

# Software Quality **Management**

**Dr. Stefan Wagner**  
Technische Universität München

Garching  
21 May 2010

# Last QOT: What quality attribute is the hardest to evaluate with tests?

"Absence of defects"

"Safety"

"Reliability"

"Usability"

2

Showing absence of defects is clearly not possible with testing. As Dijkstra's law says: "Testing can show the presence but not the absence of errors."

Safety is indeed hard to evaluate with tests alone as the system has to be safe in all cases and under all circumstances. Nevertheless, testing is suitable as one building block for providing safety evidence.

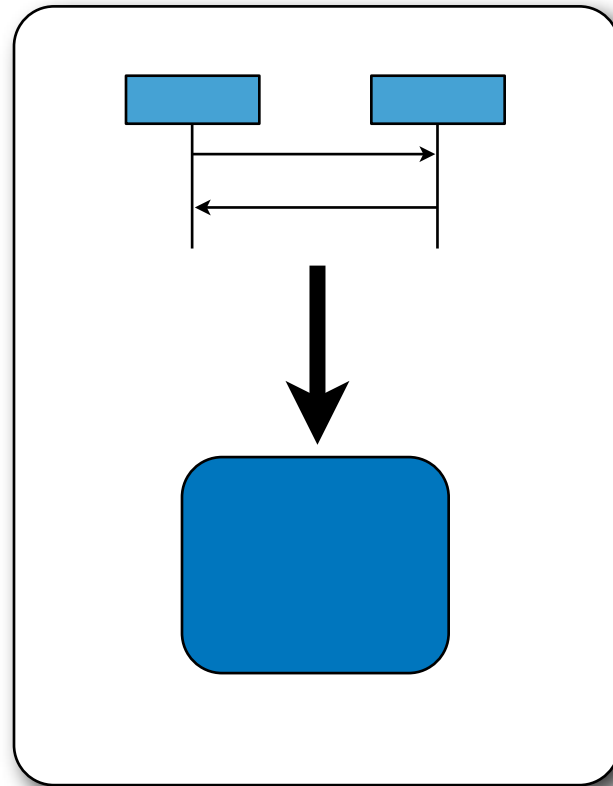
The best way of evaluating reliability are tests. The best way are field tests (or beta tests) in which the future users work with the system. Also system tests tend to be a useful means for evaluating reliability.

Usability can be tested with user tests. The Nielsen–Norman law says: "Usability is quantifiable."

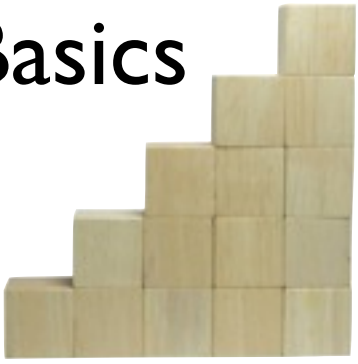
New QOT: "On which quality attribute do reviews have the most direct influence?"

