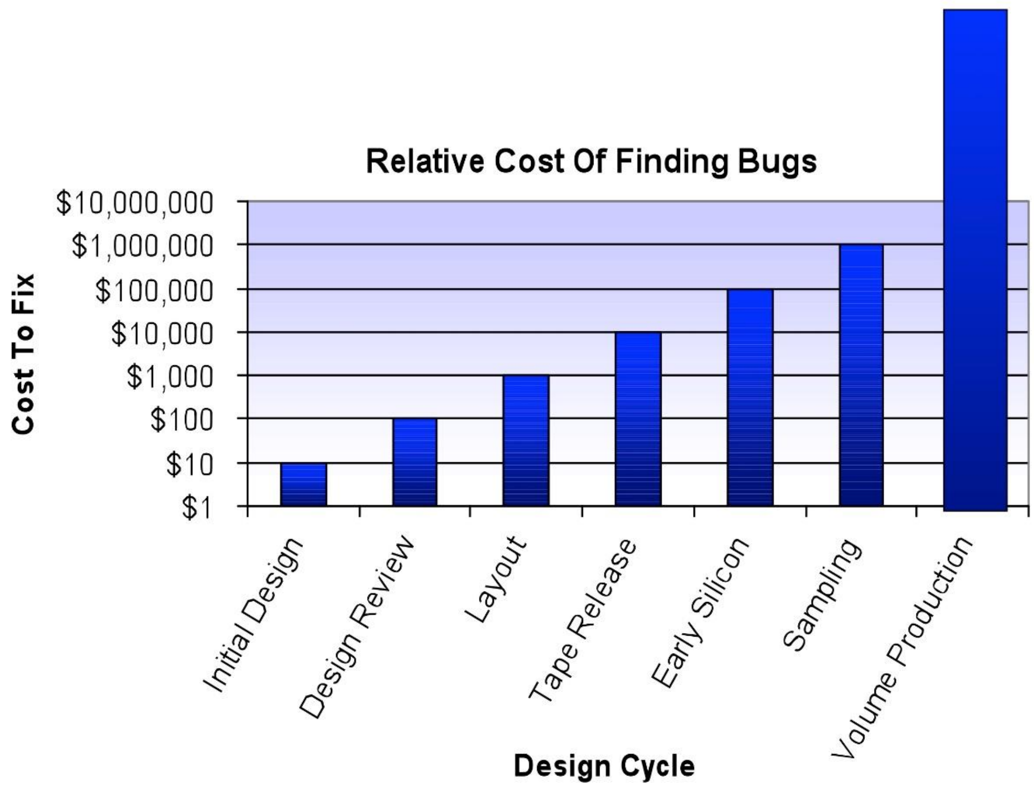
- In this talk I use the term “bug” and “defect” interchangeably, but really it should be “change”.
- A small design change, and especially a requirements change, can turn into a large amount of work downstream in the development process.
- More than once I heard that a change was “just a couple lines of RTL” and I wanted data to show what the real impact might be.

DAC 2004 verification panel

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

DAC 2004 Verification Panel, Makoto Ishii, LSI Design Division1, SoC Solution Center, SSNC, Sony.



Silicon Debug, Doug Josephson and Bob Gottlieb, (Paul Ryan)
D. Gizopoulos (ed.), Advances in Electronic Testing: Challenges and Methodologies, Springer, 2006

https://blogs.mentor.com/verificationhorizons/blog/2010/08/

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?

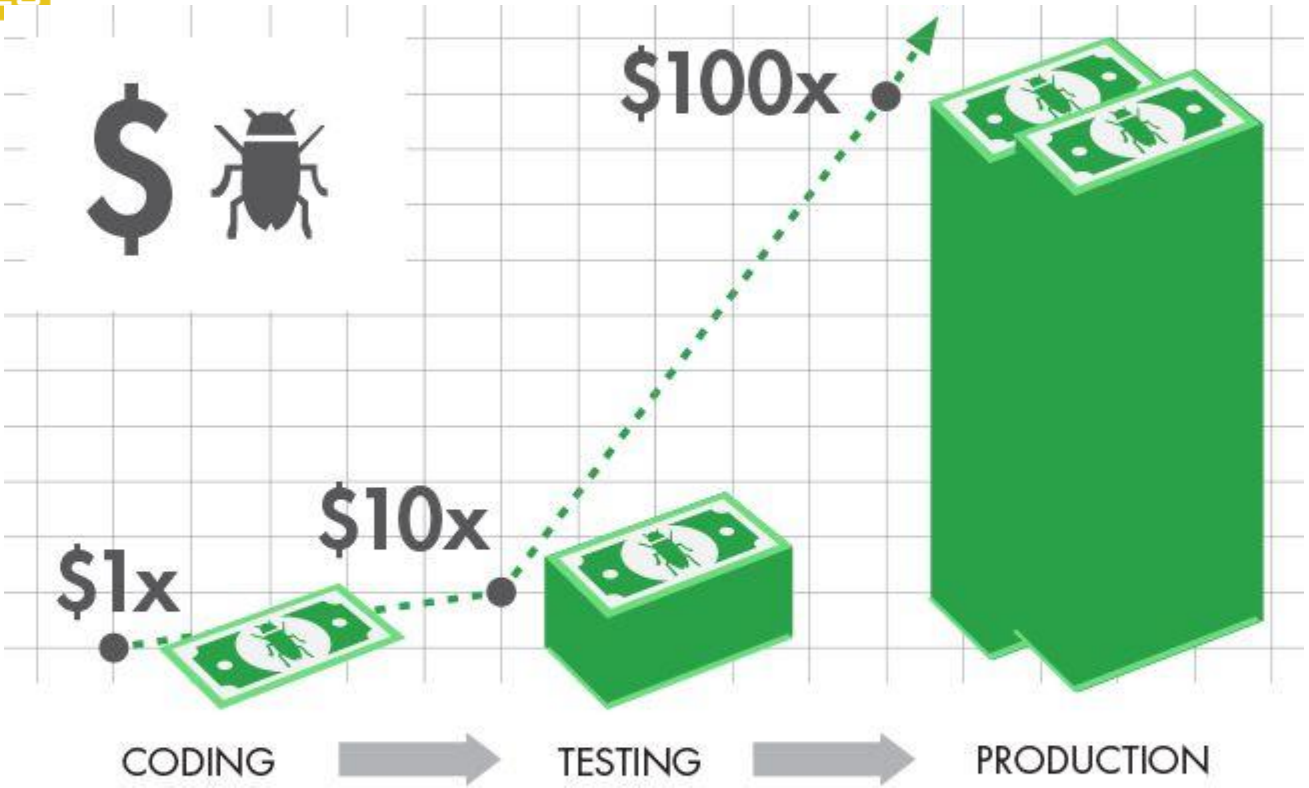
Sources of Bug Costs

Four major cost categories associated with design defects are:

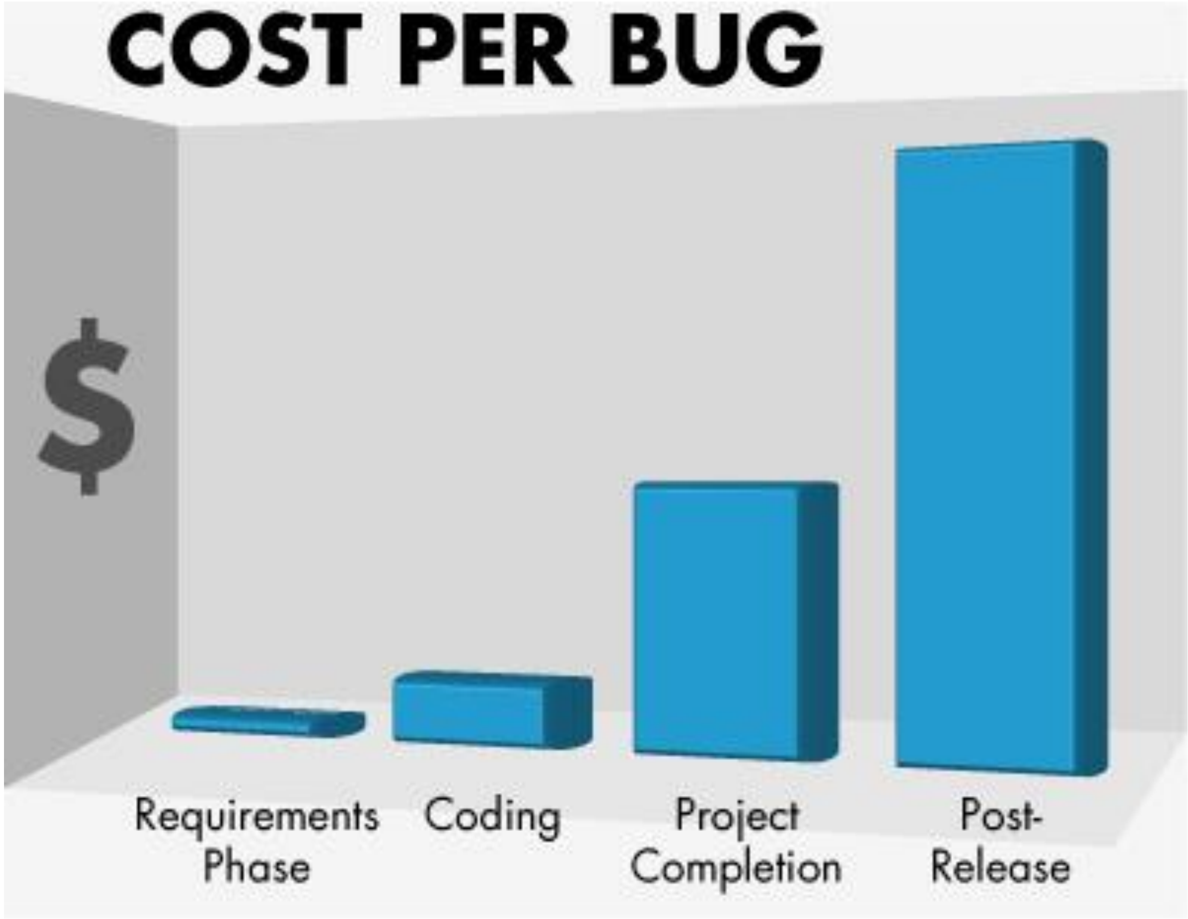
- Missed market windows.
- Liability for safety or security defects.
- Damage to a company's reputation.
- Engineering costs of finding and repairing defects during development.

Where did 10x come from?

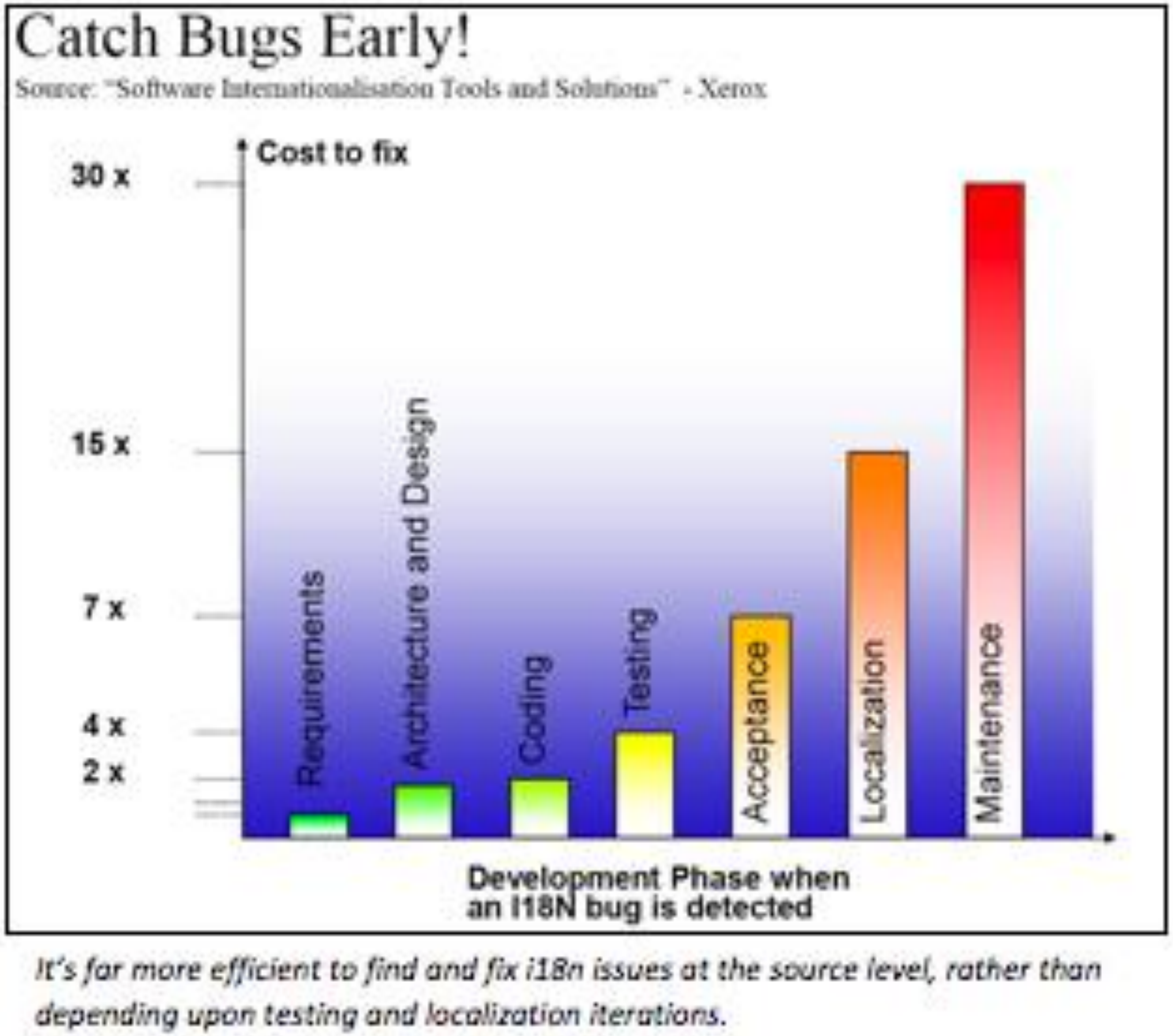
- Charts similar to the table and chart above appeared in many places, but generally without attribution.
- Hardware projects are looking more and more like software development with reference models, object-oriented testbenches, etc. – maybe we can find justification for the curve in the software world.



http://www.mathworks.com/products/polyspace/index.html?s_tid=gn_loc_drop

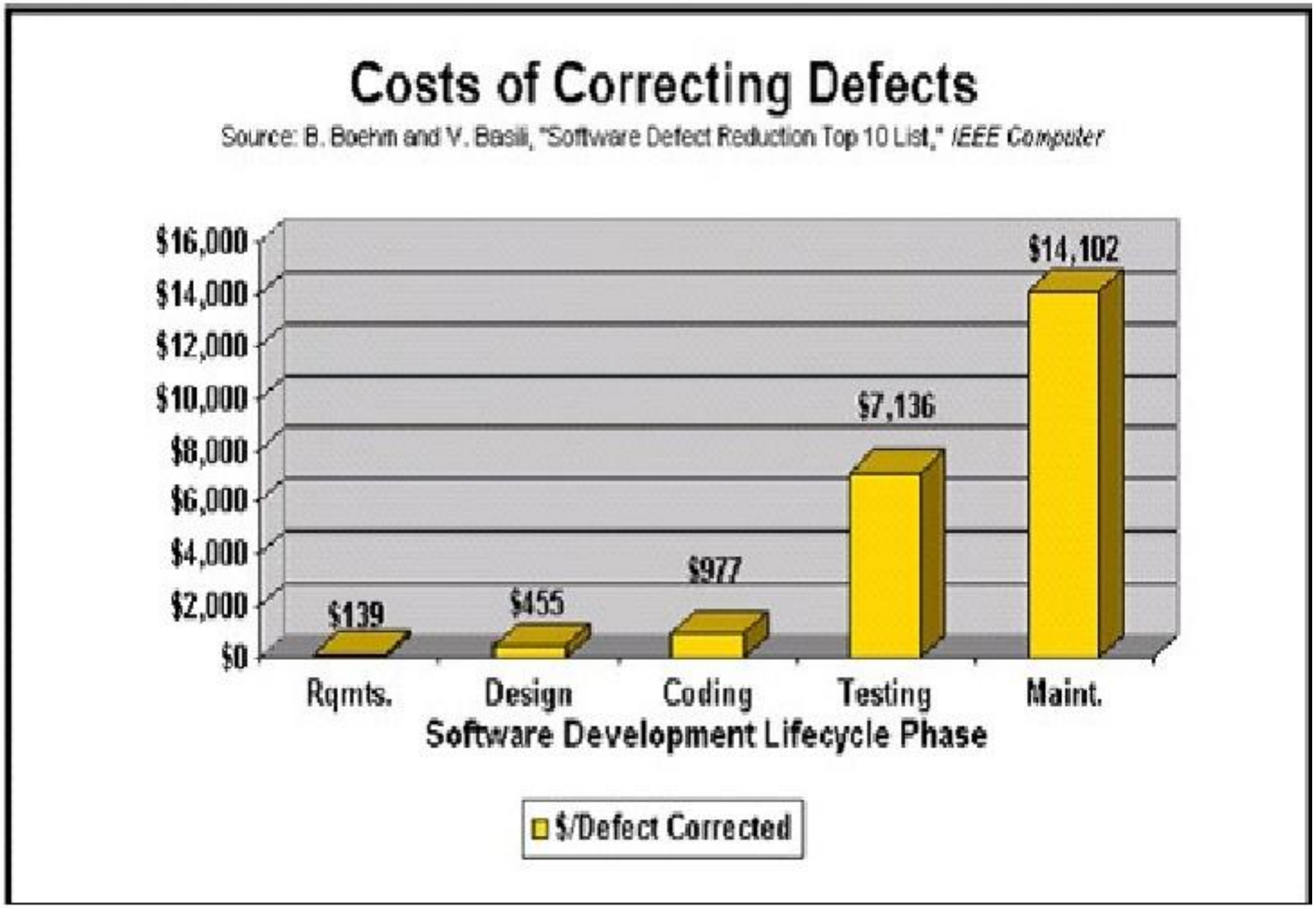


http://www.ibeta.com/qa-on-demand/risks-of-not-testing-properly/

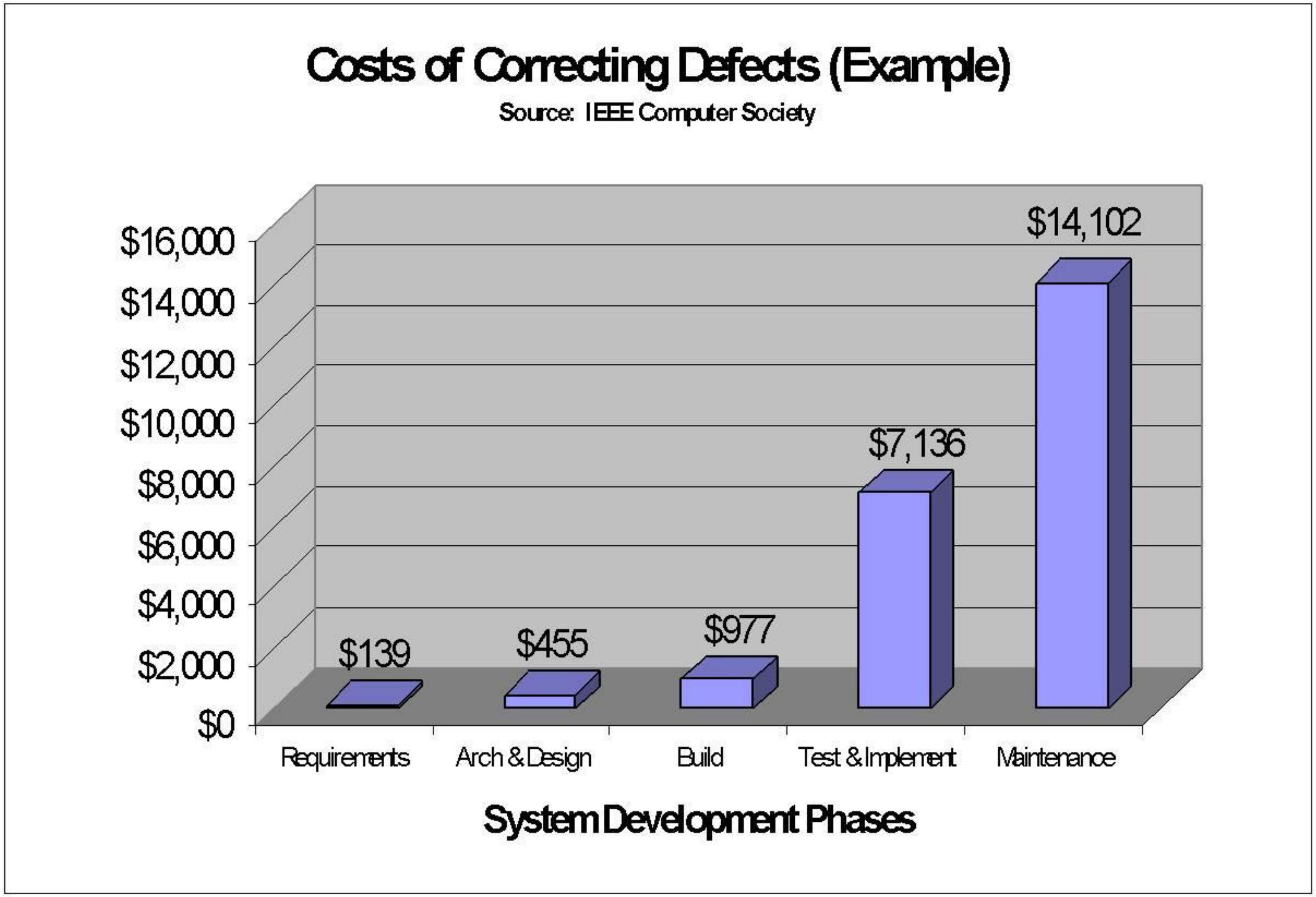


http://lingoport.com/internationalization-roi/

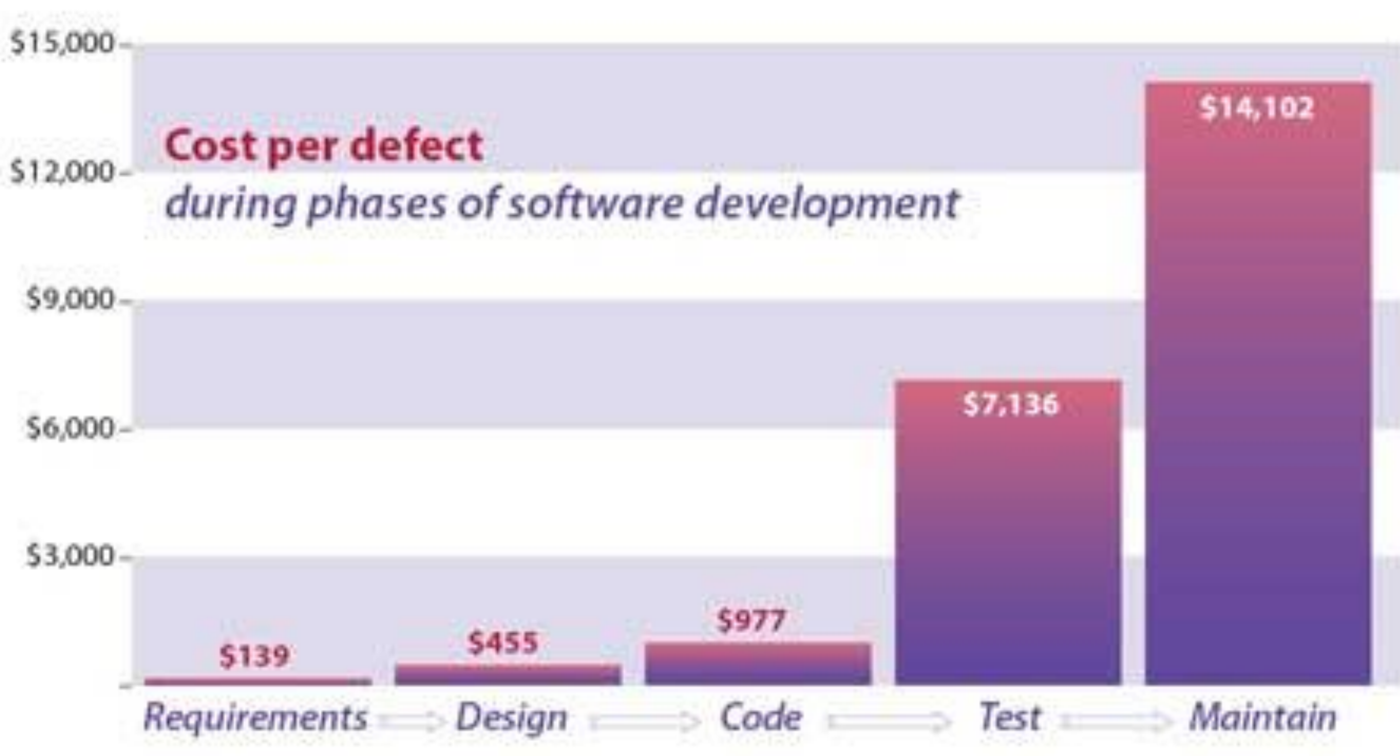
IEEE Computer: $14,102!



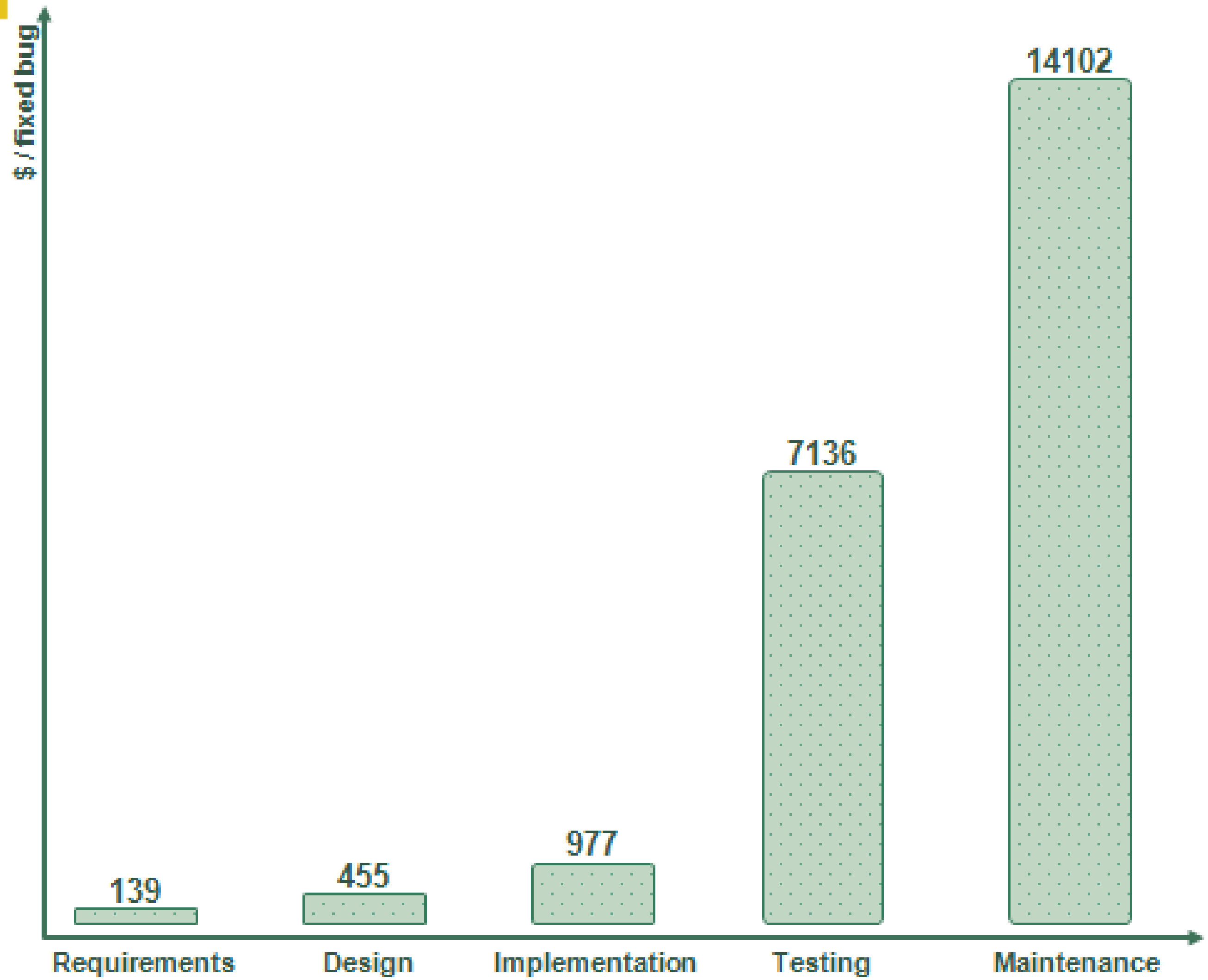
http://www.slideshare.net/mlevendusky/Cost-of-Correcting-Defects



https://f14testing.wordpress.com/2009/12/



http://blog.valiantys.com/fr/dev-tools/revue-de-code-pour-les-equipes



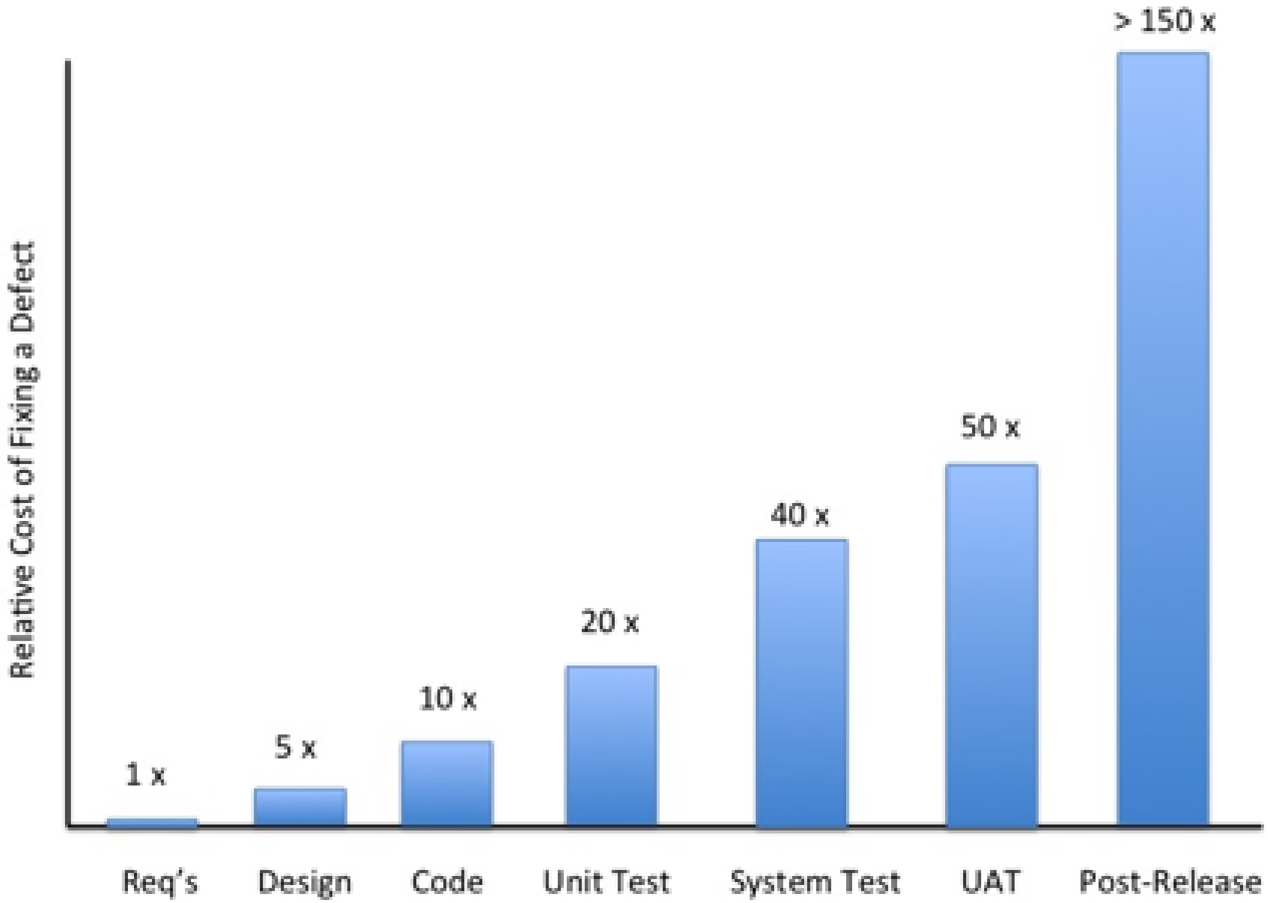
http://www.xqual.com/documentation/tutorial_test_metrics.html

