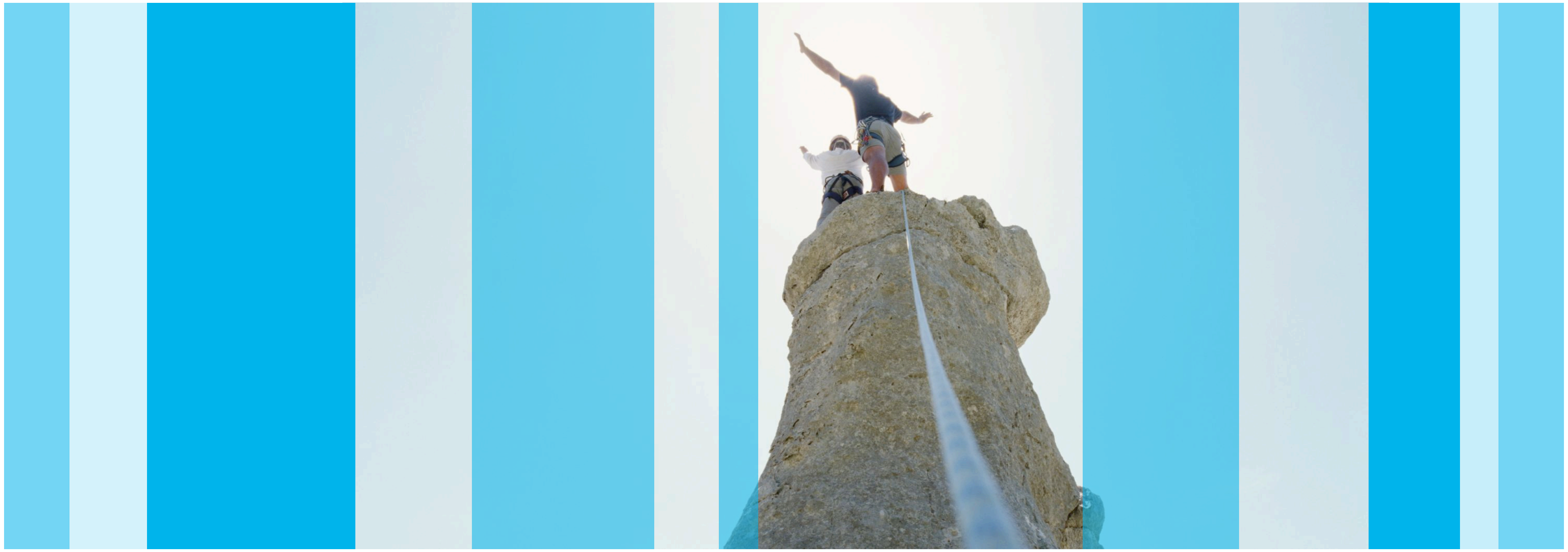
Software Improvement Group



Observing Unit Test Maturity in the Wild

Ilja Heitlager, Managing Consultant Software Risk Assessments

November 29, 2007

Arent Janszoon Ernststraat 595-H
NL-1082 LD Amsterdam
info@sig.nl
www.sig.nl

Software Improvement Group

Company

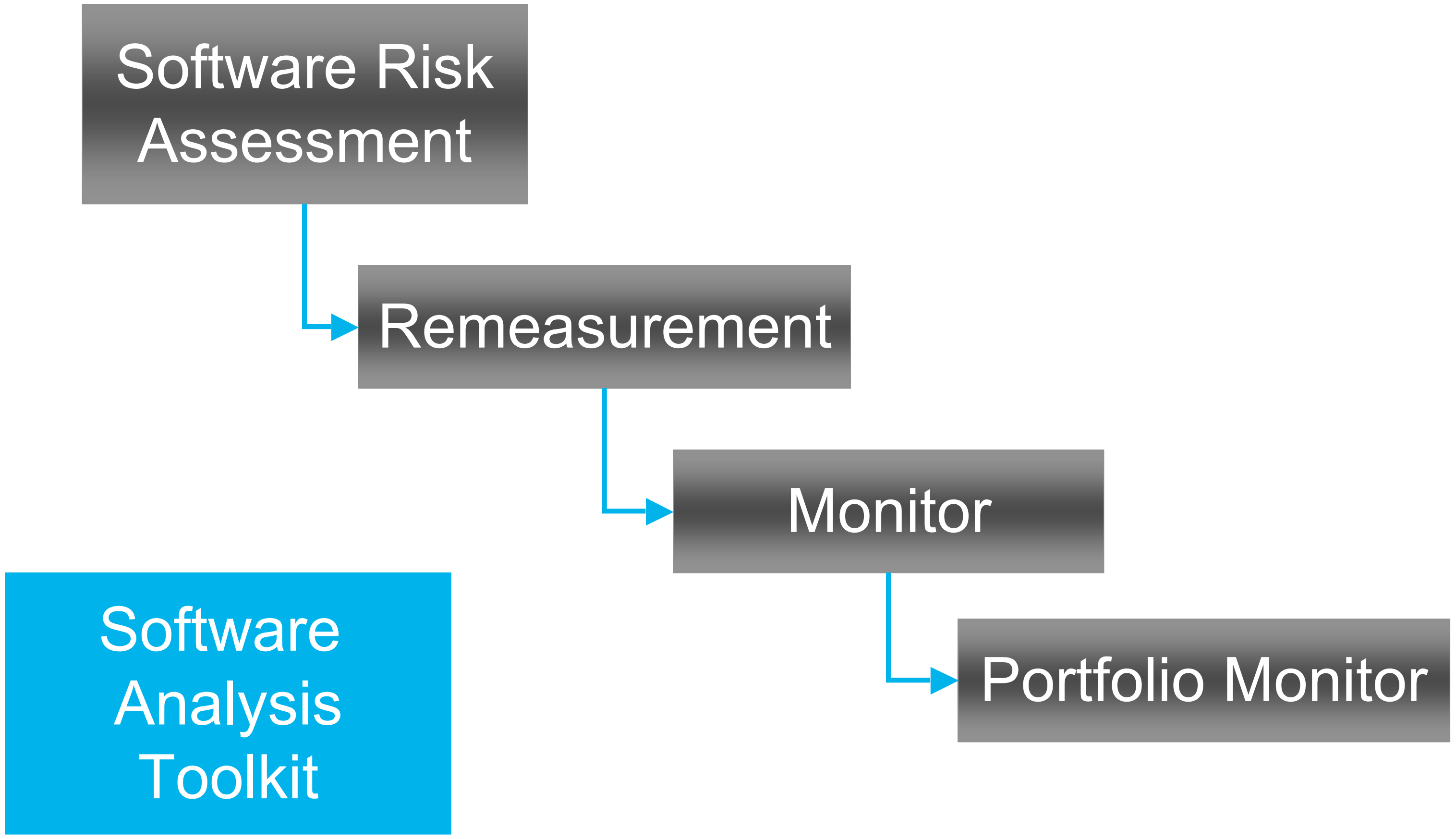
- Spin-off from CWI in 2000, self-owned, independent
- Management consultancy grounded in source code analysis
- Winner of the Innovator Award 2007

Services

- Software Risk Assessments (snapshot) and Software Monitoring (continuous)
- Toolset enables to analyze source code in an automated manner
- Experienced staff transforms analysis data into recommendations
- We analyze over 50 systems annually
- Focus on technical quality, primarily maintainability / evolvability

Our services

Software Improvement Group



Who is using our services?

Software Improvement Group

Financials / Insurance companies

ABN·AMRO

PGGM

ING

Zorg en Zekerheid

Rabobank

InterBank
KREDIET WAAR JE LANGER MEE GENIET

achmea

ZwitserLeven

Friesland Bank

delta lloyd

LeasePlan

Allianz

Government

RVR
Raden Voor Rechtsbijstand

VROM

RDW

POLITIE

cwi

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

S V B

kadaster

Logistical

KLM

DHL

TNT

EUROMAX
TERMINAL

norfolkline
DOVER - DUNKERQUE FERRIES

ProRail

IT

Getronics PinkRoccade

CENTRIC

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Exact software

IBM

CHESS

logicaCMG

Other

ENECO energie

PRICEWATERHOUSECOOPERS

CARe
SCHADESERVICE

KPMG

oxxio

Gasunie

What is unit testing

Software Improvement Group

An automated unit test is an additional unit of software that is

- reduces risk by making the system bug repellent
- fully automated and repeatable
- easy to write and maintain
- non intrusive and does no harm
- documenting
- applies to the simplest piece of software

Why unit testing?