# Constructive Quality Assurance

# Testing



**Basics**



**Product**



Quality

**Metrics and**



**Measurement**

**Process**  
Quality



**Management**

**Certifi-  
cation**



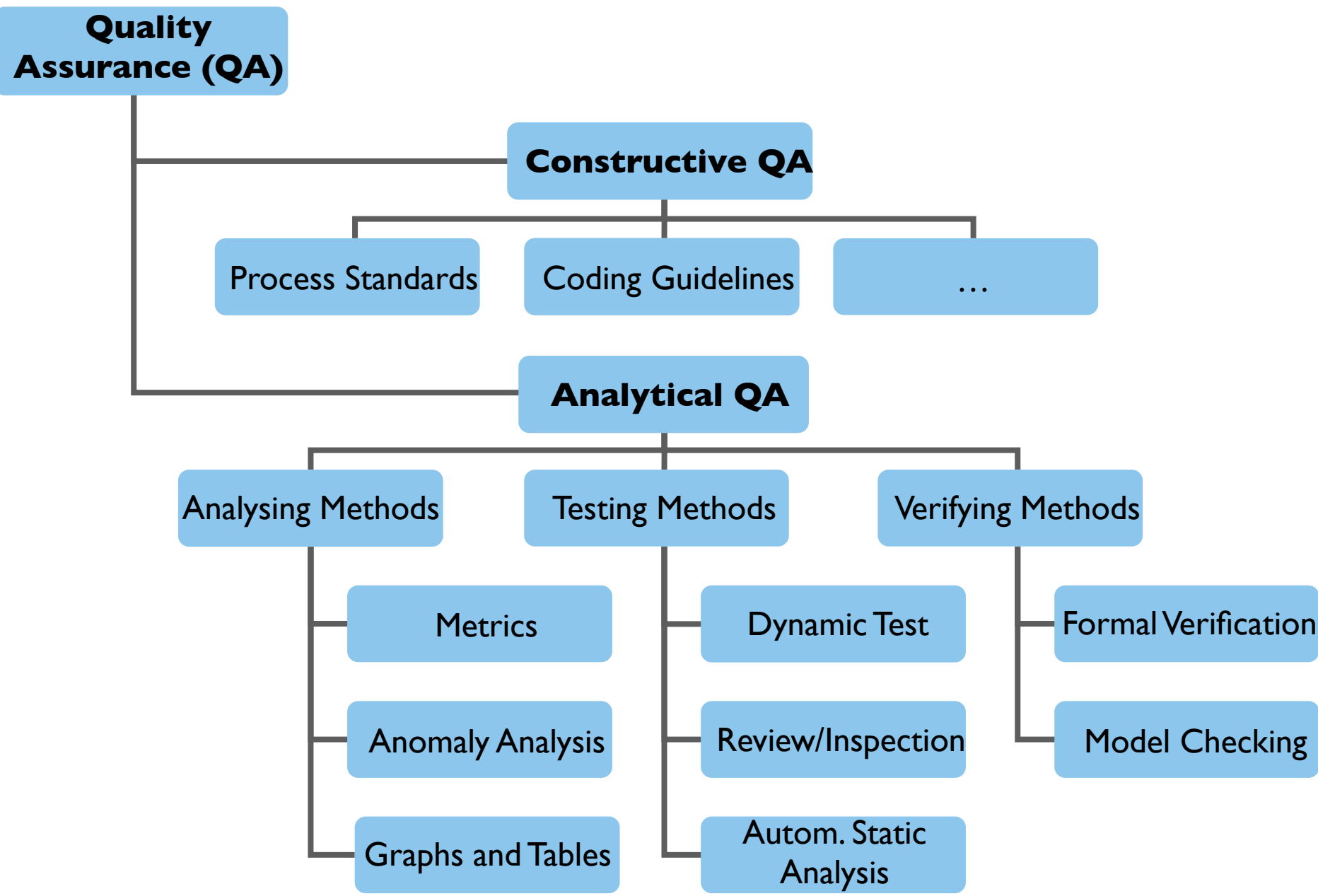
# Inspection Review Walkthrough



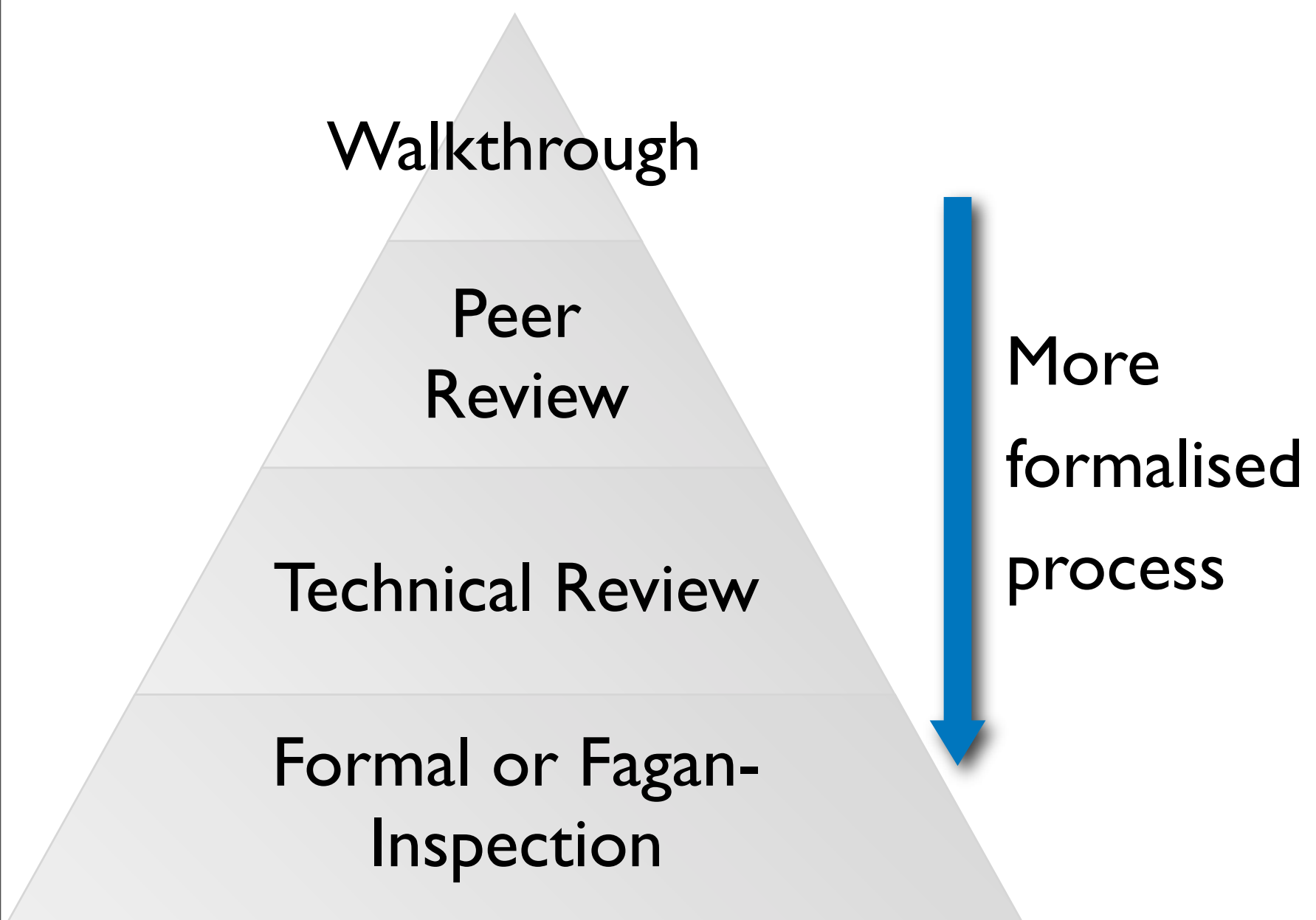
5

There is no fixed terminology, but "review" seems to be the most used umbrella term for all quality assurance methods that involve reading the contents of an artefact to find quality defects.

Walkthroughs are usually more light-weight in that the author explains the artefact. Inspections are more formalised.



Reviews and inspections are testing methods.

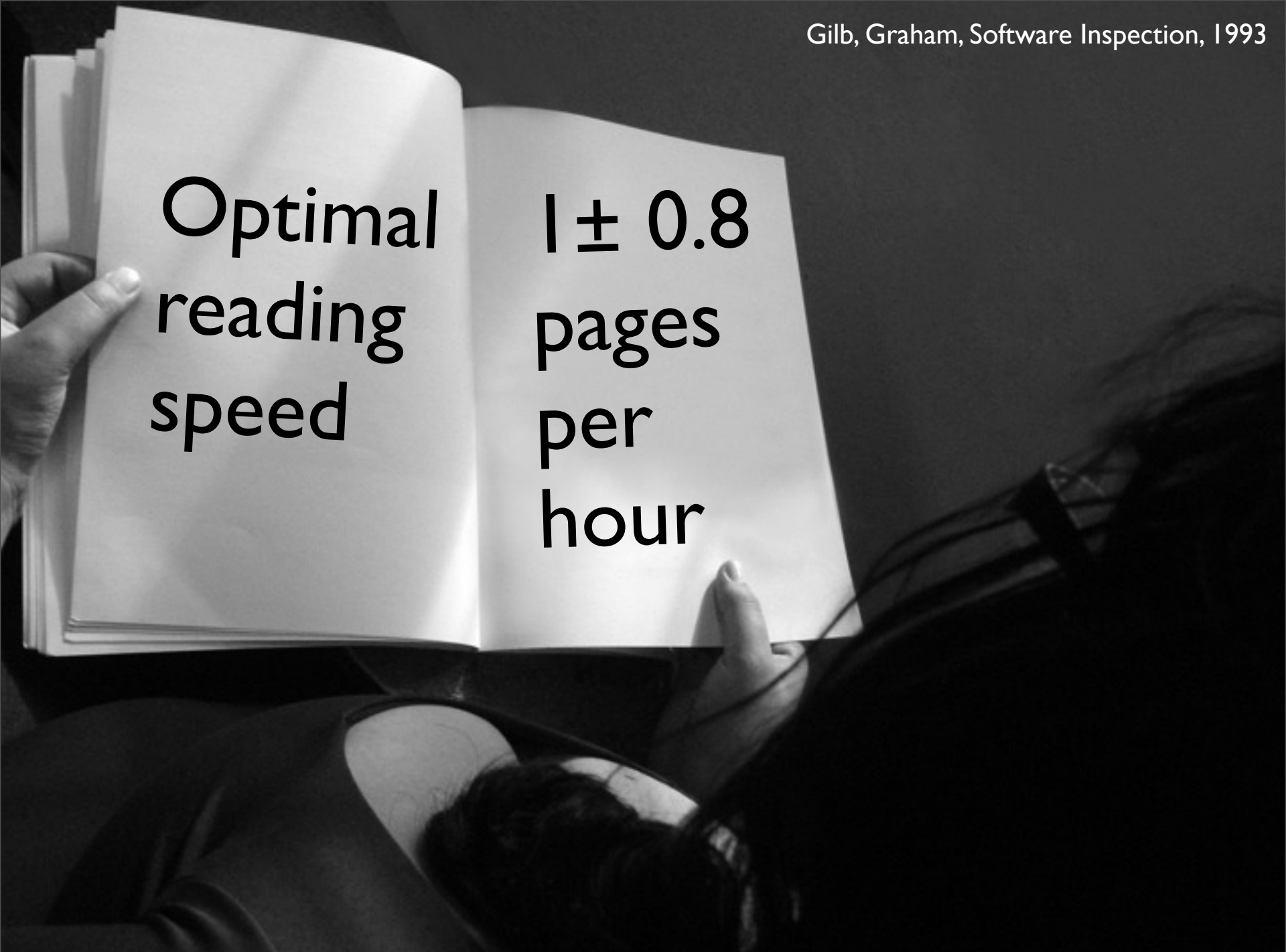


Walkthrough, also called presentation reviews, have the aim that the participants understand the contents of the analysed artefact. The author guides the group through a document and his or her thought processes, so all understand the same thing. The end should be a consensus on how to change the document.