Or maybe not so much

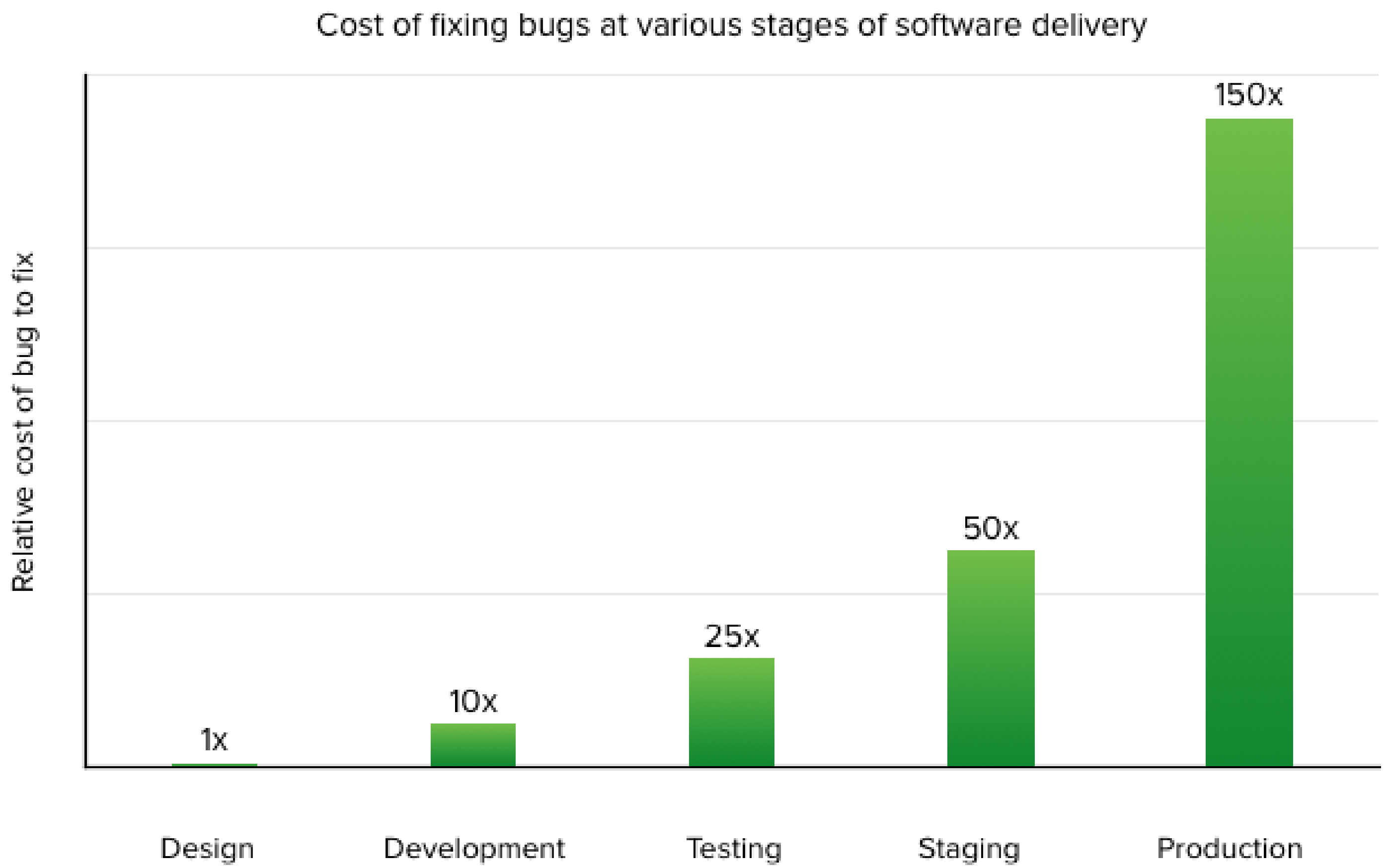
“Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase.”

- 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…

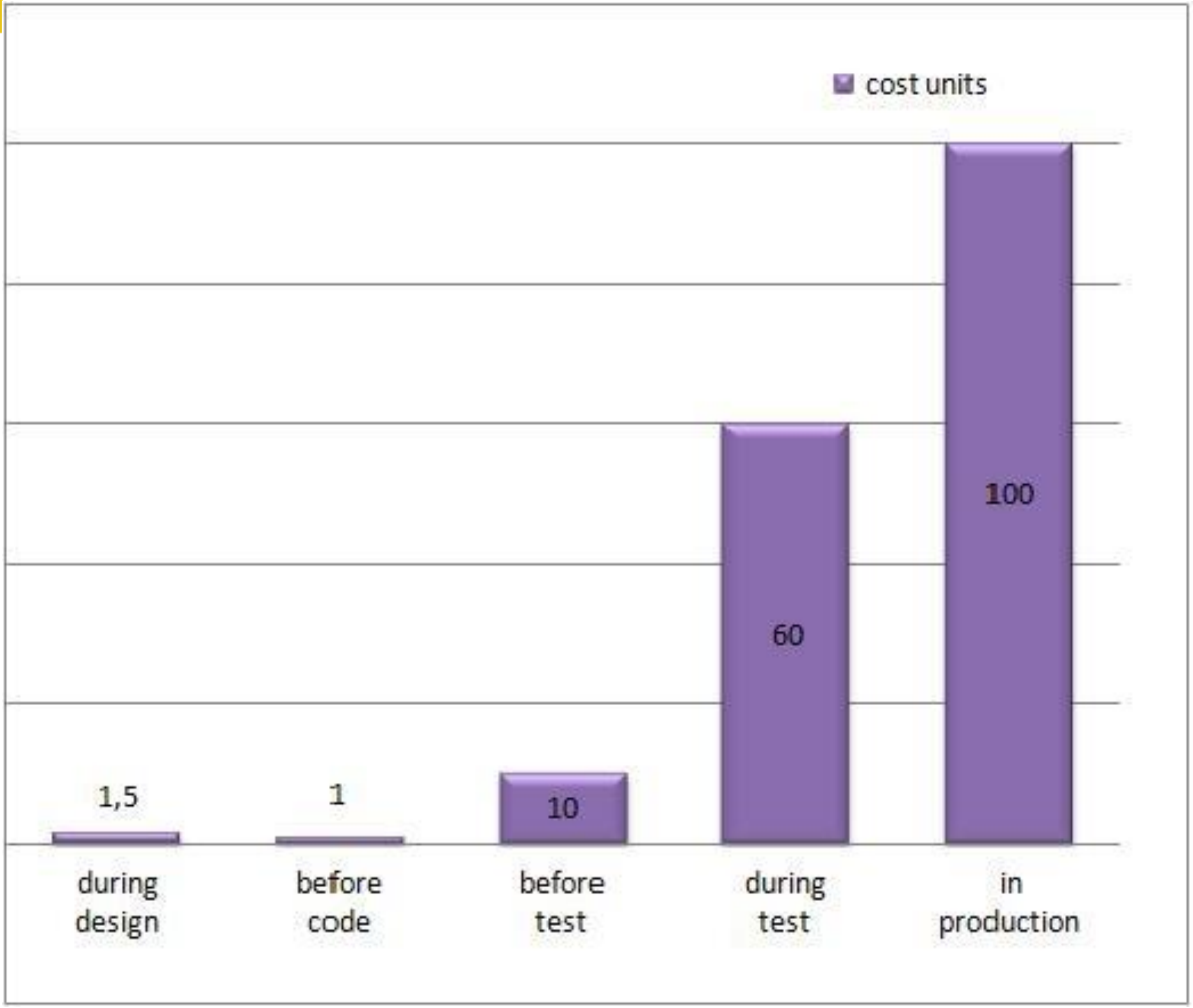
“Software Defect Reduction Top 10 List”, B. Boehm, V.R. Basili, *IEEE Computer*, January 2001.



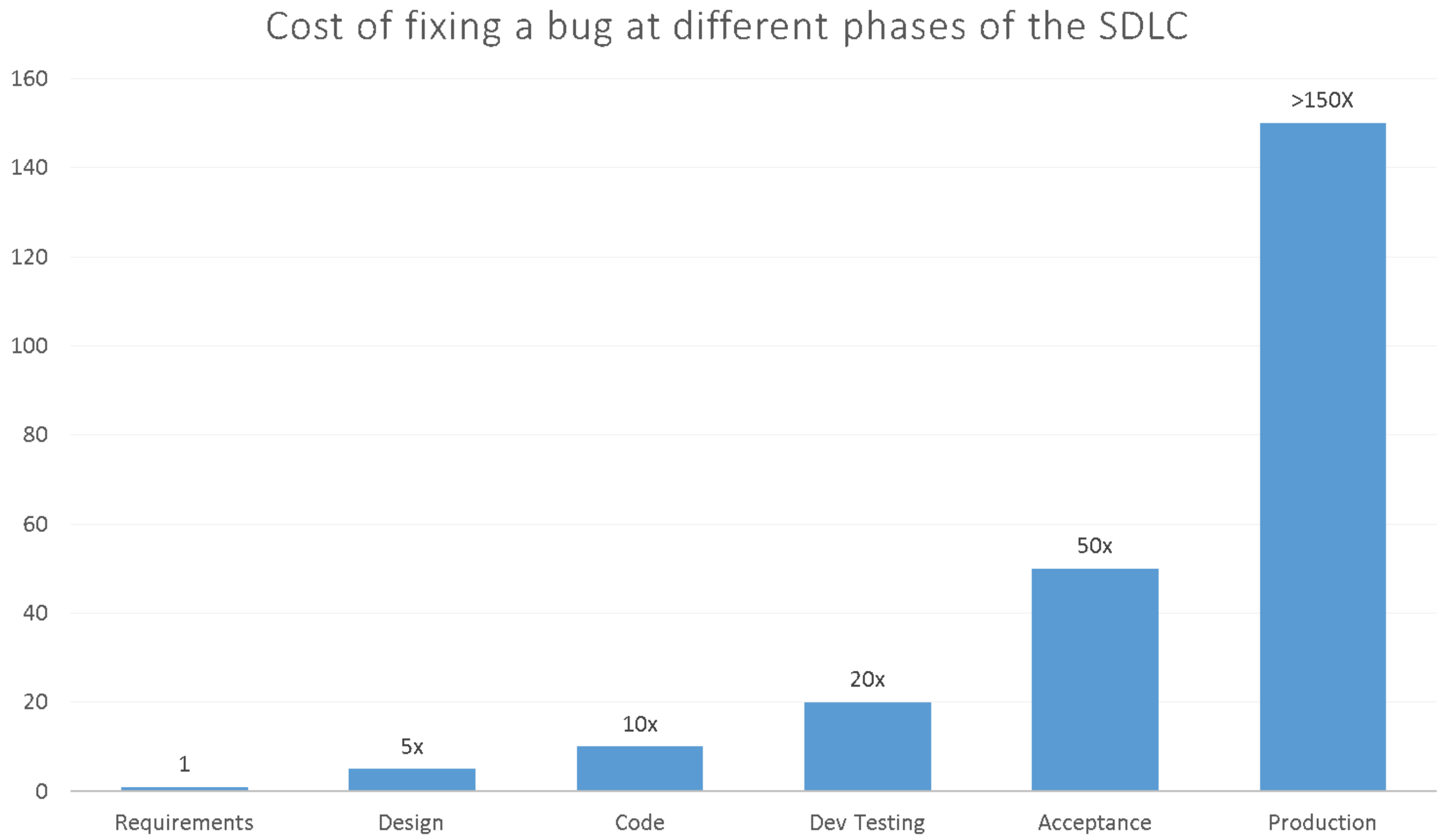
http://www.astqb.org/press-room/ISTQB_Certification_News_2015_1.html



https://raygun.io/blog/2014/01/massively-reduce-the-cost-of-bugs-with-raygun-error-tracking/

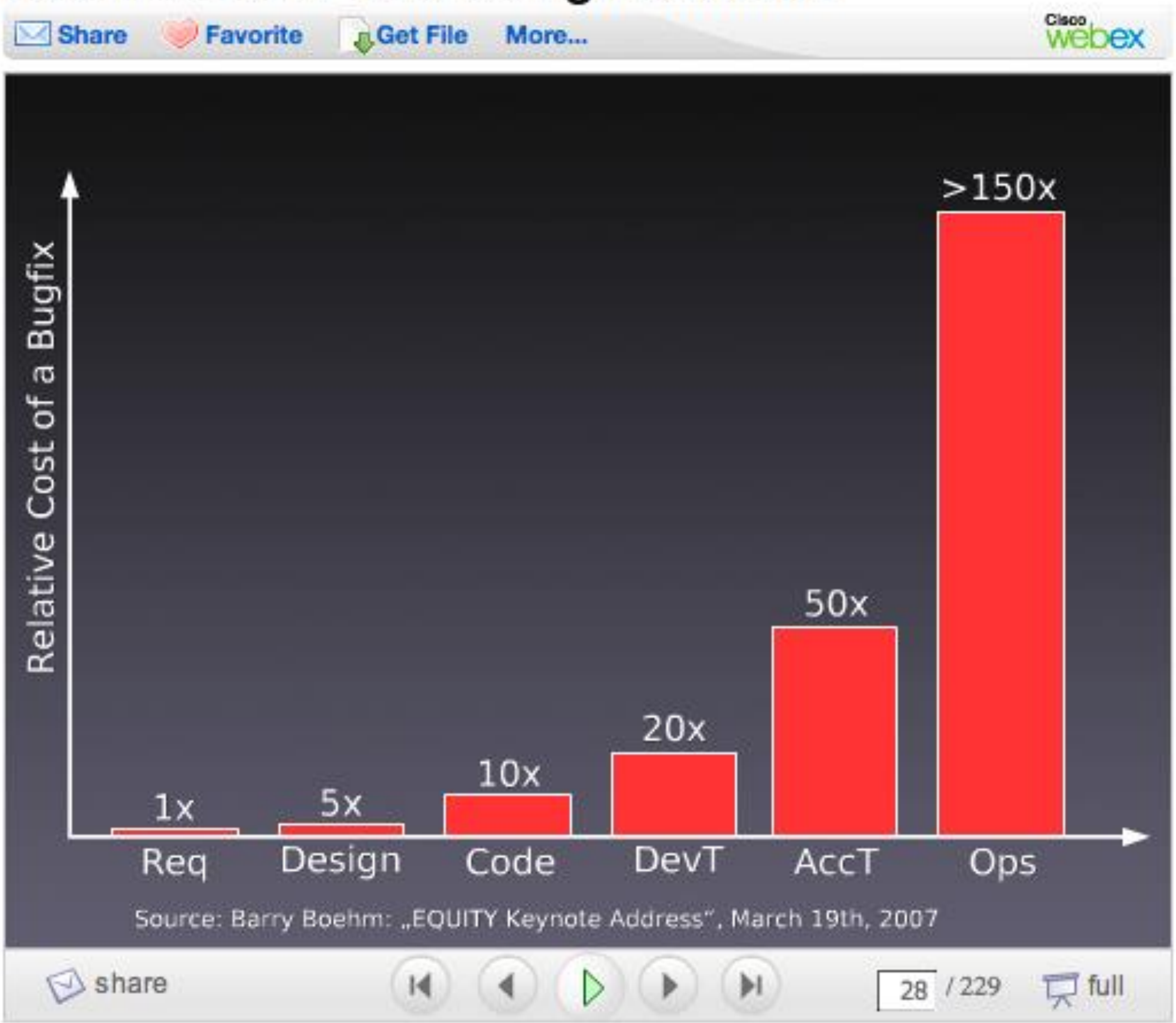


http://www.sw-engineering-candies.com/blog-1/rules-of-thumb-in-software-engineering



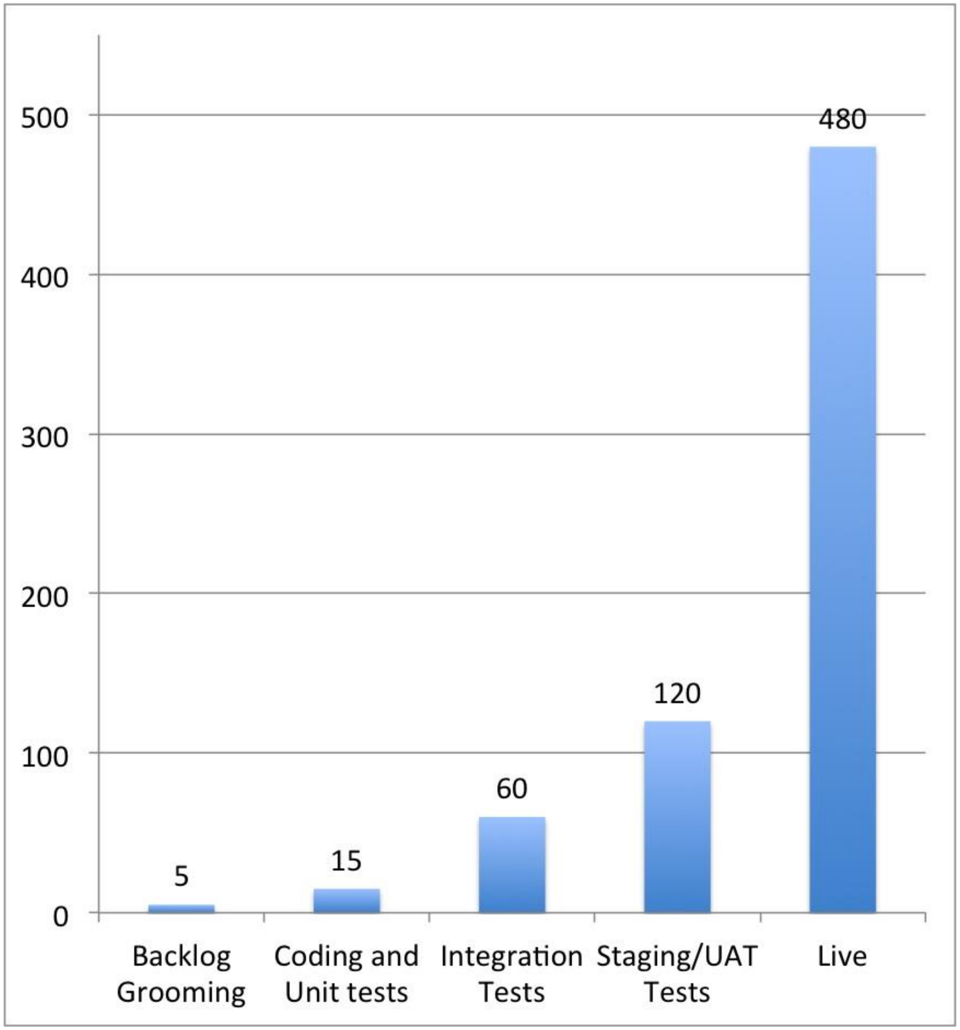
http://codedx.com/ide-integration-helps-developers-adopt-application-security-testing-tools/

Advanced OOP and Design Patterns

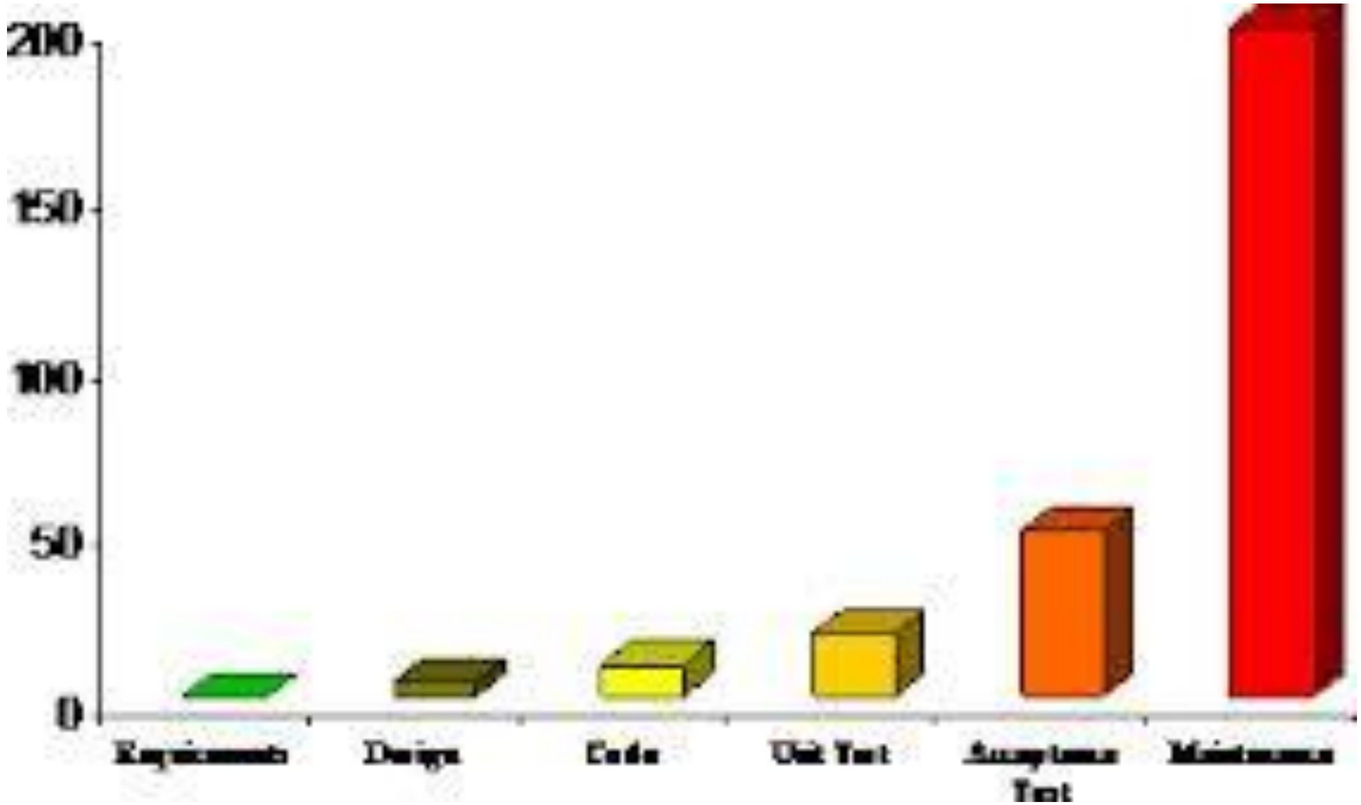


http://superwebdeveloper.com/2009/11/25/the-incredible-rate-of-diminishing-returns-of-fixing-software-bugs/

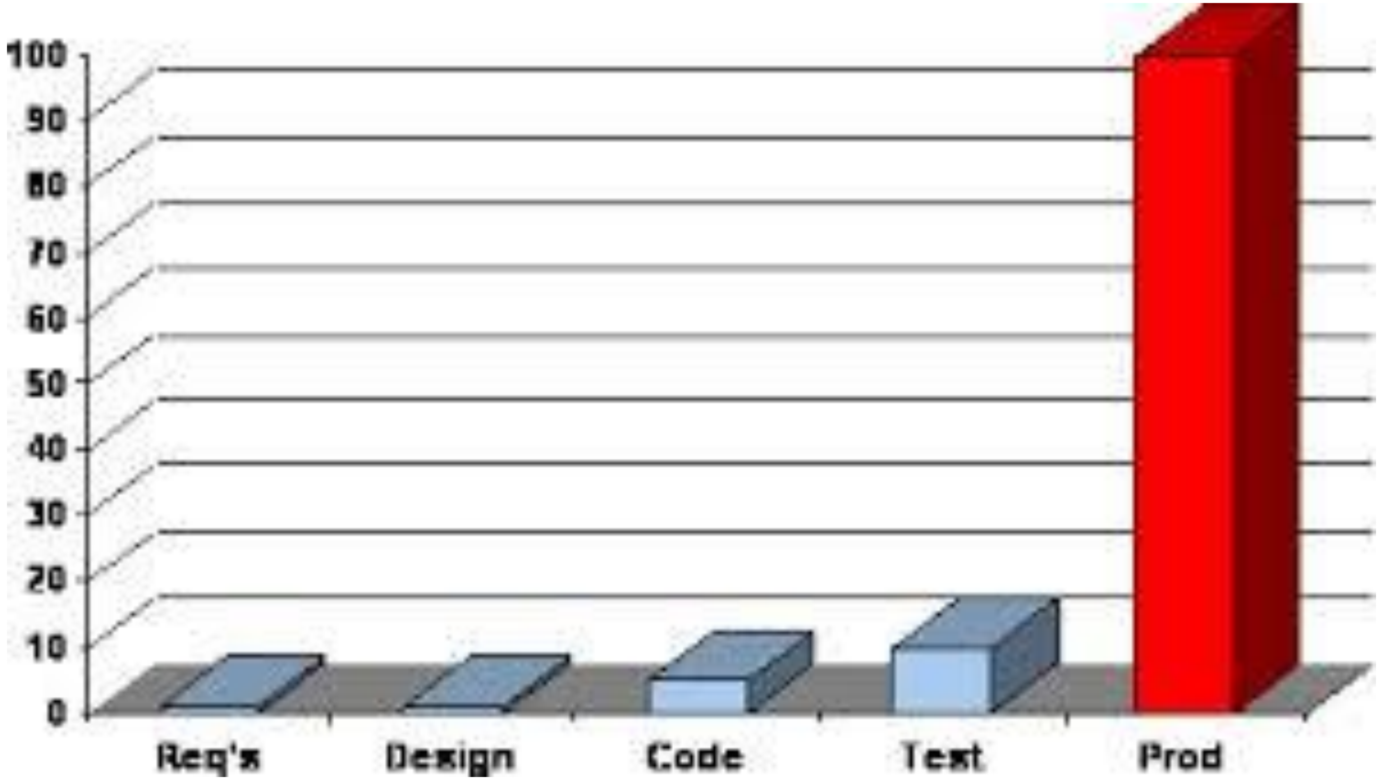
Minutes to fix



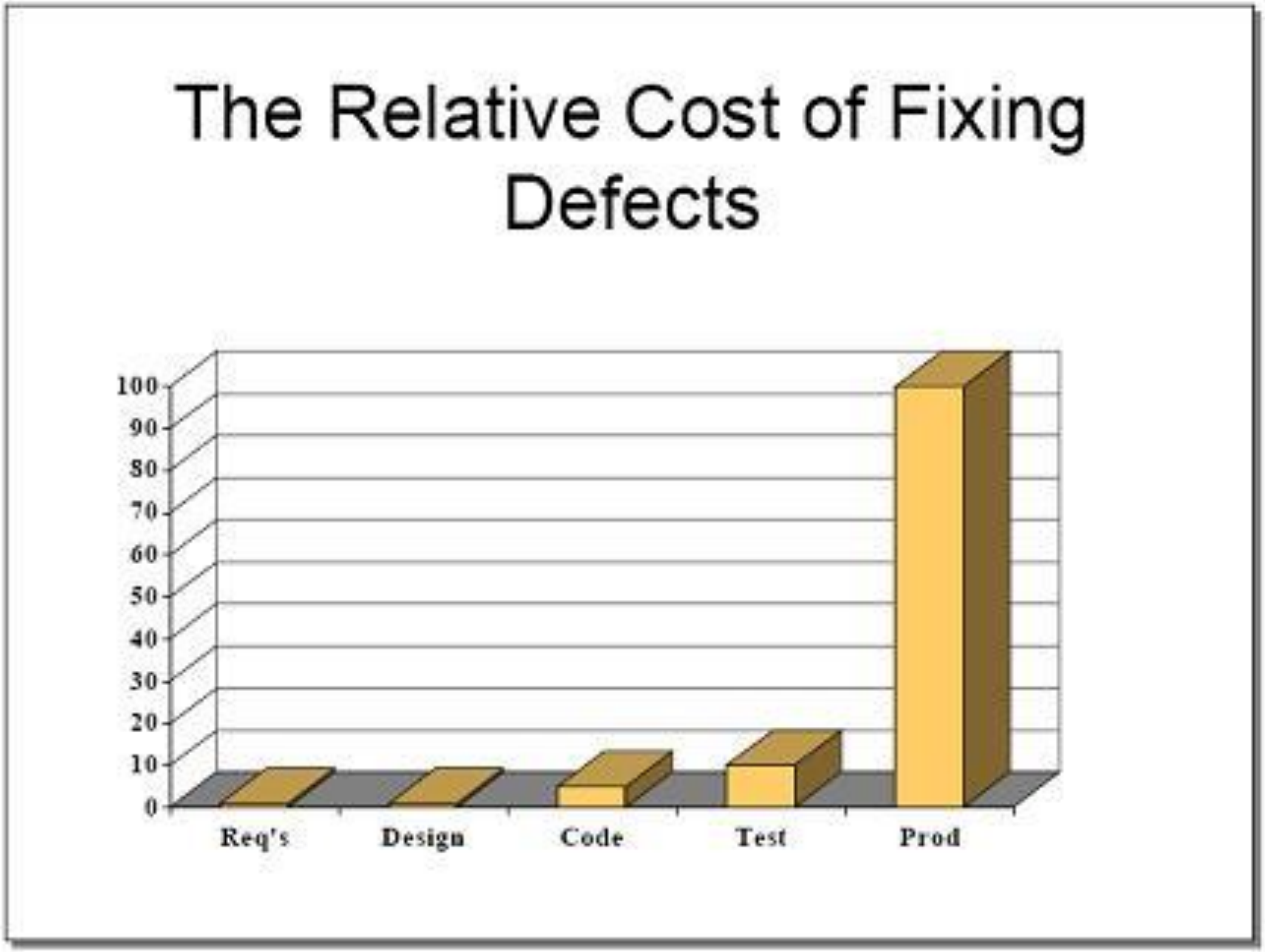
https://www.scrumalliance.org/community/articles/2013/january/quality-is-free