Software Improvement Group

Build Quality In:

If you routinely find defects in your verification process, your process is defective.

Mistake-Proof Code with Test-Driven Development

Write executable specifications instead of requirements.

Stop Building Legacy Code

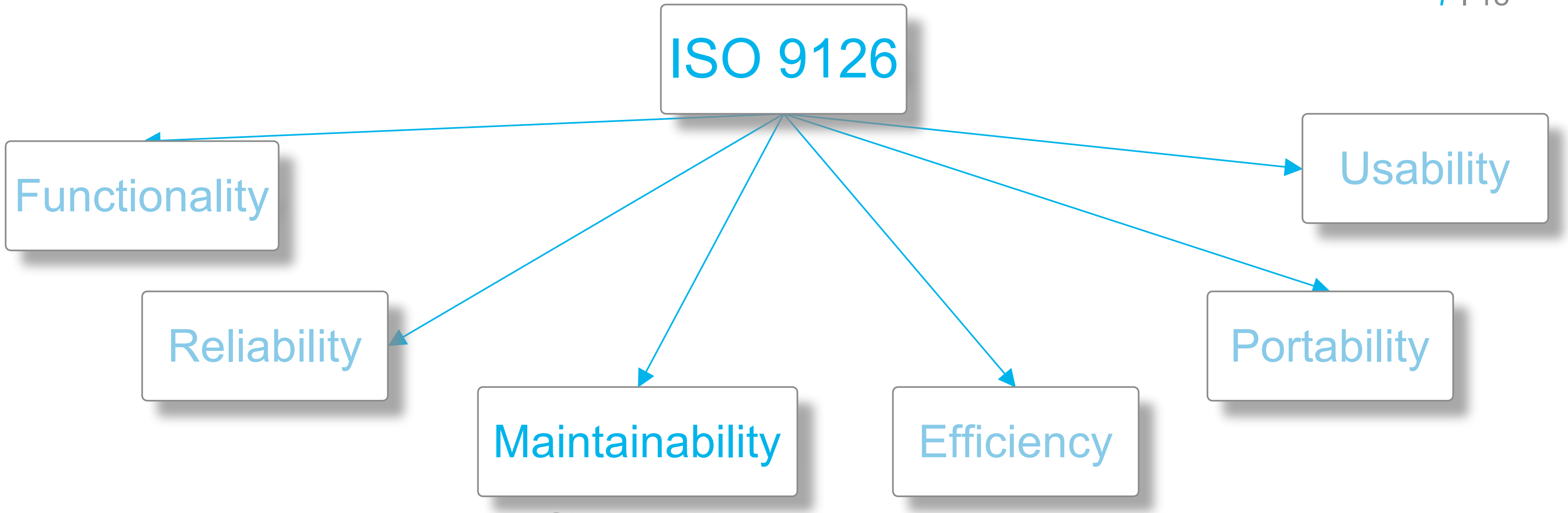
Legacy code is code that lacks automated unit and acceptance tests.

The Big Bang is Obsolete

Use continuous integration and nested synchronization.

Mary Poppendieck, Lean Software Development Principles

Role of unit testing
ISO 9126



- Changeability
- Analysability
- Stability
- Testability

Our experience base

Software Improvement Group

Organization

- public, financial, logistics
- under contract, in house, product software
- with test departments, without test departments

Architecture & Process

- under architecture, using software factories
- model driven, handwritten
- open source frameworks, other frameworks
- using use-cases/requirements
- with blackbox tools, t-map

Technology

- information systems, embedded
- webbased, desktop apps
- java, c#, 4GL’s, legacy
- latest trend: in-code asserts (java.spring)

Stage 1
No unit testing

Software Improvement Group

Observations:

- Very few organizations use unit testing
- Also brand new OO systems without any unit tests
- Small software shops and internal IT departments
- In legacy environments: programmers describe in words what tests they have done.