Peer reviews do not involve the author explaining the artefact. The author gives the artefact to one or more colleagues who read it and give feedback. The aim is to find defects and get feedback on the programming style.

Technical reviews formalise this process. They are often also management reviews or project status reviews. Here the aim is often to make decisions about the project progress. A group discusses the artefact and makes a decision about the content.

The main aim of inspections is to find defects. It involves formal individual and group checking using sources and standards. Usually there are detailed and specific rules.

A black and white photograph of a person's hands holding an open book. The book is held open, showing two pages. The left page has the text 'Optimal reading speed' and the right page has the text '1 ± 0.8 pages per hour'. The person's face is partially visible at the bottom of the frame, looking down at the book.

Optimal  
reading  
speed

$1 \pm 0.8$   
pages  
per  
hour

A surprising but well investigated fact about reviews is that the optimal reading speed is about 1 page per hour. The normal reading speed (without the aim of finding defects) is considerably higher. If you read significantly faster, you miss defects, if you read slower, you do not find more defects.



# Inspections

On average 1/3 of all faults  
Up to 93% of faults

1-2 person-hours per fault

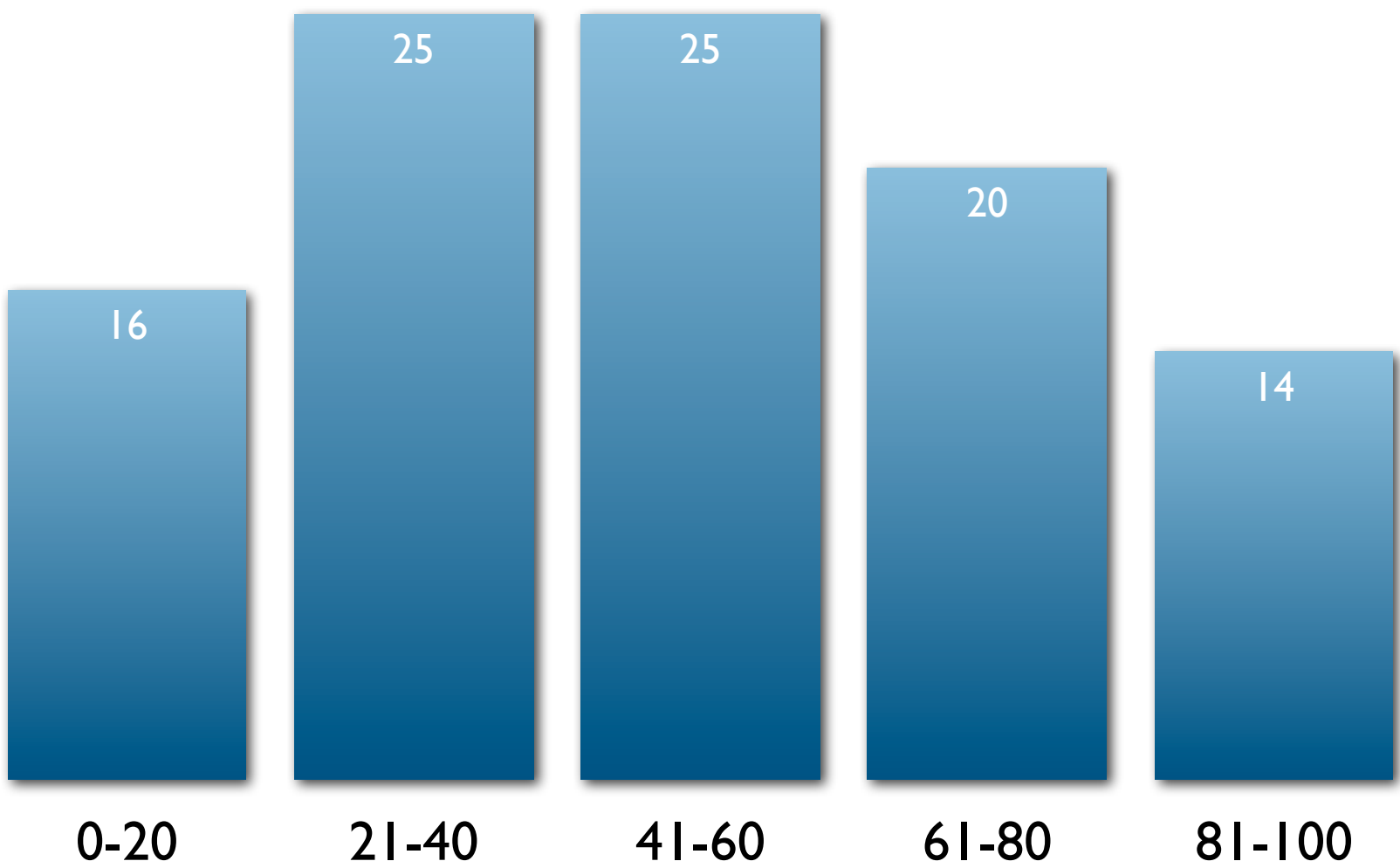
Wagner, A Literature Survey of the Quality Economics of Defect-Detection Techniques, 2006

9

- Effective and efficient
- Effectiveness
  - Are able to find up to 93% of faults
  - On average, a third of the faults are found
- Efficiency
  - Effort to detect a fault 1-2 person-hours / fault
  - Comparable to common testing methods
- But
  - Also in early phases
  - Also on requirements and design documents
  - Fault removal is the least expensive

