Software Measurement Pitfalls & Best Practices

@EricBouwers
@avandeursen
@jstvssr

Software Improvement Group
Radboud University Nijmegen
TU Delft Delft University of Technology

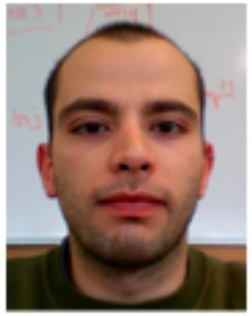
Introductions

It takes a village ...

Tiago Alves

Jose Pedro Correira

Christiaan Ypma



Miguel Ferreira

Ilja Heitlager

Tobias Kuipers

Bart Luijten

Dennis Bijlsma

And what about you?

Why talk about software measurement?

'You can't control what you can't measure.'

DeMarco, Tom. *Controlling Software Projects: Management, Measurement and Estimation*. ISBN 0-13-171711-1.

Why talk about software measurement?

‘You can’t improve what you can't measure.’

Software measurement is used for:

- Cost and effort estimation
- Productivity measures and models
- Data collection
- Reliability models
- Performance evaluation and models
- Structural and complexity metrics
- Capability-maturity assessments
- Management by metrics
- Evaluation of methods and tools
- Quality models and measures

Fenton et. al., *Software Metrics: a rigorous and practical approach*. ISBN 0534954251

Which software measures do you use?

(Software) Measurement

What is measurement?

*'Formally, we define **measurement** as a mapping from the empirical world to the formal, relational world. '*

*'A **measure** is the number or symbol assigned to an entity by this mapping in order to characterize an attribute'*

Fenton et. al., *Software Metrics: a rigorous and practical approach*. ISBN 0534954251

Entity

Attribute

Mapping

Measure

Entities

Product:

- **Specifications, Architecture diagrams, Designs, Code, Test Data, ...**

Process:

- **Constructing specification, Detailed design, Testing, ...**

Resources:

- **Personnel, Teams, Software, Hardware, Offices, ...**

Fenton et. al., *Software Metrics: a rigorous and practical approach*. ISBN 0534954251

Attributes

External

Usability, Reliability

Internal

Size, Structuredness,
Functionality

Mapping

Definition Checklist for Source Statement Counts

Definition name: *Physical Source Lines of Code (basic definition)*

Date: *8/7/92*

Originator: *SEI*

Measurement unit: **Physical source lines** [4] **Logical source statements** [ ]

	Statement type	**Definition** [4] **Data array** [ ]		**Includes**	**Excludes**
	When a line or statement contains more than one type, classify it as the type with the highest precedence.				
1	Executable	**Order of precedence ->**	1	4	
2	Nonexecutable				
3	Declarations		2	4	
4	Compiler directives		3	4	
5	Comments				
6	On their own lines		4		4
7	On lines with source code		5		4
8	Banners and nonblank spacers		6		4
9	Blank (empty) comments		7		4
10	Blank lines		8		4
11					

Park, Rober E., *Software Size Measurement: A Framework for Counting Source Statements,* Technical Report CMU/SEI-92-TR-020

Representation Condition

Attribute: Size

Measure - scale types

Scale type	Allowed operations	Example
Nominal	= , ≠	A, B, C, D, E
Ordinal	= , ≠, < , >	Small, large
Interval	= , ≠, < , > , + , -	Start date
Ratio	All	LOC
Absolute	All	-

Concept summary

Attribute (Length)

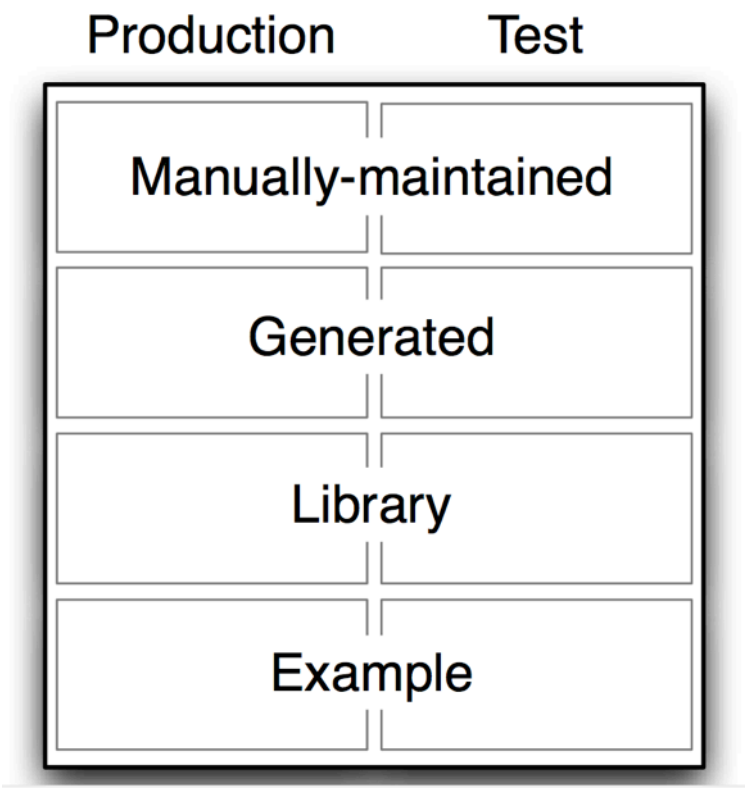
Entity (Child)

Measure (cm)

Mapping (feet on ground)

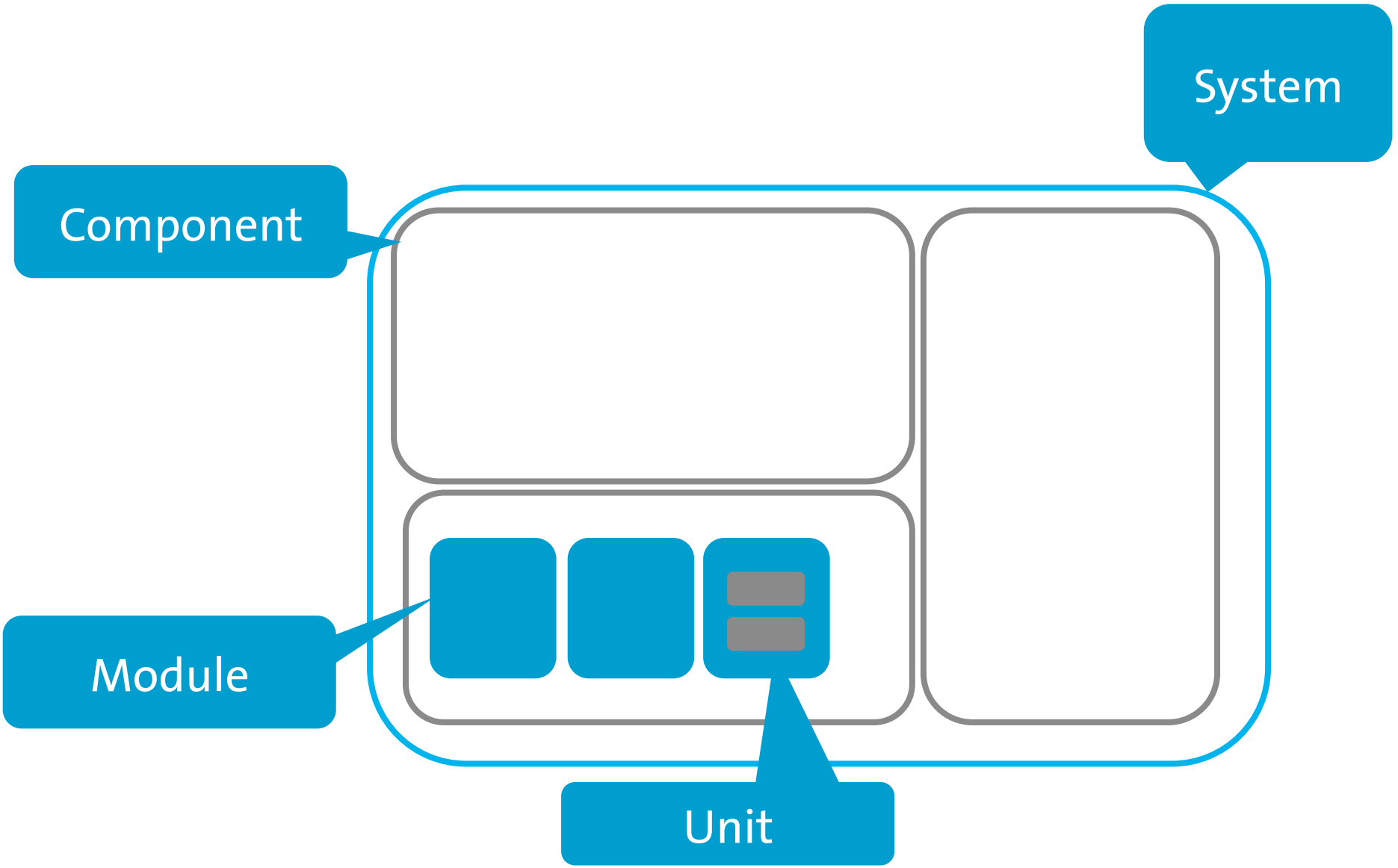
Why does this matter?

It determines what you want ...



Alves. *Categories of Source Code in Industrial Systems,* ESEM 2011: 335-338

It determines who wants it ...

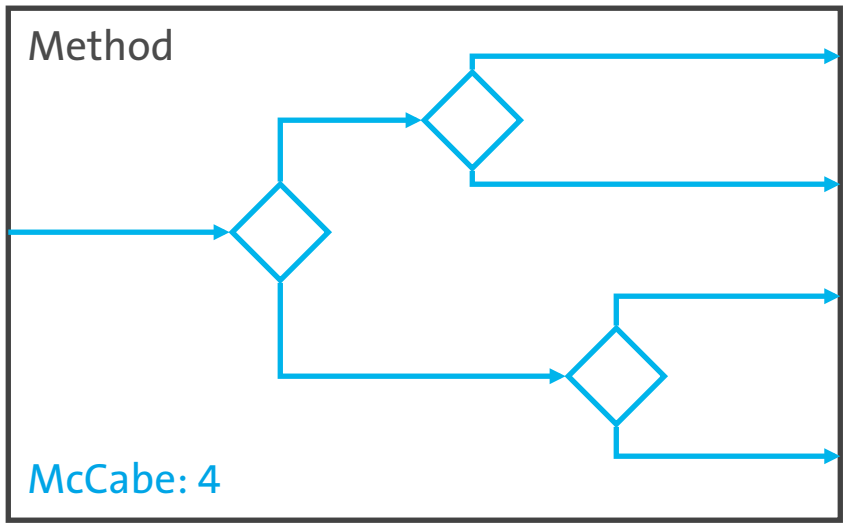


Aggregation exercise

From Unit to System

Unit measurement:
T. McCabe, *IEEE Transactions on Software Engineering*, 1976

- Academic: number of independent paths per method
- Intuitive: number of decisions made in a method
- Reality: the number of if statements (and while, for, ...)



Available data

For four projects, per unit:

- **Lines of Code**
- **McCabe complexity**

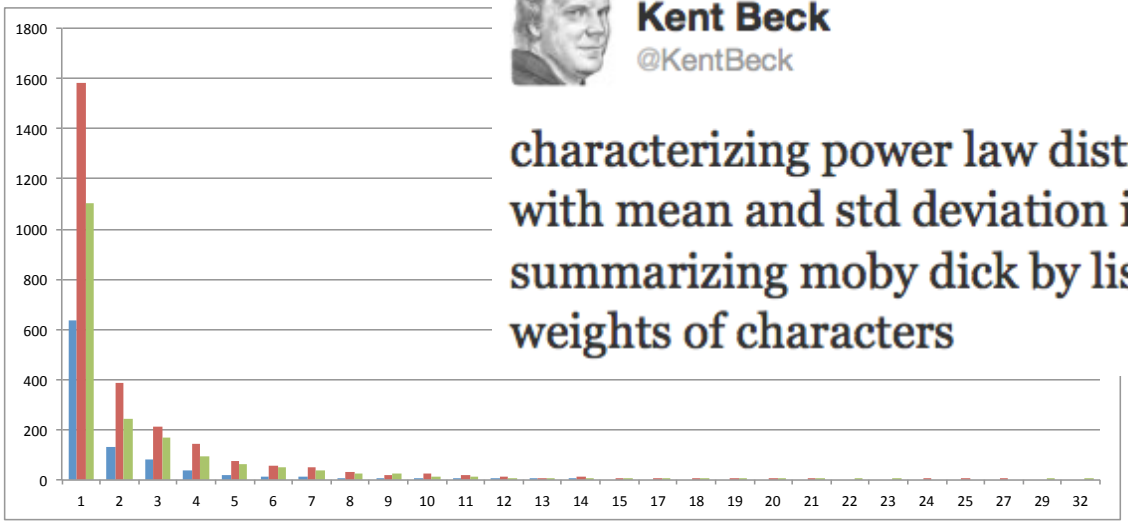
In which system is unit testing a given method the most challenging?