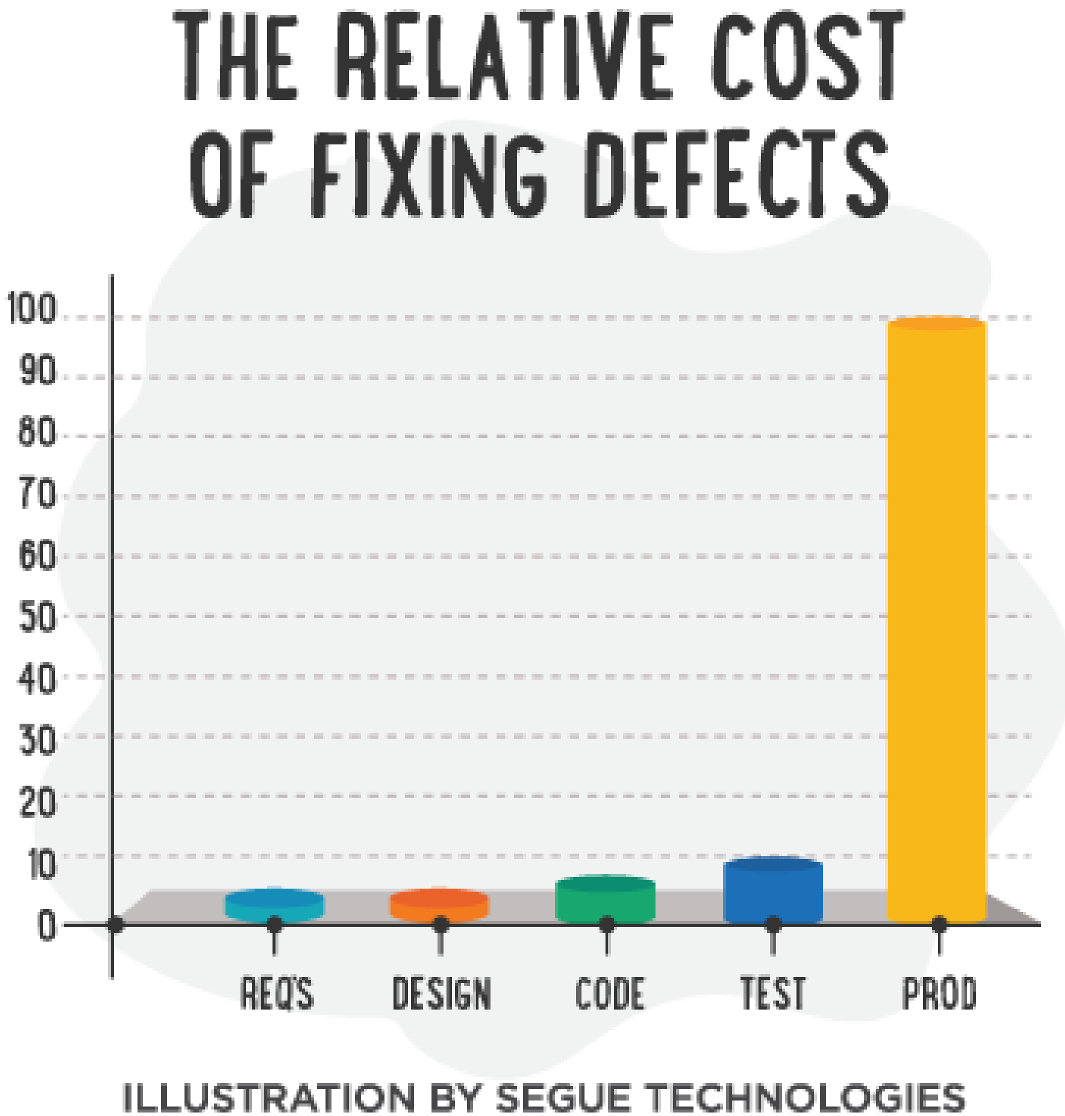
http://www.jucs.org/jucs_13_5/realising_the_benefits_of/jucs_13_5_0669_0678_hall.html



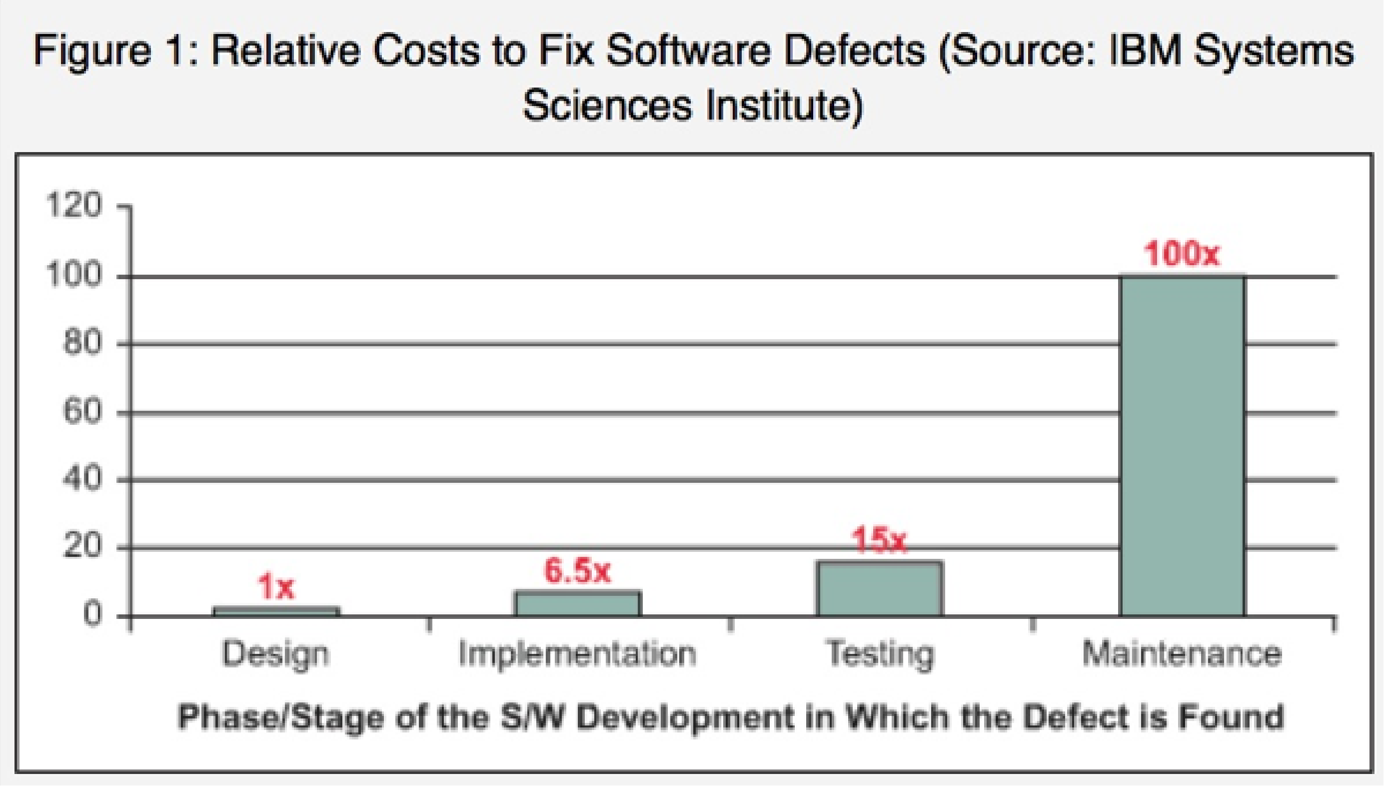
https://www.cloudreach.com/gb-en/2014/11/devsecops-aws-2/



http://jonkruger.com/blog/2008/11/20/the-relative-cost-of-fixing-defects/

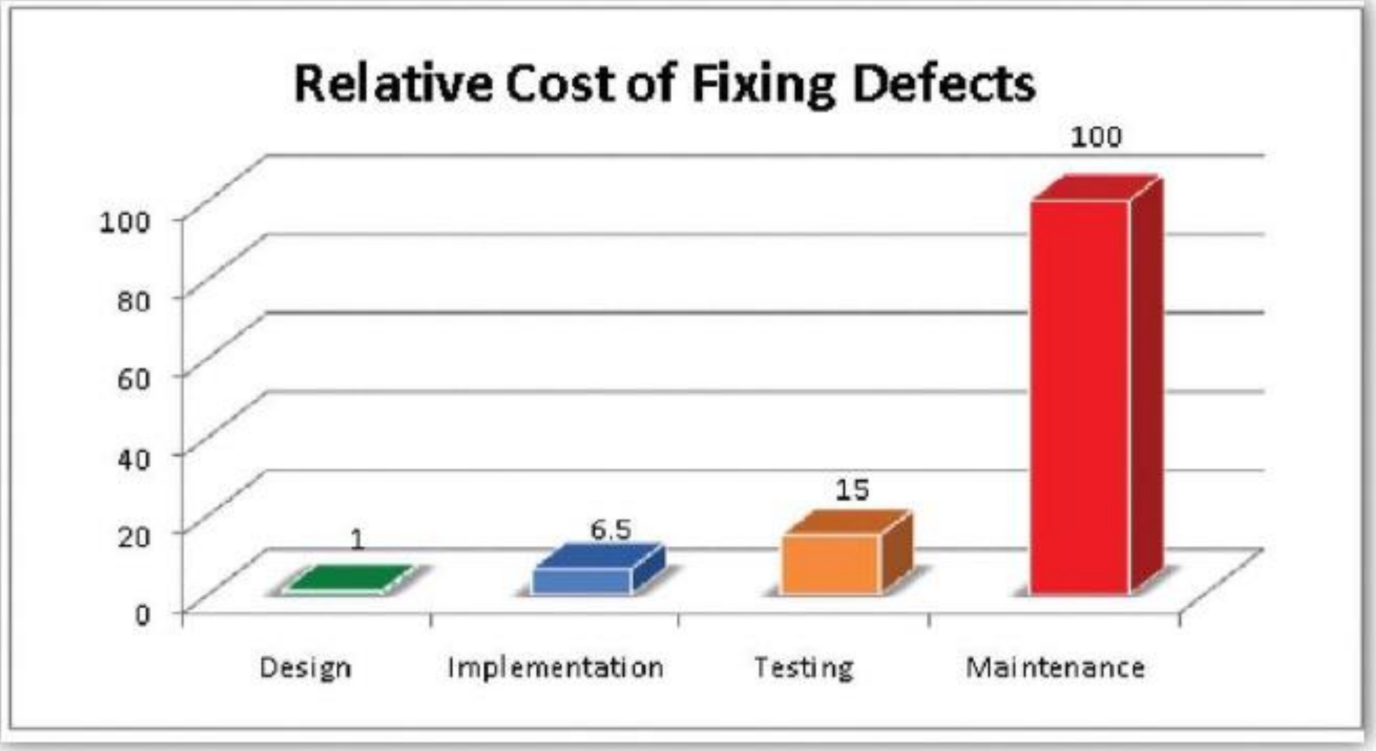


http://www.seguetech.com/blog/2014/09/05/rising-costs-defects-infographic

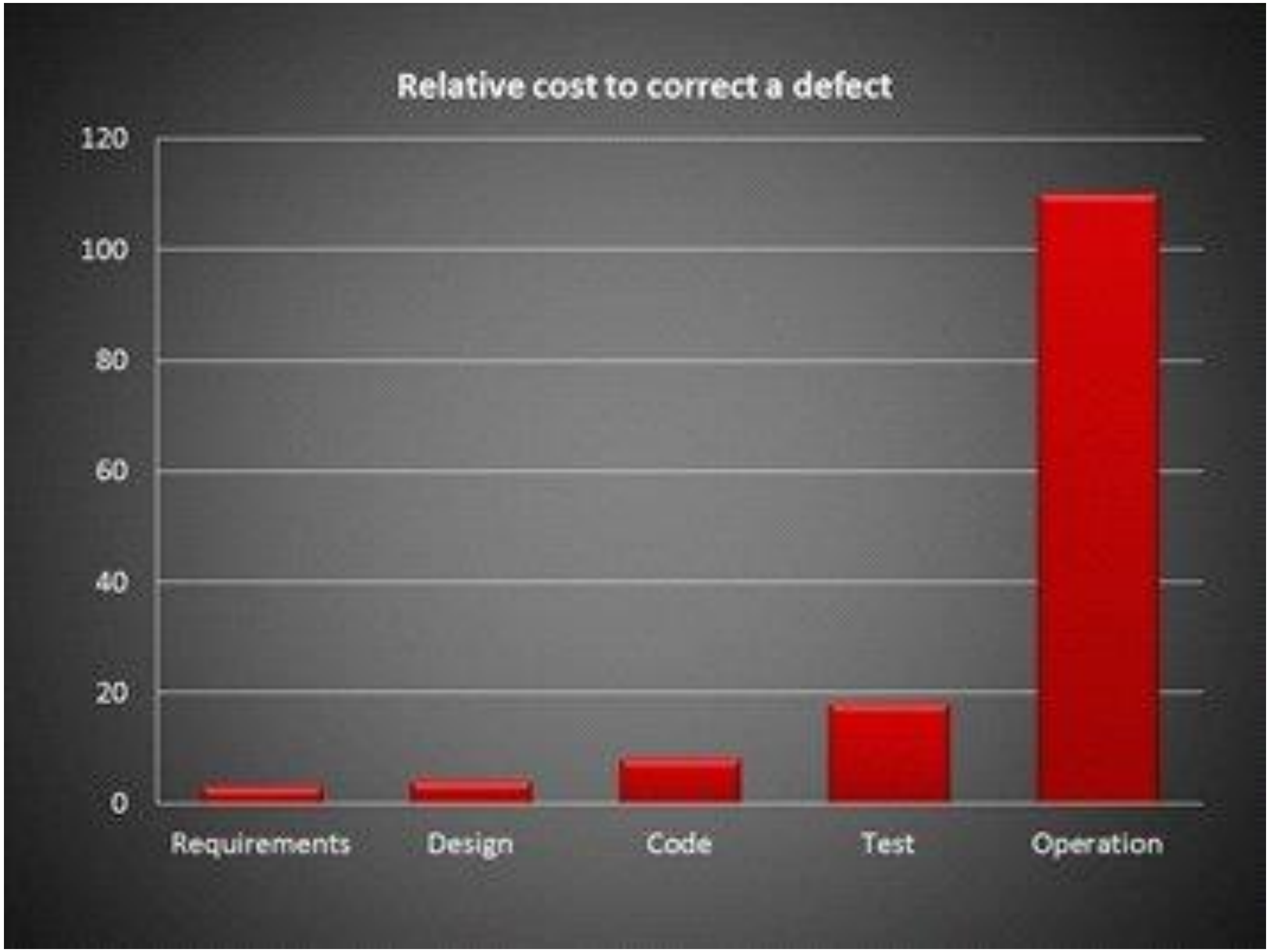


https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/enabling_devops_success_with_shift_left_continuous_testing?lang=en

Relative Cost of Fixing Defects

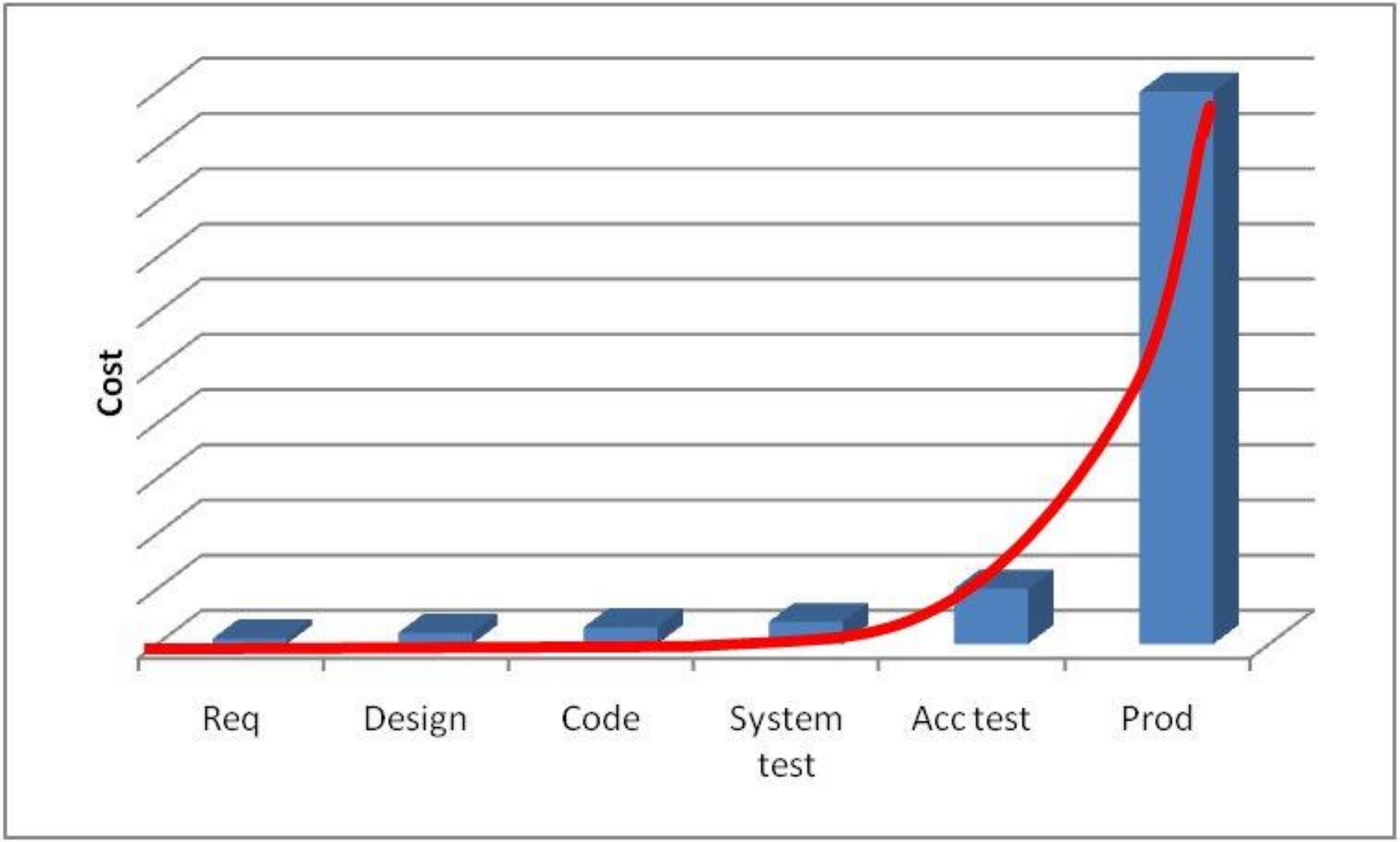


http://www.slideshare.net/drdawson/secure-software-development-life-cycle



http://www.a1qa.com/blog/test-model-and-requirement-management/

A breakthrough!



http://thesupertester.com/?p=123

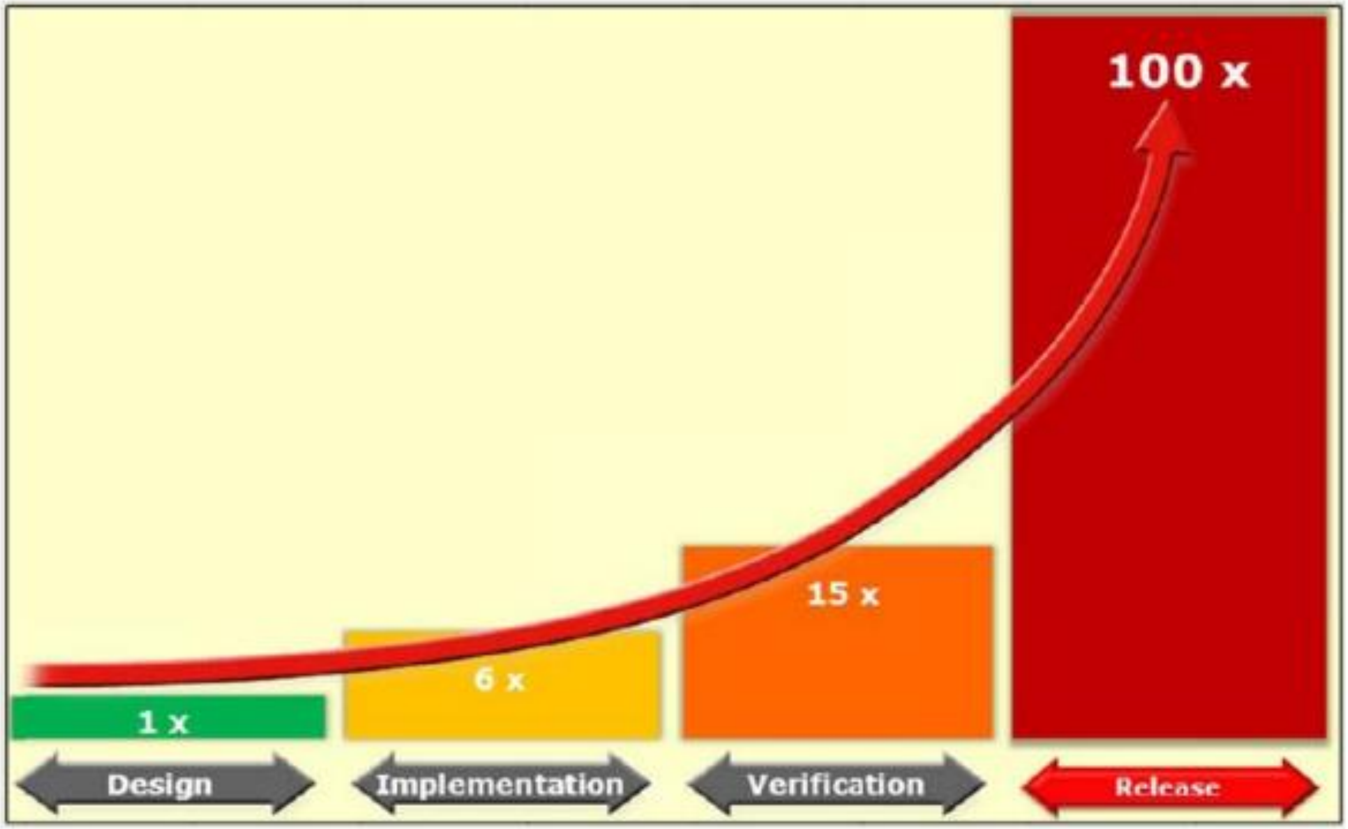
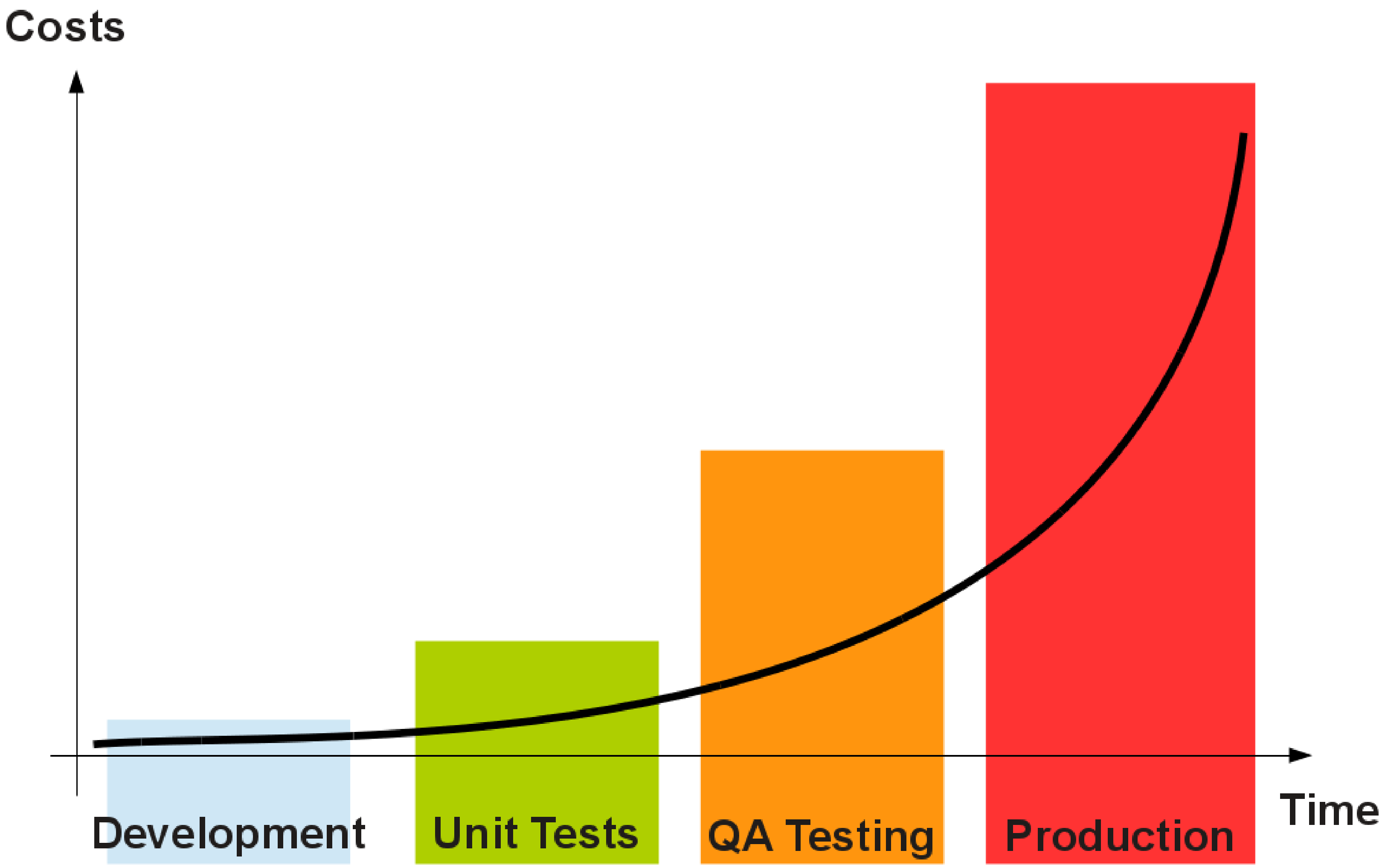
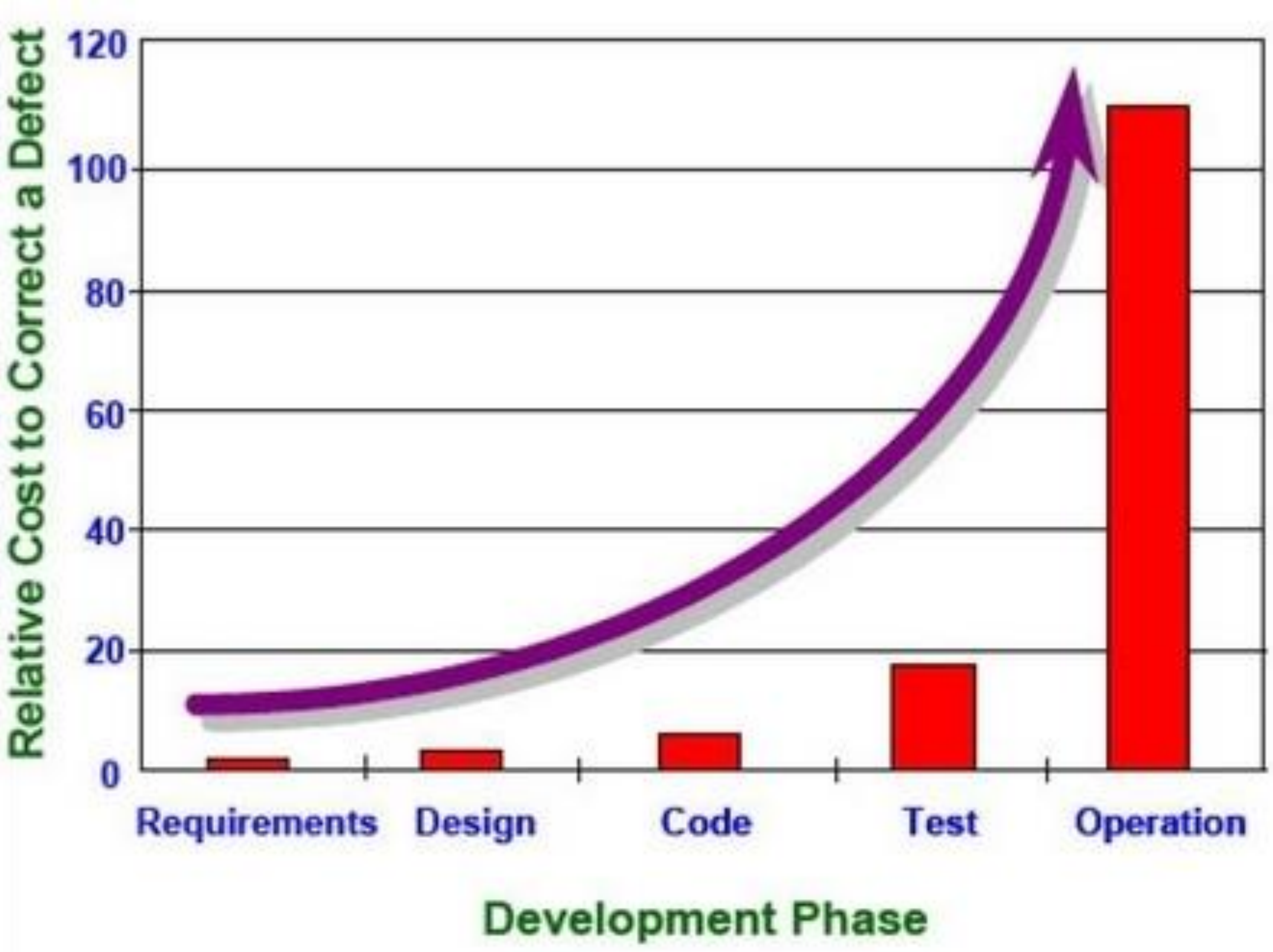


Figure 1: Cost of Bug Elimination in the Software Development Lifecycle [NIST 2002]