# Estimated review effectiveness

Percentage of defects

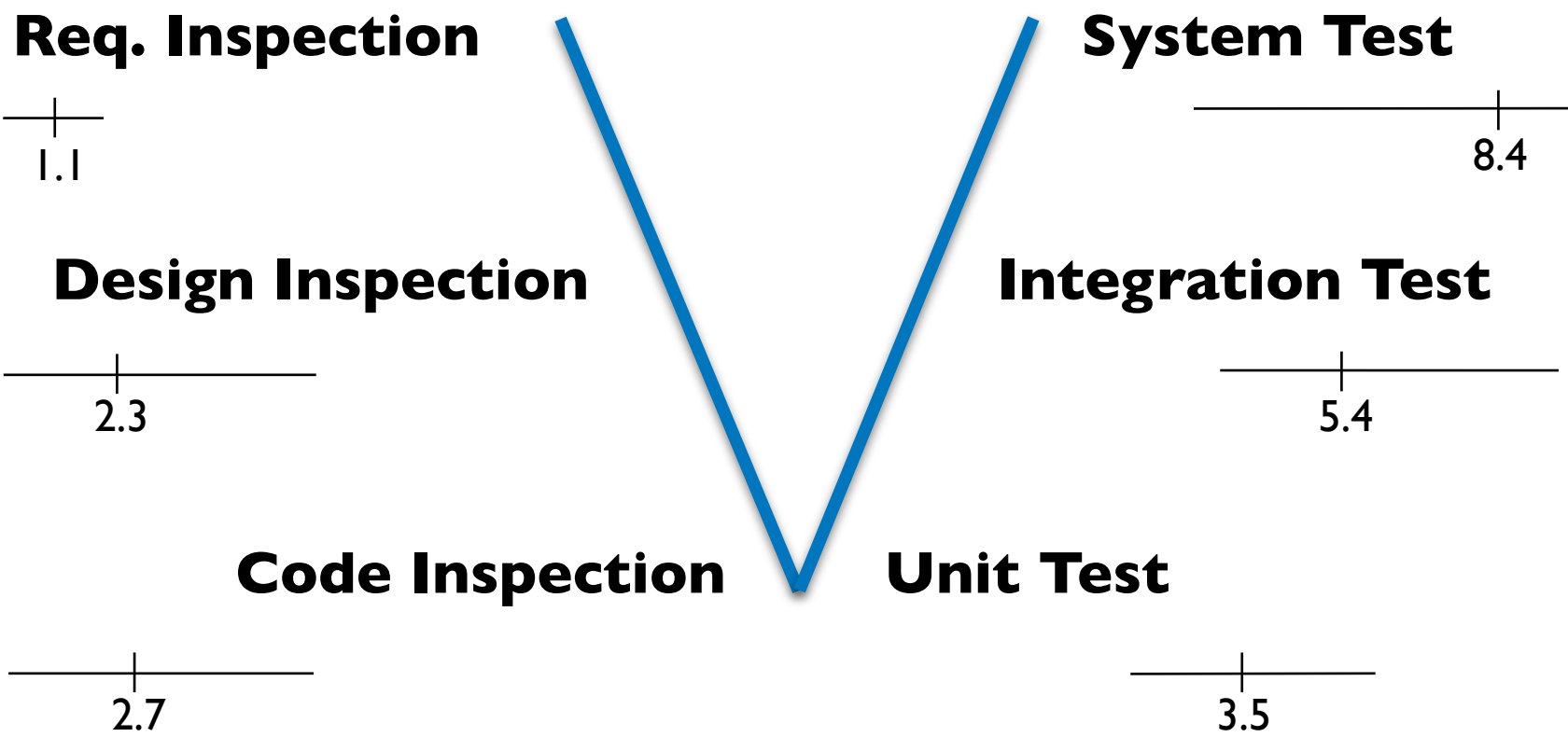


Ciolkowski, Laitenberger, Biffi, Software Reviews: The State of the Practice, 2003

In practice, reviews do not often reach the highest possible effectiveness. Most reviews seem to have an effectiveness between 20% and 60%.

# Defect-removal effort

in person-hours per defect

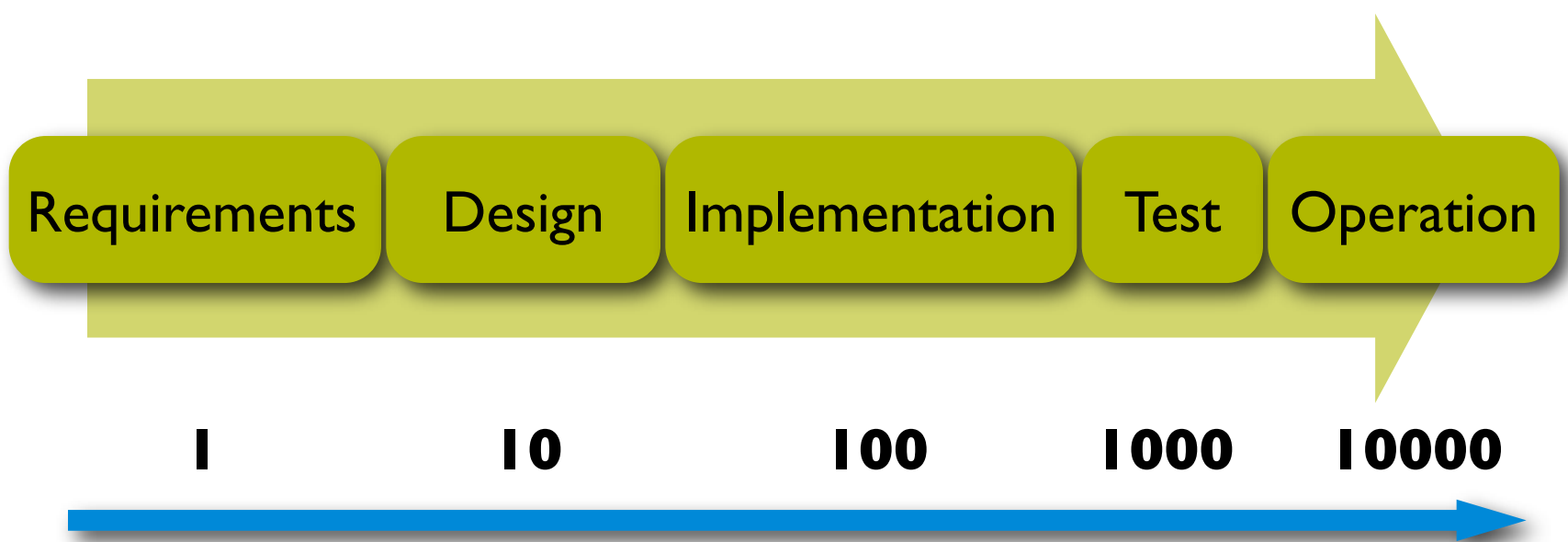


Wagner, A Literature Survey of the Quality Economics of Defect-Detection Techniques, 2006

11

The most interesting data about inspections is the defect-removal effort. It is not only interesting to look at the effort that is needed for **finding** a defect, but also how much effort is spent on **removing** it. An inspection gives you directly the cause of the problem, while testing always needs debugging first.

The highest removal effort is in system testing with up to 20 person hours per defect.