Symptoms:

- Code is instable and error-prone
- Lots of effort in post-development testing phases

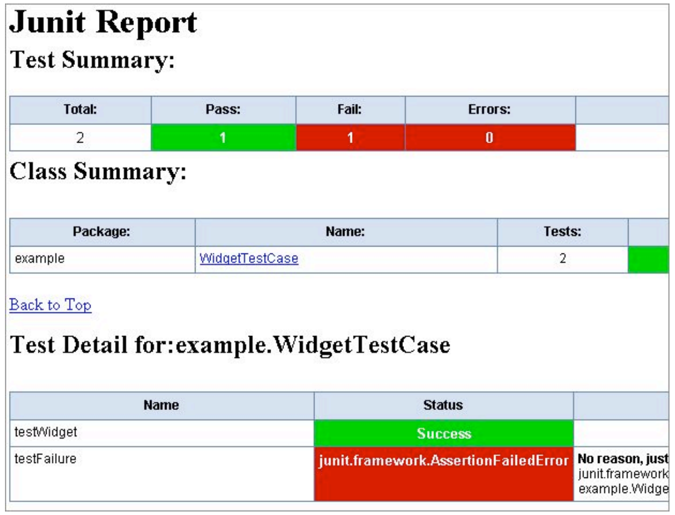
Stage 1
No unit testing

Excuses:

- “It is just additional code to maintain”
- “The code is changing too much”
- “We have a testing department”
- “Testing can never prove the absence of errors”
- “Testing is too expensive, the customer does not want to pay for it”
- “We have black-box testing”

Action

- Provide standardized framework to lower threshold
- Pay for unit tests as deliverable, not as effort



Junit Report

Test Summary:

Total:	Pass:	Fail:	Errors:	
2	1	1	0	

Class Summary:

Package:	Name:	Tests:	
example	WidgetTestCase	2	🟩

Back to Top

Test Detail for:example.WidgetTestCase

Name	Status	
testWidget	Success	
testFailure	junit.framework.AssertionFailedError	**No reason, just** junit.framework example.Widge

Stage 2
Unit test but no coverage measurement

Software Improvement Group

Observations

- Contract requires unit testing, not enforced
- Revealed during conflicts
- Unit testing receives low priority
- Developers relapse into debugging practices without unit testing
- Good initial intentions, bad execution
- Large service providers

Symptoms:

- Some unit tests available
- Excluded from daily build
- No indication when unit testing is sufficient
- Producing unit test is an option, not a requirement

Stage 2
Unit test but no coverage measurement

Excuses:

- “There is no time, we are under pressure”
- “We are constantly stopped to fix bugs”

Actions

- Start measuring coverage
- Include coverage measurement into nightly build
- Include coverage result reports into process

CLOVER
The industry standard in code coverage just got seriously better

Stage 3
Coverage, not approaching 100%

Software Improvement Group

Observations

- Coverage is measured but gets stuck at 20%-50%
- Ambitious teams, lacking experience
- Code is not structured to be easily unit-testable

Symptoms:

- Complex code in GUI layer
- Libraries in daily build, custom code not in daily build

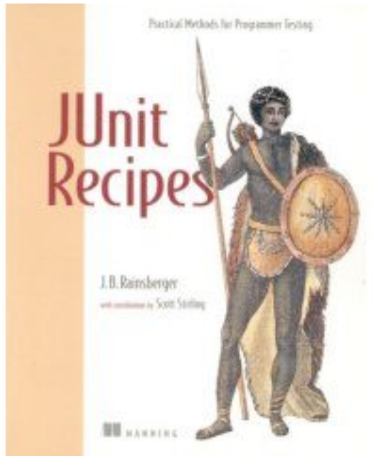
Stage 3
Coverage, not approaching 100%

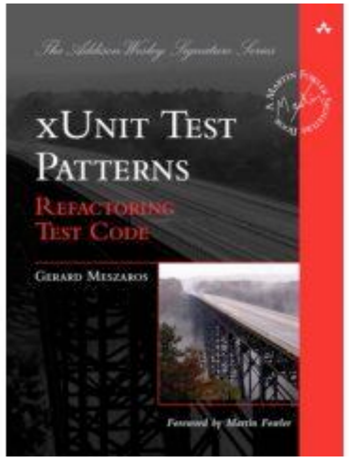
Excuses

- “we test our libraries thoroughly, that effects more customers”
- “Our software is hard to unit test”