Option 1: Summing

	Crawljax	GOAL	Checkstyle	Springframework
Total McCabe	1814	6560	4611	22937
Total LOC	6972	25312	15994	79474
Ratio	0,260	0,259	0,288	0,288

Option 2: Average

	Crawljax	GOAL	Checkstyle	Springframework
Average McCabe	1,87	2,45	2,46	1,99



Kent Beck
@KentBeck

Following

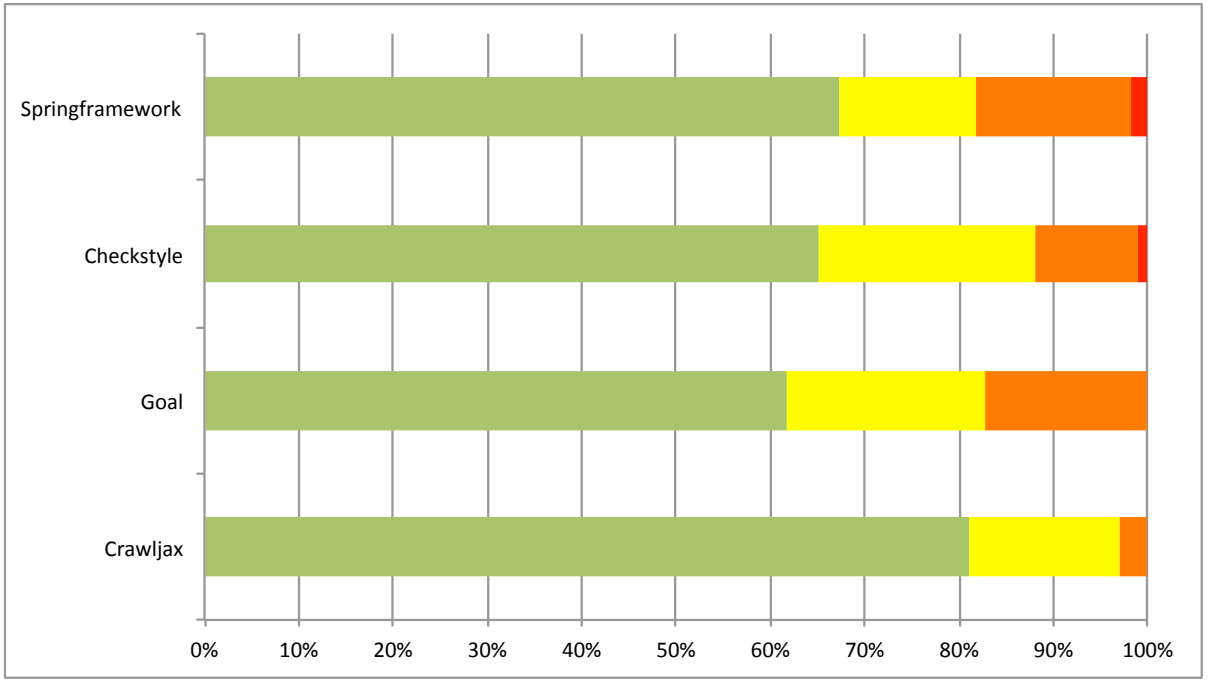
characterizing power law distributed data with mean and std deviation is like summarizing moby dick by listing the weights of characters

Option 3: Quality profile

Cyclomatic complexity	Risk category
1 - 5	Low
6 - 10	Moderate
11 - 25	High
> 25	Very high

Sum lines of code per category

Lines of code per risk category			
Low	Moderate	High	Very high
70 %	12 %	13 %	5 %



Always take into account

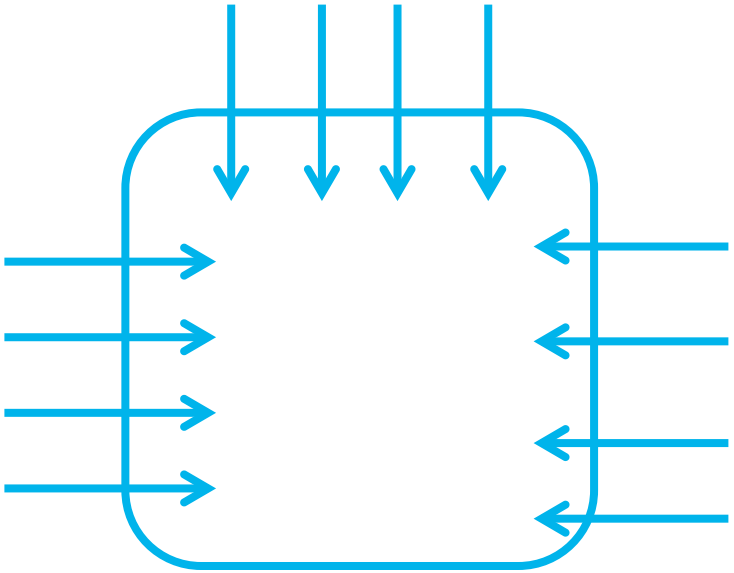
- Volume
- Explanation
- Distribution

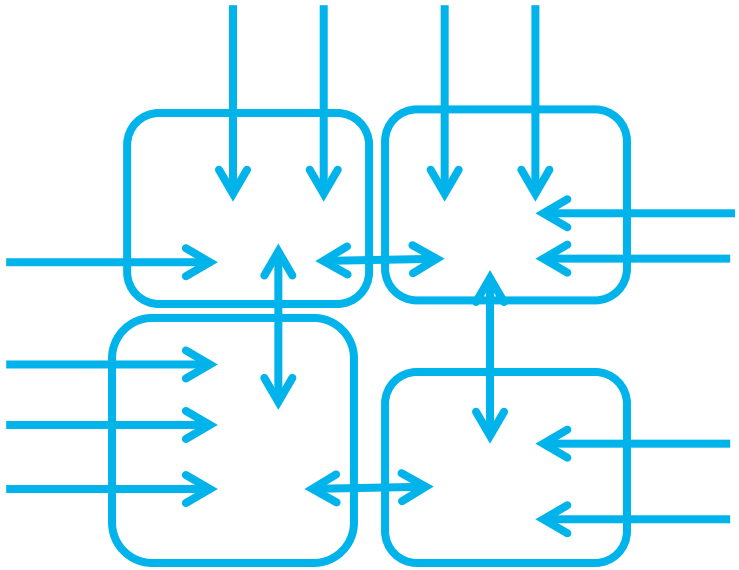
Pitfalls in using measurements

What are they and how to counter them?

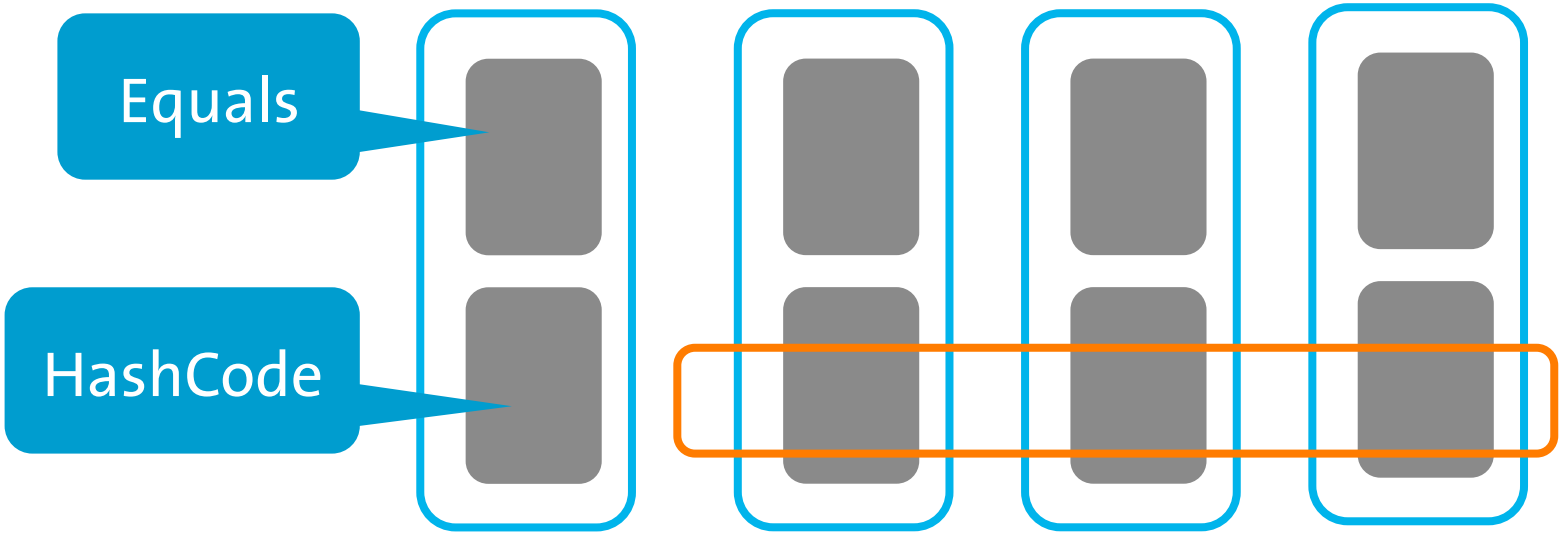
One-track metric

When only looking at size …



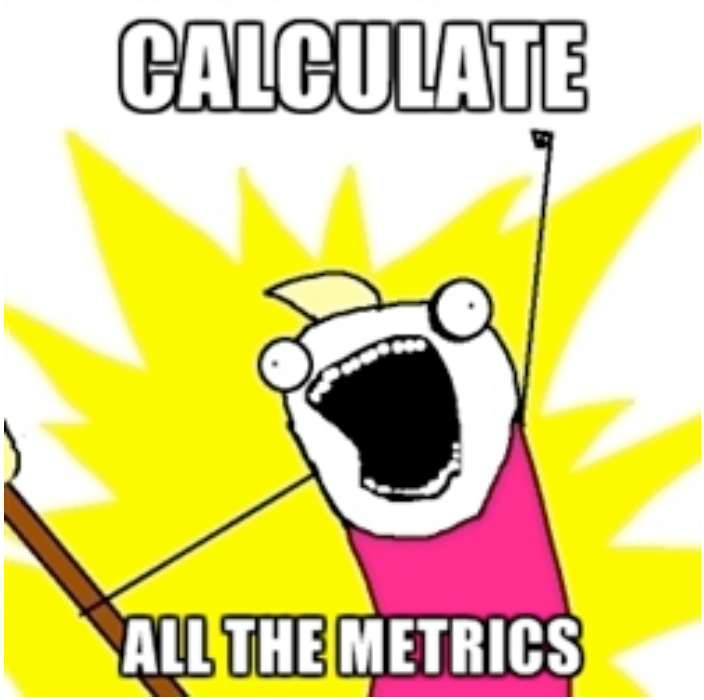


Combination shows problems on different levels



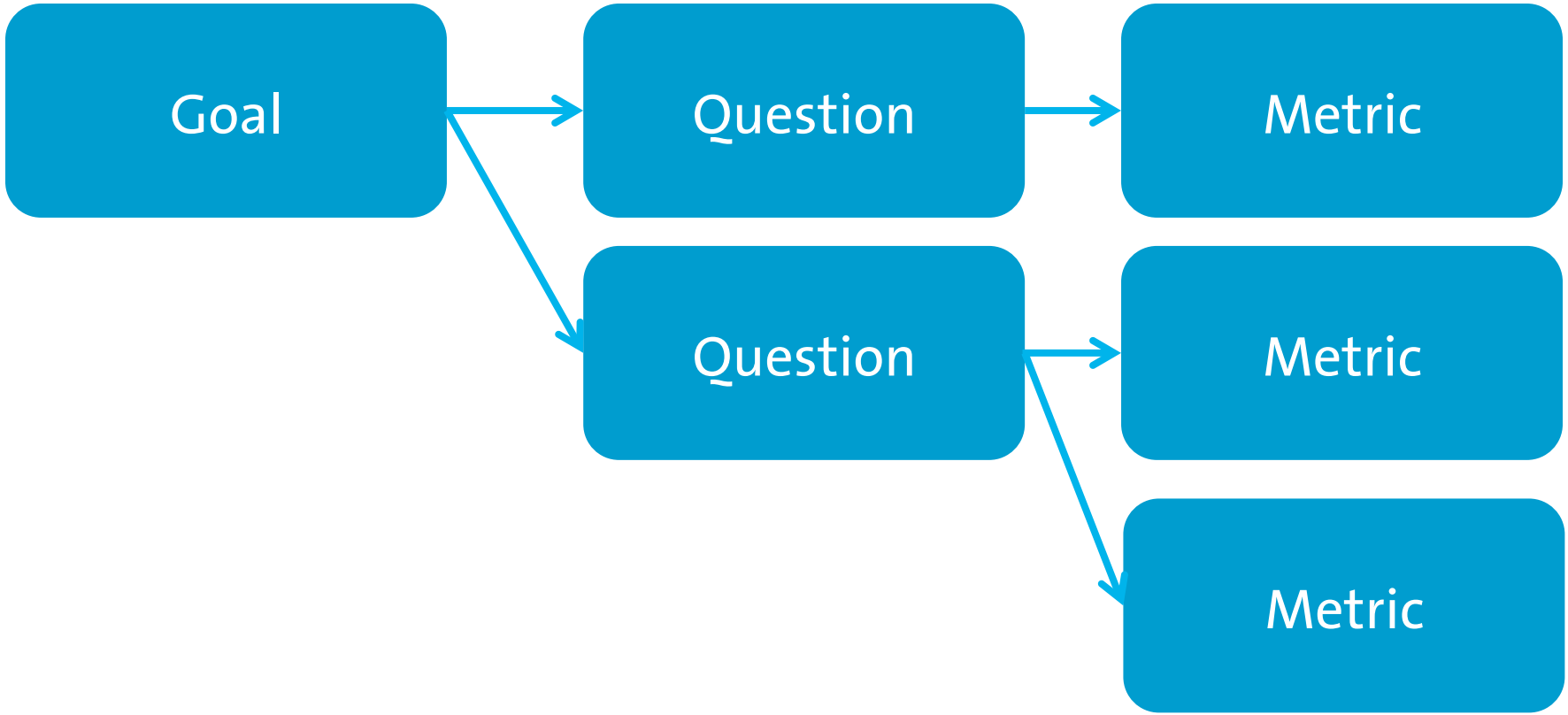

```
private String aap;
private Number noot;
private Date mies;

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((aap == null) ? 0 : aap.hashCode());
    result = prime * result + ((mies == null) ? 0 : mies.hashCode());
    result = prime * result + ((noot == null) ? 0 : noot.hashCode());
    return result;
}
```

Metrics Galore



So what should we measure?

GQM



Basili, Et. Al. , *The Goal Question Metric Approach*, Chapter in Encyclopedia of Software Engineering, Wiley, 1994.