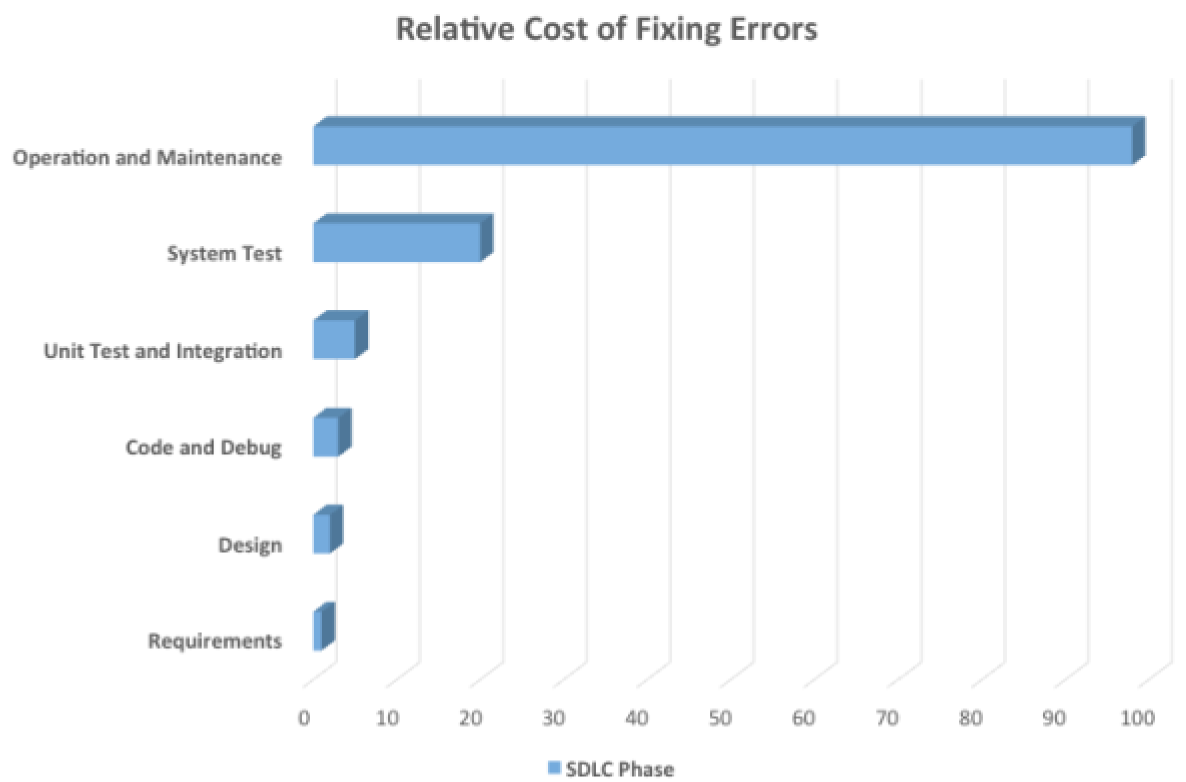
http://perfguy.blogspot.com/



http://blog.pdark.de/2012/07/21/software-development-costs-bugfixing/



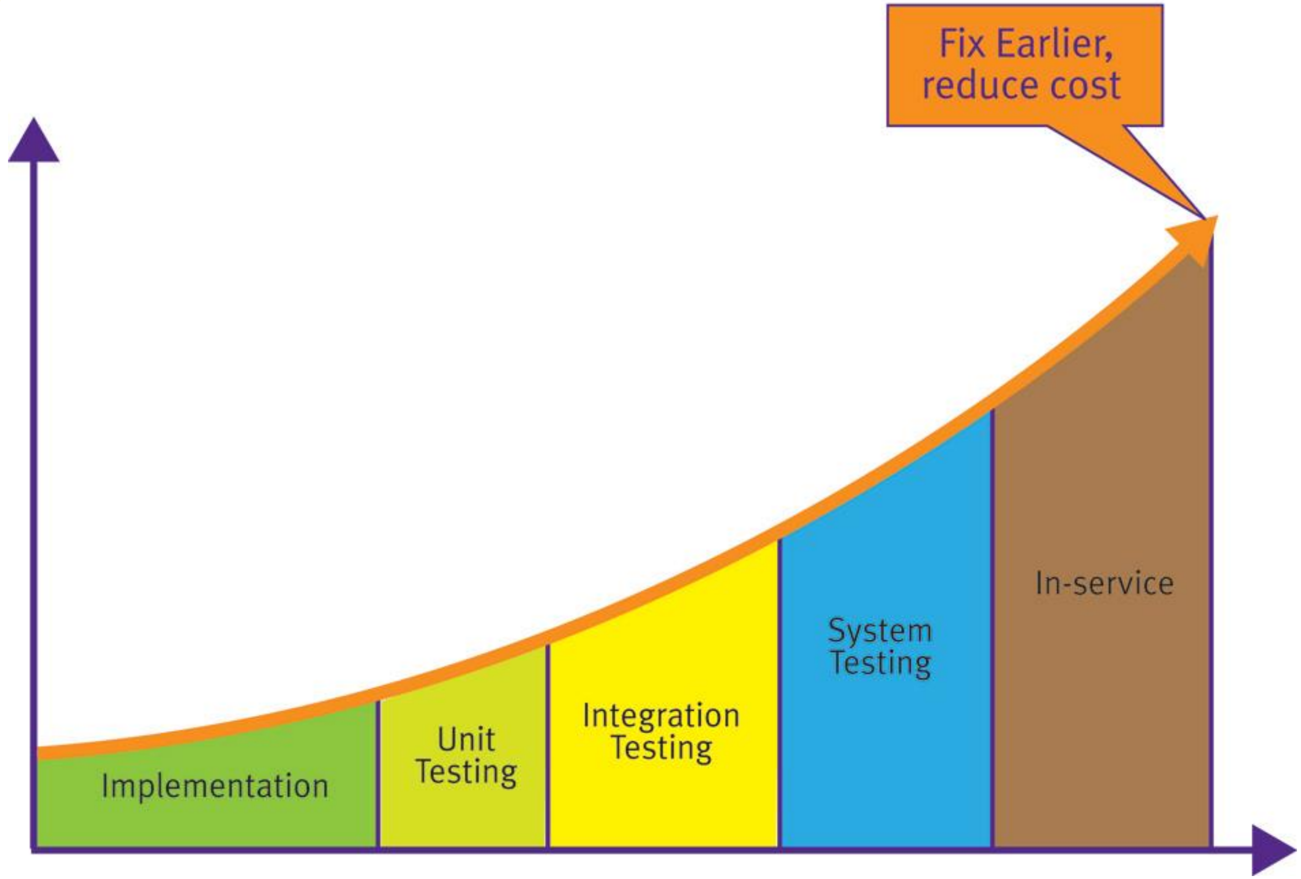
https://fcbqacorner.wordpress.com/



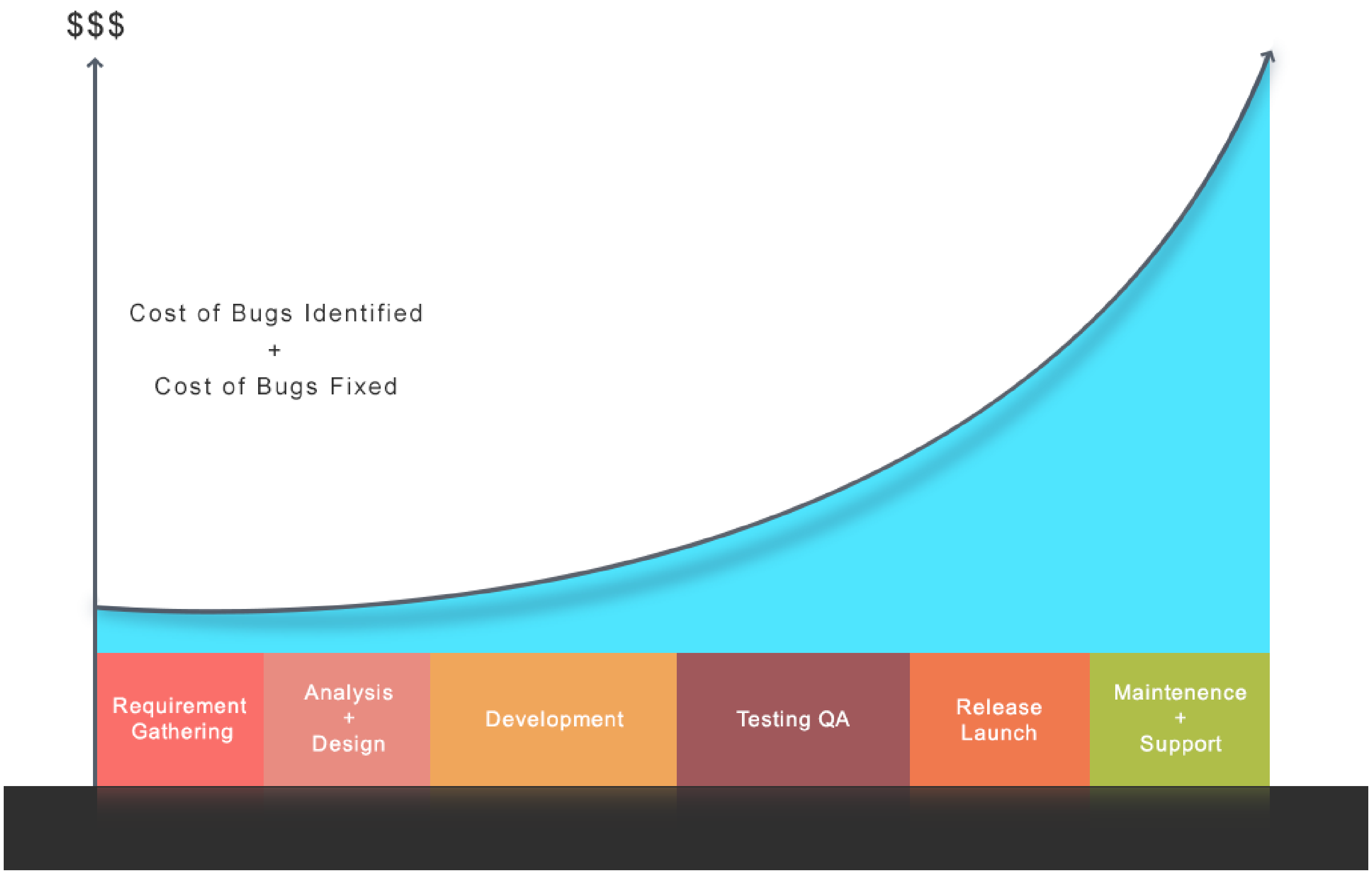
http://info.motio.com/Blog/bid/105868/Cognos-and-the-Cost-of-NOT-Testing-Your-BI

Another breakthrough!

The Cost of Defects

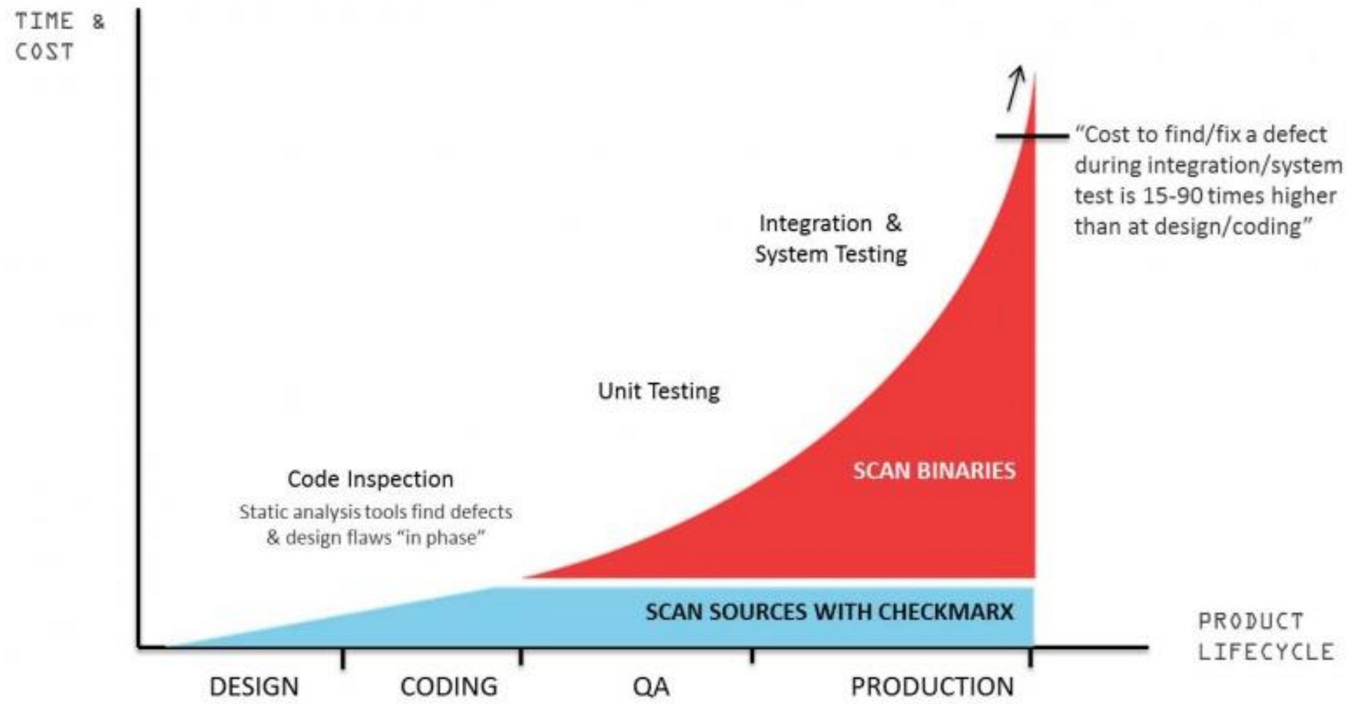


http://www.embeddedinsights.com/channels/2012/03/19/unit-test-tools-and-automatic-test-generation/



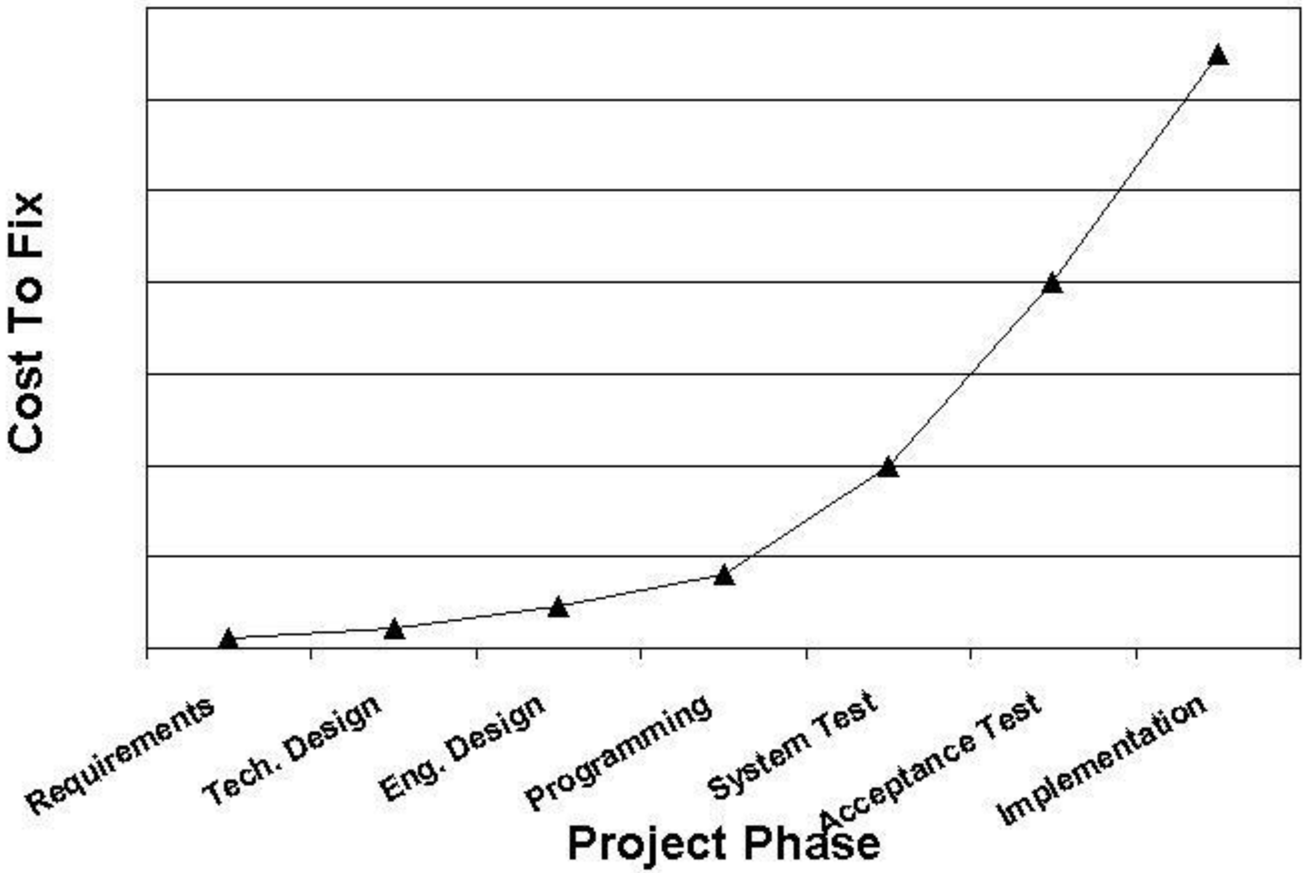
http://www.kualitatem.com/blog/economics-of-software-testing

FIXING SECURITY EARLY IN THE SDLC SAVES UP TO 90% OF THE REMEDIATION COSTS

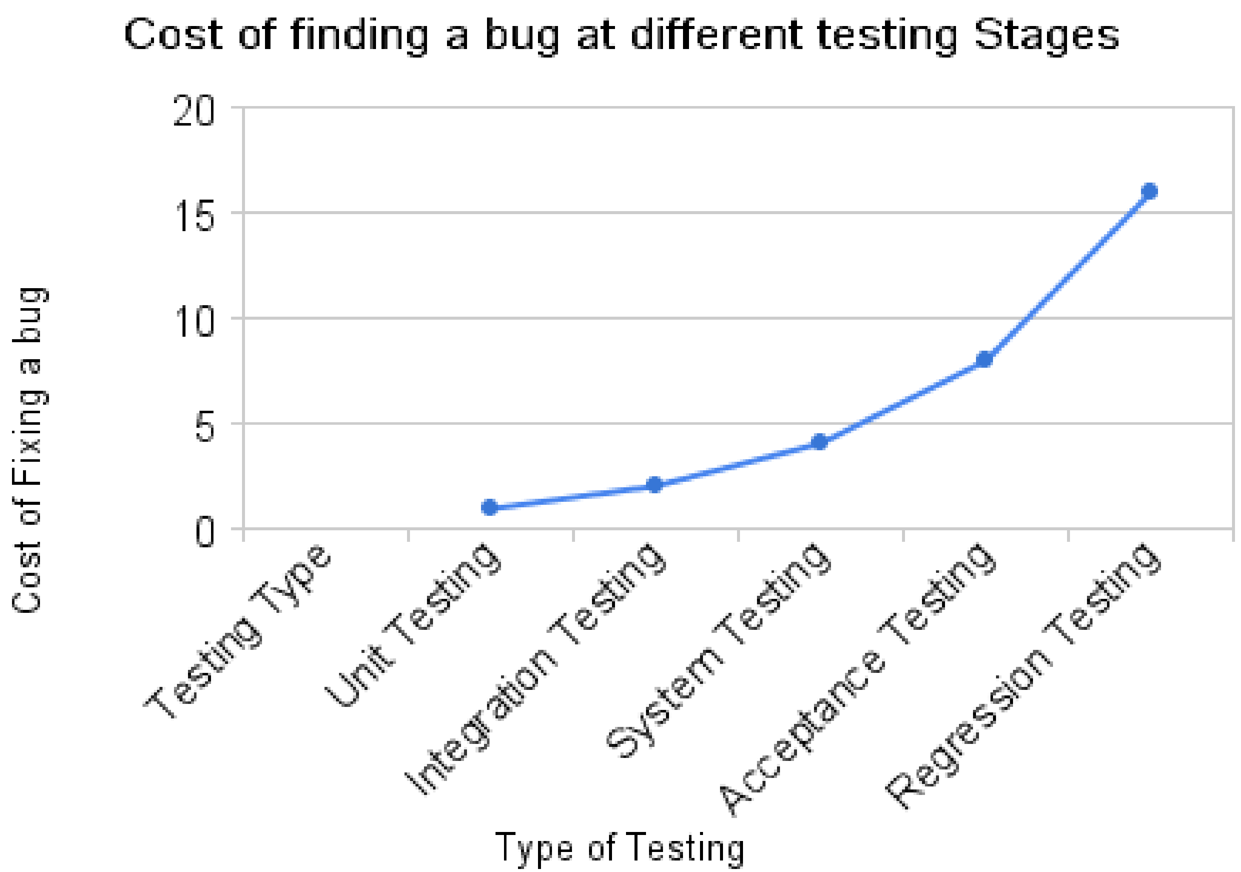


[Escalating cost to find and fix a defect or design flaw as it is discovered late in the Software Development Life Cycle (IDC, 2005)]

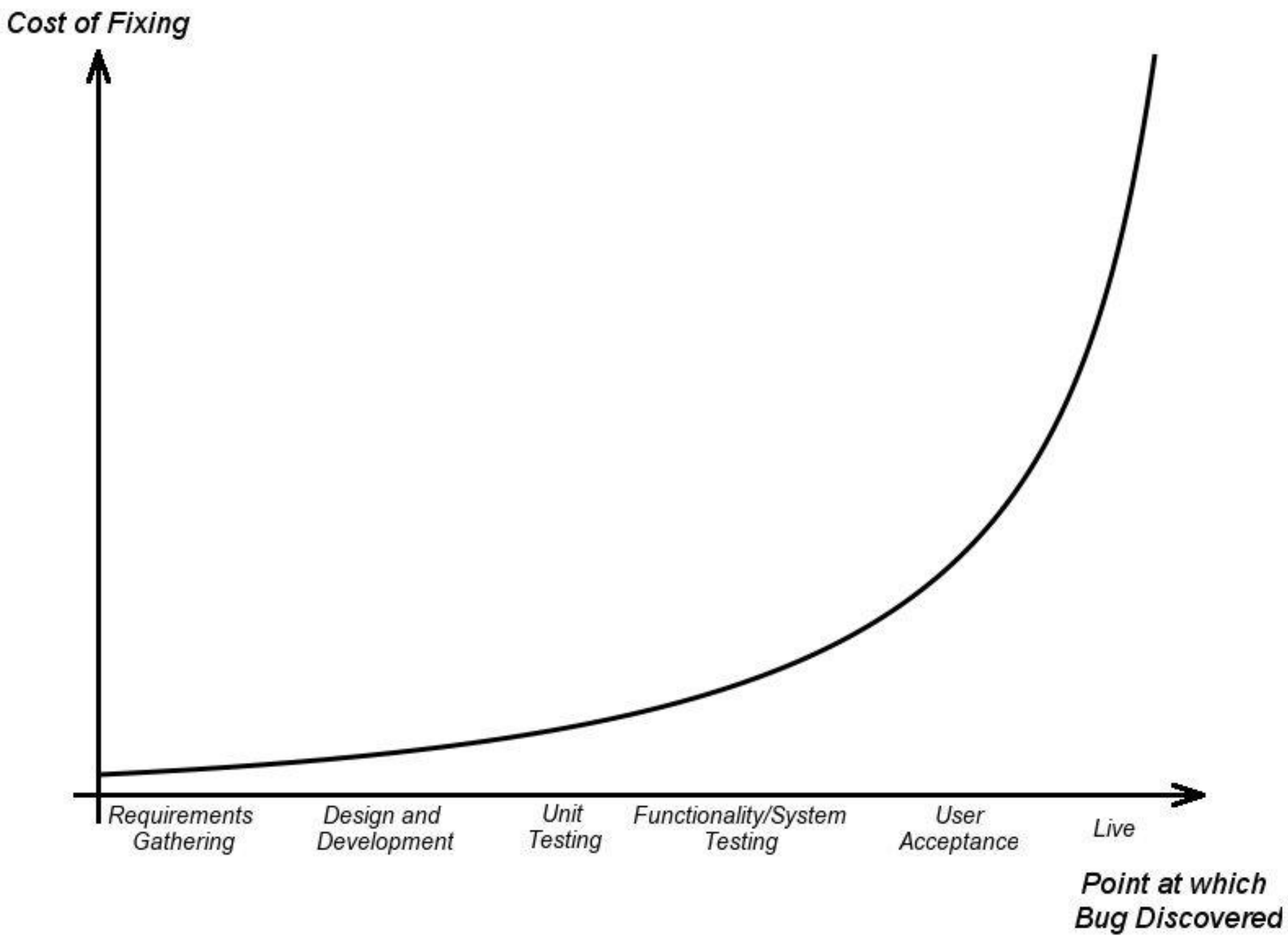
http://www.cert2connect.com/



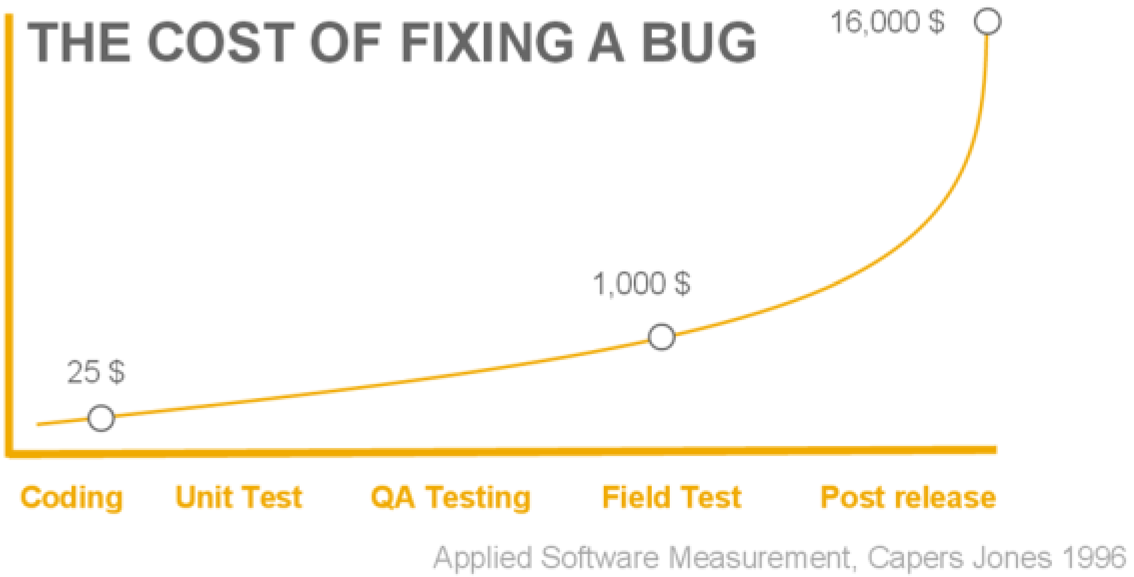
http://sqa.fyicenter.com/FAQ/Why-Bugs-in-Software/Cost_to_find_bugs.html



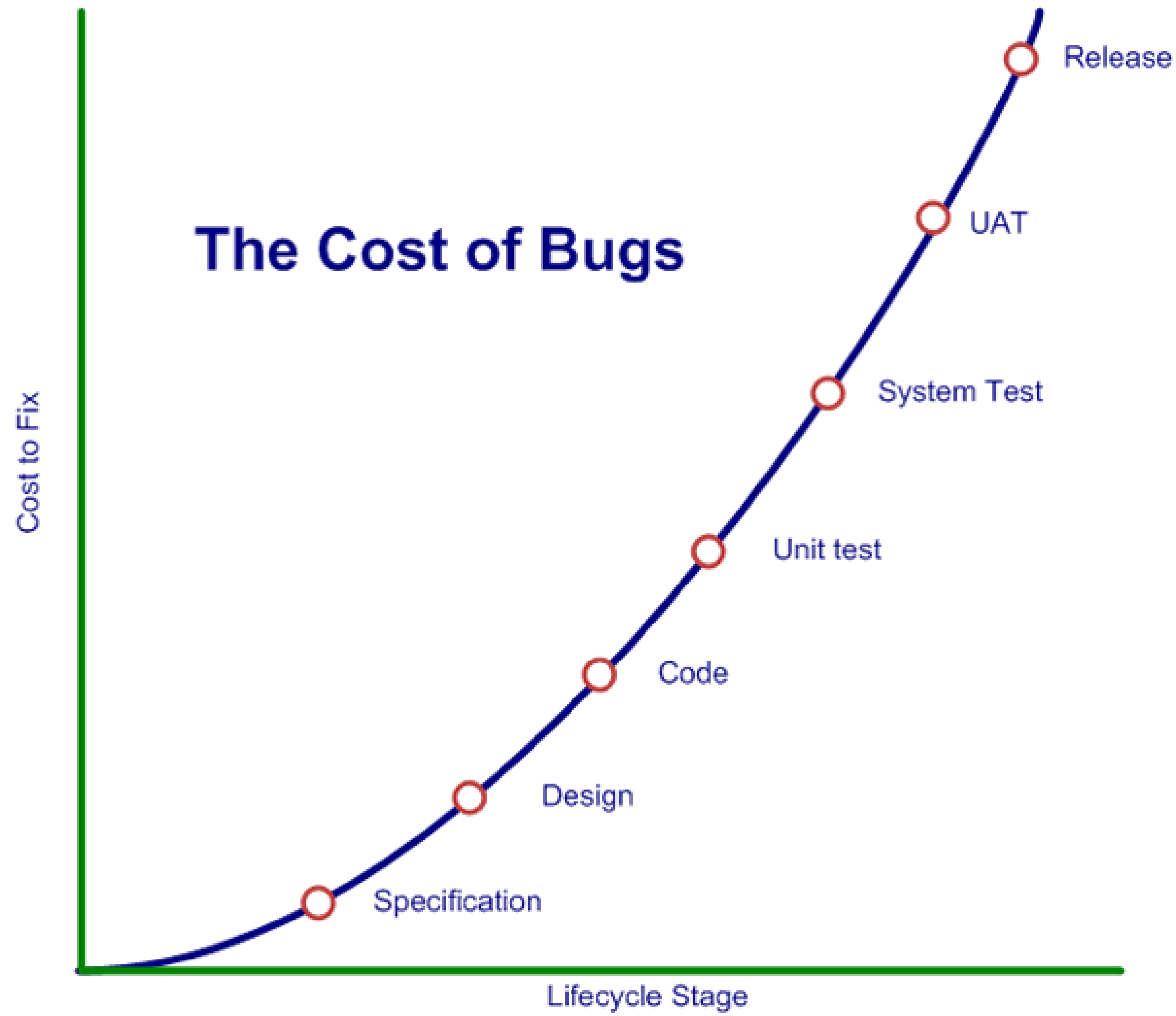
http://www.theautomatedtester.co.uk/blog/2008.htm



