



Software Improvement Group



Benchmark-based Software Product Quality

Tiago L. Alves

July 2011 T +31 20 314 0950
info@sig.eu
www.sig.eu

Background



2 | 41

PhD Student (University of Minho) since 2007

Software Improvement Group



3 | 41

Who are we?

- Highly specialized research company for quality of software, founded in 2000 as a spin-off of the Centre for Mathematics and Information Technology
- Independent and therefore able to give objective advice
- Decorated with the Innovator Award 2007 and ICT Regie Award 2008

What do we do?

- Fact-based consultancy supported by our automated toolset for source code analysis
- Assessment across technologies by use of technology-independent methods

Our mission: We give you control over your software.

Services



4 | 41



Software Risk Assessment

- In-depth investigation of software quality and associated business risks
- Answers to specific research questions



Software Monitoring

- Continuous measurement, feedback, and development consultancy
- Guard quality from start to finish



Software Product Certification

- Five levels of technical quality (maintainability)
- Evaluation by SIG, certification by TÜV Informationstechnik

Who is using our services?



Software Improvement Group

5 | 41

Financial and Insurance companies



Government



kadaster

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Belastingdienst

Raad voor Rechtsbijstand

POLITIE



Logistical



ProRail



IT

Getronics
PinkRocca



Exact[®]
software



Other



PricewaterhouseCoopers



SWISSLEX



Alcatel-Lucent

Electrabel

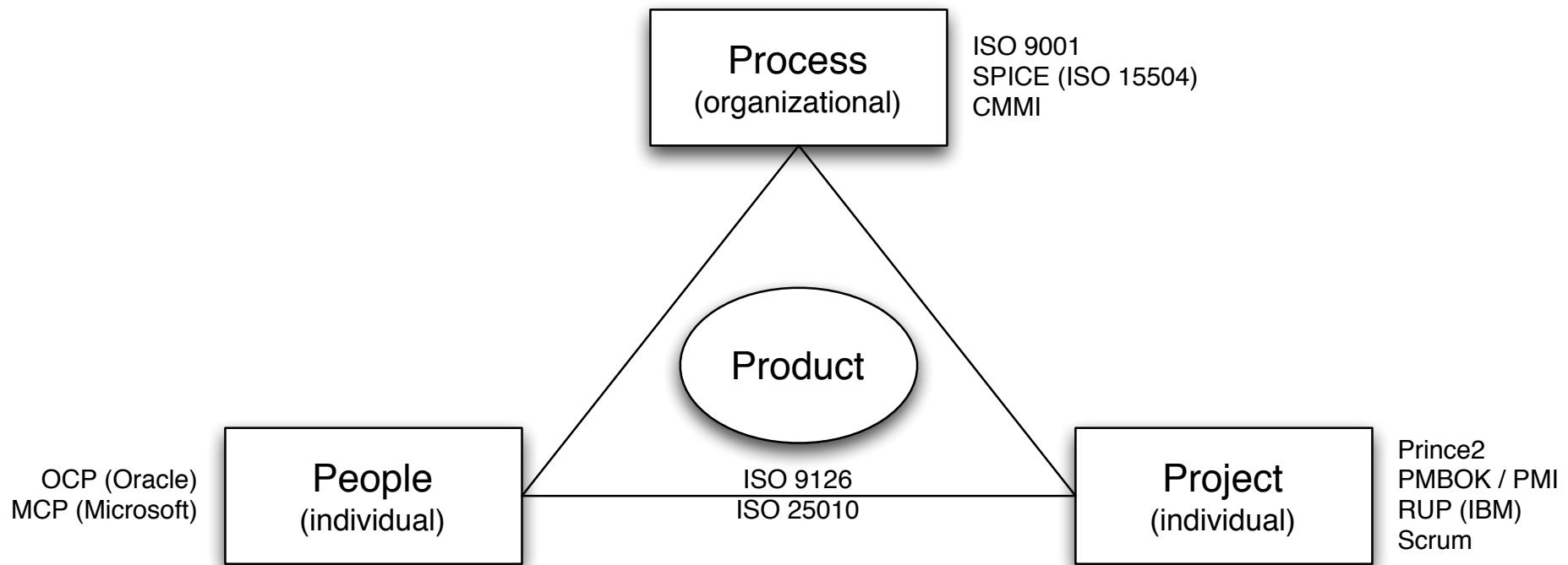
Bermuda triangle of software quality assurance