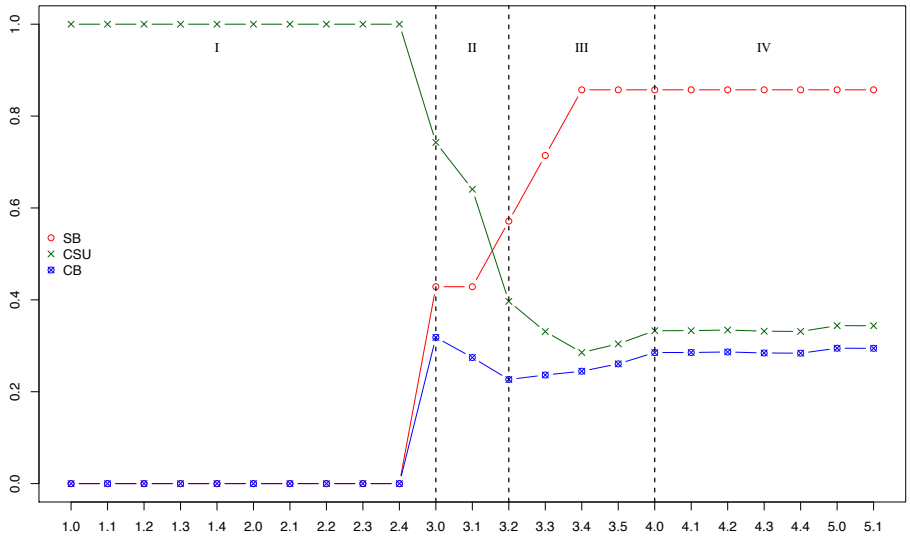
GQM - Example

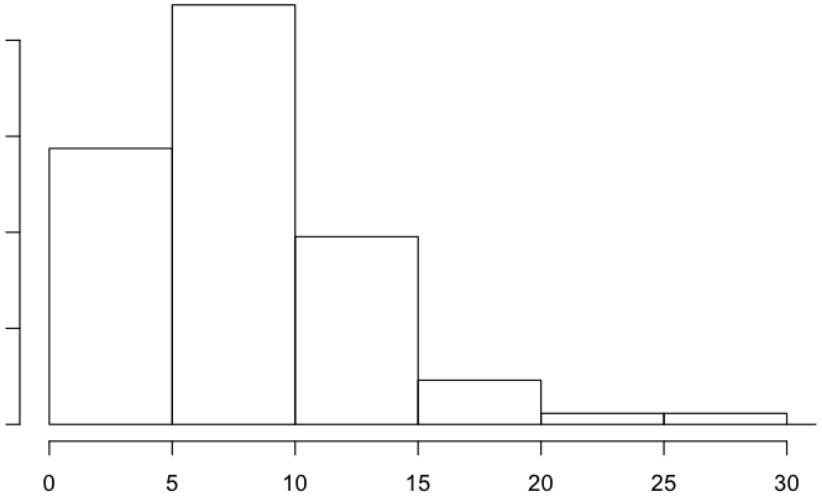
Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question	Q1	What is the current change request processing speed?
Metrics	M1 M2 M3	Average cycle time Standard deviation % cases outside of the upper limit
Question	Q2	Is the (documented) change request process actually performed?
Metrics	M4 M5	Subjective rating by the project manager % of exceptions identified during reviews

http://www.cs.umd.edu/users/mvz/handouts/gqm.pdf

Treating the metric

Metric in a bubble





Software Quality as a goal

Software Quality waves

1970s

- Higher-order languages

1980s

- Design methods and tools (CASE)

1990s

- Process

2000s

- Product

2010s

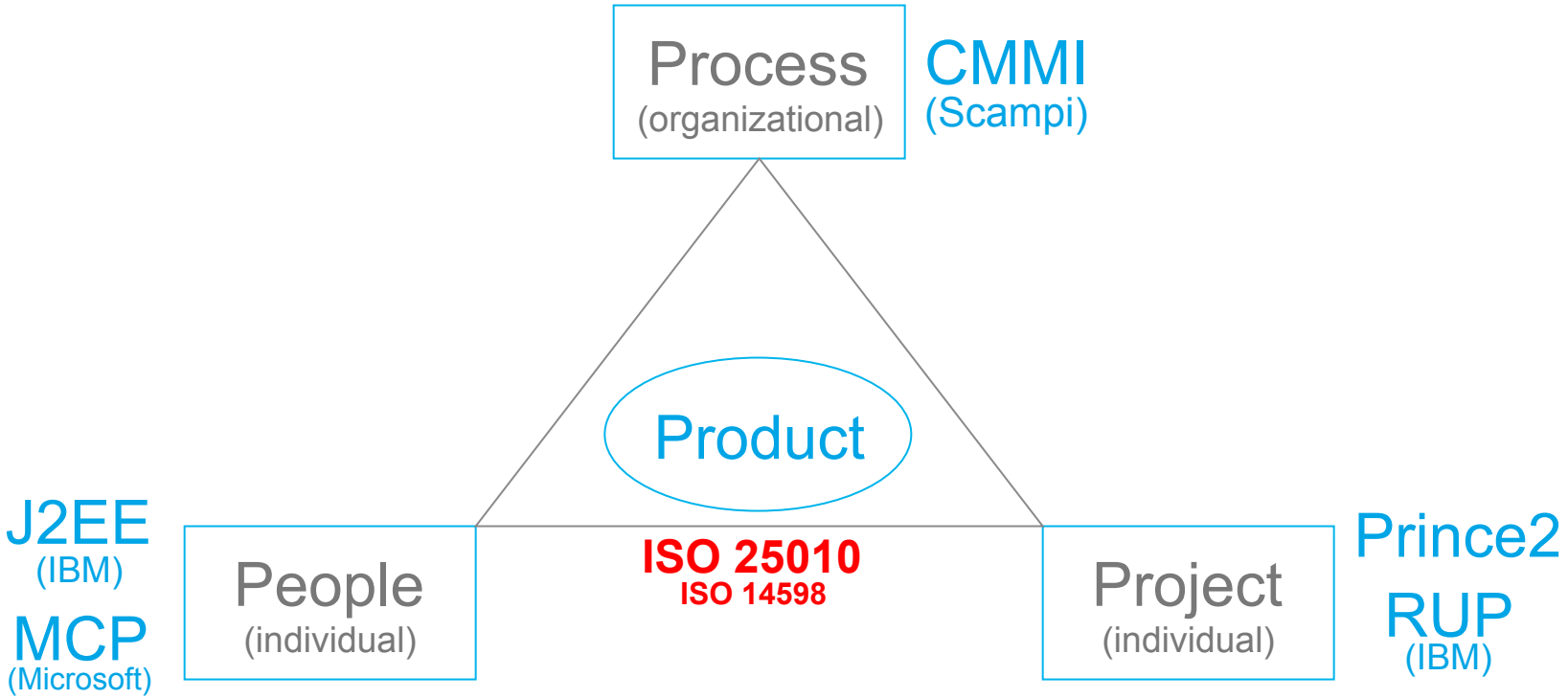
- ?

"Focus on the product to improve quality"

"The CMM [..] is not a quality standard"

Bill Curtis
co-author of the *Capability Maturity Model (CMM)*,
the *People CMM*, and the *Business Process Maturity Model*.
In "CMMI: Good process doesn't always lead to good quality"
Interview for TechTarget, June 2008
http://tinyurl.com/process-vs-quality

The Bermuda Triangle of Software Quality



ISO 25010

Software product quality standards

Previous: separate standards

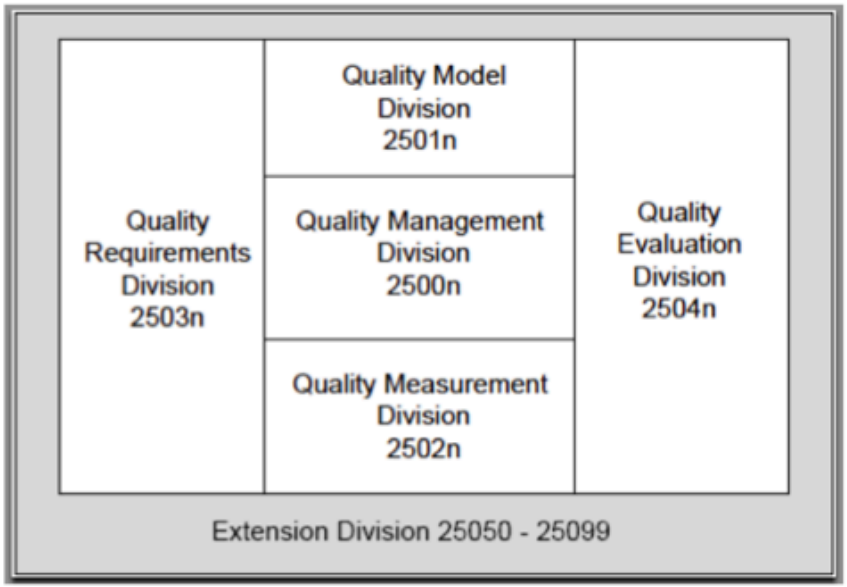
ISO/IEC 9126

1. Quality model
2. External metrics
3. Internal metrics
4. Quality in use

ISO/IEC 14598

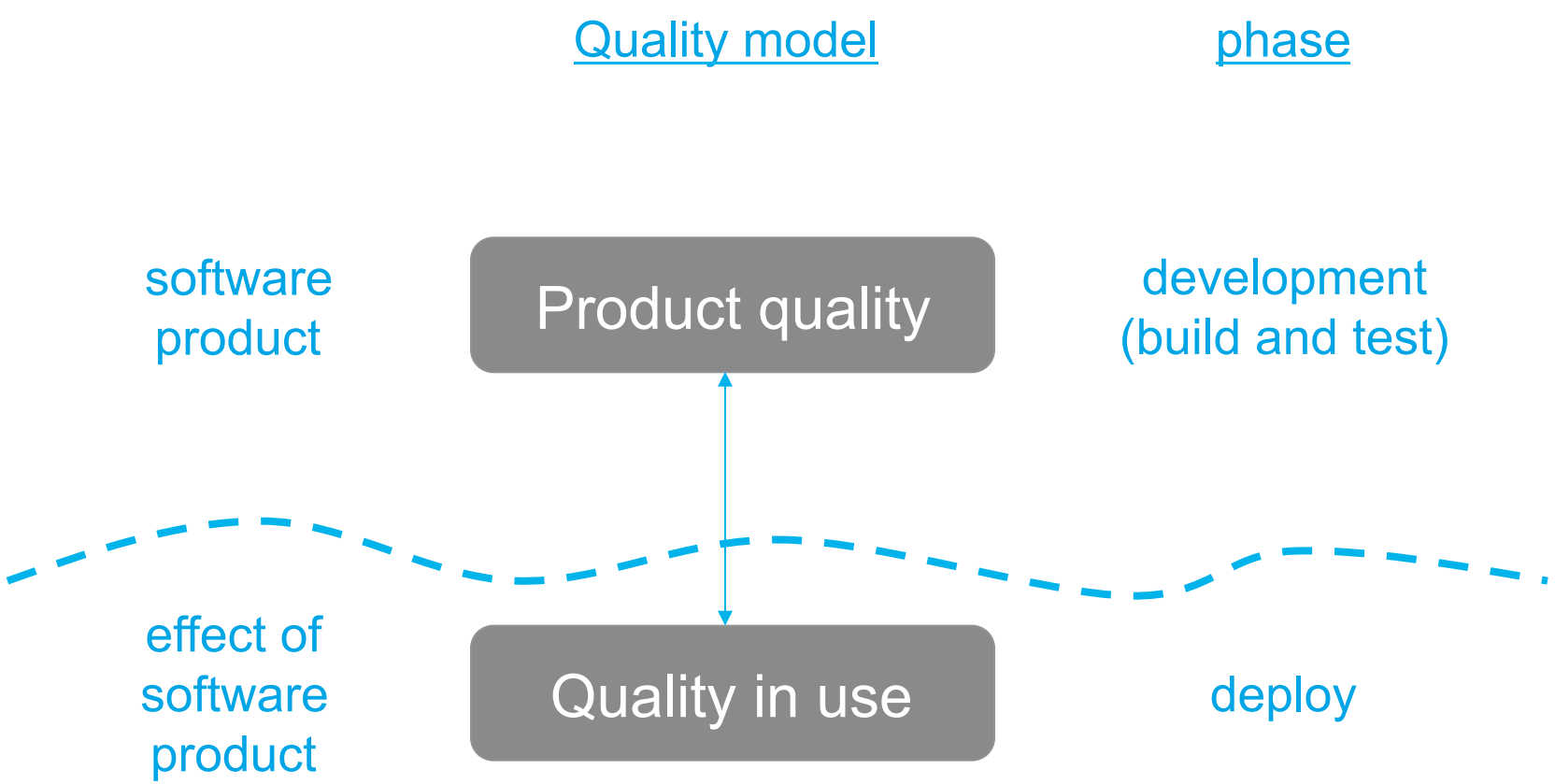
1. General overview
2. Planning and management
3. Process for developers
4. Process for acquirers
5. Process for evaluators
6. Documentation of evaluation modules