## Defect Costs

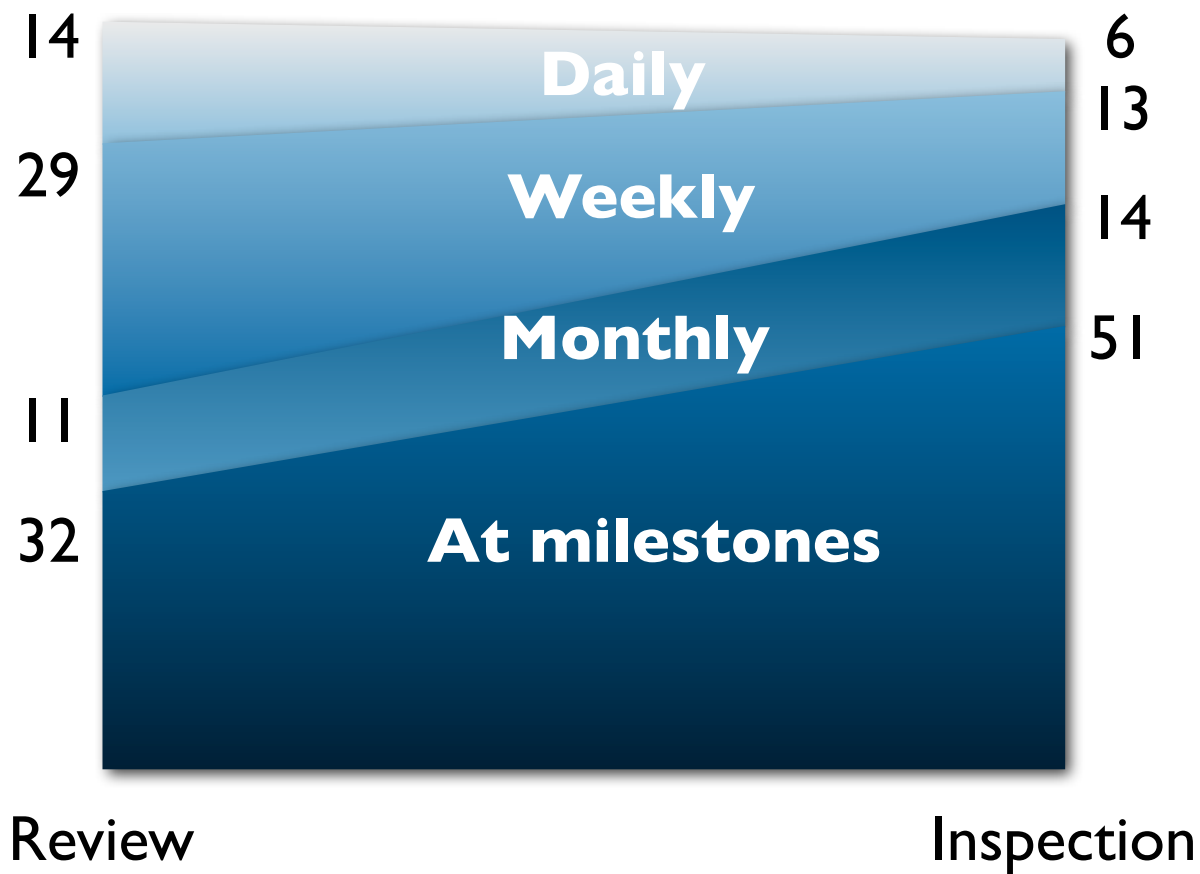
Boehm, Software Engineering Economics, 1981

12

- It is always better to prevent a defect than to remove it
- The earlier a defect is found, the less expensive it is.
- Defect costs here include finding and removing the defect as well as further costs (loss of reputation).
- There is a ten-fold increase from phase to phase. Hence, investing early pays heavily.

# How often do you review?

Percentage of respondents



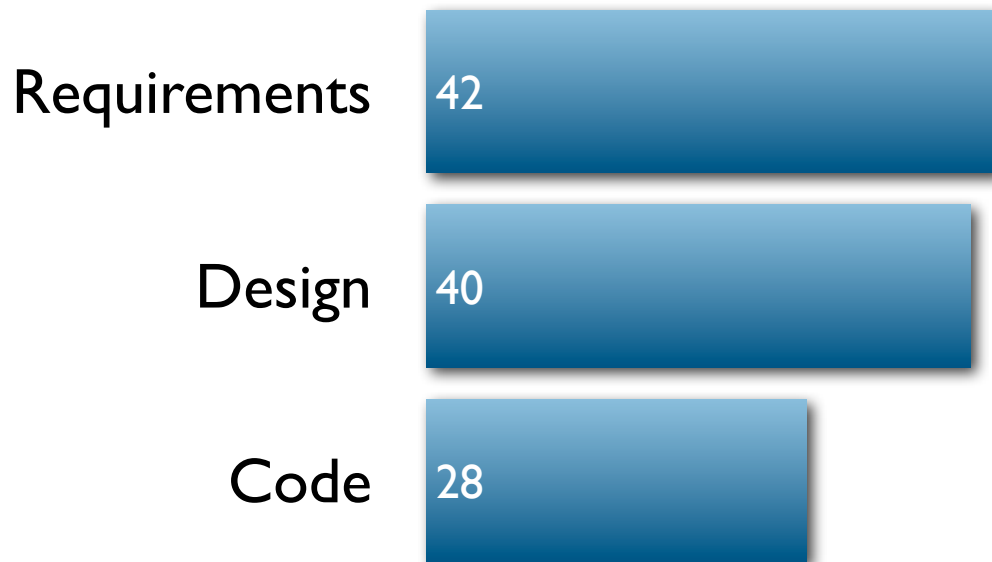
Wagner et al., Quality Models in Practice, 2010

Most reviews and inspections are done at specific milestones in the development process, i.e., only a small number of times.

Some companies, however, use reviews and even inspections on a daily basis.

# Regular reviews of artefacts

Percentage of respondents



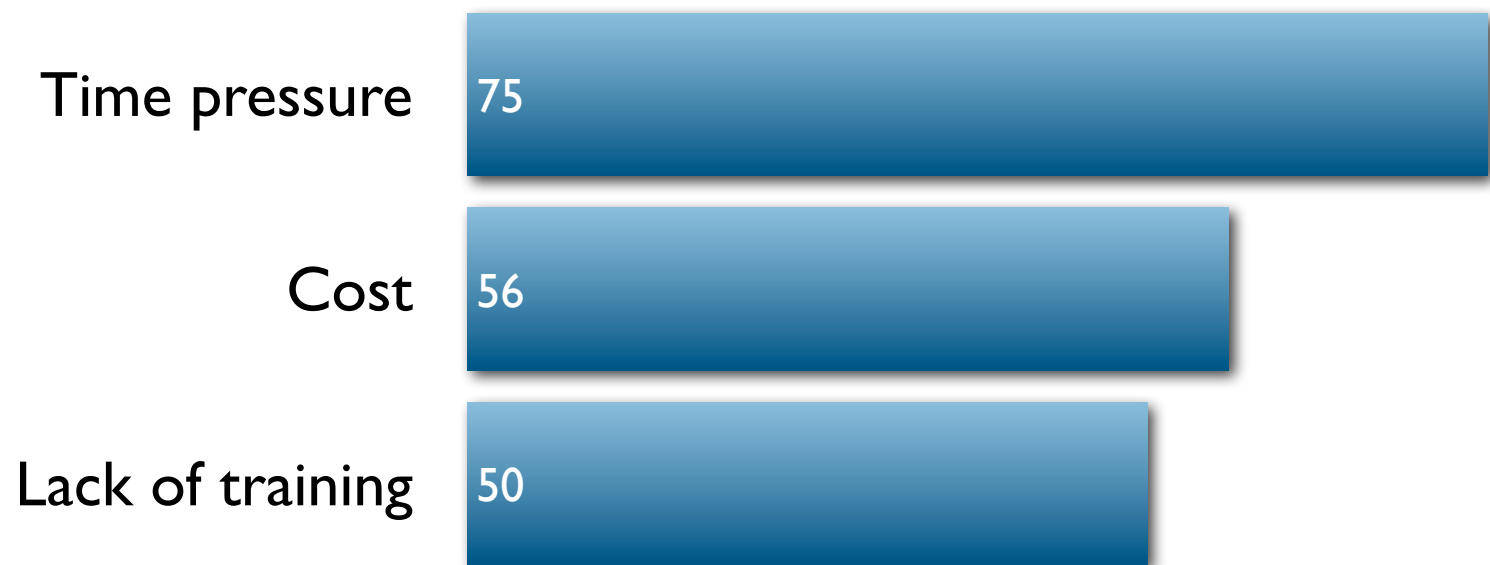
Ciolkowski, Laitenberger, Biffi, Software Reviews: The State of the Practice, 2003

14

Overall, reviews are not well adopted in practice.  
Less than a third of the companies perform regular code reviews.  
The situation is not much better for requirements and design.