Software Improvement Group



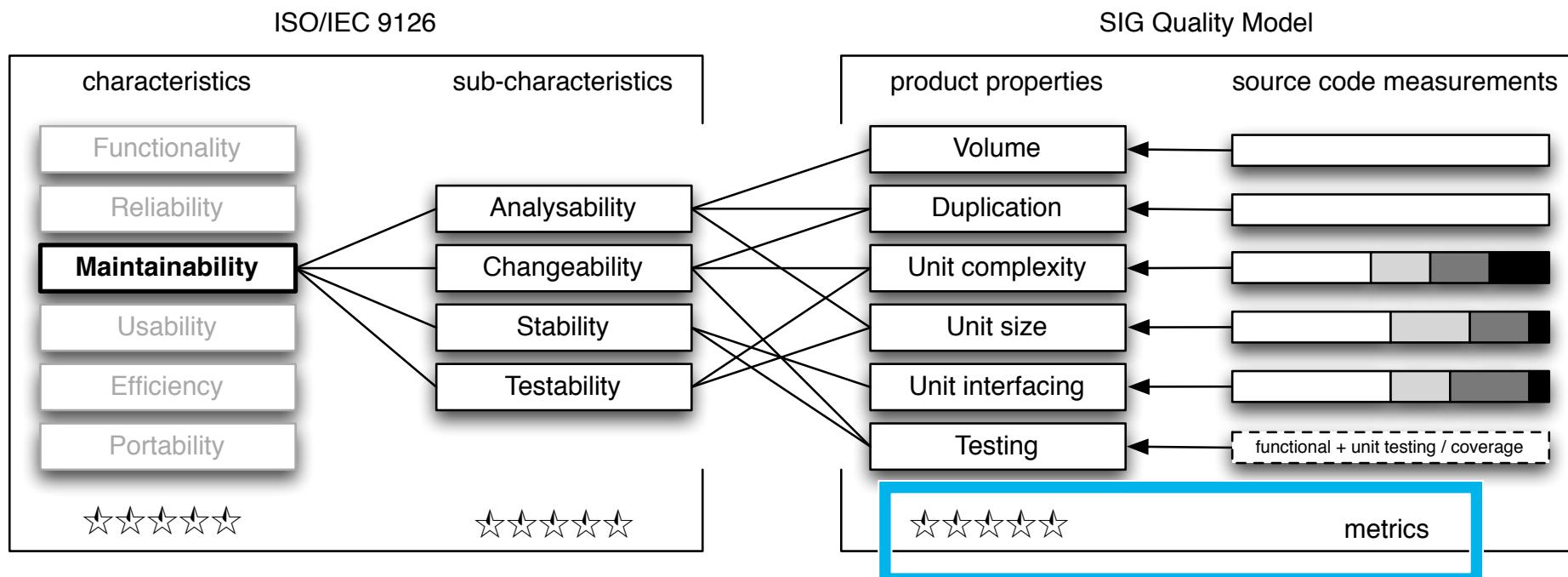
6 | 41



SIG Quality Model for Maintainability (operationalization of the ISO 9126)