Currently: ISO 25000-series, a coherent family of standards

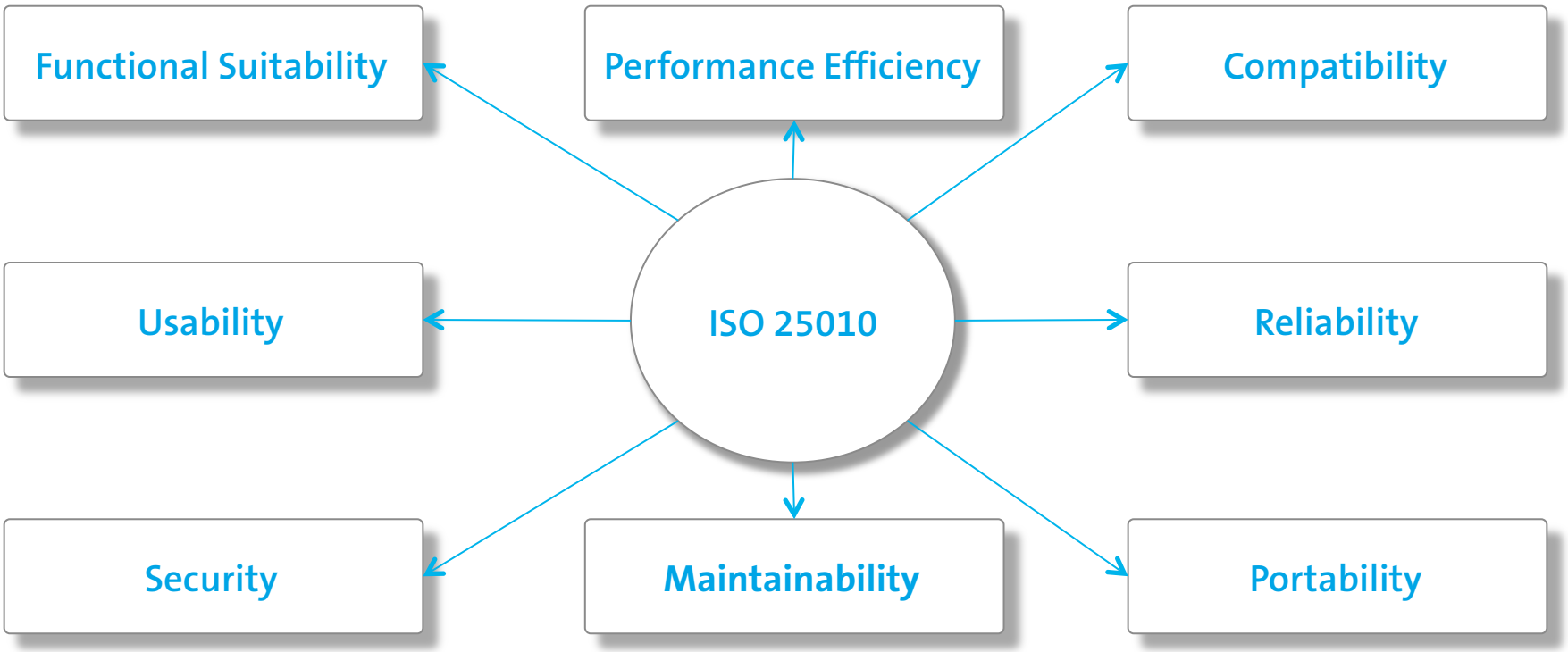


ISO International Organization for Standardization

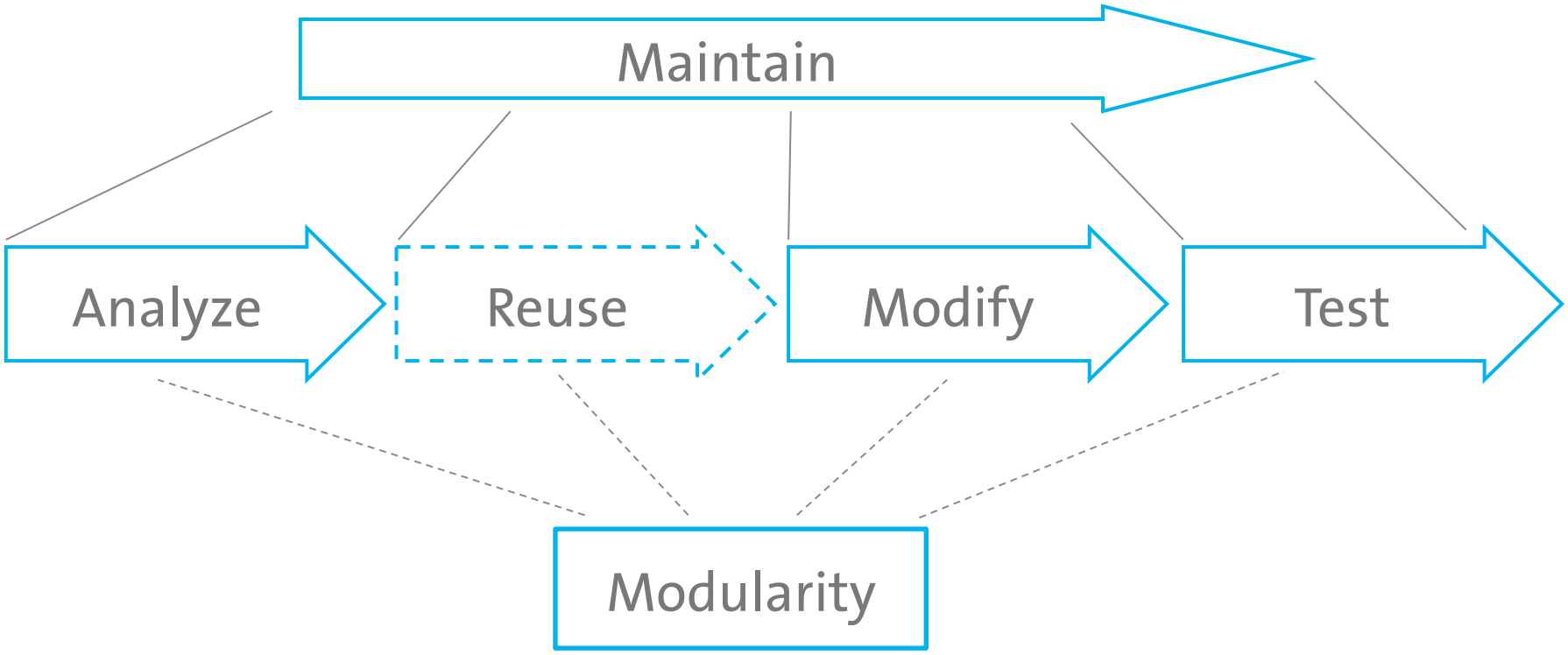
ISO/IEC 25010 Quality perspectives



ISO/IEC 25010 *Software product quality characteristics*



The sub-characteristics of maintainability in ISO 25010



A Practical Model for Measuring Maintainability

Some requirements

- Simple to understand
- Allow root-cause analyses
- Technology independent
- Easy to compute

Heitlager, et. al. *A Practical Model for Measuring Maintainability, QUATIC 2007*

Suggestions for metrics?

Measuring ISO 25010 maintainability using the SIG model

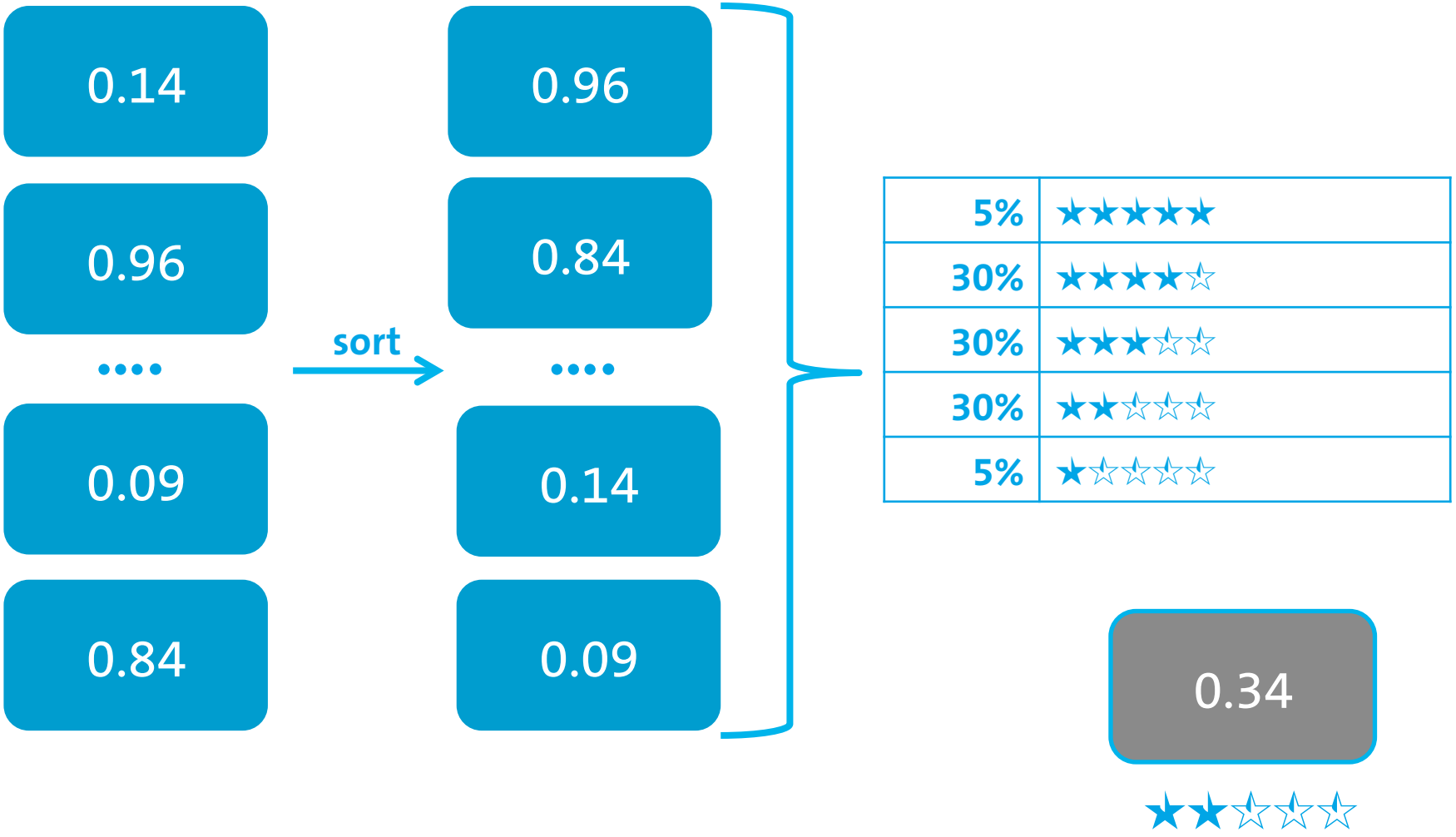
	Volume	Duplication	Unit size	Unit complexity	Unit interfacing	Module coupling	Component balance	Component independence
Analysability	X	X	X				X	
Modifiability		X		X		X		
Testability	X			X				X
Modularity						X	X	X
Reusability			X		X			

From measurement to rating

A benchmark based approach

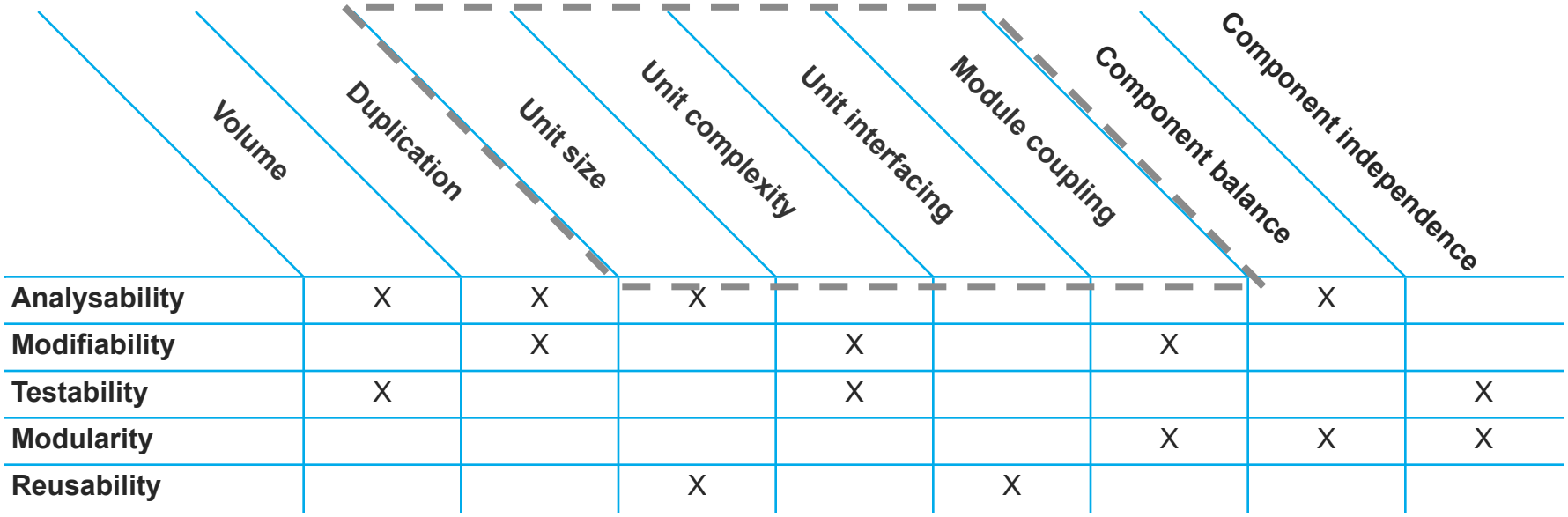
Embedding

A bechmark based approach



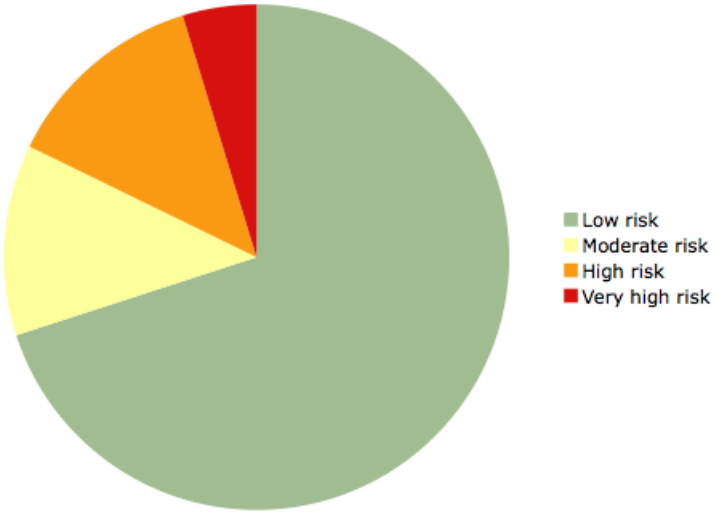
Note: example thresholds

From measurements to ratings



Remember the quality profiles?

1. Compute source code metrics per method / file / module
2. Summarize distribution of measurement values in “Quality Profiles”



Cyclomatic complexity	Risk category
1 - 5	Low
6 - 10	Moderate
11 - 25	High
> 26	Very high

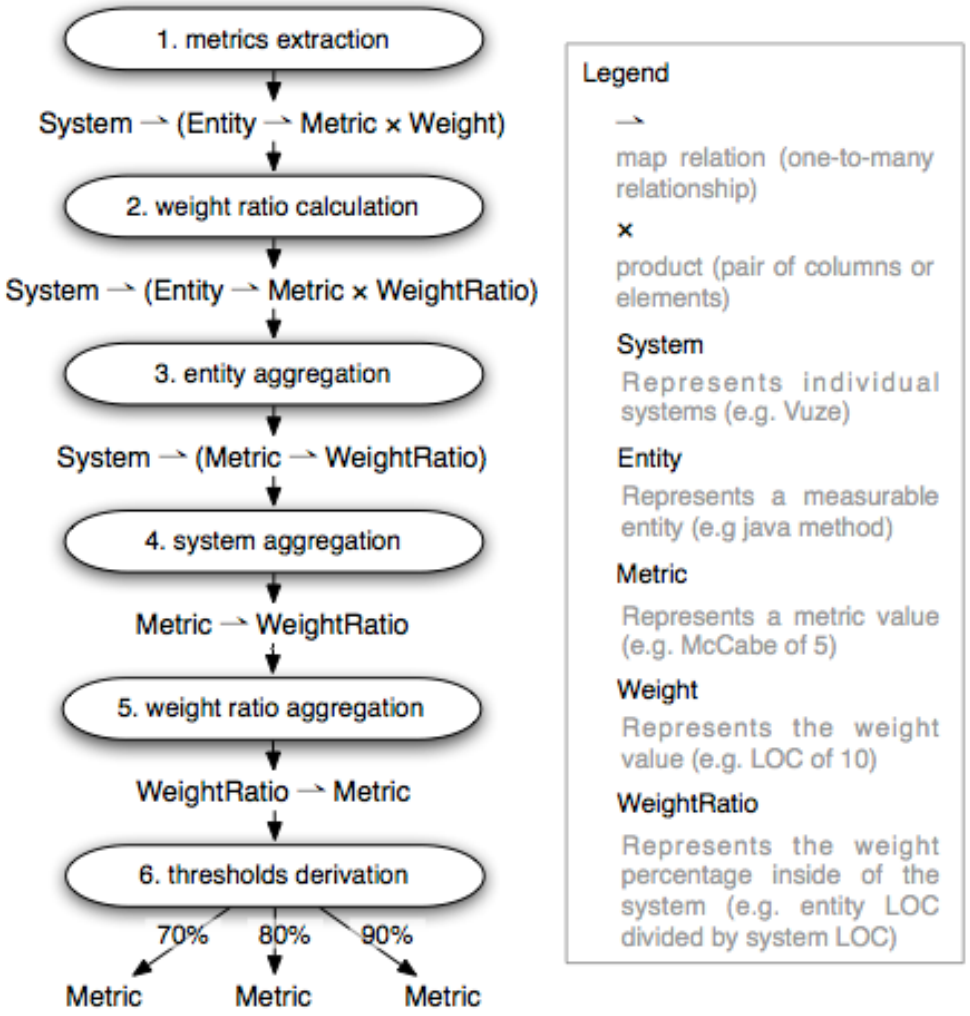
Sum of lines of code per category

Lines of code per risk category			
Low	Moderate	High	Very high
70 %	12 %	13 %	5 %

Heitlager, et. al. *A Practical Model for Measuring Maintainability, QUATIC 2007*