Actions:

- Refactor code to make it more easily testable
- Teach advance unit testing patterns
- Invest in set-up and mock-up




The Addison-Wesley Signature Series
xUnit Test Patterns
Refactoring Test Code
Gerard Meszaros
Foreword by Martin Fowler

Stage 4
Approaching 100%, but no test quality

Software Improvement Group

Observations

- Formal compliance with contract
- Gaming the metrics
- Off-shored, certified, bureaucratic software factories

Symptoms:

- Empty tests
- Tests without asserts.
- Tests on high-level methods, rather than basic units

- Need unit tests to test unit tests

Stage 4
Approaching 100%, but no test quality

Software Improvement Group

Anecdotes:

- "We have generated our unit tests (at first this seems a stupid idea)"
- Tell me how you measure me, and I tell you how I behave

Action:

- Measure test quality (statically)
- Number of asserts per unit test
- Number of statements tested per unit test
- Ratio of number of execution paths versus number of tests

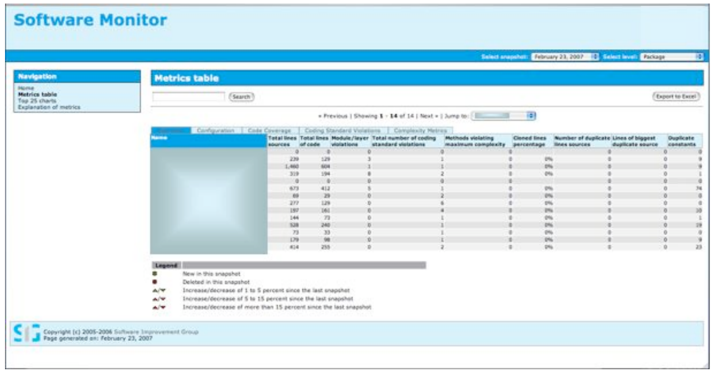
Stage 5
Measuring test quality

Software Improvement Group

Enlightenment:

- “We don’t know how to do without”

- Measure statically:
  - Production code incorporated in tests
  - number of assert and fail statements
  - low complexity of tests (not too many ifs)



Metrics table

« Previous | Showing 1 - 14 of 14 | Next » | Jump to:

Name	Total lines sources	Total lines of code	Module/layer violations	Total number of coding standard violations	Methods violating maximum complexity	Cloned lines percentage	Number of duplicate lines sources	Lines of biggest duplicate source	Duplicate constants
	0	0	0	0	0		0	0	0
	239	129	3	1	0	0%	0	0	9
	1,460	604	1	1	0	0%	0	0	9
	319	194	8	2	0	0%	0	0	1
	0	0	0	0	0		0	0	0
	673	412	5	1	0	0%	0	0	74
	69	29	0	2	0	0%	0	0	0
	277	129	0	6	0	0%	0	0	0
	197	161	0	4	0	0%	0	0	10
	144	73	0	1	0	0%	0	0	1
	528	240	0	1	0	0%	0	0	19
	73	33	0	1	0	0%	0	0	0
	179	98	0	1	0	0%	0	0	9
	414	255	0	2	0	0%	0	0	23

Legend	
●	New in this snapshot
●	Deleted in this snapshot
▲/▼	Increase/decrease of 1 to 5 percent since the last snapshot
▲/▼	Increase/decrease of 5 to 15 percent since the last snapshot
▲/▼	Increase/decrease of more than 15 percent since the last snapshot

Copyright (c) 2005-2006 Software Improvement Group
Page generated on: February 23, 2007

- The process
  - part of daily automated build
  - “stop the line process”, fix bugs first by adding more tests
  - happy path and exceptions testing
  - code first, test first, either way

Conclusion

Software Improvement Group

Unit test is an essential technique, however

- Unit testing is a close interaction between:
  - Software architecture
  - Testing framework
  - Process
  - and Code monitoring (both static and dynamic)
- There is no immediate acceptance
- We observe and classify in five stages of acceptance
- Each stage has its own excuses and counter actions
- Companies of various backgrounds and sizes are at different levels