7 | 41

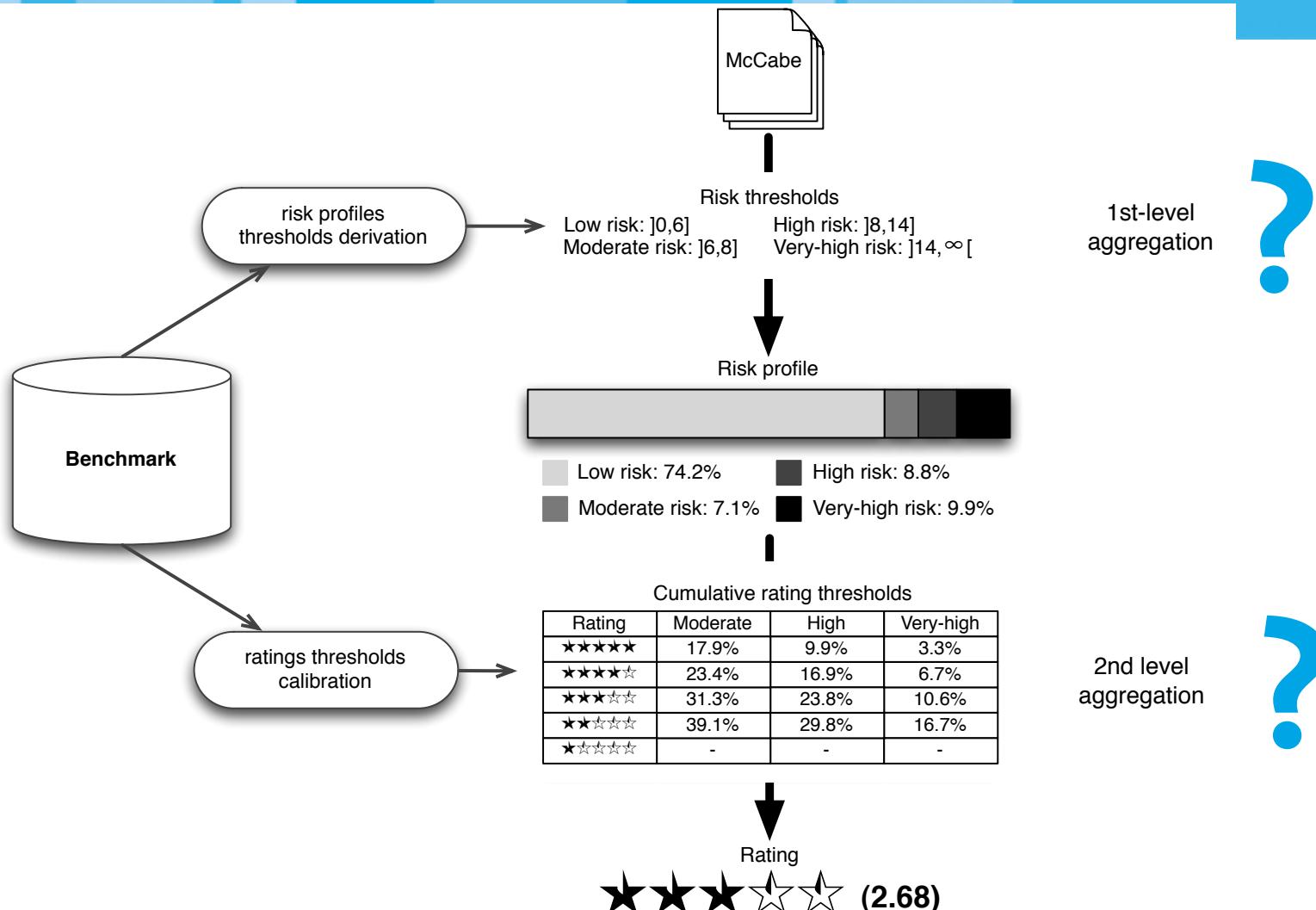


Benchmarking metrics to ratings



Software Improvement Group

8 | 41





Software Improvement Group

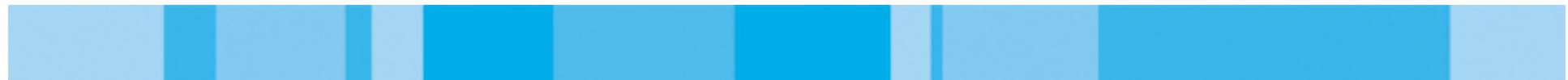


9 | 41

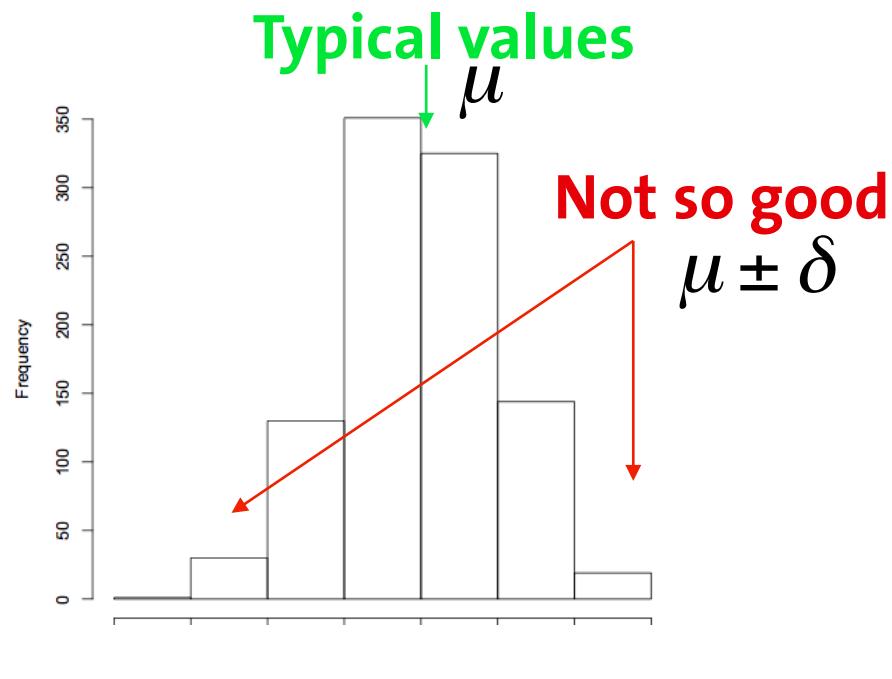
Part I

Derivation of risk thresholds

How to derive thresholds? Life sciences vs. software sciences



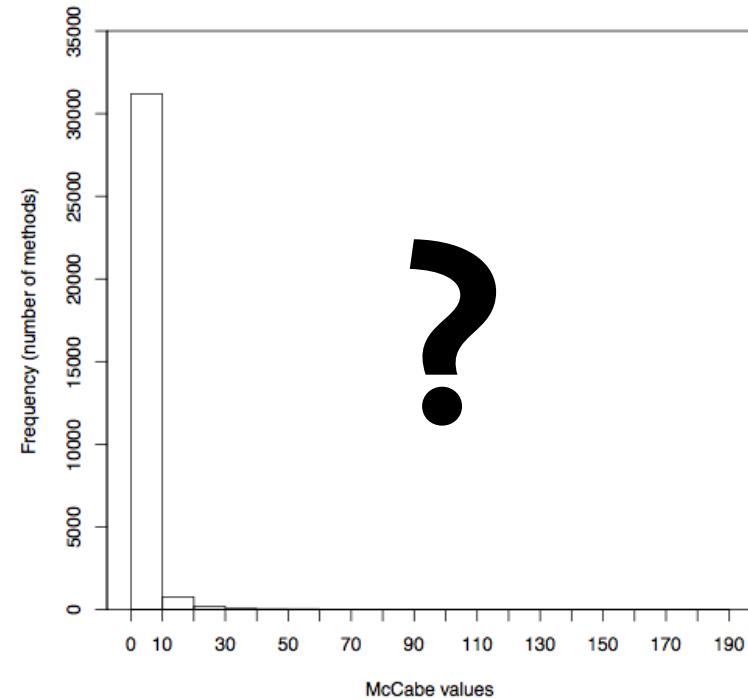
Cholesterol levels



Malnutrition - anxiety,
depression, suicide

Heart attack

Complexity



10 | 41

Derivation of risk thresholds: requirements



11 | 41

Requirements

1. Respect the statistical properties of the metric (scale and distribution)
2. Based on data analysis from a representative set of systems (benchmark)
3. Repeatable, transparent, and of straightforward execution.
4. Enable traceability of results