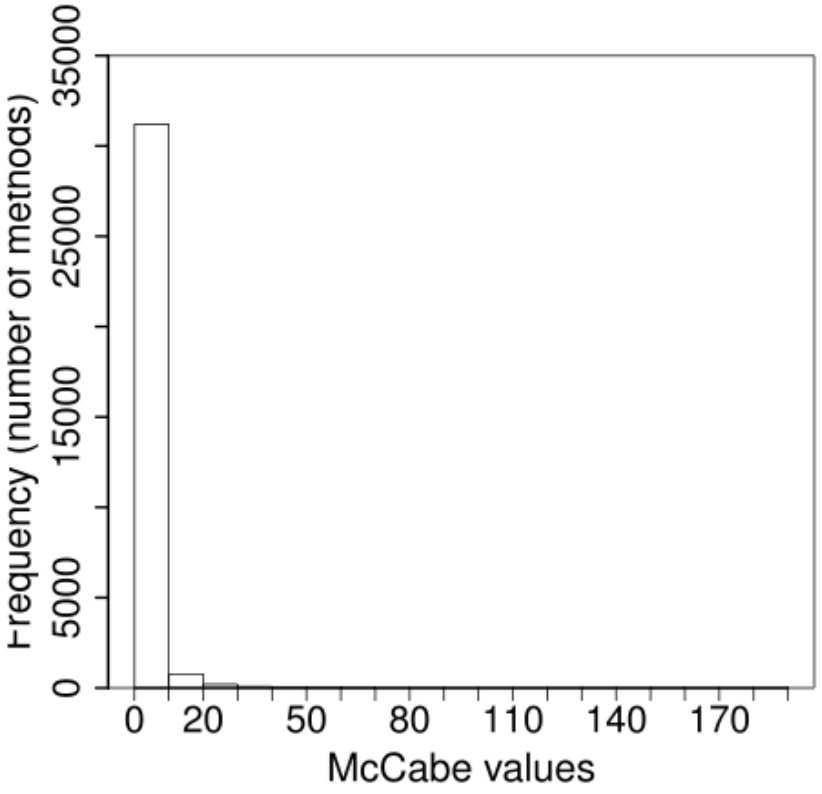
‘First level’ calibration

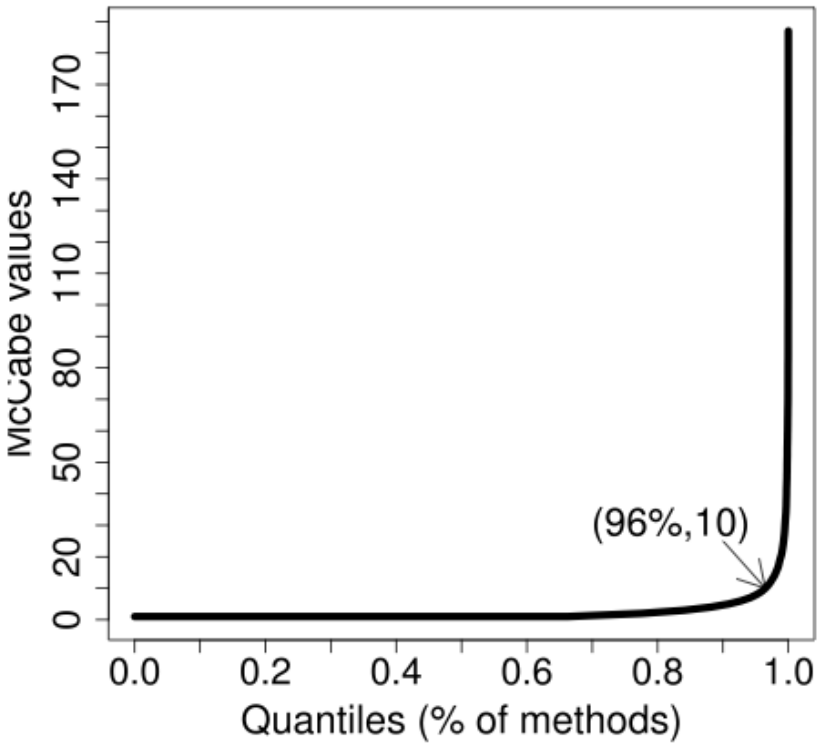
The formal six step proces



Alves, et. al., *Deriving Metric Thresholds from Benchmark Data, ICSM 2010*

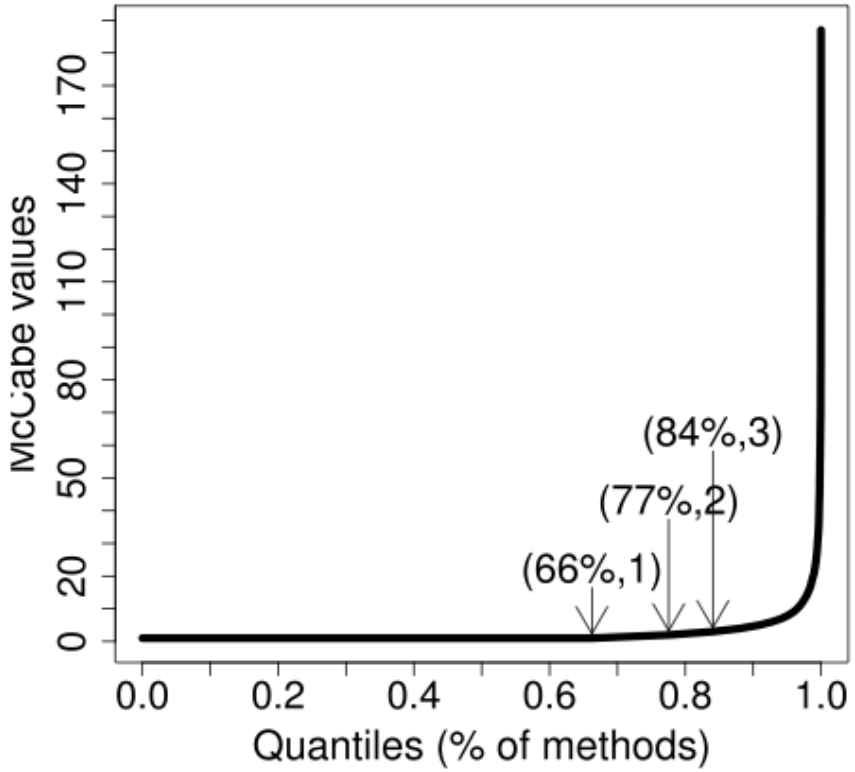
Visualizing the calculated metrics

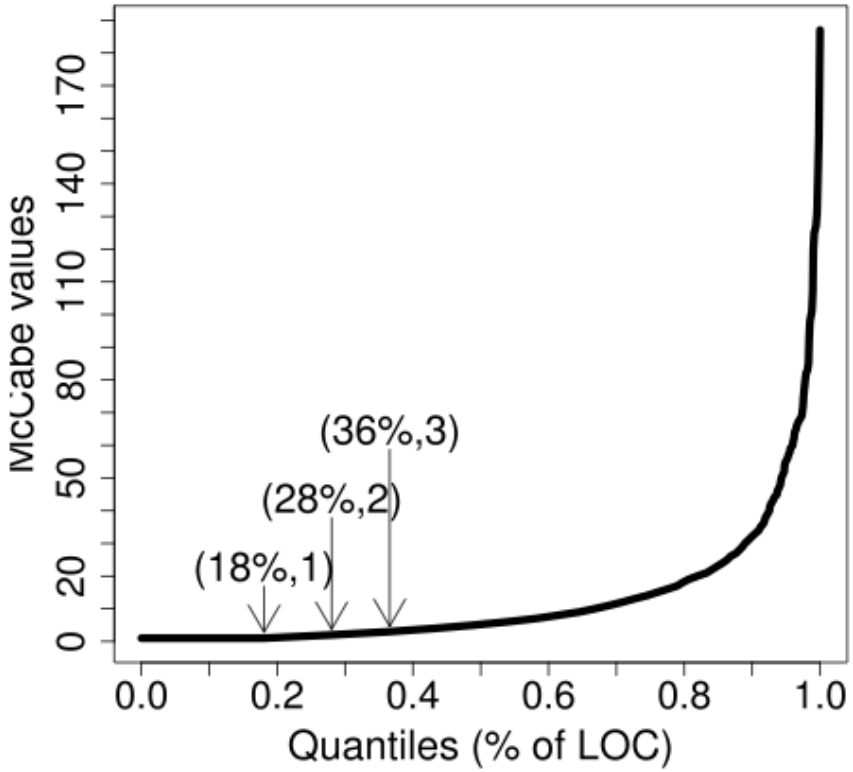




Alves, et. al., *Deriving Metric Thresholds from Benchmark Data, ICSM 2010*

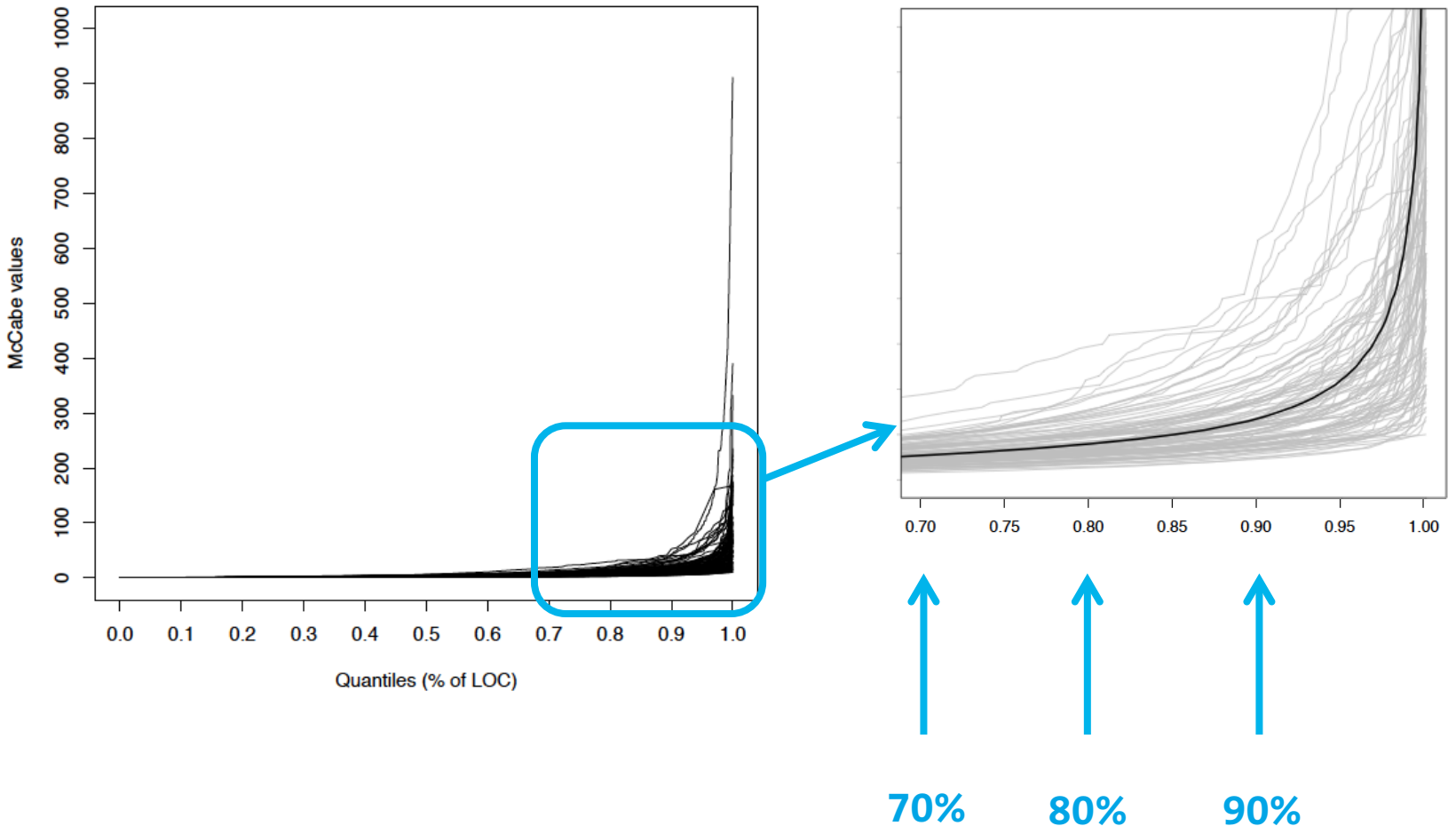
Choosing a weight metric





Alves, et. al., *Deriving Metric Thresholds from Benchmark Data, ICSM 2010*

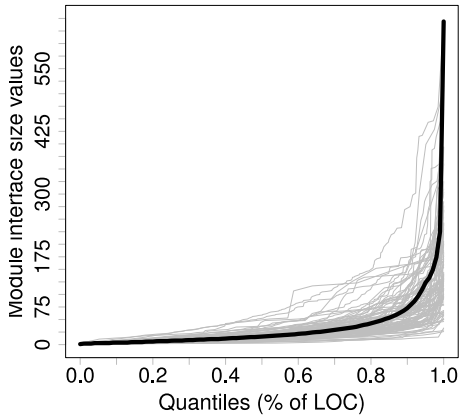
Calculate for a benchmark of systems



Alves, et. al., *Deriving Metric Thresholds from Benchmark Data, ICSM 2010*

SIG Maintainability Model
Derivation metric thresholds

1. Measure systems in benchmark
2. Summarize all measurements
3. Derive thresholds that bring out the metric's variability
4. Round the thresholds