http://habrahabr.ru/post/206294/

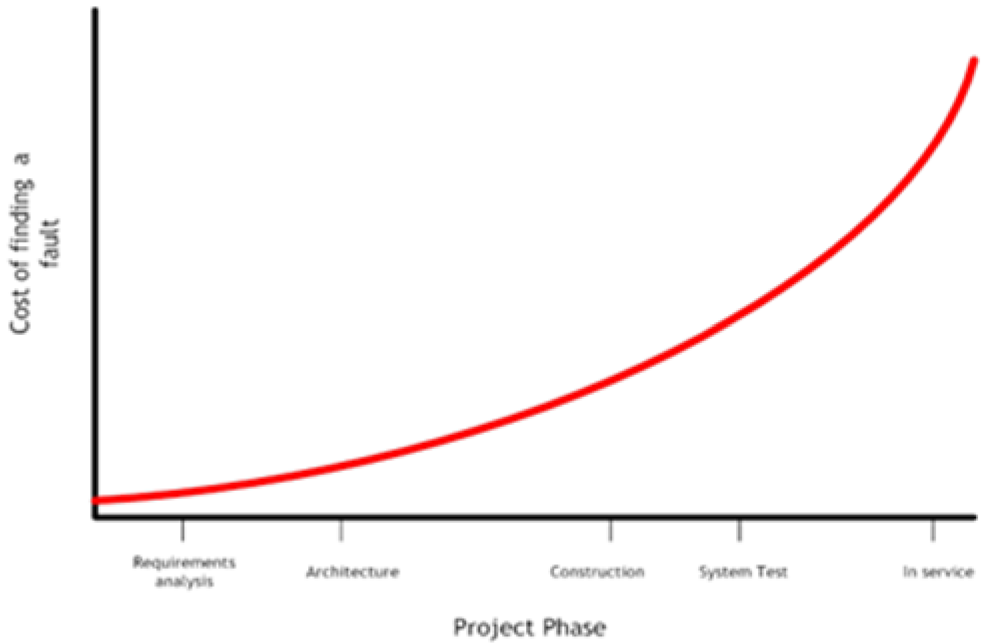


http://blog.revolunet.com/

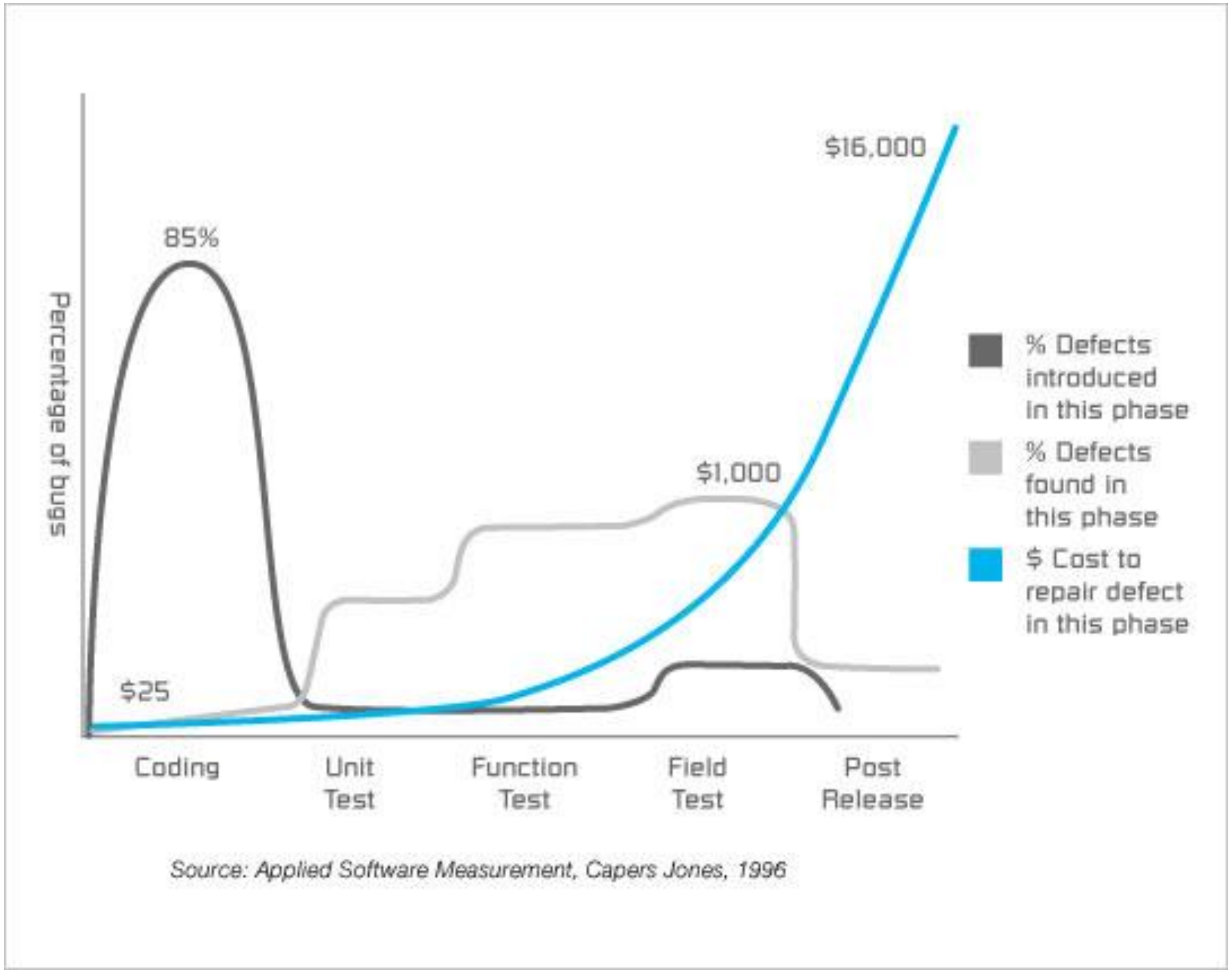


http://www.artigonal.com/software-artigos/introducao-a-testes-automatizados-7282245.html

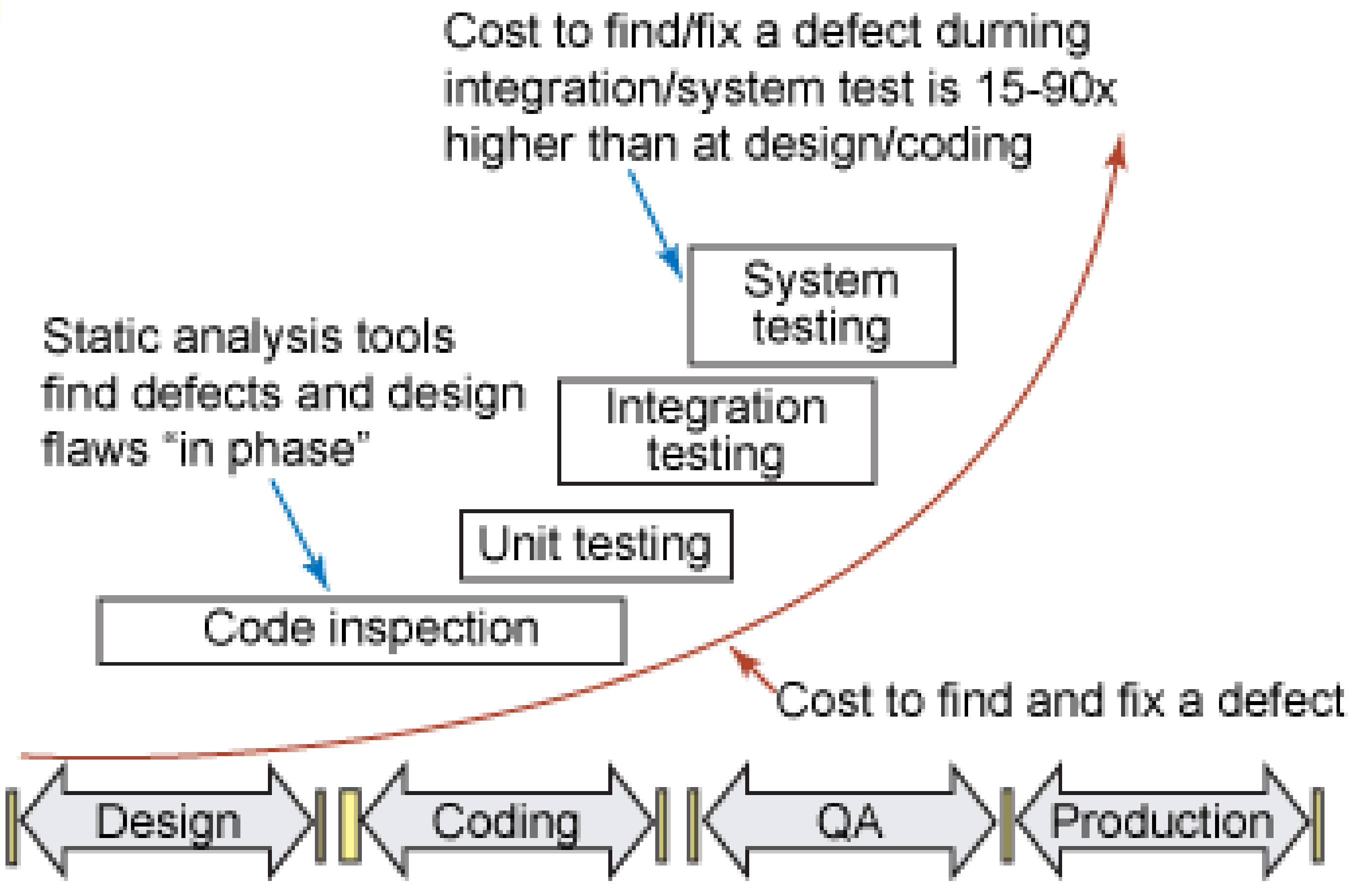
http://lighthousetechnologies.com/blog/software-testing-bug-y-hunter



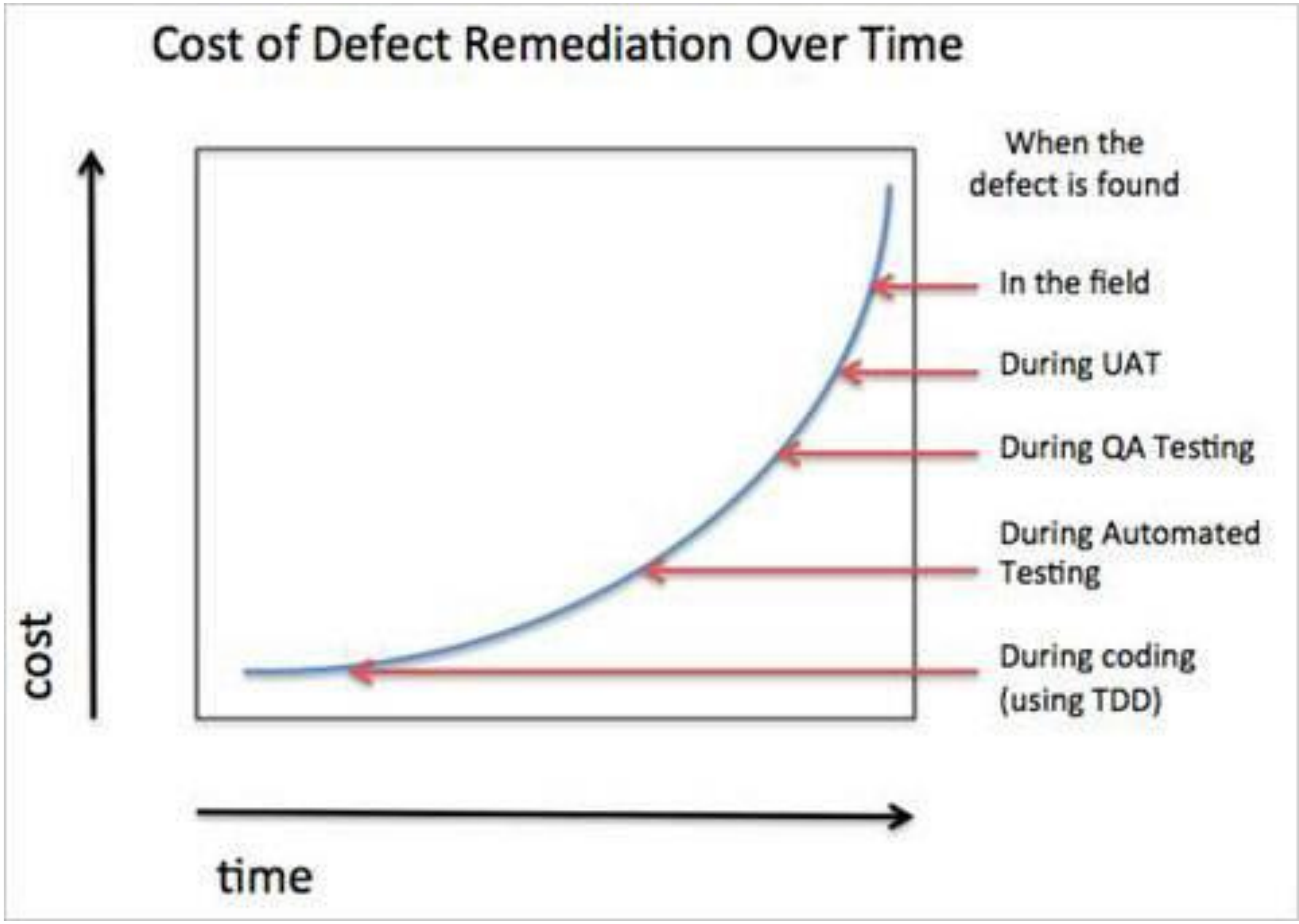
https://blog.feabhas.com/tag/quality/



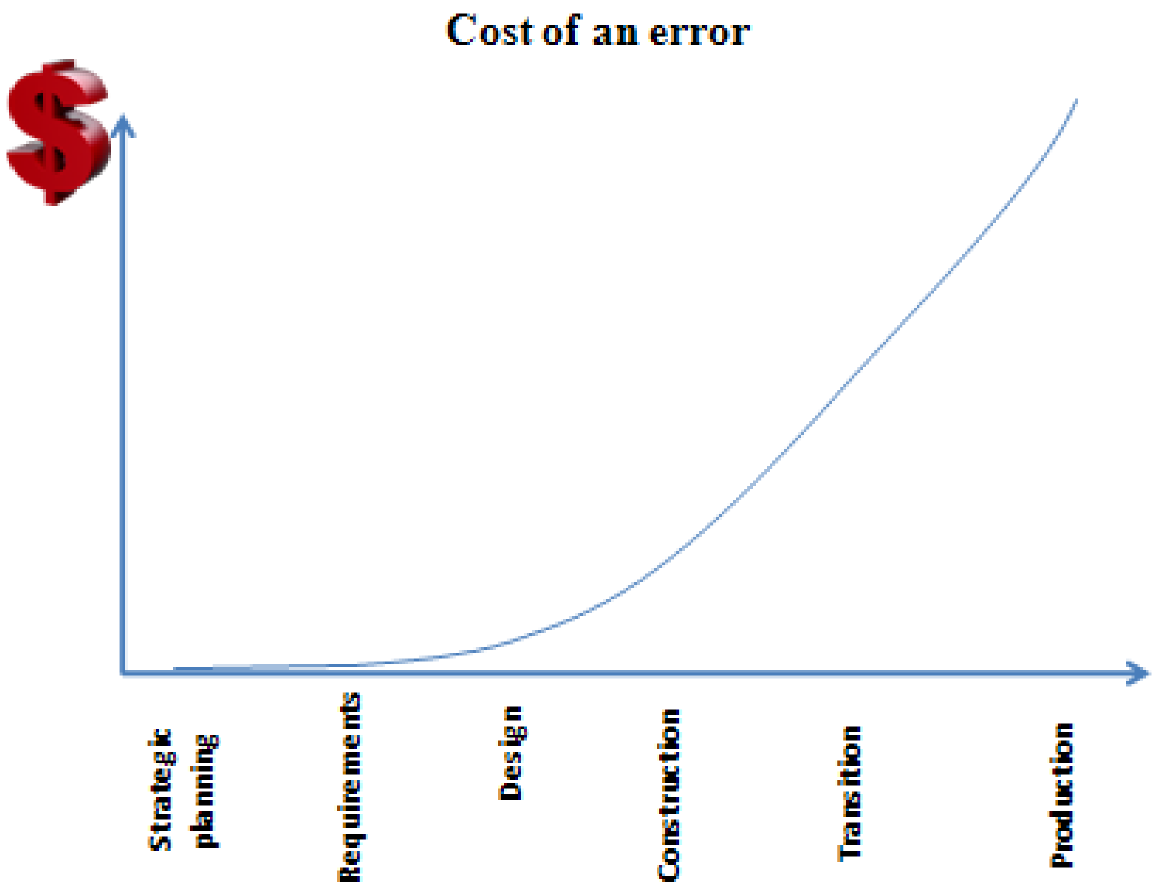
https://utbrudd.bouvet.no/2012/03/09/the-vicious-release-circle/



http://www.ibm.com/developerworks/rational/library/08/0429_gutz1/



http://searchsoftwarequality.techtarget.com/tip/Continuous-integration-Quality-from-the-start-with-automated-regression



https://enectoux.wordpress.com/tag/business-architecture/

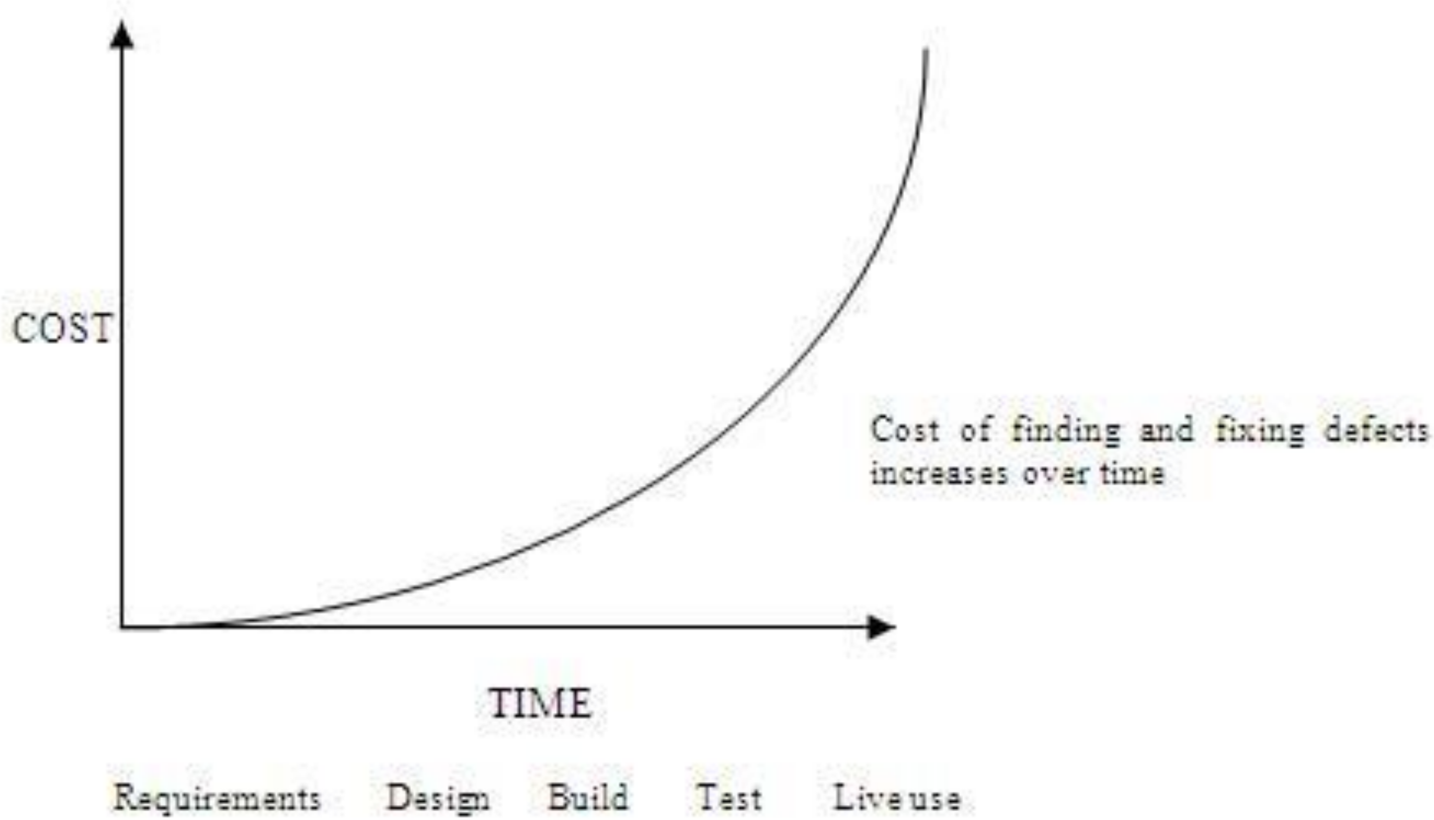
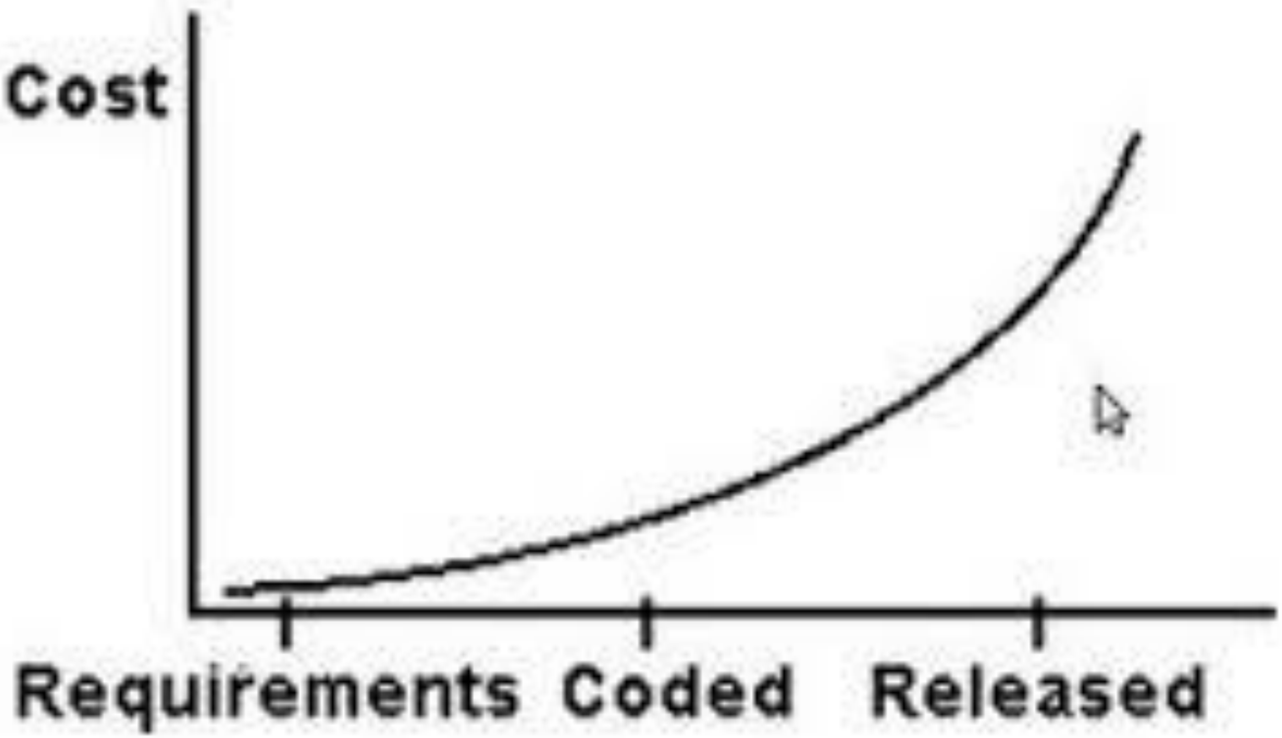
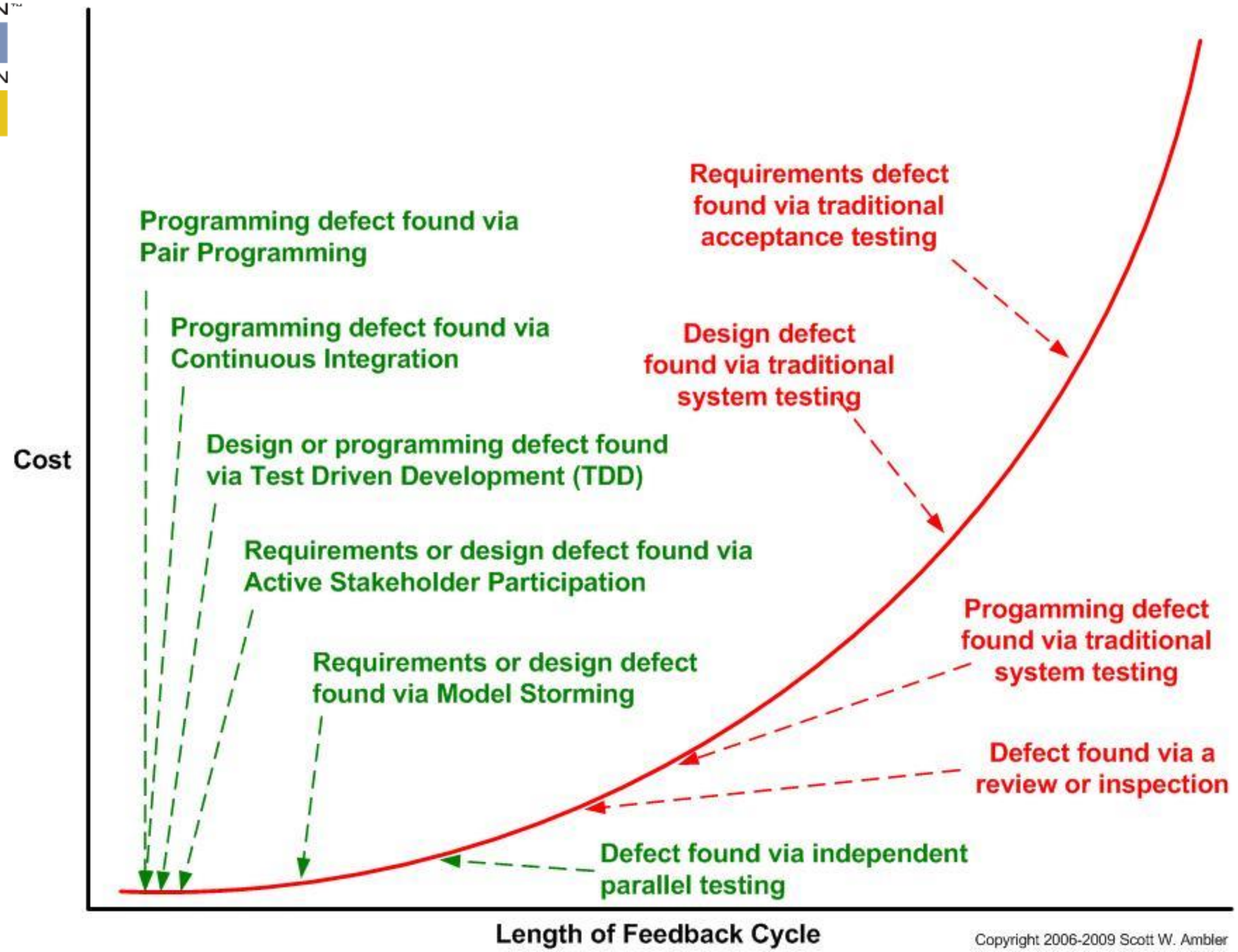


FIGURE 1.2

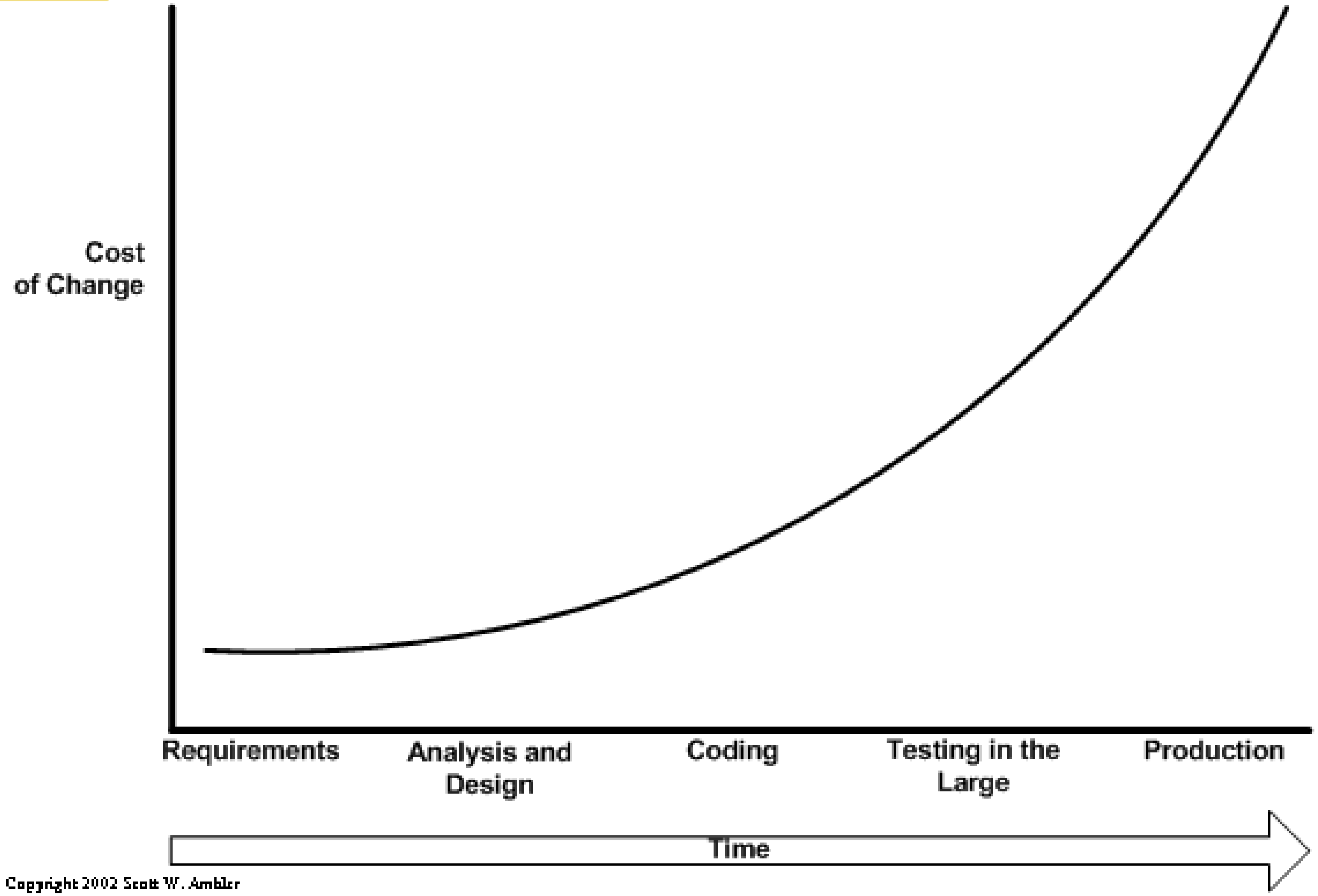
http://istqbexamcertification.com/what-is-the-cost-of-defects-in-software-testing/