Experimental benchmark



Software Improvement Group



12 | 41

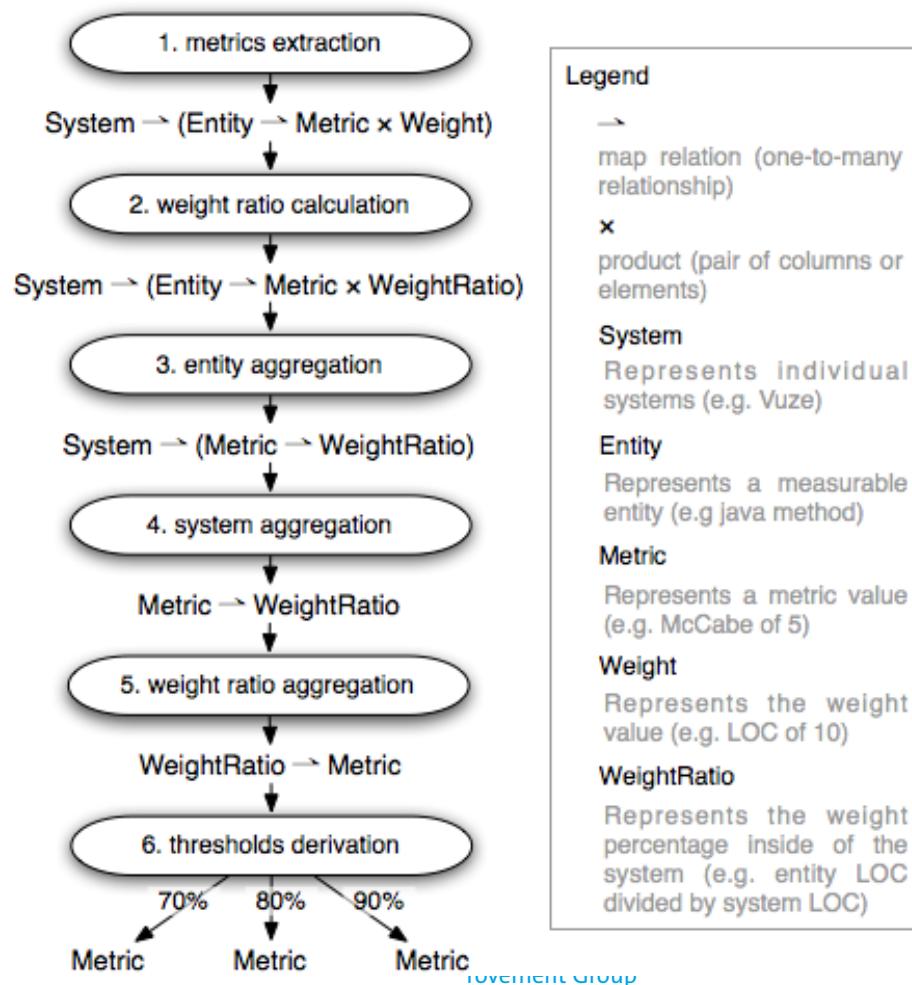
Technology	License	n	LOC
Java	Proprietary	60	8,435K
	OSS	22	2,756K
C#	Proprietary	17	794K
	OSS	1	10K
	Total	100	11,996K