Derive & Round

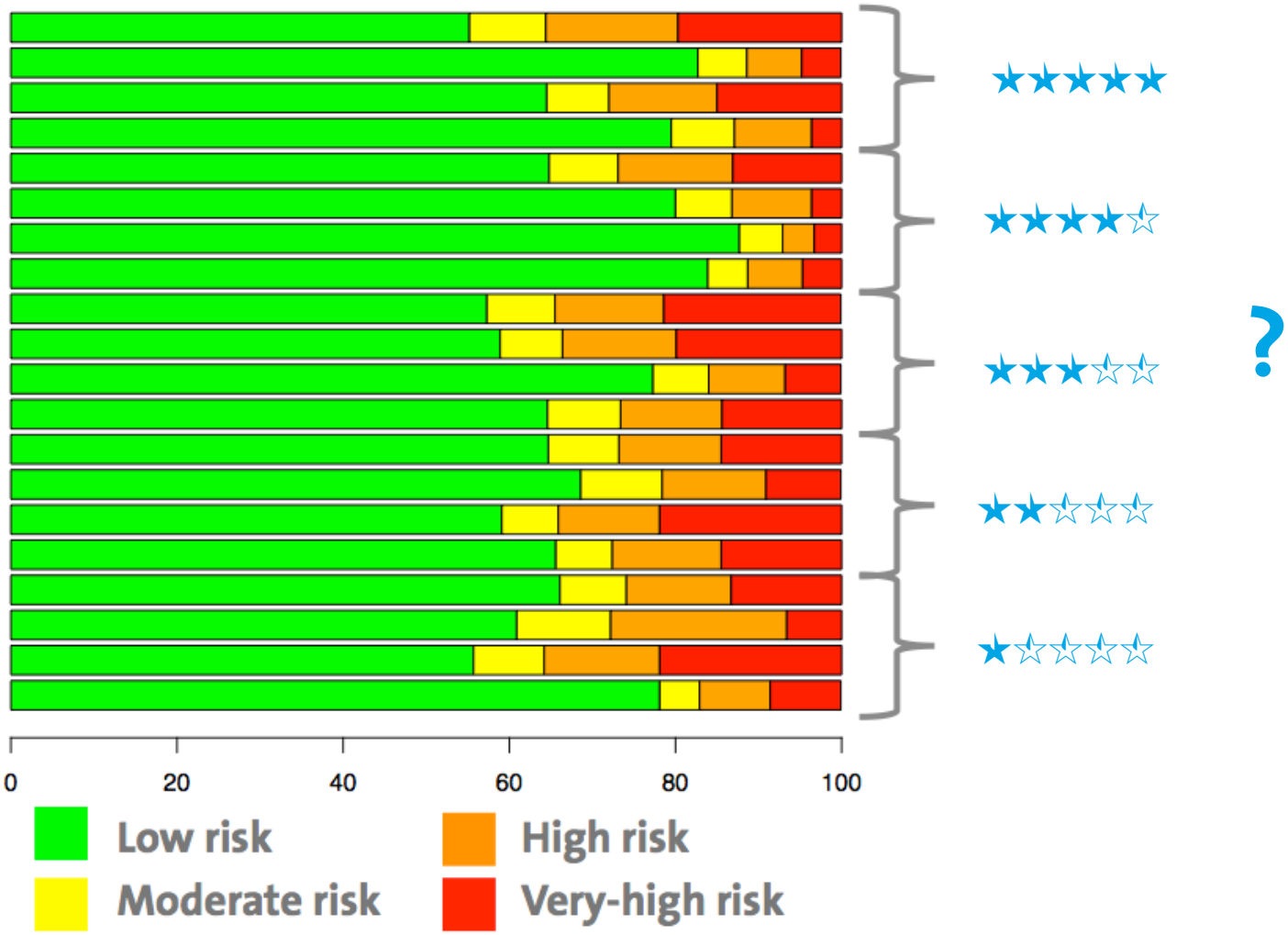
Cyclomatic complexity	Risk category
1 - 5	Low
6 - 10	Moderate
11 - 25	High
> 26	Very high

Alves, et. al., *Deriving Metric Thresholds from Benchmark Data, ICSM 2010*

‘Second level’ calibration

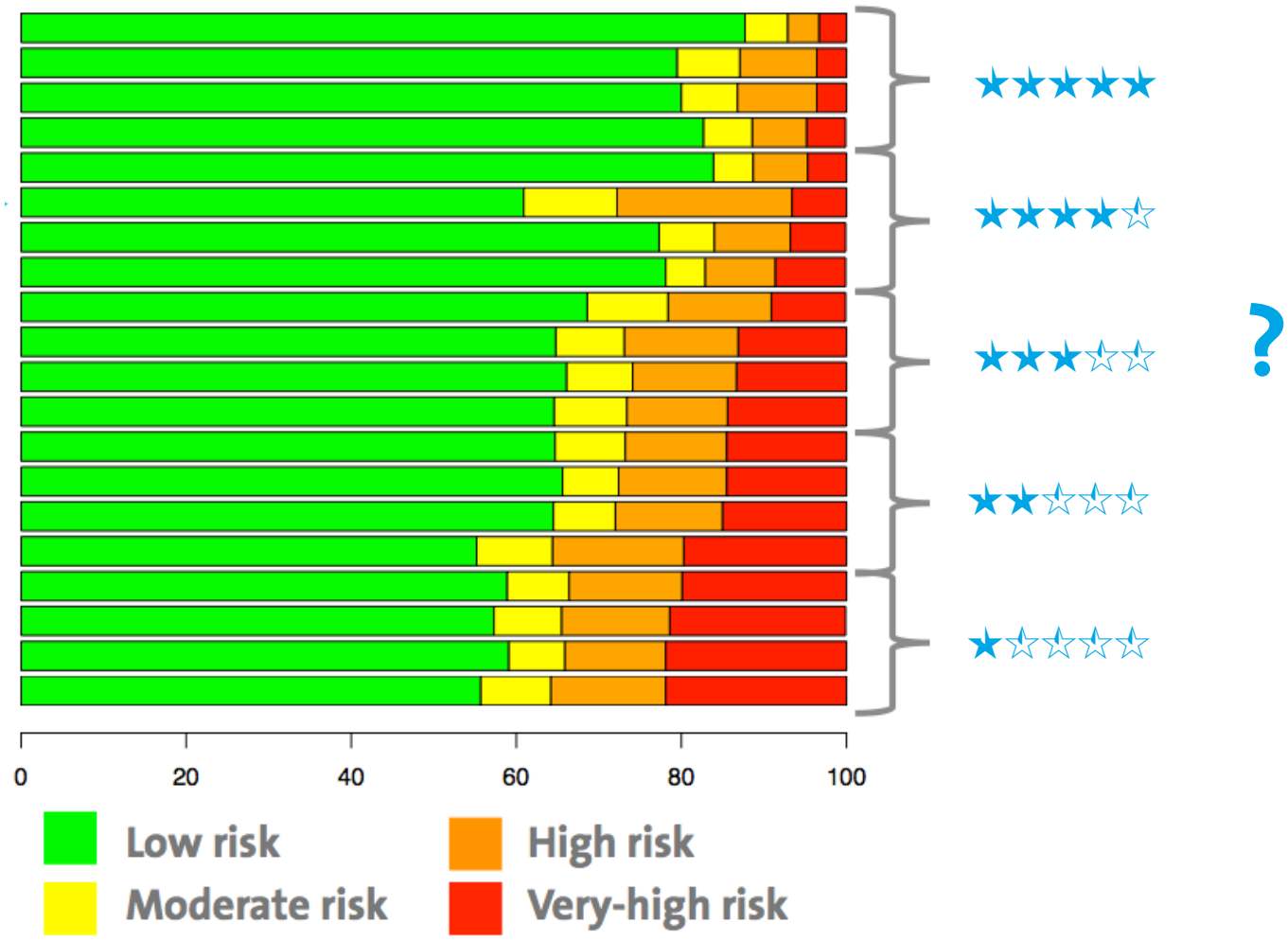
How to rank quality profiles?

Unit Complexity profiles for 20 random systems



Alves, et. al., *Benchmark-based Aggregation of Metrics to Ratings*, IWSM / Mensura 2011

Ordering by highest-category is not enough!



Alves, et. al., *Benchmark-based Aggregation of Metrics to Ratings*, IWSM / Mensura 2011

A better ranking algorithm

Require: $riskprofiles : (Moderate \times High \times VeryHigh)^*$, $partition^{N-1}$

1: $thresholds \leftarrow ()$
2: $ordered[Moderate] \leftarrow order(riskprofiles.Moderate)$
3: $ordered[High] \leftarrow order(riskprofiles.High)$
4: $ordered[VeryHigh] \leftarrow order(riskprofiles.VeryHigh)$
5: **for** $rating = 1$ **to** $(N-1)$ **do**
6: $i \leftarrow 0$
7: **repeat**
8: $i \leftarrow i + 1$
9: $thresholds[rating][Moderate] \leftarrow ordered[Moderate][i]$
10: $thresholds[rating][High] \leftarrow ordered[High][i]$
11: $thresholds[rating][VeryHigh] \leftarrow ordered[VeryHigh][i]$
12: **until** $distribution(riskprofiles, thresholds[rating]) \geq partition[rating]$ **or** $i \geq length(riskprofiles)$
13: $index \leftarrow i$
14: **for all** $risk$ in $(Moderate, High, VeryHigh)$ **do**
15: $i \leftarrow index$
16: $done \leftarrow False$
17: **while** $i > 0$ **and not** $done$ **do**
18: $thresholds.old \leftarrow thresholds$
19: $i \leftarrow i - 1$
20: $thresholds[rating][risk] \leftarrow ordered[risk][i]$
21: **if** $distribution(riskprofiles, thresholds[rating]) < partition[rating]$ **then**
22: $thresholds \leftarrow thresholds.old$
23: $done \leftarrow True$
24: **end if**
25: **end while**
26: **end for**
27: **end for**
28: **return** $thresholds$

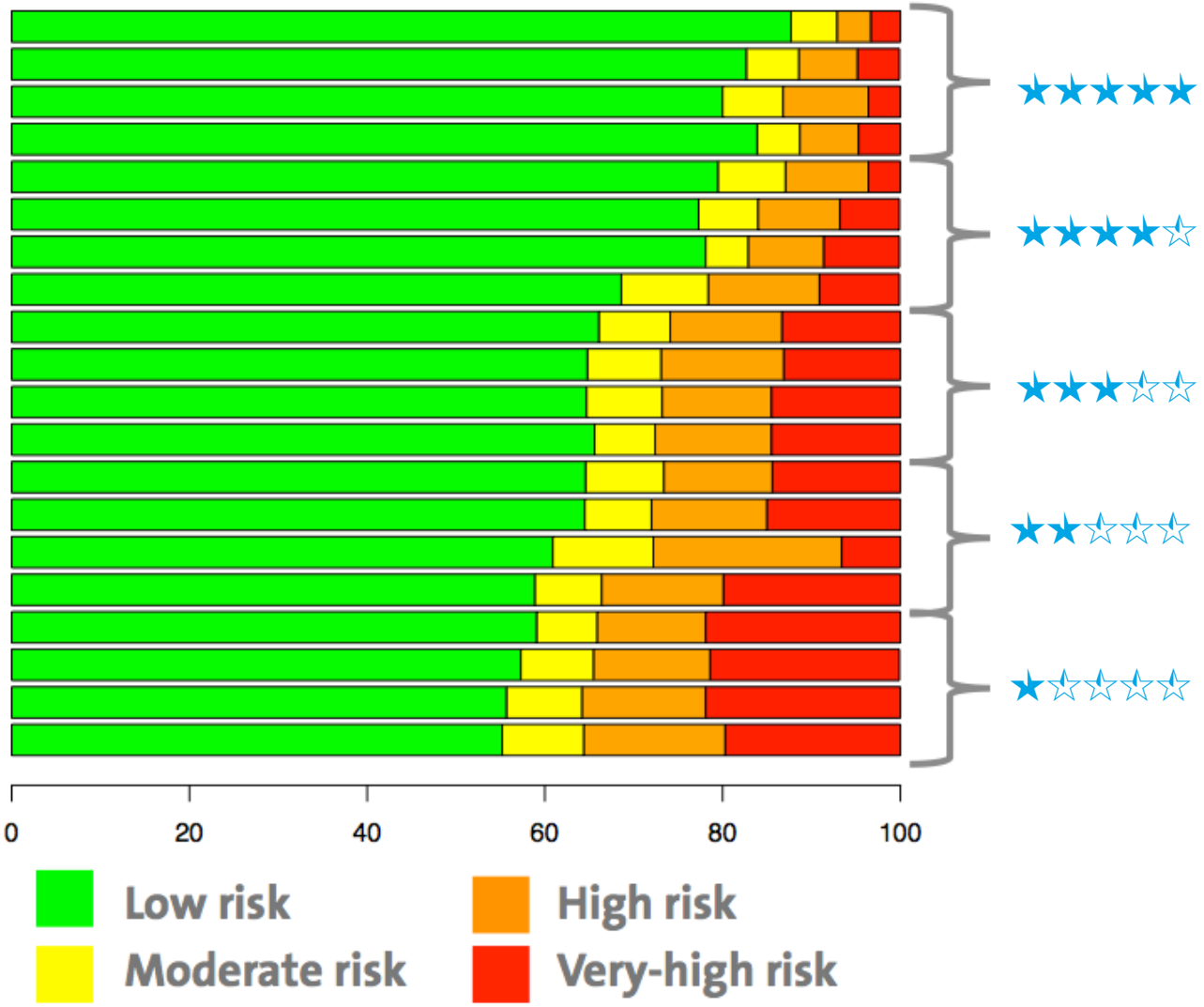
Order categories

Define thresholds of given systems

Find smallest possible thresholds

Alves, et. al., *Benchmark-based Aggregation of Metrics to Ratings*, IWSM / Mensura 2011

Which results in a more natural ordering



Alves, et. al., *Benchmark-based Aggregation of Metrics to Ratings*, IWSM / Mensura 2011

Second level thresholds

Unit size example

Star rating	Low risk]0, 30]	Moderate risk]30, 44]	High risk]44, 74]	Very-high risk]74, ∞[
★★★★★	-	19.5	10.9	3.9
★★★★☆	-	26.0	15.5	6.5
★★★☆☆	-	34.1	22.2	11.0
★★☆☆☆	-	45.9	31.4	18.1

Alves, et. al., *Benchmark-based Aggregation of Metrics to Ratings*, IWSM / Mensura 2011

SIG Maintainability Model
Mapping quality profiles to ratings

1. Calculate quality profiles for the systems in the benchmark
2. Sort quality profiles
3. Select thresholds based on 5% / 30% / 30% / 30% / 5% distribution