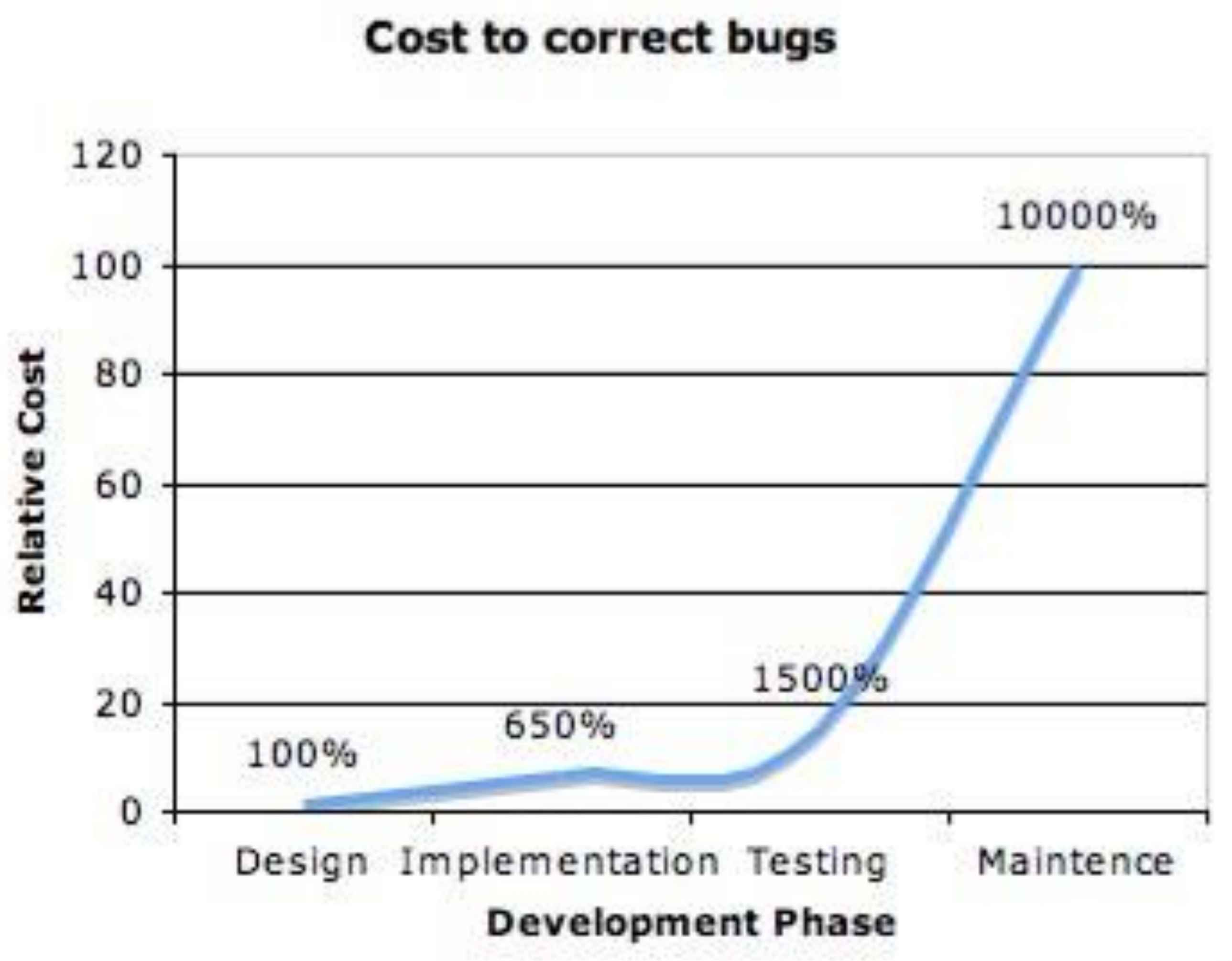
http://naree9-testing.blogspot.com/2009/07/bug-cost.html



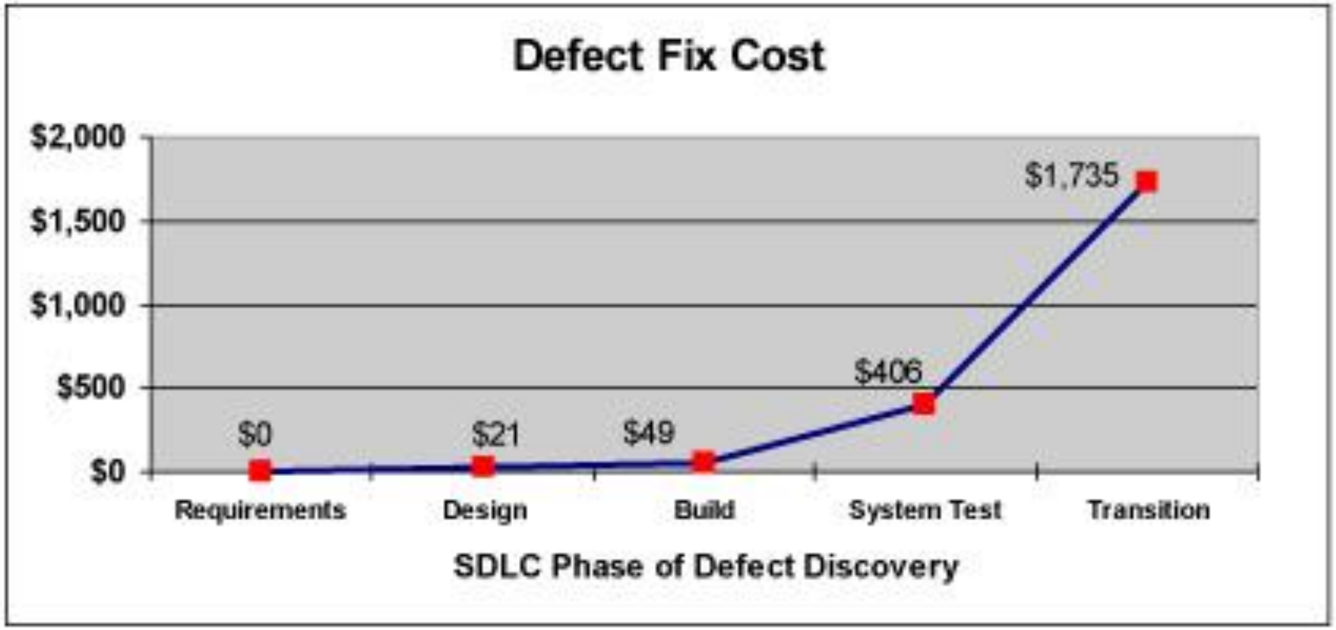
http://www.agilemodeling.com/essays/costOfChange.htm



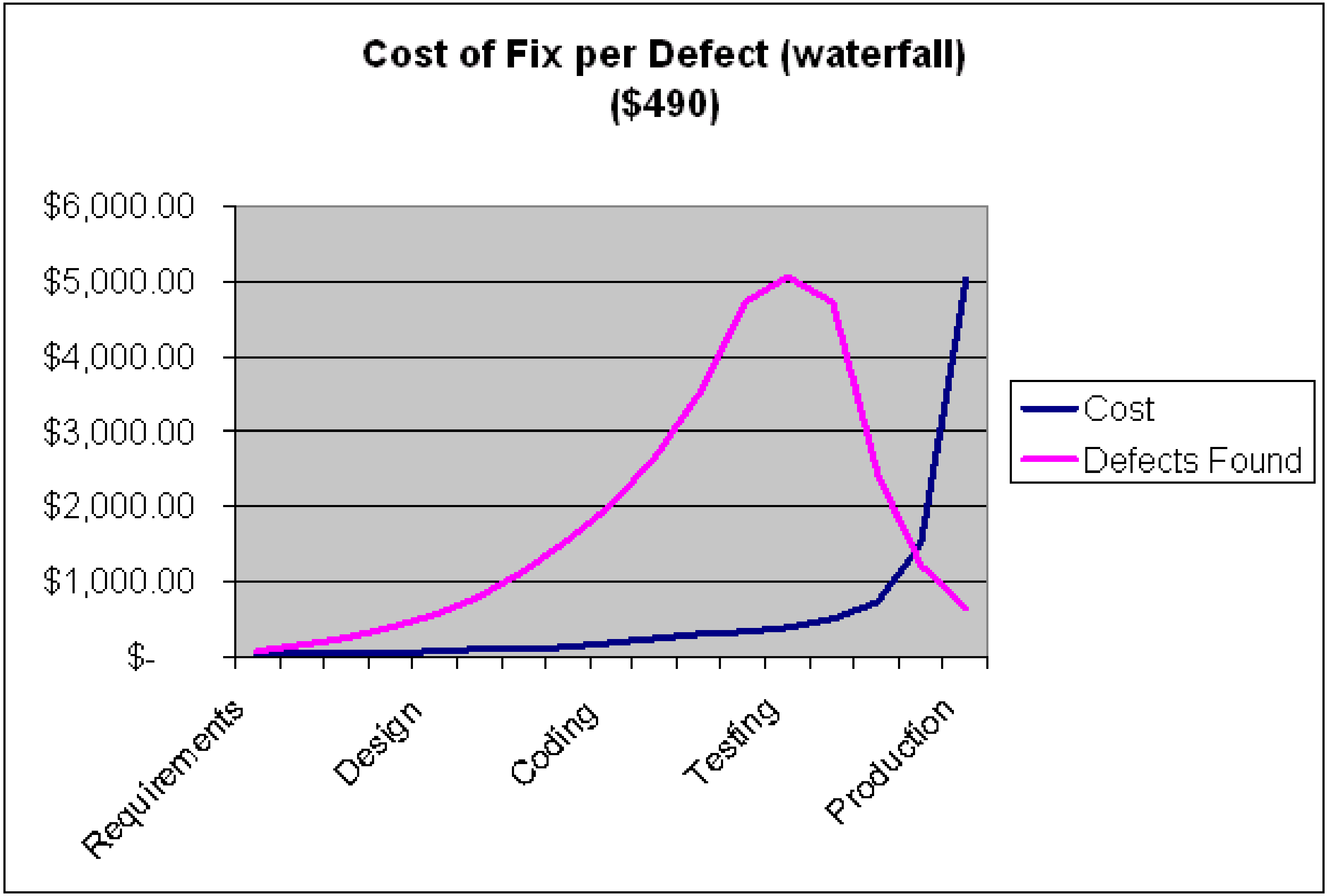
http://www.agilemodeling.com/essays/costOfChange.htm



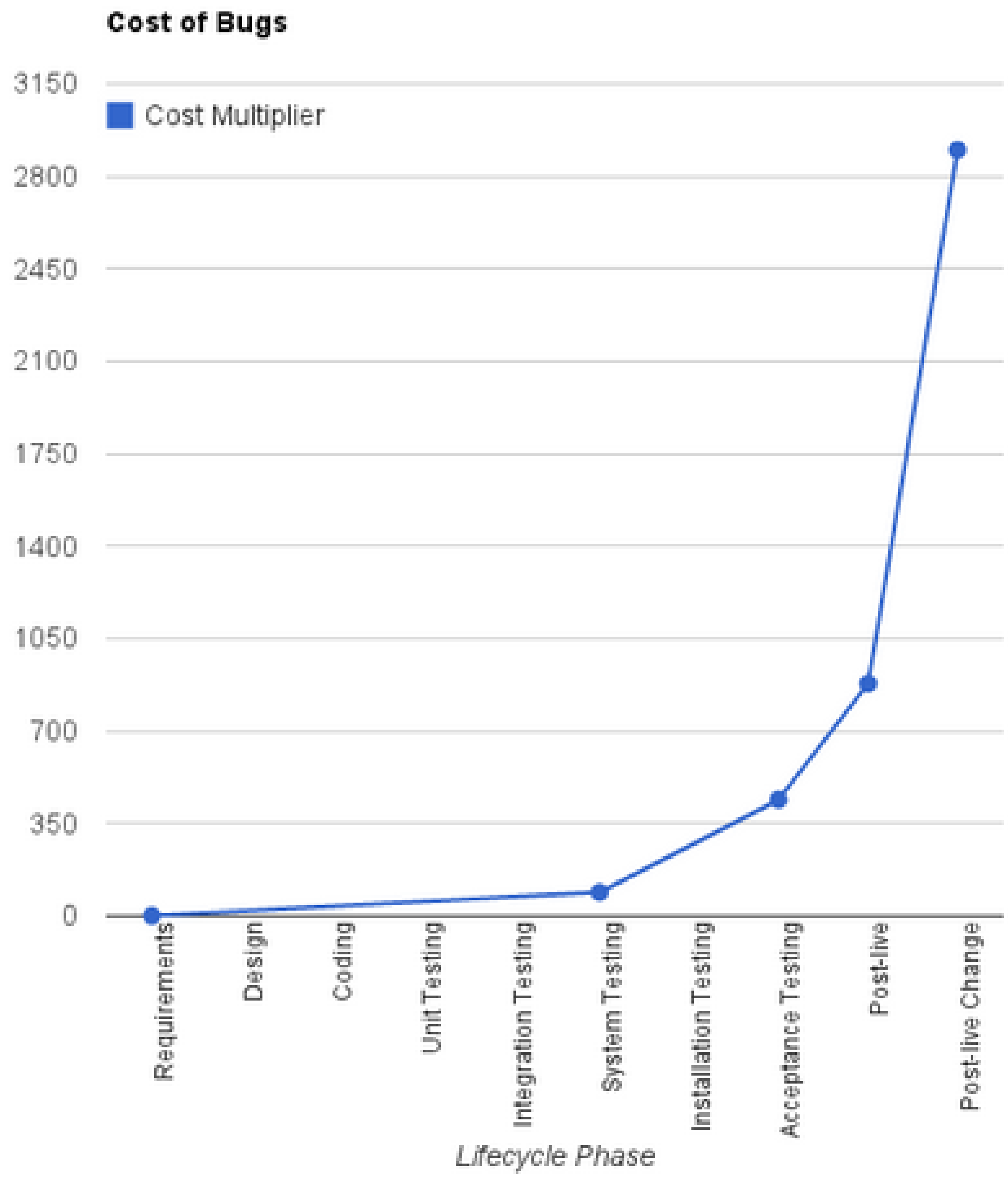
http://www.trustiv.co.uk/2014/08/what-performance-test-tools-are-being-used-%E2%80%98shift-left%E2%80%99



http://www.softwaretestpro.com/Item/5584/The-Job-of-Testing/Testing-Process

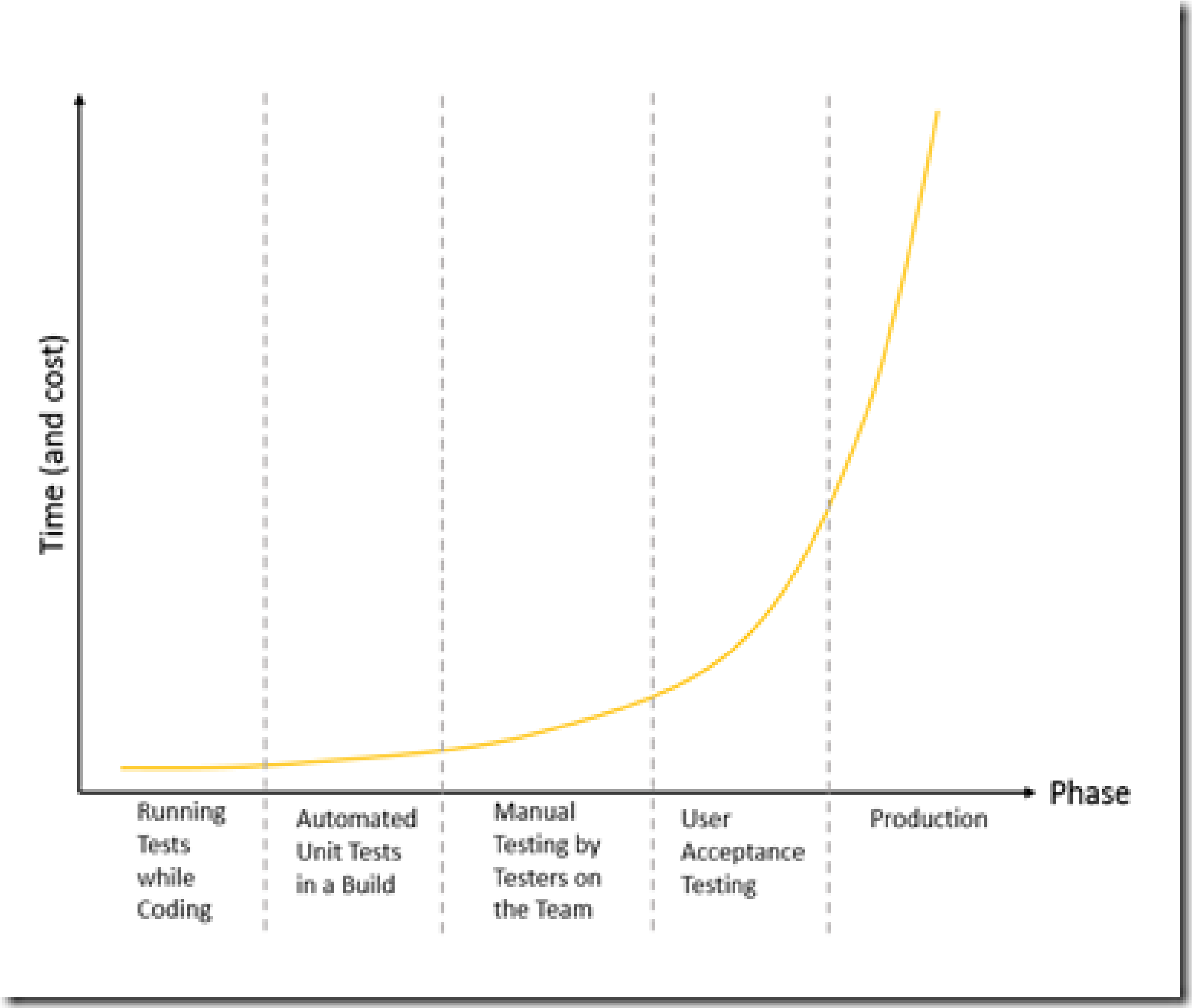


http://www.solutionsiq.com/the-math-behind-agile-and-automation/

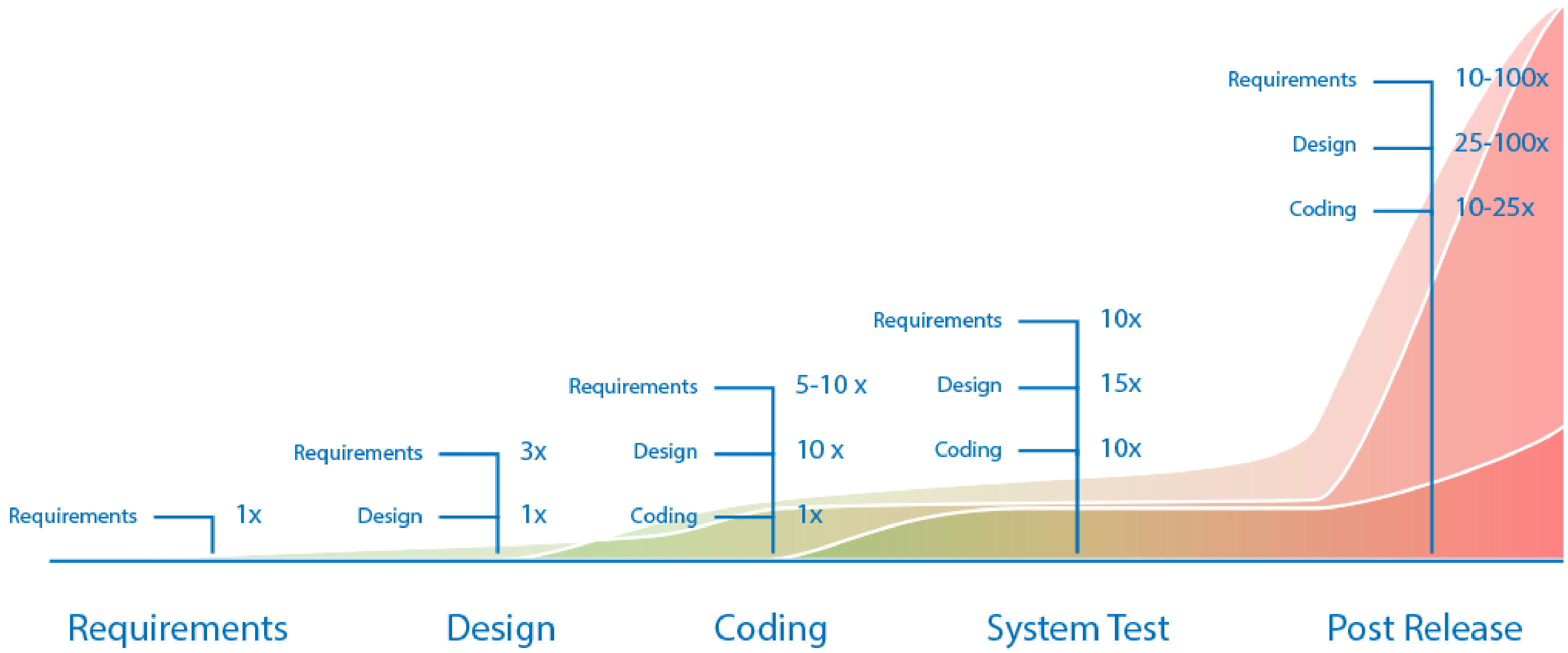


http://www.mediacurrent.com/blog/why-qa-your-website

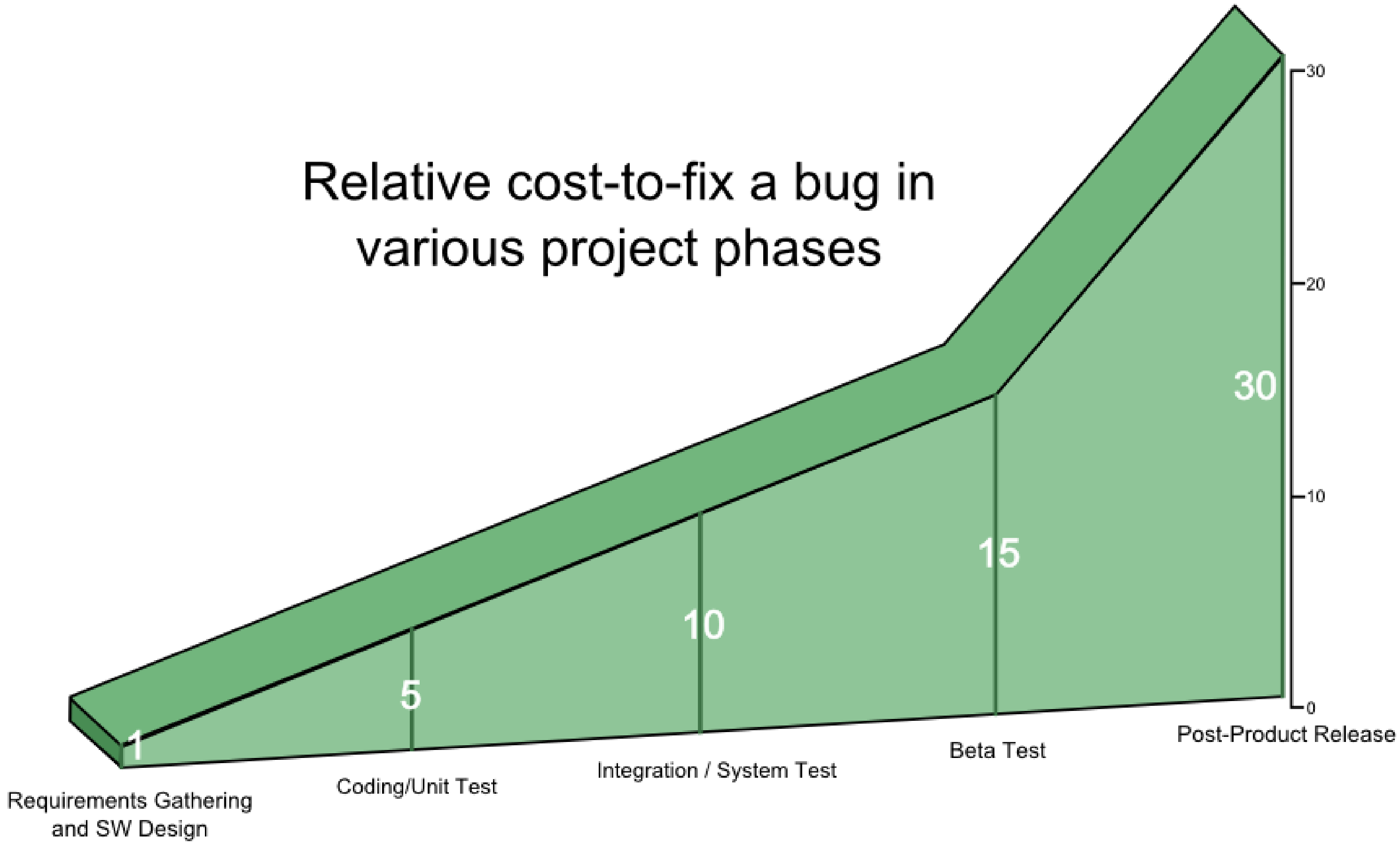
DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES 2016



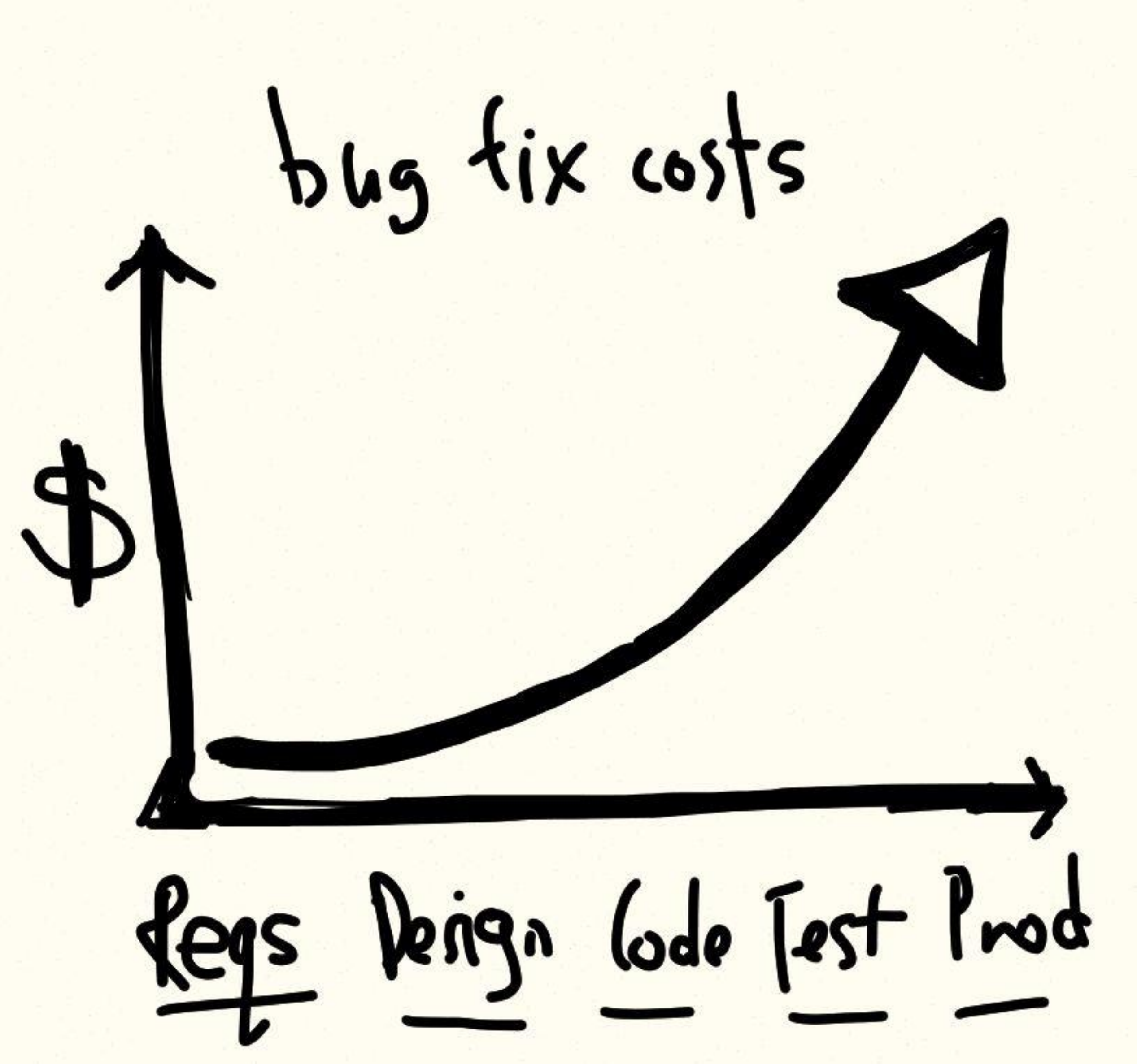
http://www.colinsalmcorner.com/post/why-you-absolutely-need-to-unit-test



https://www.arcanys.com/dedicated-developer/software-tester/

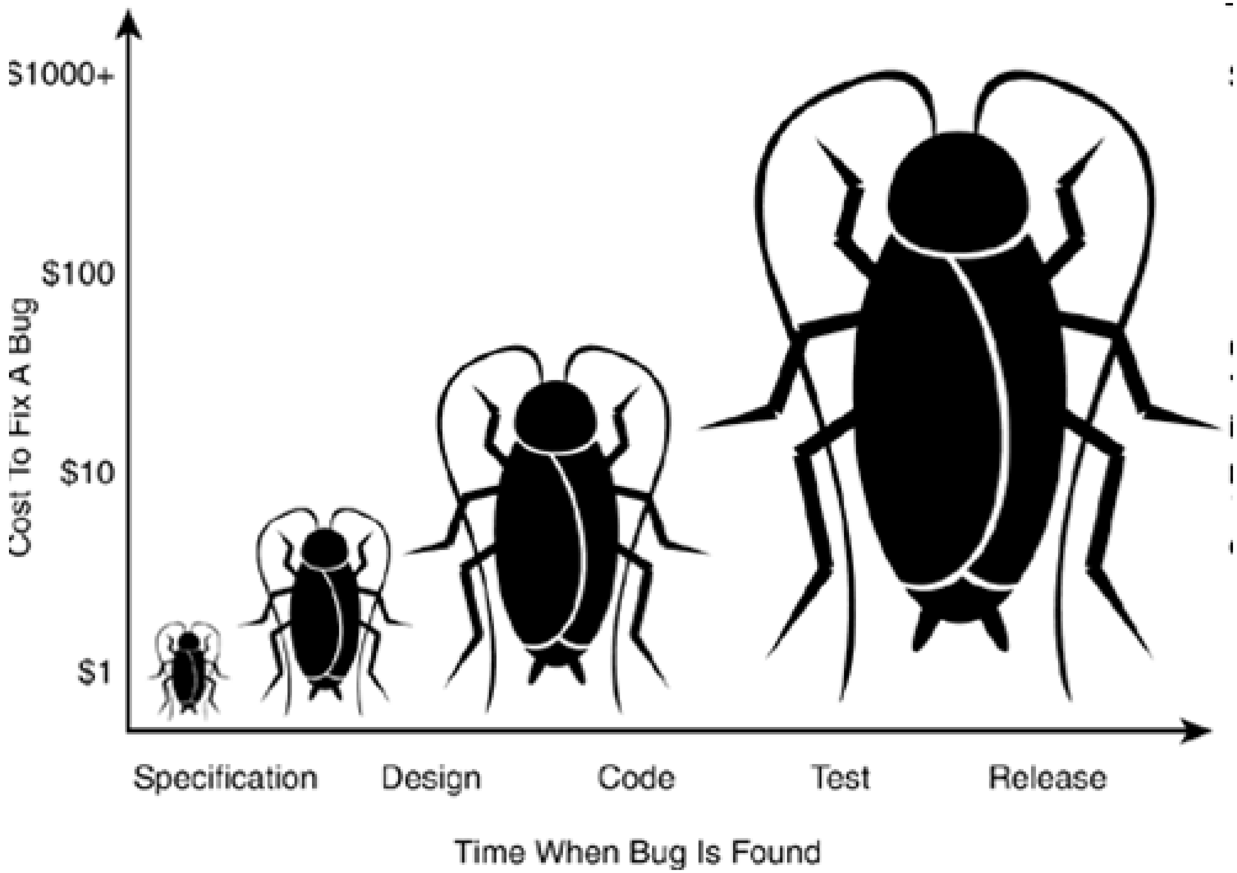


http://profiler-and-tracer.com/product/net-profiler-and-tracer-description/



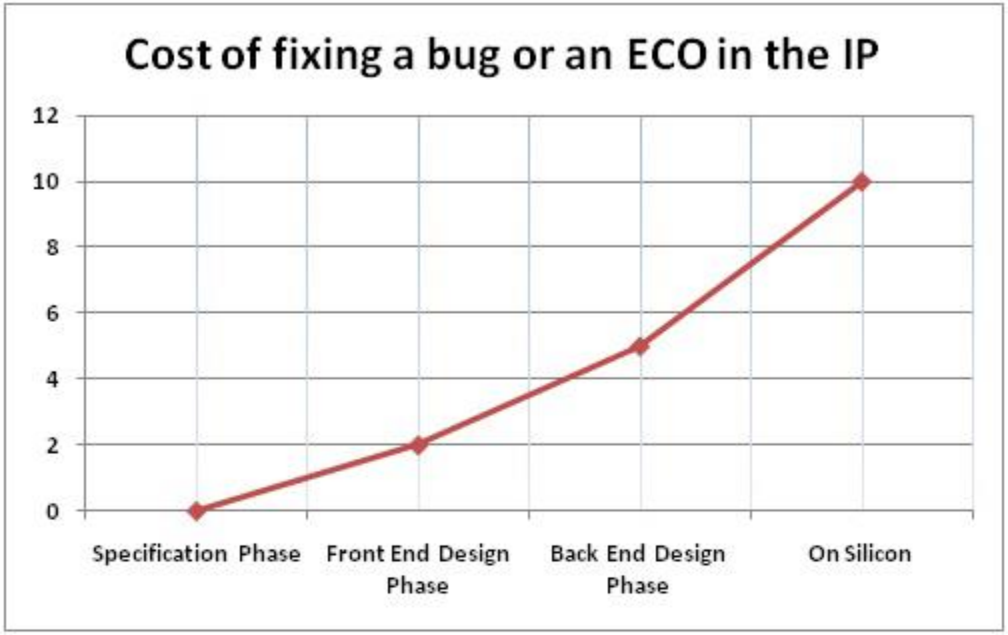
http://watirmelon.com/2013/05/

my personal favorite

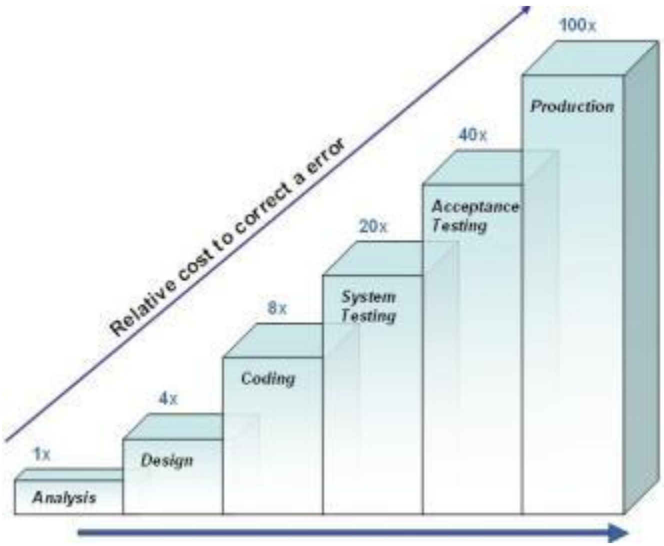


Patton, R. (2005). Software Testing (2nd ed.).