# Obstacles to using reviews

Percentage of respondents



Ciolkowski, Laitenberger, Biffi, Software Reviews: The State of the Practice, 2003

15

The major obstacles that people in practice see are time pressure, cost, and lack of training.

## **Group work**

You are responsible to introduce inspections at your company.  
How do you convince your colleagues?

15 minutes

Design poster

Short presentation



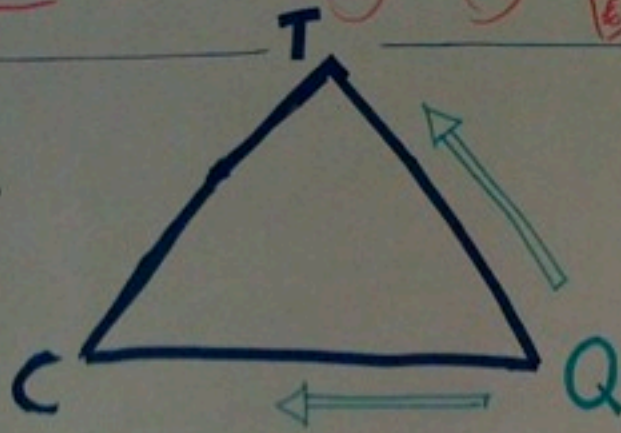
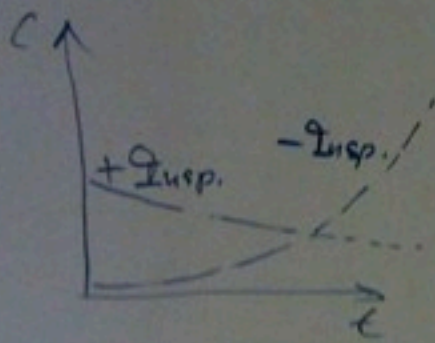
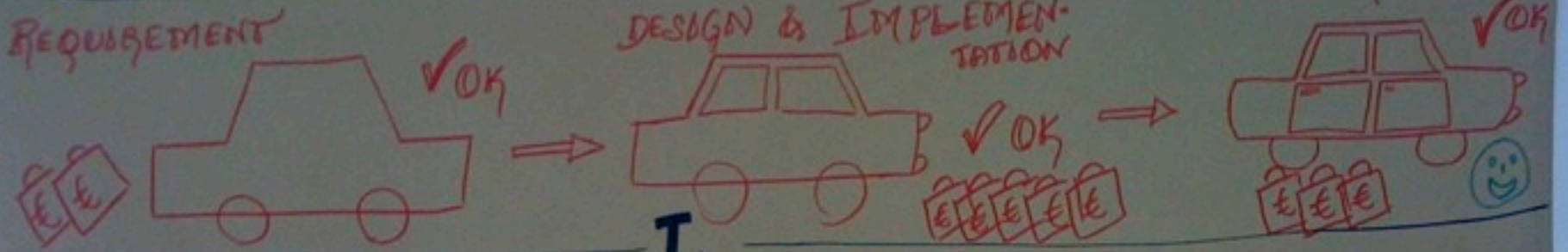
# Why Inspections?



- ① Save Time Later
- ① Save ~~Cost~~ Money
- ① Earlier Feedback
- ① More Readable Code

Early investment pays off later  
Early feedback on the quality  
Improves readability

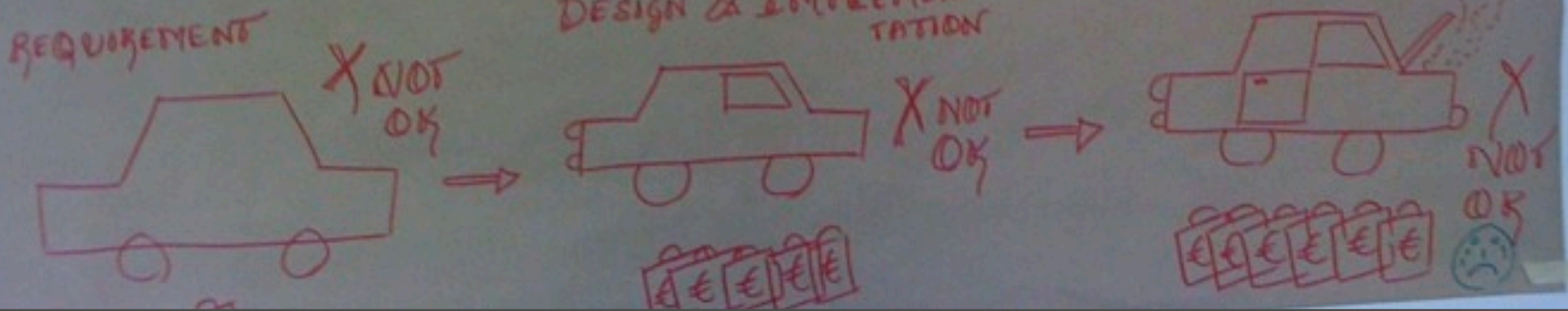
# WITH INSPECTION \*



ROI:  
- INVEST  
- RETURN

REVIEW & INSPECTION  
OVERALL COST ↓,  
QUALITY ↑,

# WITHOUT INSPECTION \*



Early investment pays off later.  
Overall quality is higher although costs are lower.

# Advantages

## Personal :-

- 1) Increased knowledge of coding skills and design skills.
- 2) Increased insight on the mapping between requirements & implementation.
- 3) Some errors are found earlier
  - a) Code redundancy is increased.
  - b) faster bug fixing.
  - c) Personal timelines