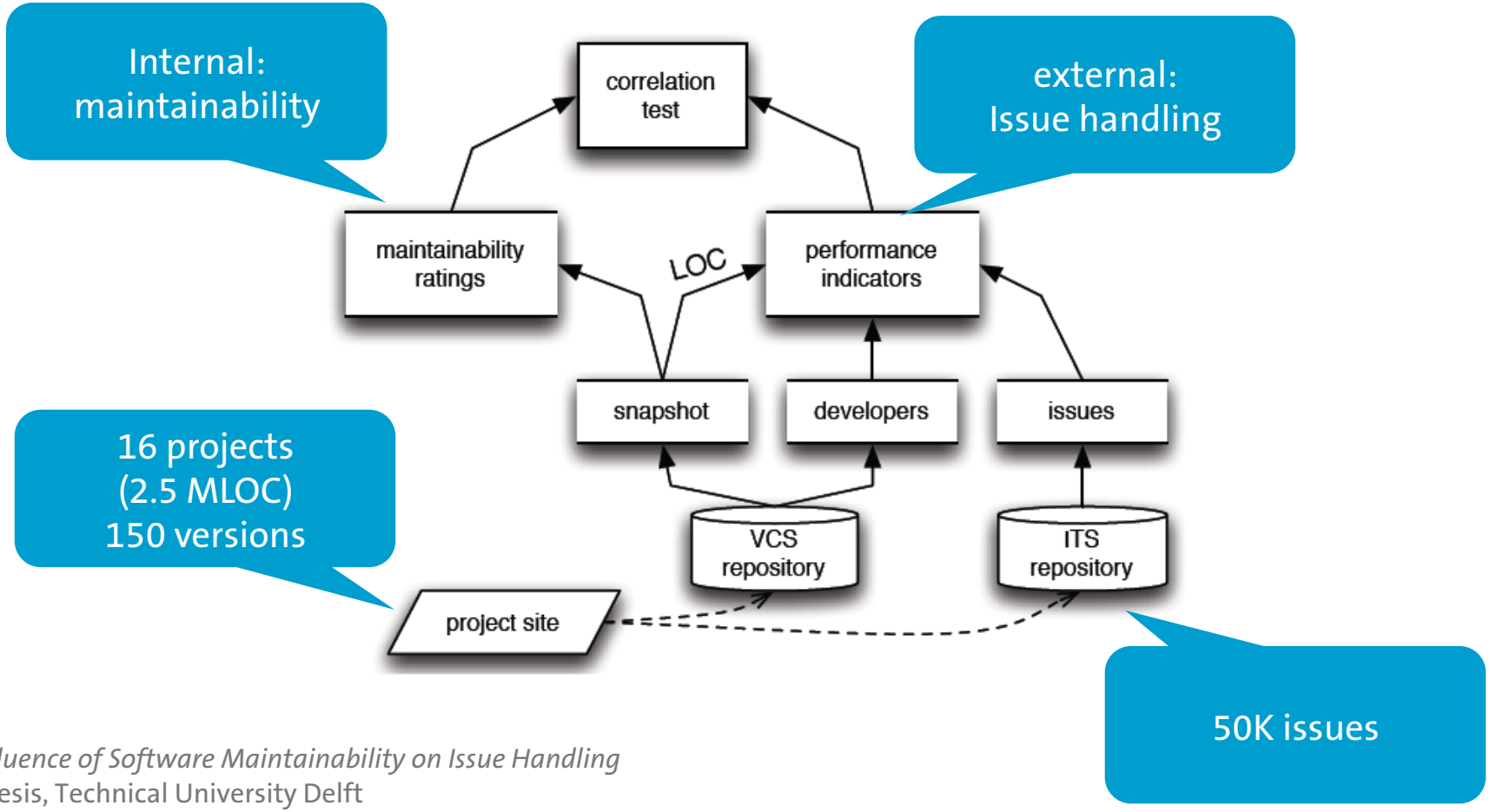
Alves, et. al., *Benchmark-based Aggregation of Metrics to Ratings*, IWSM / Mensura 2011

SIG measurement model

Putting it all together

Does it measure maintainability?

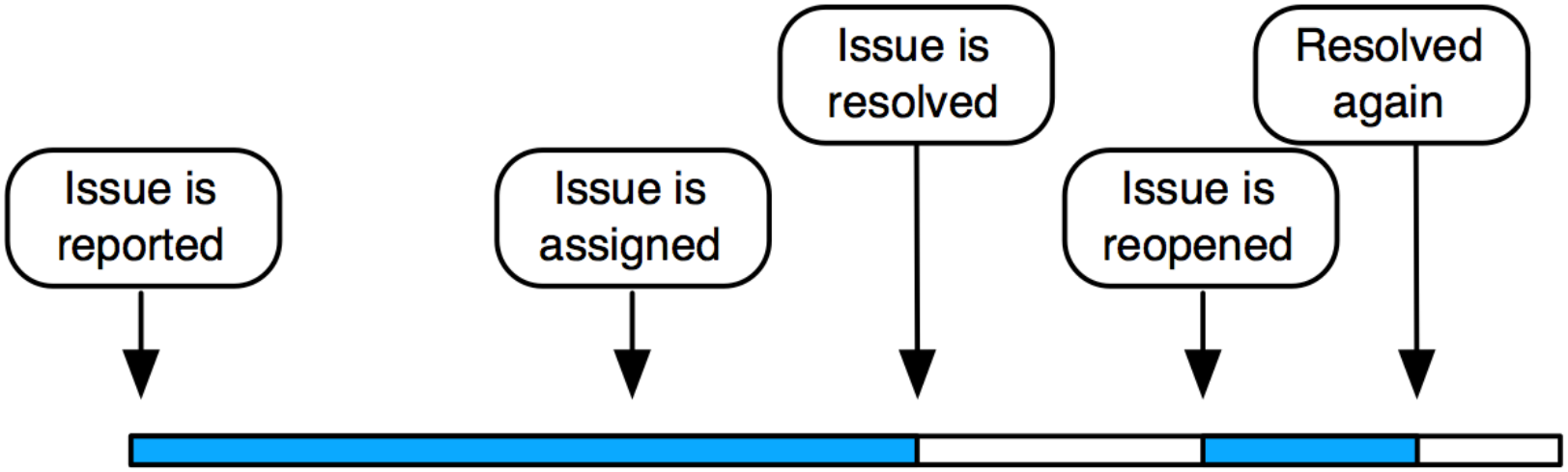
SIG Maintainability Model
Empirical validation



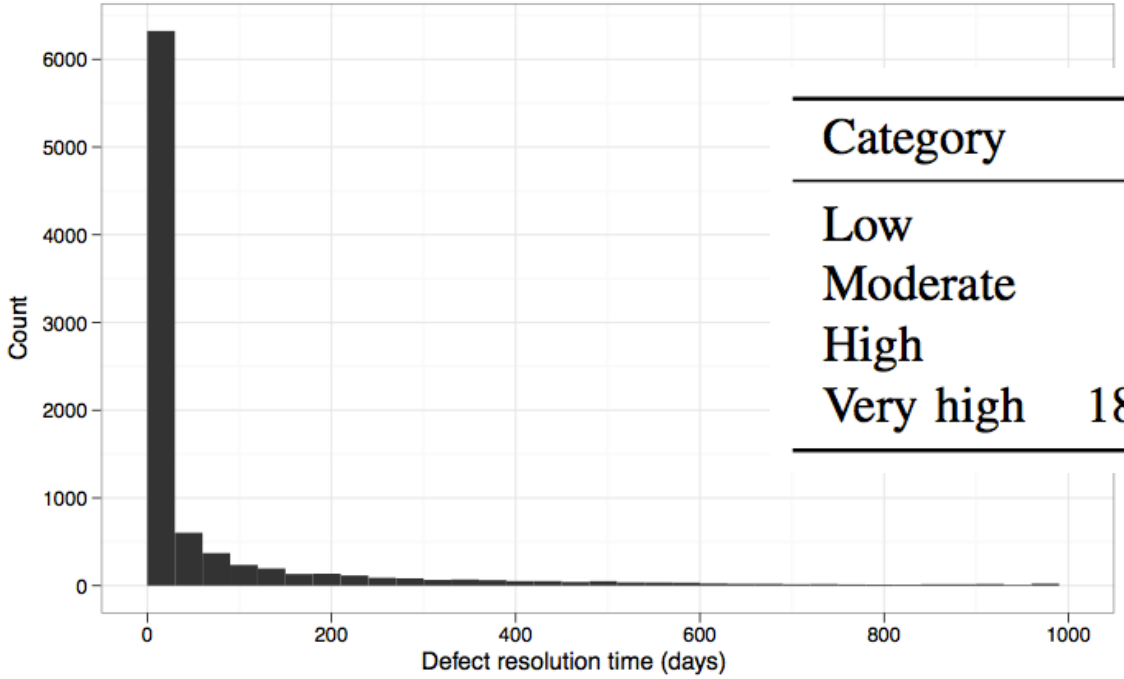
- *The Influence of Software Maintainability on Issue Handling*
  MSc thesis, Technical University Delft
- *Indicators of Issue Handling Efficiency and their Relation to Software Maintainability,*
  MSc thesis, University of Amsterdam
- *Faster Defect Resolution with Higher Technical Quality of Software, SQM 2010*

Empirical validation

The life-cycle of an issue



Empirical validation
Defect resolution time



Category	Thresholds	
Low	0 - 28 days	(4 weeks)
Moderate	28 - 70 days	(10 weeks)
High	70 - 182 days	(6 months)
Very high	182 days or more	

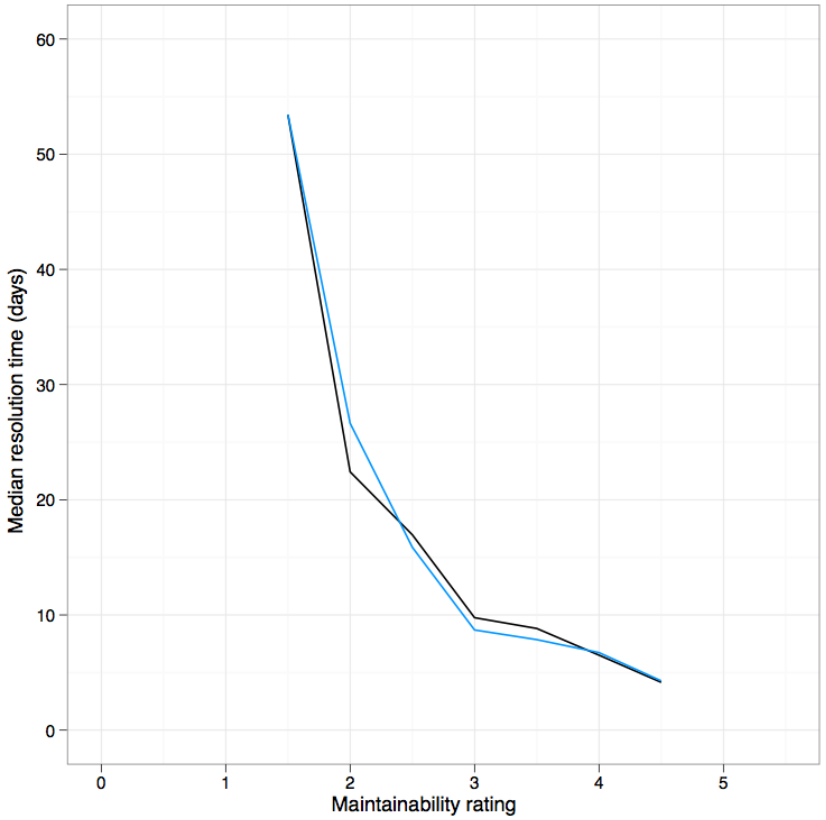
Luijten et.al. *Faster Defect Resolution with Higher Technical Quality of Software,* SQM 2010

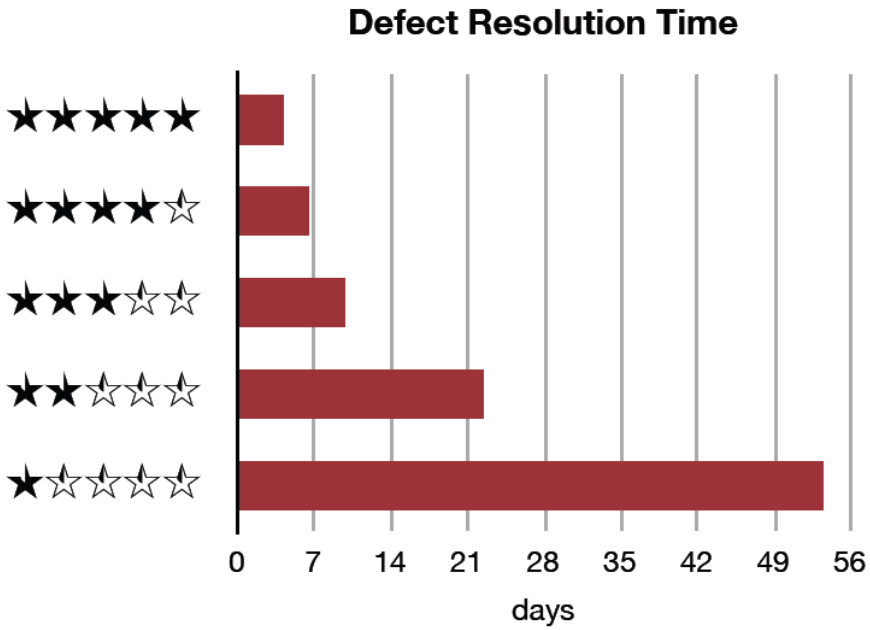
Empirical validation
Quantification

Defect resolution vs.	ρ_s	p-value
Volume	0.29	0.003
Duplication	0.31	0.002
Unit size	0.51	0.000
Unit complexity	0.51	0.000
Unit interfacing	-0.14	0.897
Module coupling	0.51	0.000
Analysability	0.51	0.000
Changeability	0.64	0.000
Stability	0.41	0.000
Testability	0.53	0.000
Maintainability	0.62	0.000

Luijten et.al. *Faster Defect Resolution with Higher Technical Quality of Software,* SQM 2010

Empirical validation
Quantification



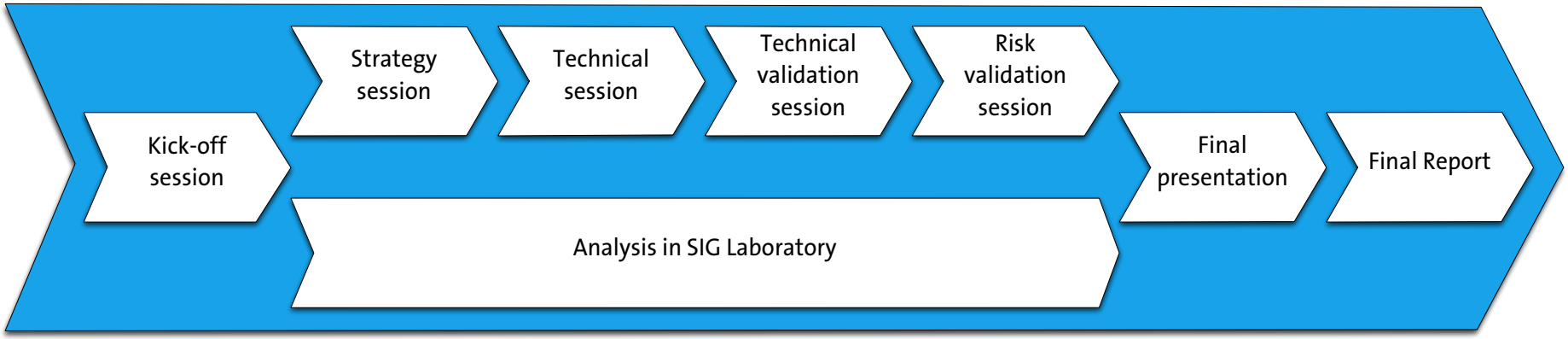


Bijlsma. *Indicators of Issue Handling Efficiency and their Relation to Software Maintainability,* MSc thesis, University of Amsterdam

Does it measure maintainability?

Is it useful?