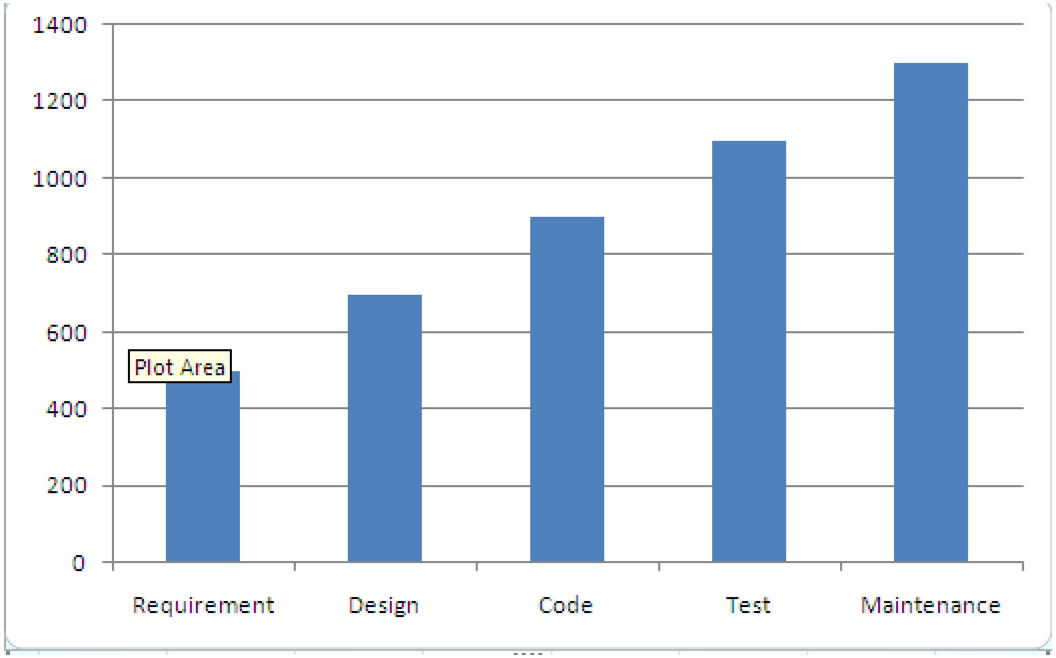
outliers



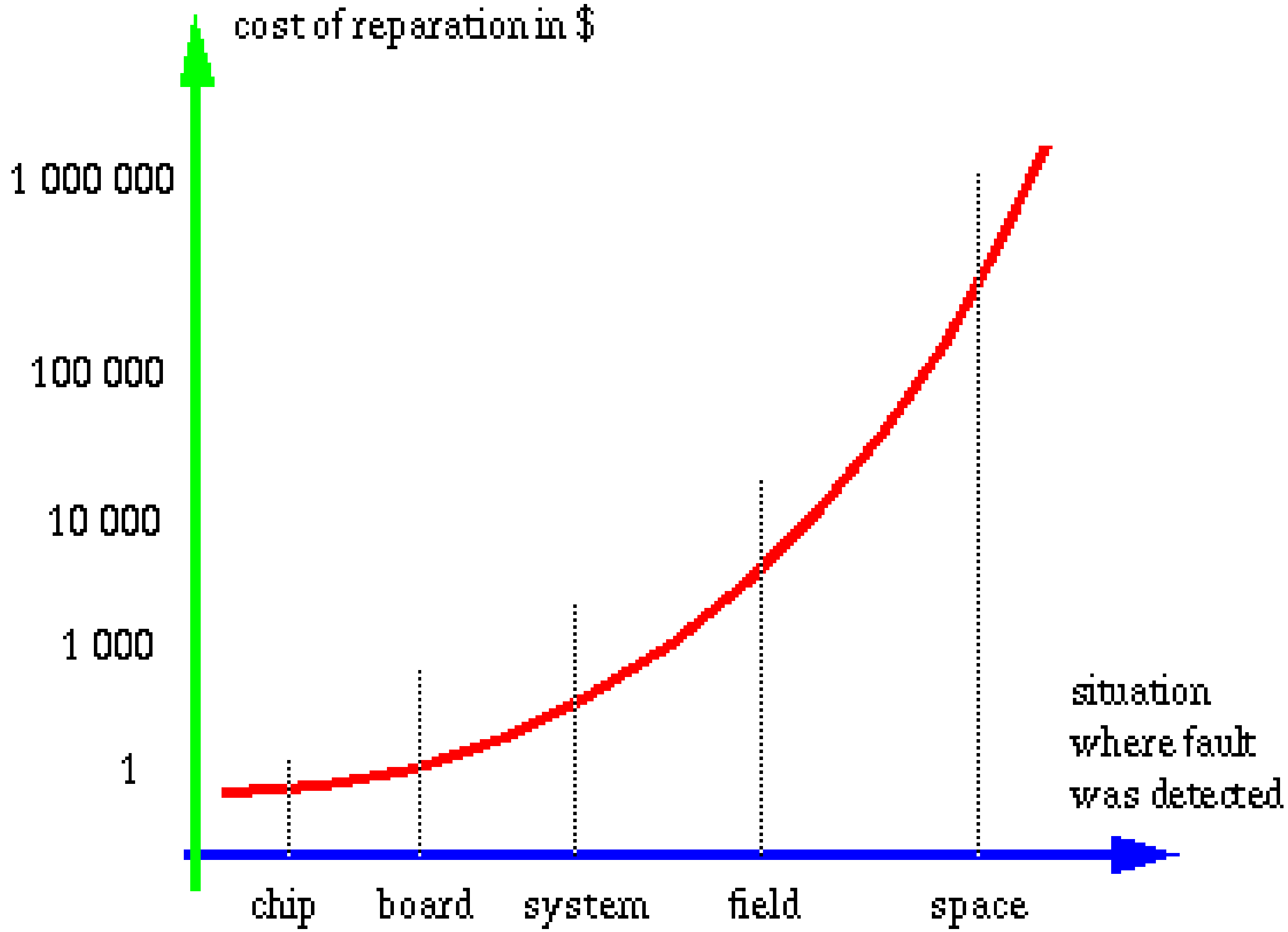
http://chipdesignmag.com/display.php?articleId=5268



http://archive.newsweaver.com/qualtech/newsweaver.ie/qualtech/e_article00118136164e4.html?x=b11,0,w



http://arashiqe.blogspot.com/2012/07/software-cost-of-defects.html



http://www.pldworld.com/_hdl/1/www.ireste.fr/fdl/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:

NIST 2002

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.

Requirements Gathering and Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/RAISE System Test	Early Customer Feedback/Beta Test Programs	Post-product Release
1X	5X	10X	15X	30X

Planning Report 02-3 “The Economic Impacts of Inadequate Infrastructure for Software Testing”

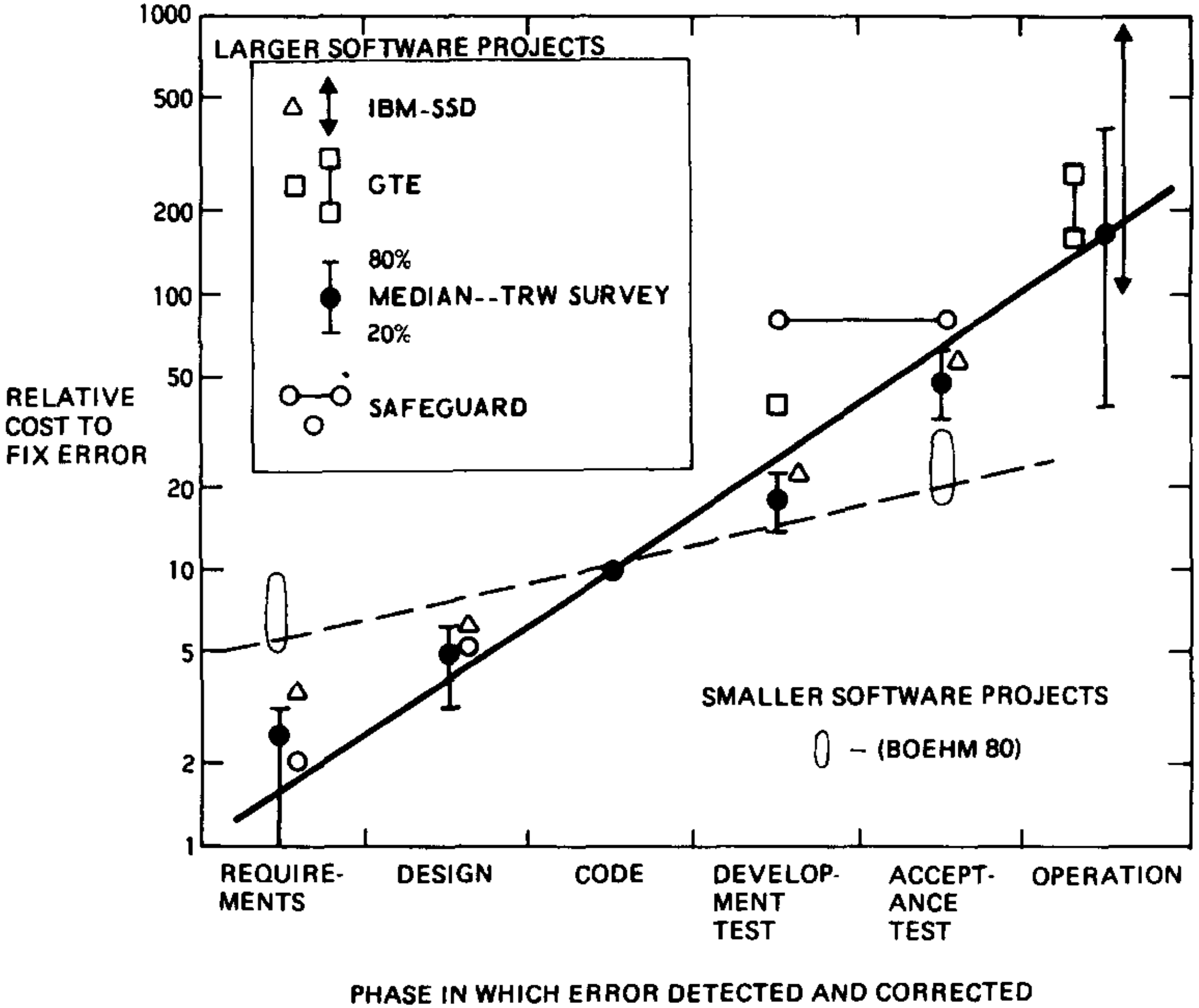
Prepared by: RTI for National Institute of Standards & Technology Program Office, Strategic Planning and Economic Analysis Group May 2002

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…

http://www.sicpers.info/2012/09/an-apology-to-readers-of-test-driven-ios-development/

source of all cost-to-fix charts



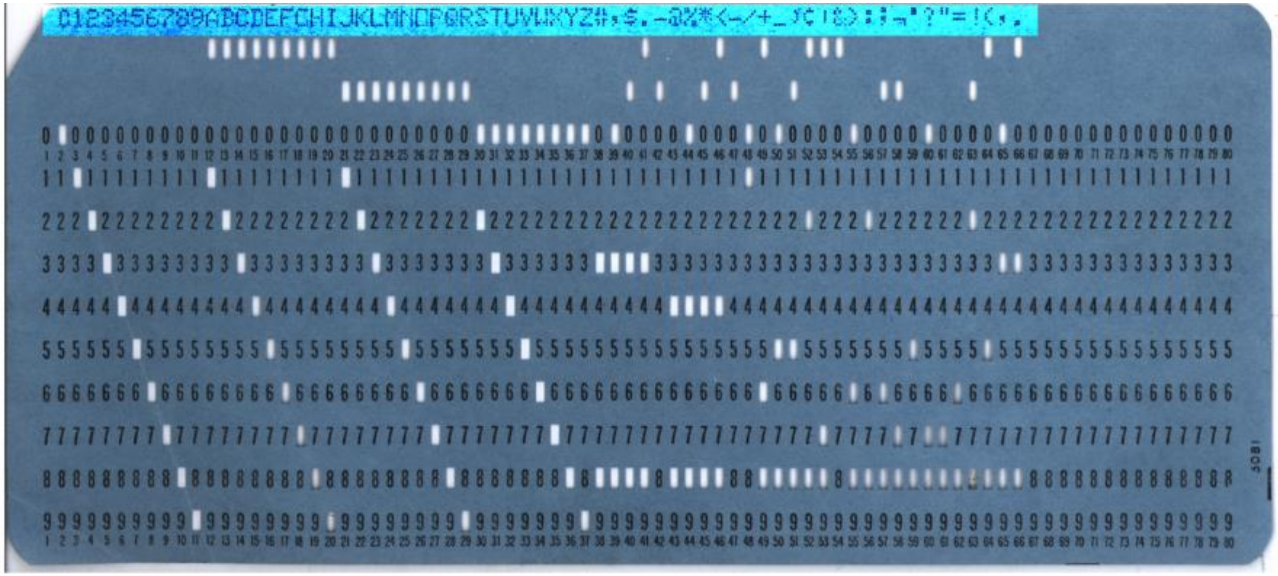
The studies in the chart

In Barry Boehm's *Software Engineering Economics* (1981), he identifies the souces 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]"

https://en.wikipedia.org/wiki/File:IBM_card_punch_029.JPG



https://en.wikipedia.org/wiki/File:Blue-punch-card-front-horiz_top-char-contrast-stretched.png

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.

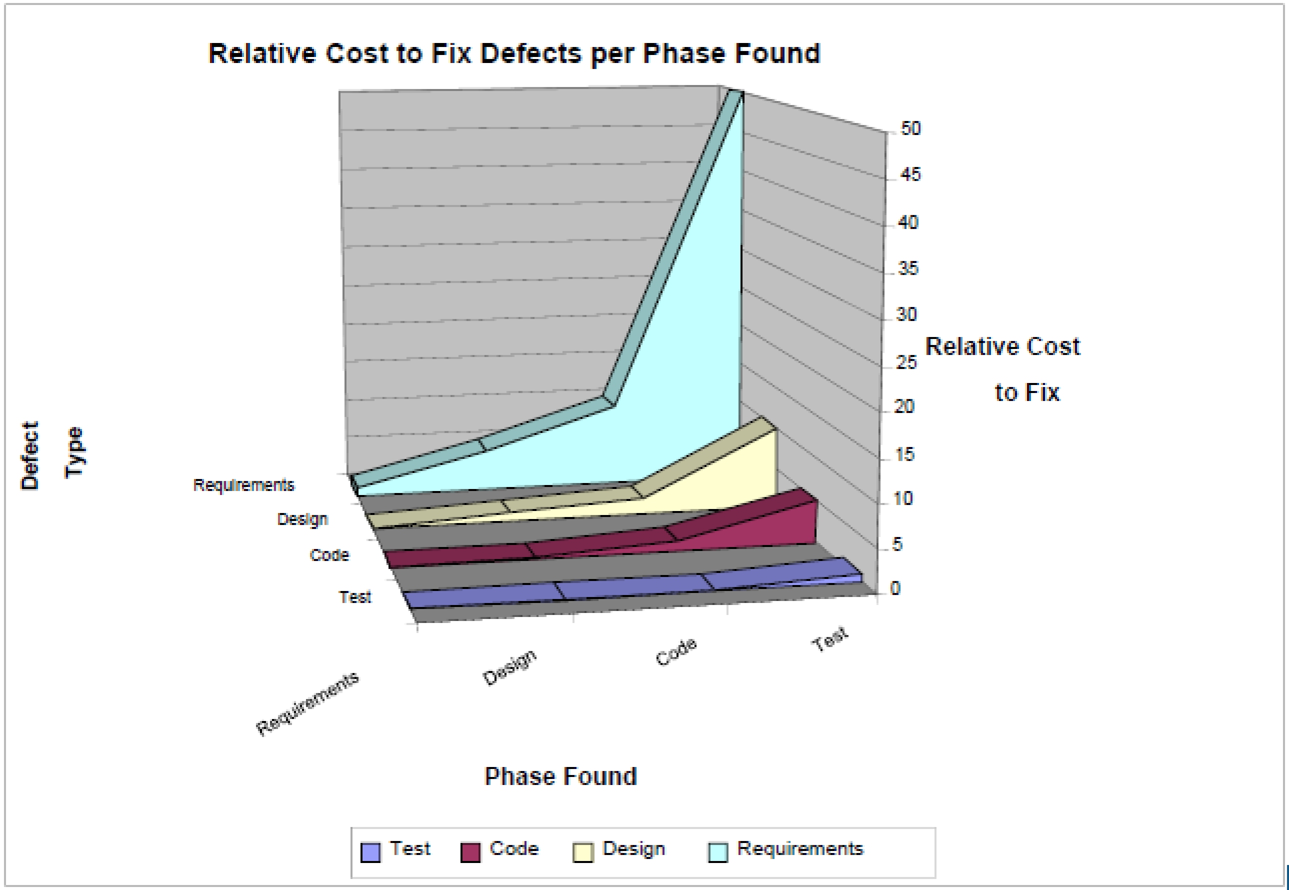
NIST 2002 (with phases)

Table 5-2. Preliminary Estimates of Relative Cost Factors of Correcting Errors as a Function of Where Errors Are Introduced and Found (Example Only)

	Where Errors are Found				
Where Errors are Introduced	**Requirements Gathering and Analysis/ Architectural Design**	**Coding/ Unit Test**	**Integration and Component/ RAISE System Test**	**Early Customer Feedback/Beta Test Programs**	**Post-product Release**
Requirements Gathering and Analysis/ Architectural Design	1.0	5.0	10.0	15.0	30.0
Coding/Unit Test		1.0	10.0	20.0	30.0
Integration and Component/ RAISE System Test			1.0	10.0	20.0

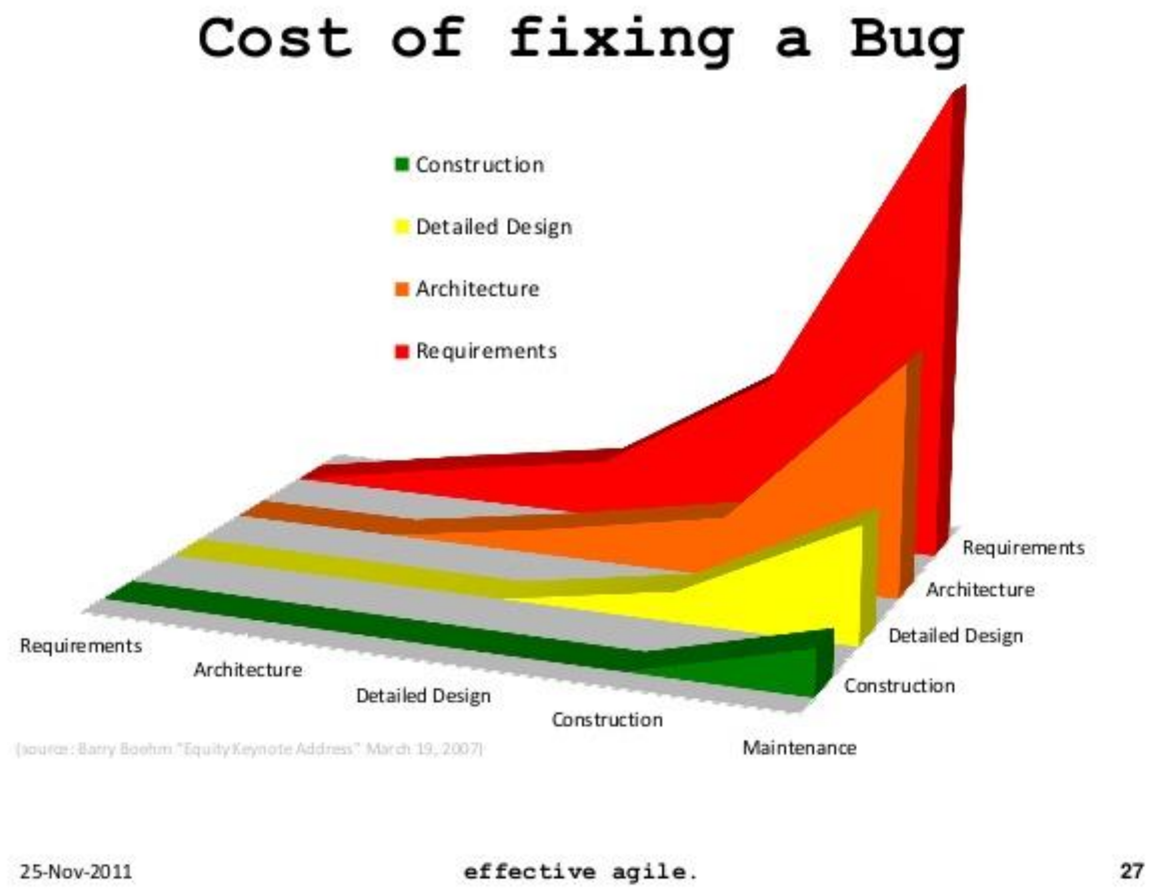
Planning Report 02-3 "The Economic Impacts of Inadequate Infrastructure for Software Testing"
Prepared by: RTI for National Institute of Standards & Technology Program Office, Strategic Planning and Economic Analysis Group May 2002

Importance of Early Verification, Validation, & Test

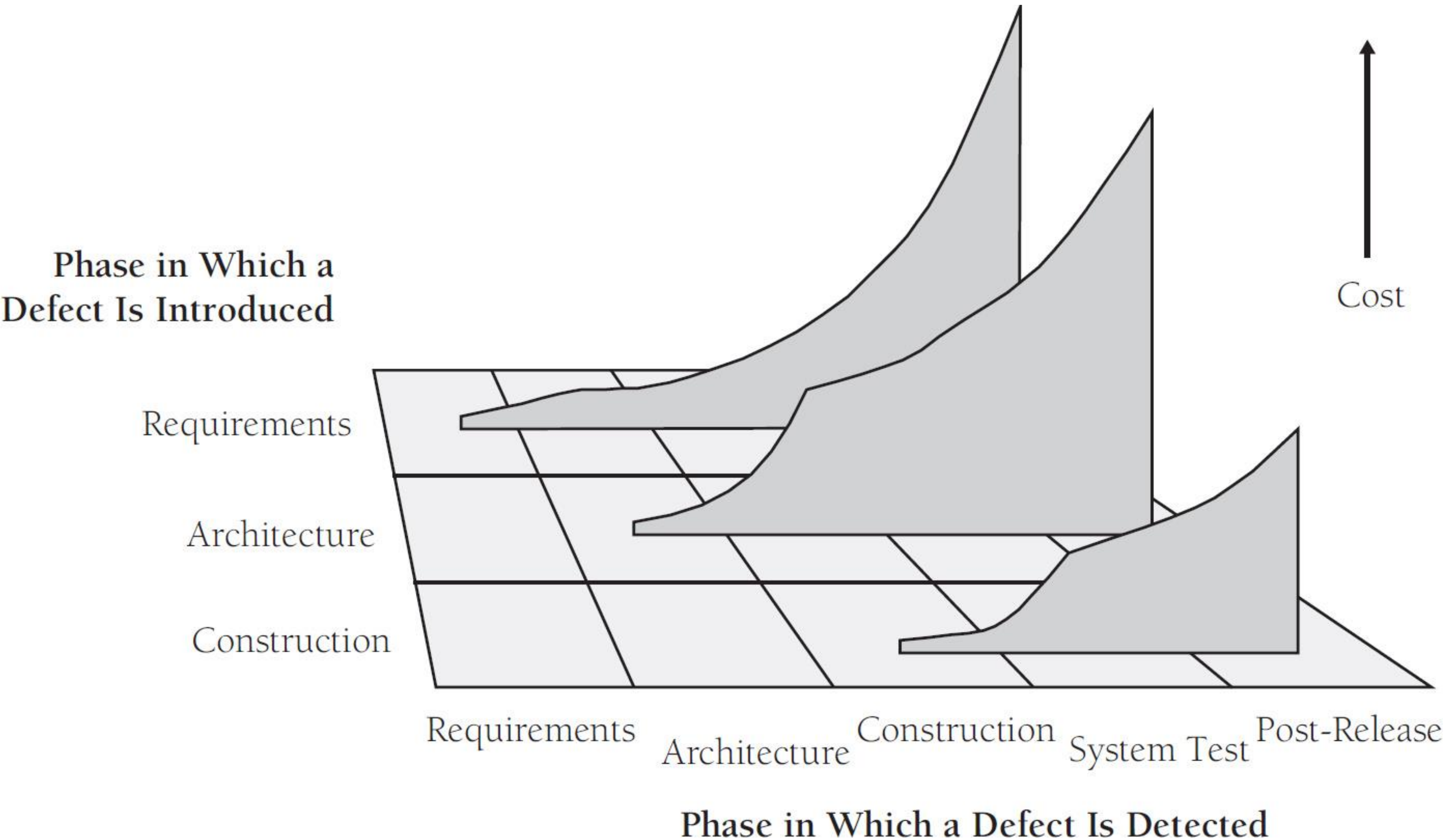


Source: Return on Investment for Independent Verification & Validation, NASA, 2004.

http://ww2.distek.com/casestudy/modeling-simulation-paper/



http://www.slideshare.net/BosniaAgile/empiricism-with-scrum-by-ralph-jocham

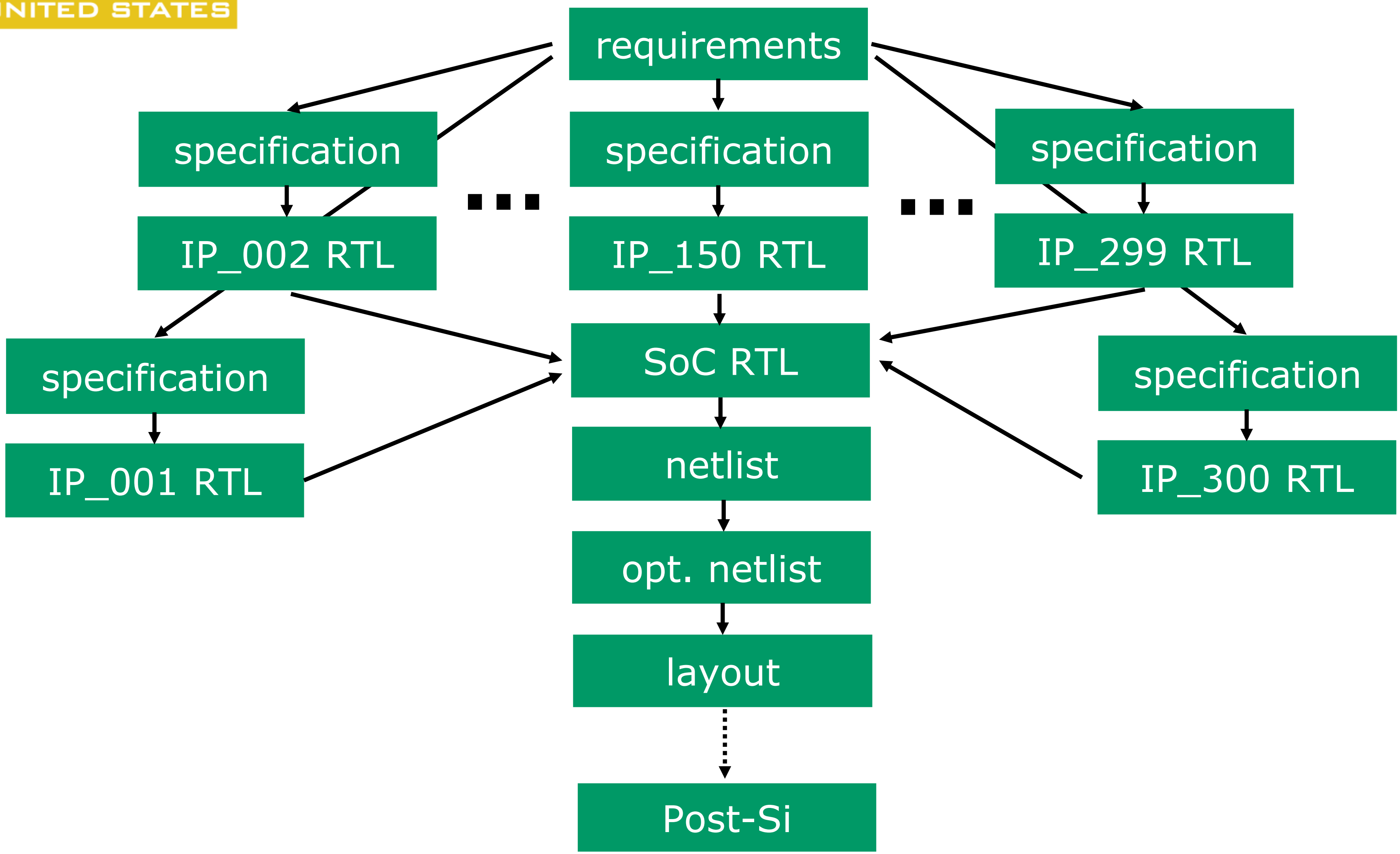

Phase in Which a Defect Is Introduced
Requirements
Architecture
Construction
Cost
Requirements
Architecture
Construction
System Test
Post-Release
Phase in Which a Defect Is Detected