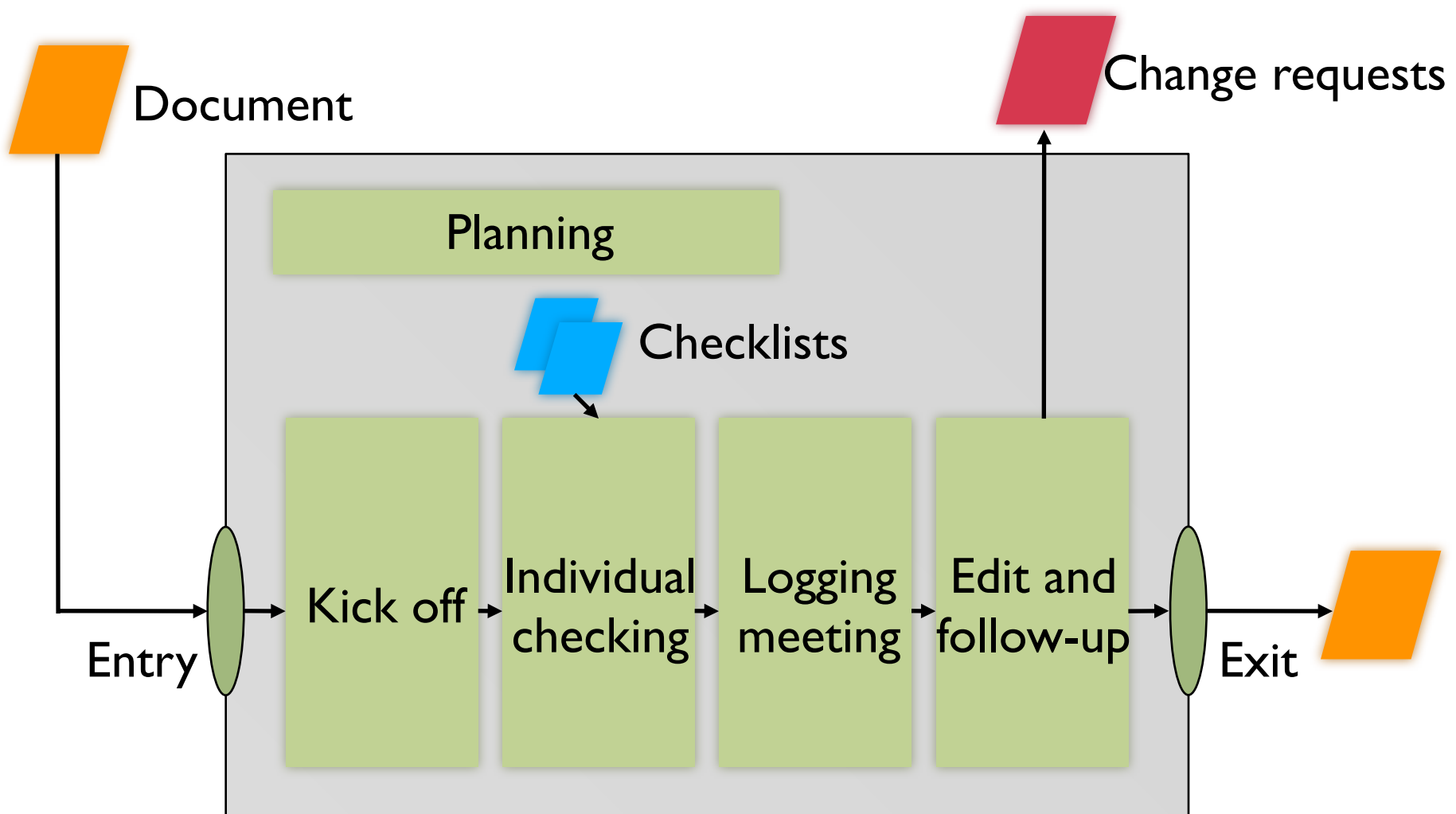
+ faster integration of new employees.

## Enterprise :-

- 1) Lower cost when bugs found early.
- 2) Better customer feedback (external review)
- 3) More insight on the project progress.
- 4) Regular reviews leads to smaller and even more solvable problems.
- 5) Failure recognition of previous errors (in past projects)

The skills of the involved people increase.  
New employees learn fast in reviews.  
Early investment pays off later.  
Better control over the project in early phases.

# Inspection process



Gilb, Graham, Software Inspection, Addison-Wesley, 1993

20

The planning step involves all organisational tasks, e.g., who needs to take part? What will be inspected?

Then it is checked whether the document fulfils the entry criteria, e.g., specific automatic code checks have been performed, the code compiles.

In the kick off meeting, all participants come together to discuss how the inspection will be done. They will receive all necessary material.

Afterwards, all participants check the document individually for defects (or issues). Most often, checklists are used to drive this checking.

In the logging meeting, the individual issues are logged. Sometimes this also involves joint checking.

The edit and follow-up meeting is responsible for issuing change requests for found defects. Here, the document can be scheduled for re-inspection.

If the document fulfils the exit criteria, it is successfully inspected.

# Reading techniques

- Checklist-based Reading
- Perspective-based Reading
- Defect-based Reading
- Usage-based Reading

Basili et al., The Empirical Investigation of Perspective-Based Reading, 1996

21

- Checklist-based reading
  - Defect checking using a checklist
  - Coding guidelines
- Perspective-based reading
  - Reading from the point of view of different roles
  - Designer, developer, maintainer, user, or tester
- Defect-based reading
  - Searching for specific defect classes
  - Incorrect function, interface fault, or type fault
- Usage-based reading
  - Reading following the use cases
  - Needs prioritised use cases

# android open source project



## All open changes

ID	Subject	Owner	Project	Branch	Updated	V	R
I618baaff	<a href="#">debugger: Show function names in tombstone backtraces</a>	Garmin	platform/system/core	master	4:55 PM		
I4c34f617	<a href="#">Fix error detection for invalid command line arguments.</a>	Raphael Moll	platform/sdk	master	4:23 PM	✓	✓
I5aa97b96	<a href="#">Don't allow invalid Uris to be added as observers.</a>	Garmin	platform/frameworks/base	master	2:39 PM		
I914897bf	<a href="#">Fix window leak problems in settings.</a>	Johan Redestig	platform/packages/apps/Settings	master	9:06 AM	✓	-1
I28aa897f	<a href="#">Removed Calls to depracted APIs. Added type Arguments for Type Safety</a>	Christian Mehlmauer	platform/frameworks/base	master	9:05 AM		✗
I2617122e	<a href="#">Start app with several activities with intent filter [Main Launcher] was broken</a>	Johan Redestig	platform/frameworks/base	master	9:04 AM	✓	
I22f9e91c	<a href="#">Pass attachment mime-type from Email app to viewer app</a>	Mirko Nasato	platform/packages/apps/Email	master	8:59 AM		+1
I88c0ce08	<a href="#">Updated Sensor Samples to newest API</a>	Christian Mehlmauer	platform/development	master	8:56 AM	✓	
I17d64997	<a href="#">Changed Sample Sensor Implementation. Cleaned up Android Samples</a>	Christian Mehlmauer	platform/development	master	8:55 AM		✗
Ia855b8b6	<a href="#">Adds the ability to customize the fast scroller drawables</a>	Johan Redestig	platform/frameworks/base	master	8:53 AM		
Iff898474	<a href="#">the mediaserver will crash in snd_pcm_bytes_to_frames while close handle in writ</a>	KennyGong	platform/hardware/alsa_sound	master	8:04 AM		-1
Ib22cde5a	<a href="#">void: reimplement get_uevent_param in the correct way</a>	Chih-Wei Huang	platform/system/core	master	3:22 AM		✗
I69b88989	<a href="#">Porting Unsubmitted Donsut InputMethod Tests</a>	Brian Muramatsu	platform/cts	eclair	3:17 AM		
If973b2ad	<a href="#">Build SDL from sources directly.</a>	David Turner	platform/external/gemu	master	3:08 AM	✓	
I5e0cc1c8	<a href="#">Allow POST with auth (issue 4326).</a>	Jean-Baptiste Quen	platform/libcore	master	2:20 AM	✓	+1
I93d74b65	<a href="#">network: wireless: bcm4329: Add "HANG" event and console monitoring</a>	Dmitry Schmidt	kernel/msm	android-msm-2.6.32	2:13 AM		
Ifa1b9907	<a href="#">Fix data type of RIL_UNSOL_RESPONSE_NEW_BROADCAST_SMS</a>	Johan Redestig	platform/hardware/ril	master	2:08 AM	✓	✓
I9b9b26d1	<a href="#">Fix data type of RIL_UNSOL_RESPONSE_NEW_BROADCAST_SMS in RIL.</a>	Johan Redestig	platform/frameworks/base	master	2:08 AM	✓	+1
I16334dd8	<a href="#">Add a separate queue for non-interactive users</a>	Nico Saitembien	tools/gerrit	master	1:55 AM		
Ie13671e6	<a href="#">Solution for issue of inflating big compressed files.</a>	Sinker	platform/frameworks/base	master	1:06 AM		✗ ✗
Ib0f0184d	<a href="#">Perform zipalign-on-install of applications.</a>	Steve Kondik	platform/frameworks/base	master	1:05 AM	✓	✗
Iee841605	<a href="#">Enabling cell broadcast (SMS-CB) support in the platform.</a>	Johan Redestig	platform/frameworks/base	master	12:29 AM	✓	
I38b06e12	<a href="#">Re-align use of signed/unsigned/long use off_t and re-align format parameters.</a>	Scott Turner	platform/build	master	May 18	✓	
I7fd7867e	<a href="#">PM: eadysuspend: rename work function for readability</a>	vikram.pandita	kernel/common	android-2.6.32	May 18		+1
I9b1b8ce9	<a href="#">Fix "route add default dev &lt;iface&gt;" behaviour</a>	Dani Baeyens	platform/system/core	master	May 18	✓	+1