Software Risk Assessment

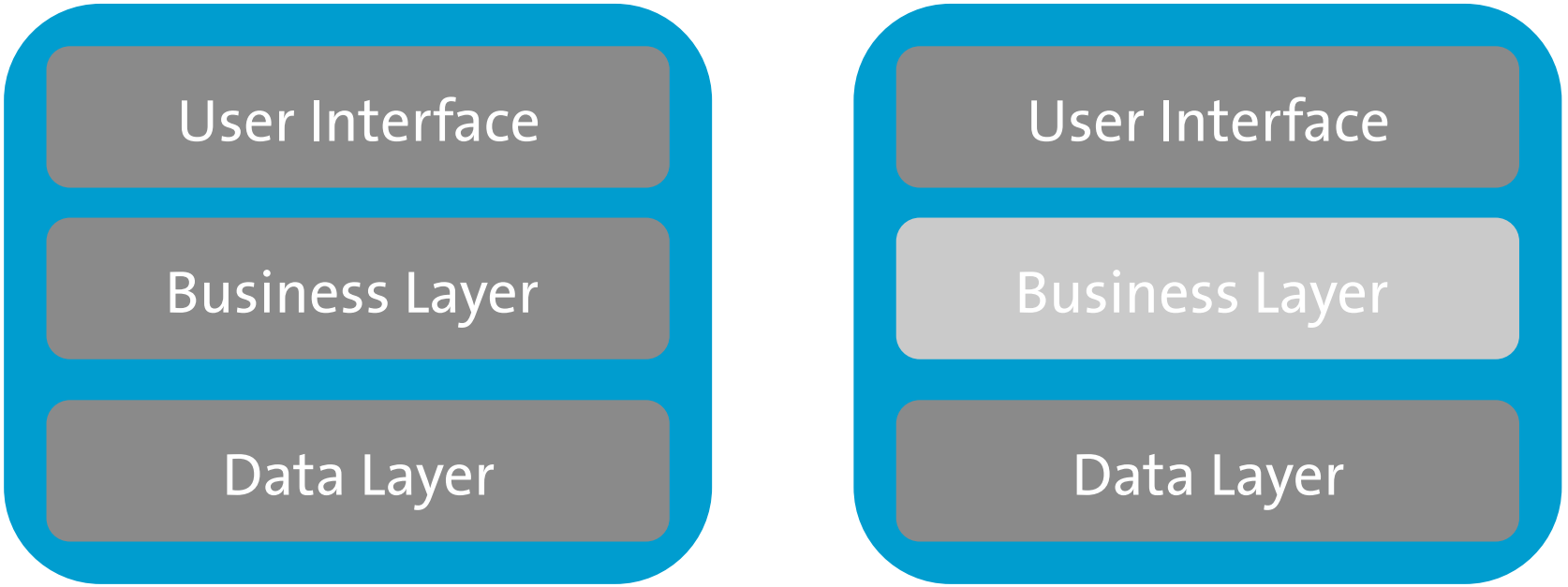


Example

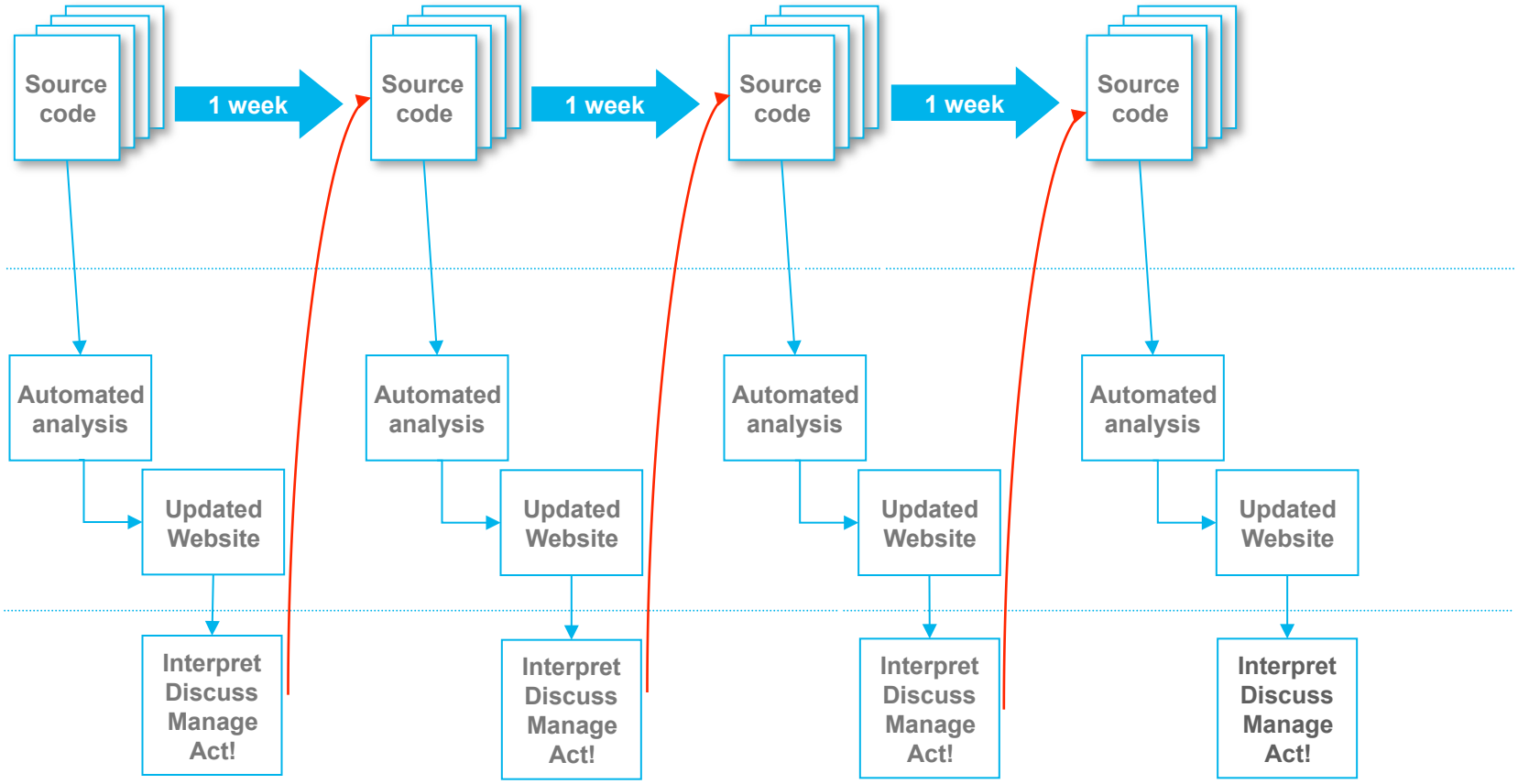
Which system to use?

Example

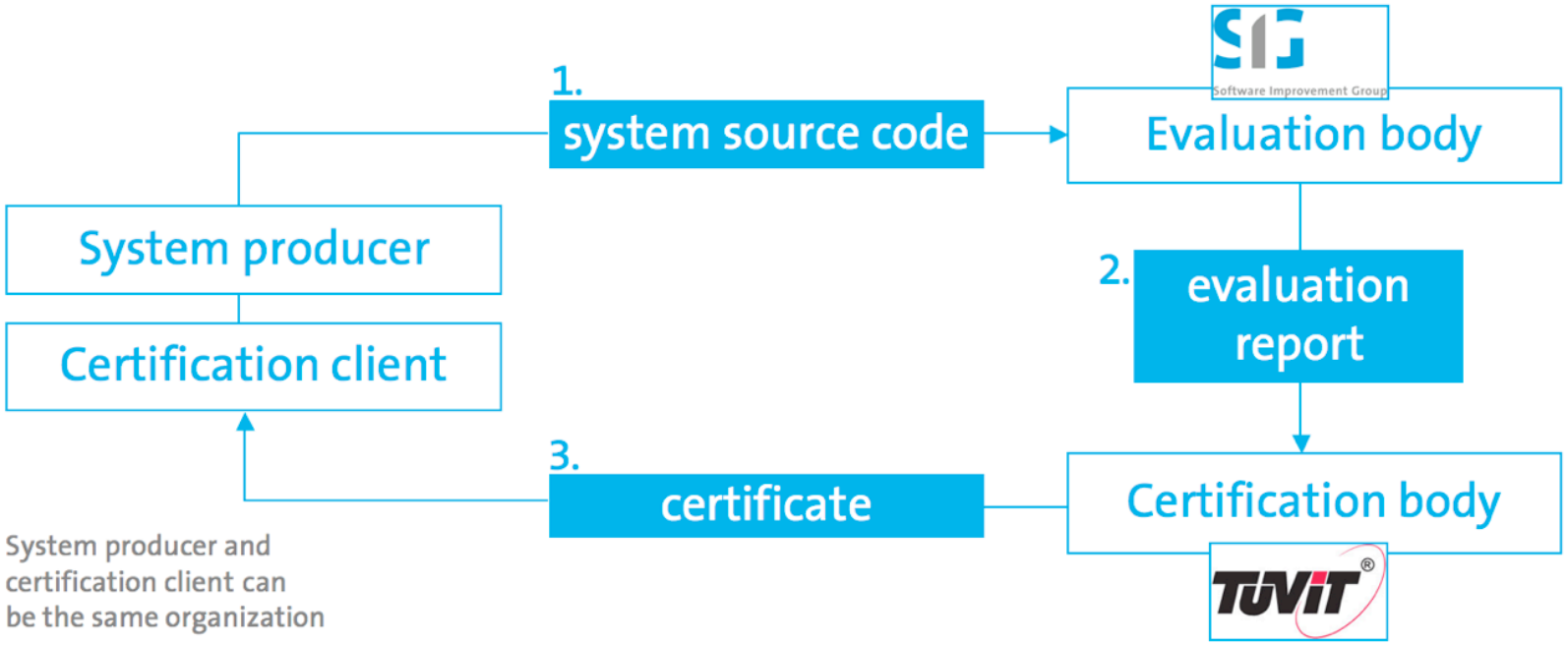
Should we accept delay and cost overrun, or cancel the project?



Software Monitoring



Software Product Certification



Summary

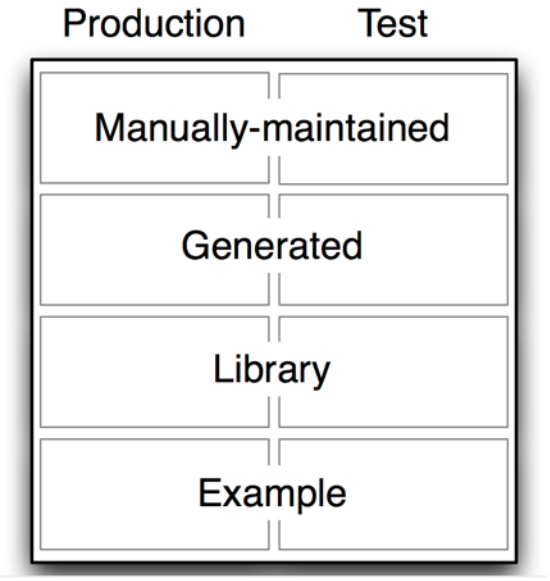
Measurement challenges

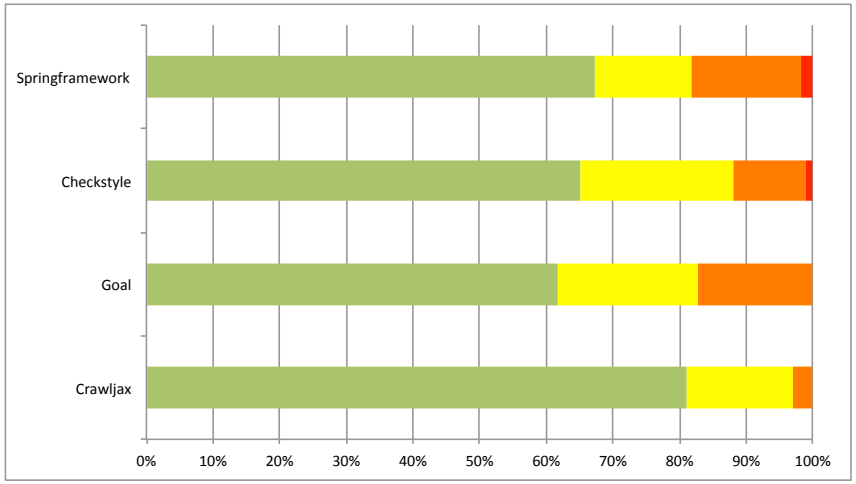
Entity

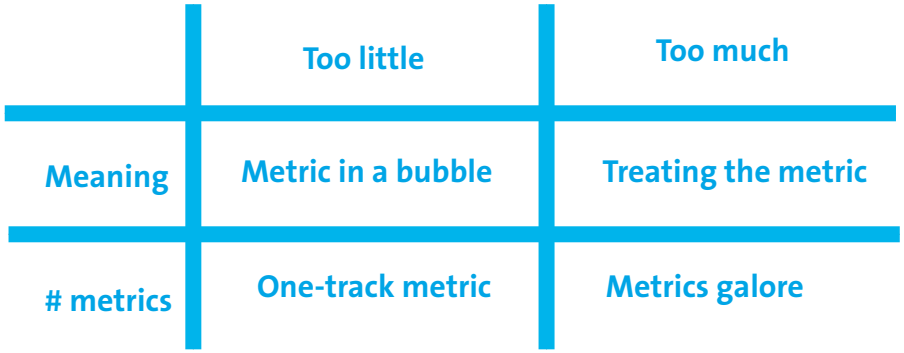
Attribute

Mapping

Measure

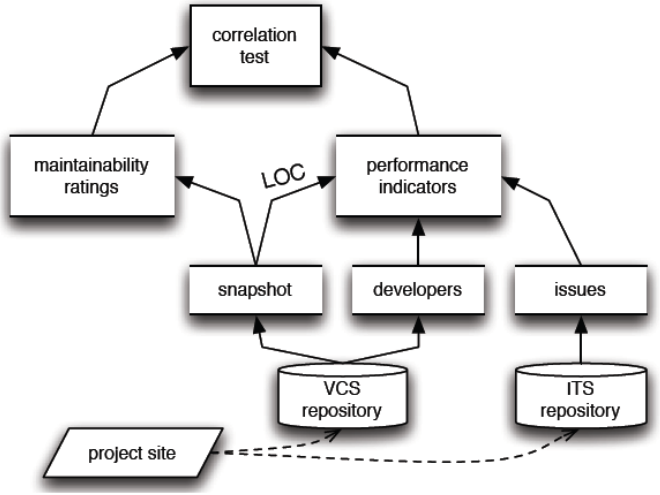


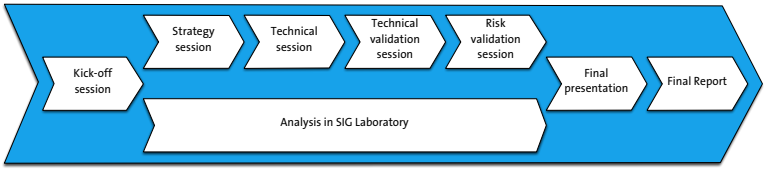




A benchmark based model

	Volume	Duplication	Unit size	Unit complexity	Unit interfacing	Module coupling	Component balance	Component independence
Analysability	X	X	X				X	
Modifiability		X		X		X		
Testability	X			X				X
Modularity						X	X	X
Reusability			X		X			





The most important things to remember

Goal

Entity – Attribute – Mapping

Context

Validate in
theory and practice