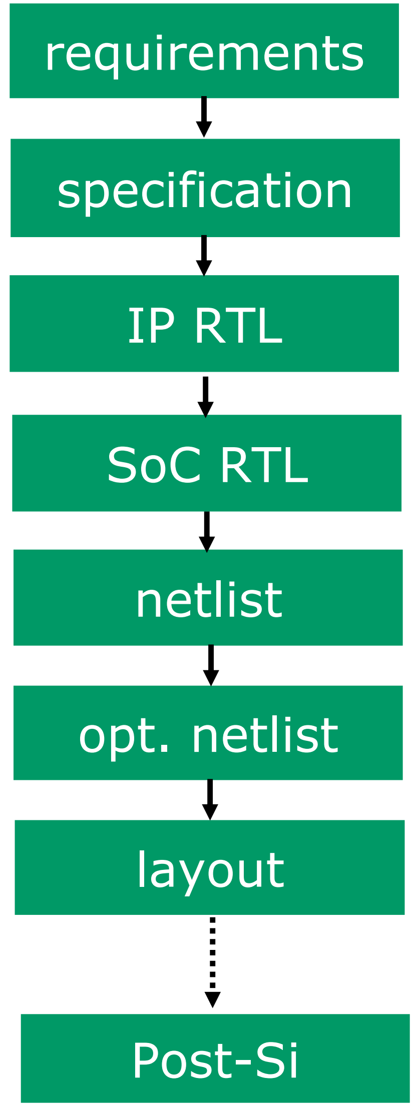
SoC waterfall development

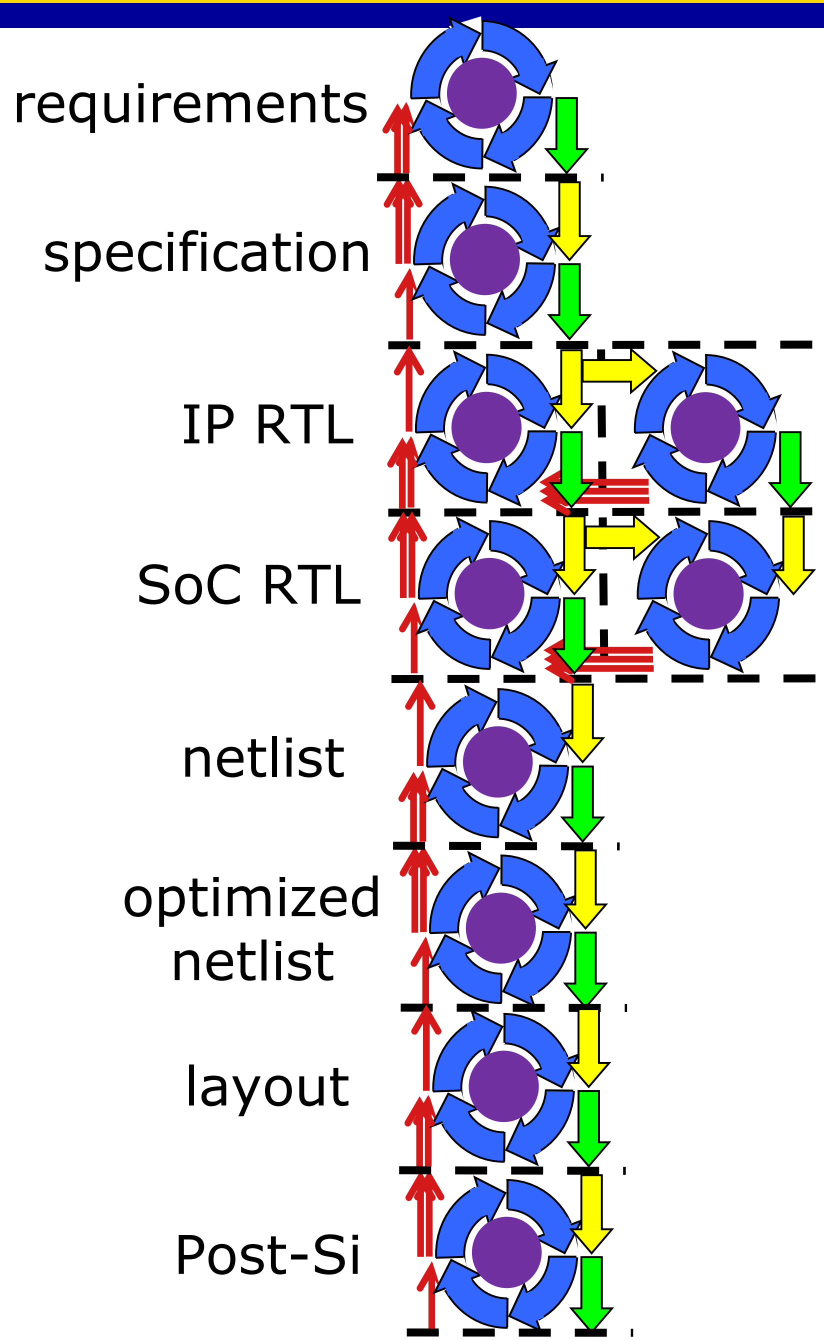
- While out of favor in software development, SoC hardware refinement and physical implementation is necessarily mostly waterfall development.
- Moreover, the development phases generally involve different teams at different sites performing quite different tasks. The cost of deliveries across phase 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
specification
specification
IP_002 RTL
IP_150 RTL
IP_299 RTL
specification
SoC RTL
specification
IP_001 RTL
netlist
IP_300 RTL
opt. netlist
layout
Post-Si


requirements
specification
IP RTL
SoC RTL
netlist
opt. netlist
layout
Post-Si

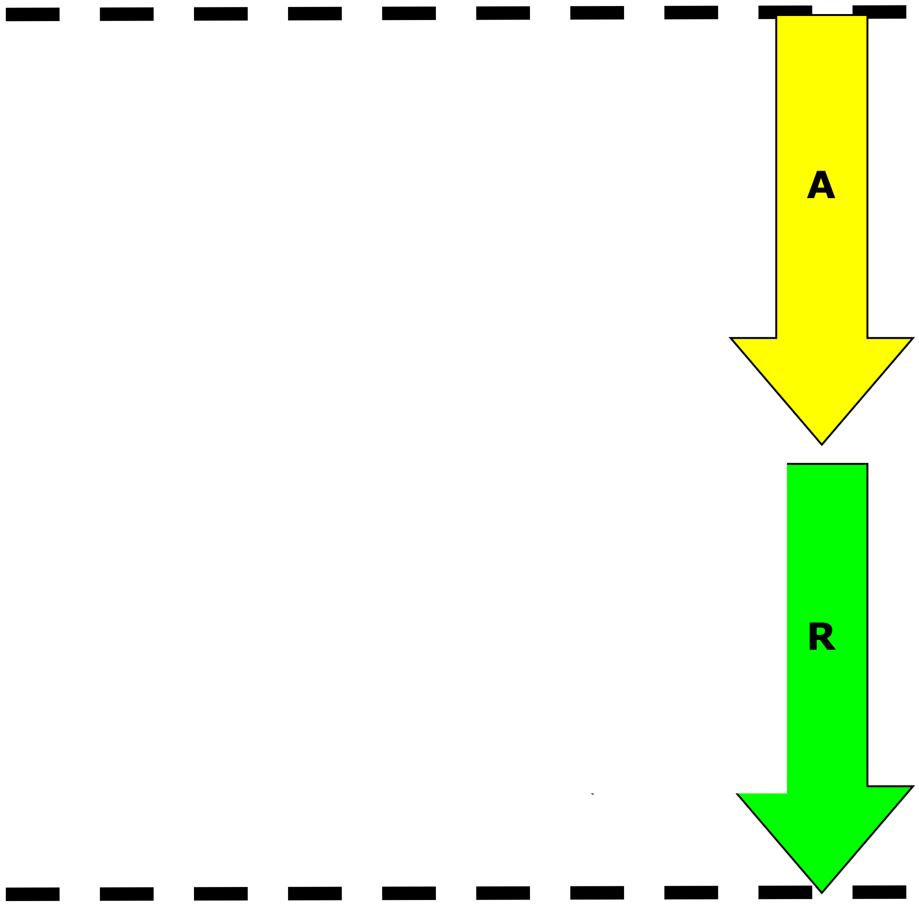


requirements
specification
IP RTL
SoC RTL
netlist
optimized netlist
layout
Post-Si

Stage vs. entire design flow

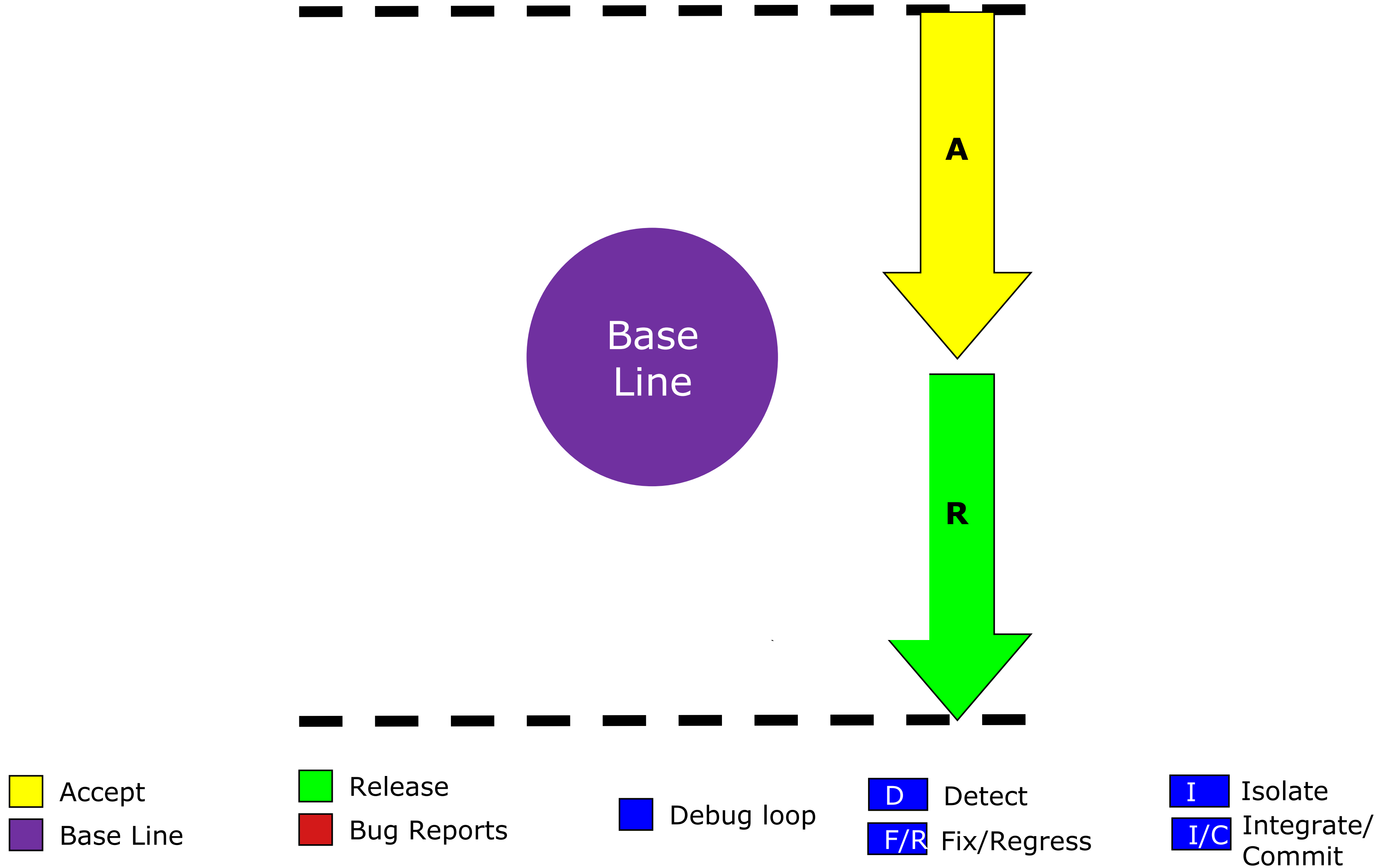
- Modeling cost can be done at the project-level or within a single stage:
  - Decisions to increase quality upstream (e.g., shifting resources into IP verification) must typically be done by management at the project level.
  - Within a stage there are many opportunities to improve efficiency, and the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

Accepts and releases

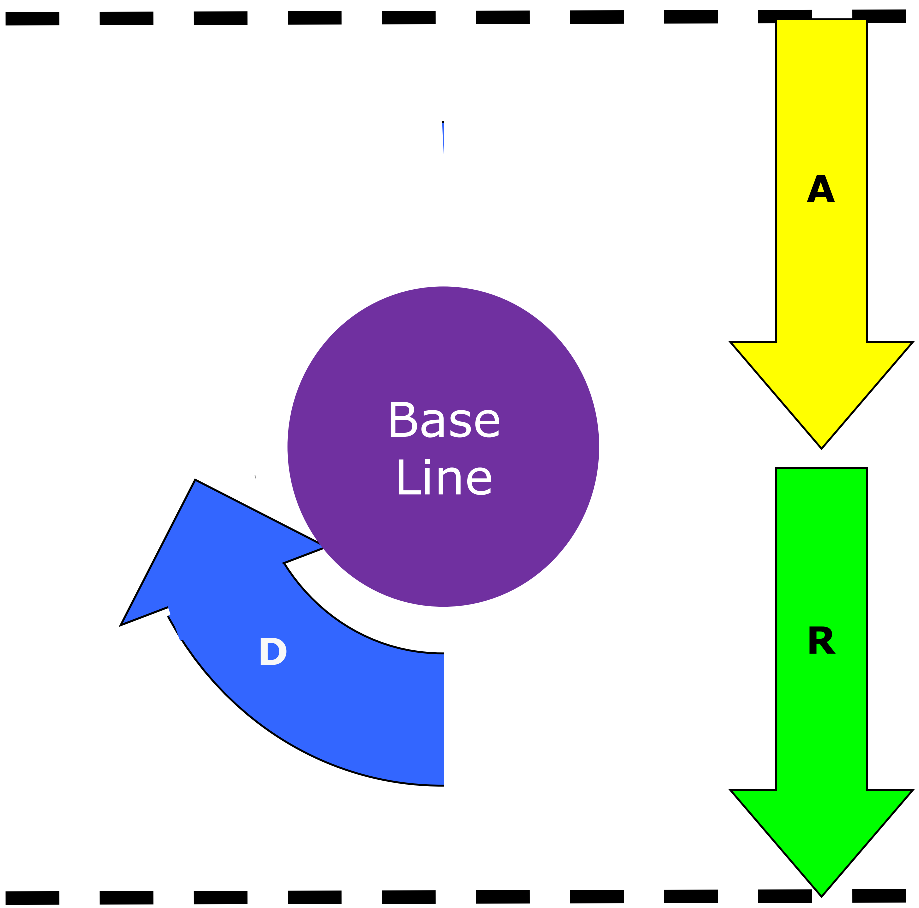


Accept	Release	Debug loop	D Detect	I Isolate
Base Line	Bug Reports		F/R Fix/Regress	I/C Integrate/ Commit

Baseline work

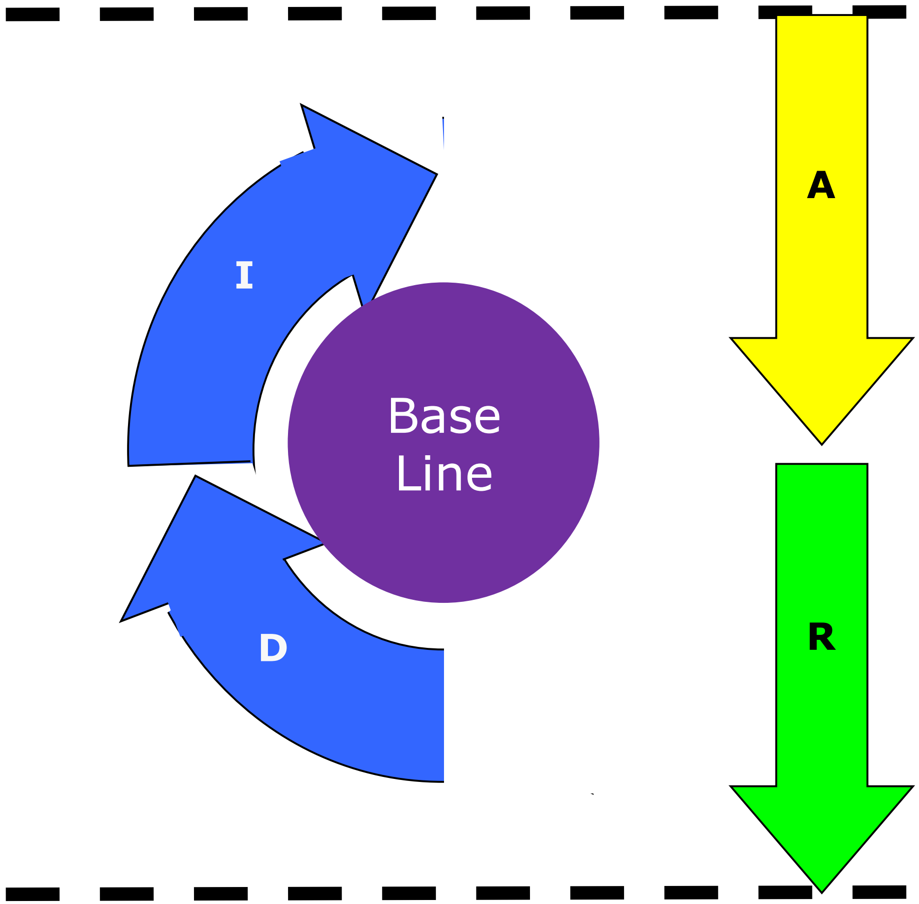


Debug loop: Detect



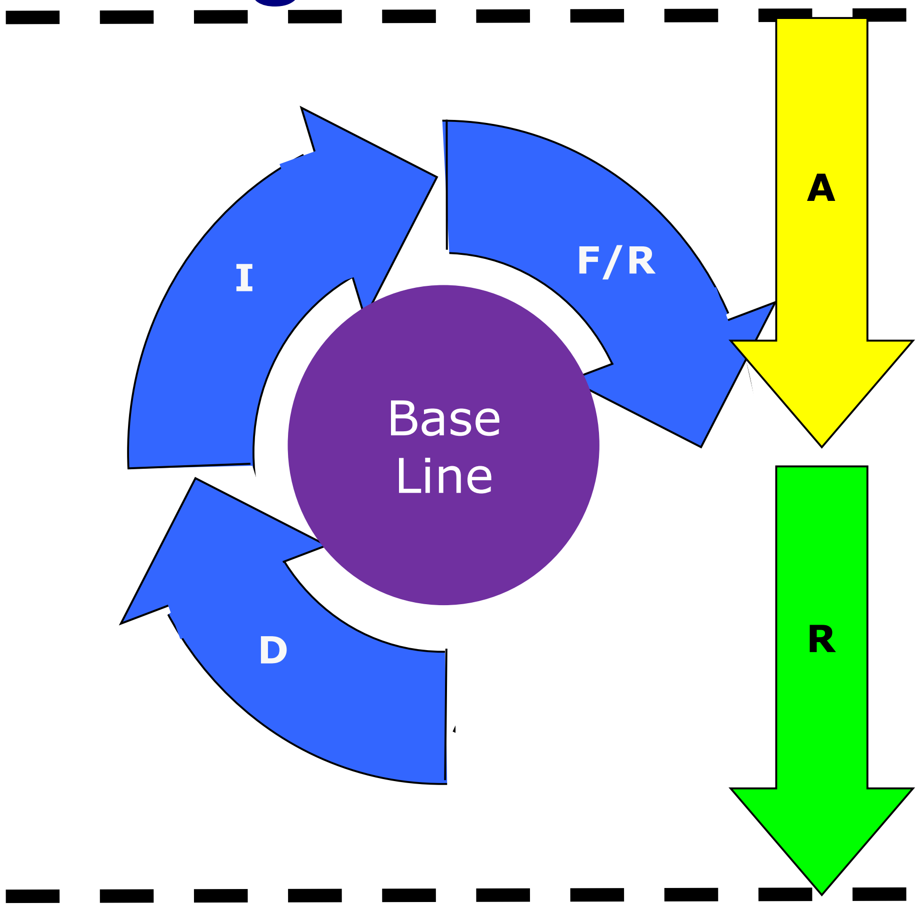
Accept
Base Line
Release
Bug Reports
Debug loop
D Detect
F/R Fix/Regress
I Isolate
I/C Integrate/ Commit

Debug loop: Isolate



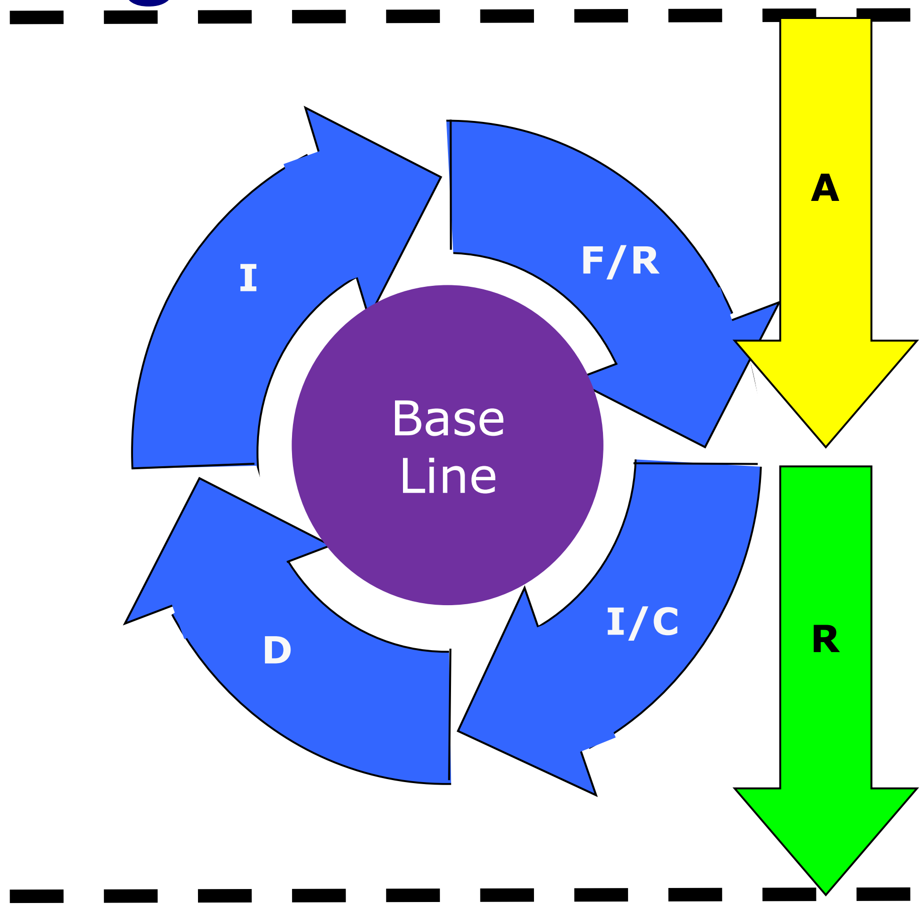
Accept
Base Line
Release
Bug Reports
Debug loop
D Detect
F/R Fix/Regress
I Isolate
I/C Integrate/ Commit

Debug loop: Fix/Regress



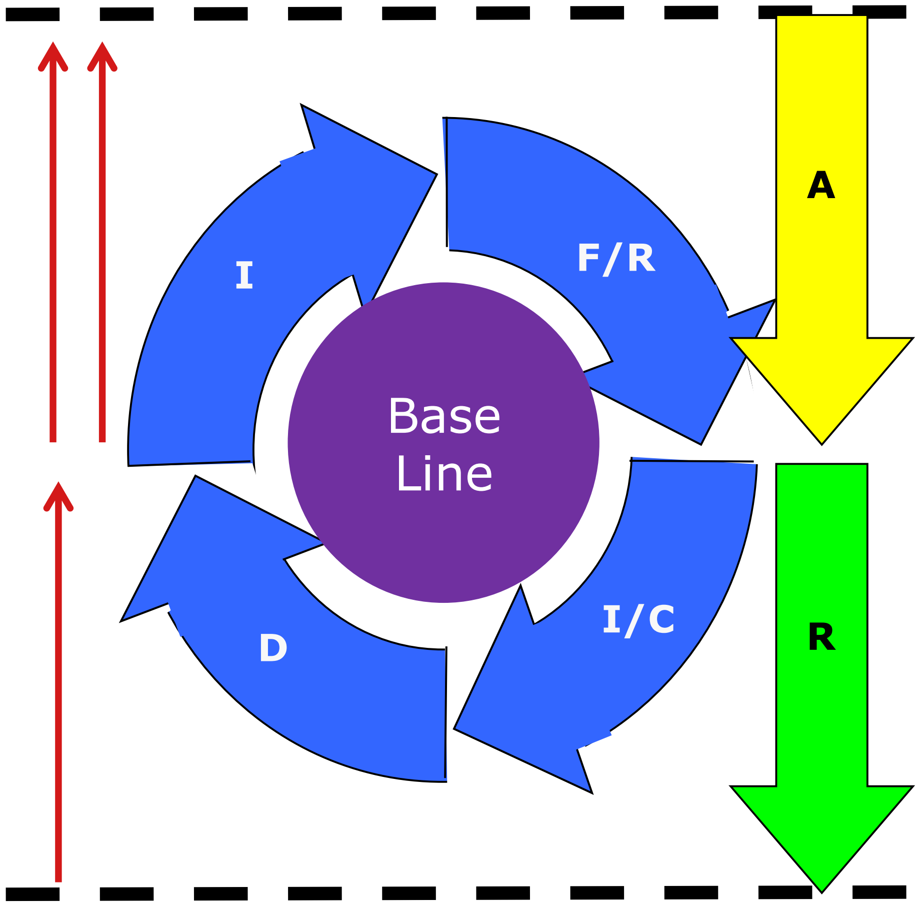
Accept	Release	Debug loop	D Detect	I Isolate
Base Line	Bug Reports		F/R Fix/Regress	I/C Integrate/ Commit

Debug loop: Integrate/Commit



Accept	Release	Debug loop	D	Detect	I	Isolate
Base Line	Bug Reports		F/R	Fix/Regress	I/C	Integrate/ Commit

Bugs and communication



	Accept		Release		Debug loop	D	Detect	I	Isolate
	Base Line		Bug Reports			F/R	Fix/Regress	I/C	Integrate/ Commit

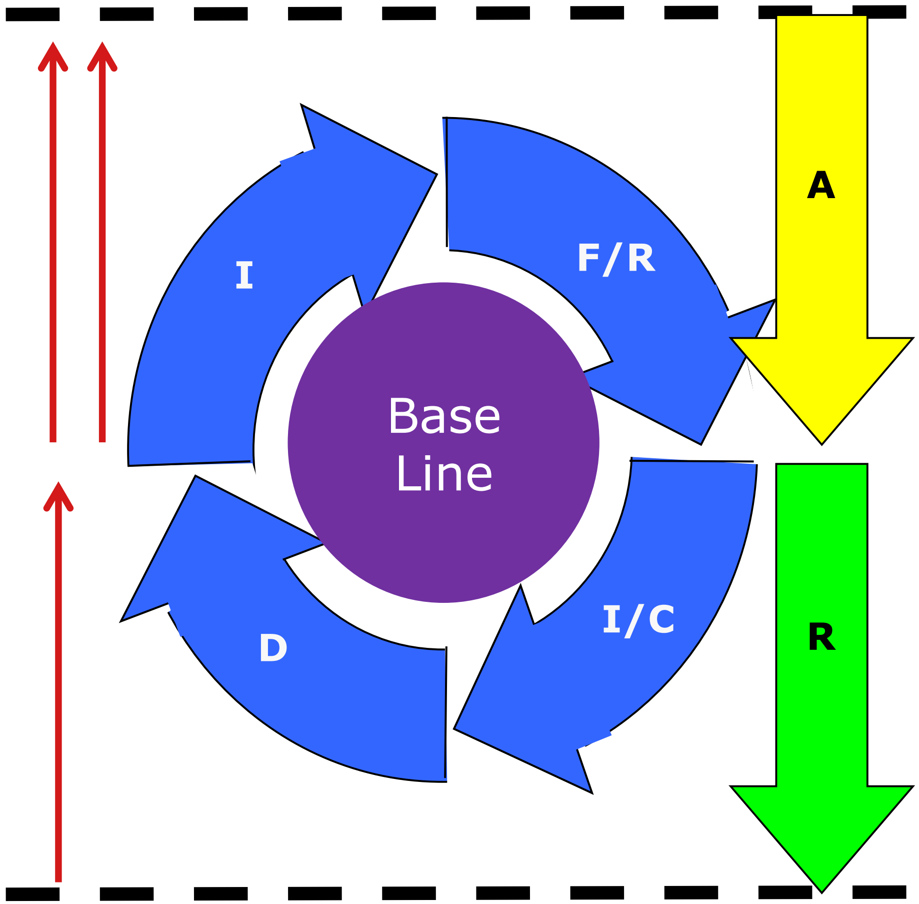
Collecting data

- 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
D Detect
F/R Fix/Regress
I Isolate
I/C Integrate/ Commit

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."

"Software Defect Reduction Top 10 List", B. Boehm, V.R. Basili, *IEEE Computer*, January 2001.

DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES 2016