Next >

The Android open source project uses Gerrit as web based reviewing tool  
[review.source.android.com](http://review.source.android.com)

# Change I8a221cad: Adds support for UBFX to JIT and Disassembler

Change-Id:	I8a221cad8fc13024983e0d6ea401770c90df901c
Owner	<a href="#">David Butcher</a>
Project	<a href="#">platform/system/core</a>
Branch	master
Uploaded	Dec 4, 2009 6:08 PM
Updated	May 19, 2010 6:43 PM
Status	Review in Progress

[Permalink](#) 

Adds support for UBFX to JIT and Disassembler

This introduces UBFX instruction generation abilities to the Pixelflinger JIT, and also modifies the component extraction function to generate the instruction.

The extract function contains defines to prevent generation of UBFX on pre-v7 cores. The JIT itself retains the ability to produce the instruction even on v5/6.

This patch only generates UBFX when MOV, AND or BIC can't be used. Based on the TRM, this appears to be faster on A9 than using UBFX in all cases.

On startup, Pixelflinger JITs three chunks of code. UBFX improves these as follows:

```
00000077:03515104_00000000_00000000
(Blends a single colour into an RGB565 buffer.)
  Before: 27 inst/pixel, After: 24 inst/pixel, Improvement: 12.5%
00000077:03545404_00000A01_00000000
(Blends RGBA8888 texture into an RGB565 buffer using alpha.)
  Before: 30 inst/pixel, After: 27 inst/pixel, Improvement: 11.1%
00000077:03545404_00000A04_00000000
(Blends RGB565 texture into an RGB565 buffer using alpha.)
  Before: 29 inst/pixel, After: 27 inst/pixel, Improvement: 7.4%
```

Reviewer	Verified	Code Review
<a href="#">David Butcher</a>		
<a href="#">Mathias Agopian</a>		✓ Looks good to me, approved
<a href="#">Martyn Capewell</a>		+1 Looks good to me, but someone else must approve
<a href="#">Jean-Baptiste Queru</a>	✗	Fails

- Need Verified +1 (Verified)

## Dependencies

▼ Patch Set 1 [8a221cad8fc13024983e0d6ea401770c90df901c](#) ([gitweb](#))

Author	<a href="#">Martyn Capewell</a> <martyn.capewell@arm.com> Dec 4, 2009 5:44 PM
Committer	<a href="#">Dave Butcher</a> <david.butcher@arm.com> Dec 4, 2009 5:58 PM
Download	<a href="#">repo download</a>   <a href="#">checkout</a>   <a href="#">pull</a>   <a href="#">cherry-pick</a>   <a href="#">patch</a>   <a href="#">diff</a>
	<a href="#">repo download platform/system/core 12709/1</a> <a href="#">diff</a>

	File Path	Comments	Diff
▶ M	<a href="#">libpixelflinger/codeflinger/ARMAsembler.cpp</a>		<a href="#">Side-by-Side</a> <a href="#">Unified</a>

Overview of a change with description, change set, and responsible reviewers.



Change I39a4eb2b: samples/ApiDemos/src/com/example/android/apis/app/MyPreference.java

Patch History

Ignore Whitespace:   Syntax Coloring  Whitespace Errors  
 Intra-line Difference  Show Tabs  
 Show Full File

[MenuInflateFromXml.java](#)

[Up to change](#)

[NotificationDisplay.java](#)

Old Version (Download)	New Version (Download)
(... skipping 148 common lines ...)	
149 super.writeToParcel(dest, flags);	149 super.writeToParcel(dest, flags);
150	150
151 // Save the click counter	151 // Save the click counter
152 dest.writeInt(clickCounter);	152 dest.writeInt(clickCounter);
153 }	153 }
154	154
155 public SavedState(Parcelable superState) {	155 public SavedState(Parcelable superState) {
156 super(superState);	156 super(superState);
157 }	157 }
158	158
	159 @SuppressWarnings("unused")
	<b>Romain Guy</b> Remove this warning, this is used. 7:08 AM
	<b>Christian Mehlmauer</b> I know its used, but eclipse does not know. Thats why I added the ... 7:40 AM
	<b>Romain Guy</b> Then it's not needed, please remove it. 7:43 AM
159 public static final Parcelable.Creator<SavedState> CREATOR =	160 public static final Parcelable.Creator<SavedState> CREATOR =
160 new Parcelable.Creator<SavedState>() {	161 new Parcelable.Creator<SavedState>() {
161 public SavedState createFromParcel(Parcel in) {	162 public SavedState createFromParcel(Parcel in) {
162 return new SavedState(in);	163 return new SavedState(in);
163 }	164 }
164	165
165 public SavedState[] newArray(int size) {	166 public SavedState[] newArray(int size) {
166 return new SavedState[size];	167 return new SavedState[size];
167 }	168 }
168 };	169 };
(... skipping 3 common lines ...)	

[MenuInflateFromXml.java](#)

[Up to change](#)

[NotificationDisplay.java](#)

Side-by-side diff view of the change in one file with inline comments from the reviewers.



The Google logo is displayed in its characteristic multi-colored font. The letters are: 'G' (blue), 'O' (red), 'O' (yellow), 'g' (blue), 'l' (green), and 'e' (red). Each letter has a slight 3D effect with a shadow.

# Mondrian

# **Inspection Review Walkthrough**