Derivation of risk thresholds: methodology

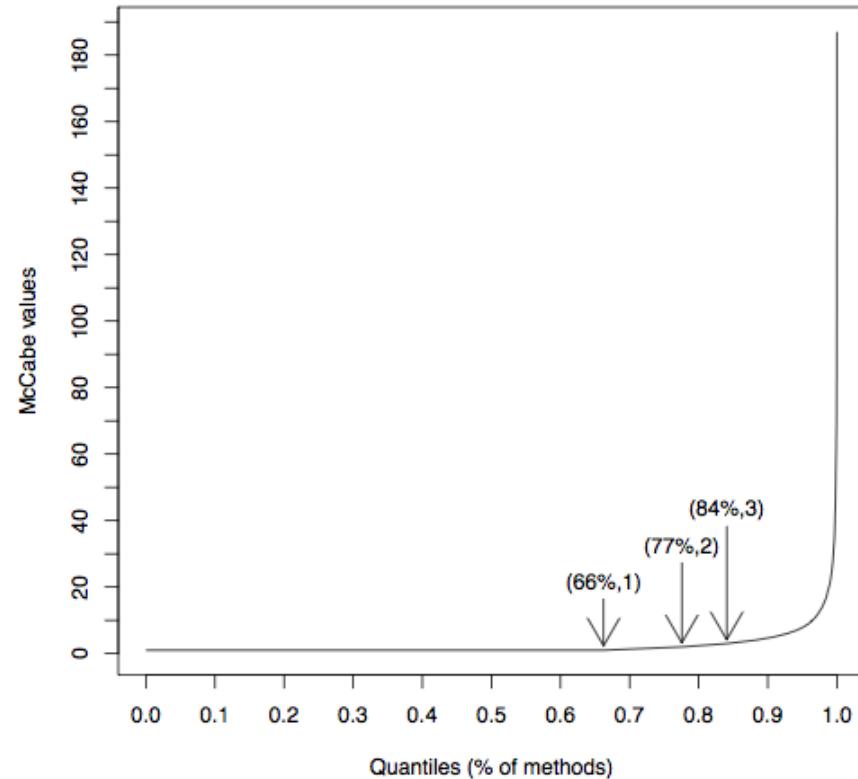
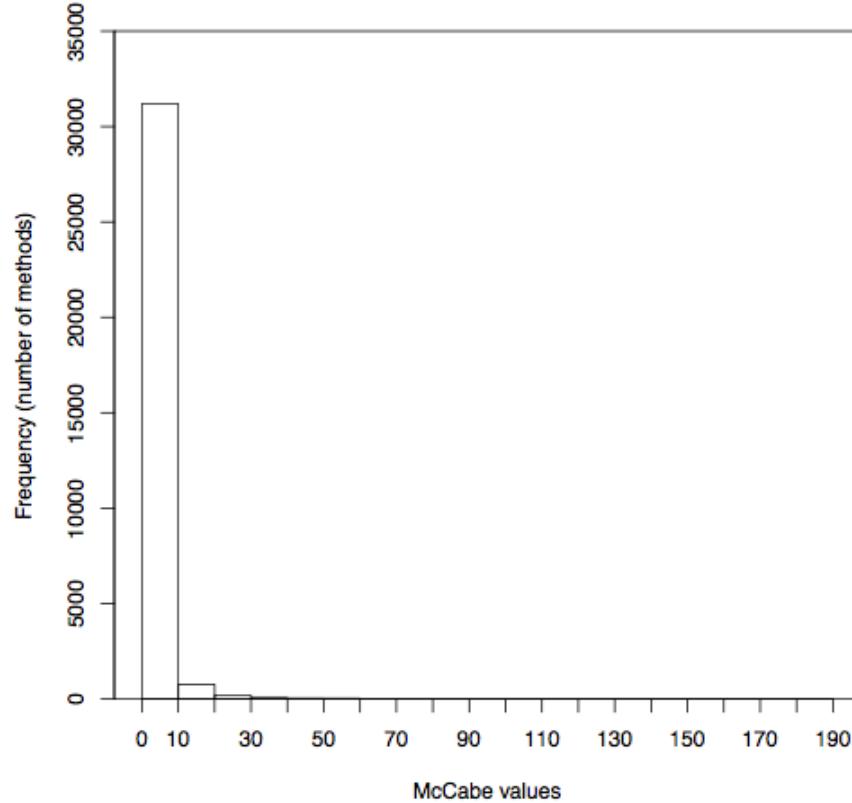


Software Improvement Group

13 | 41



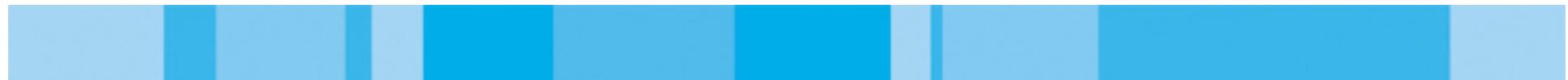
Derivation of risk thresholds: background Histogram vs. Quantile plots



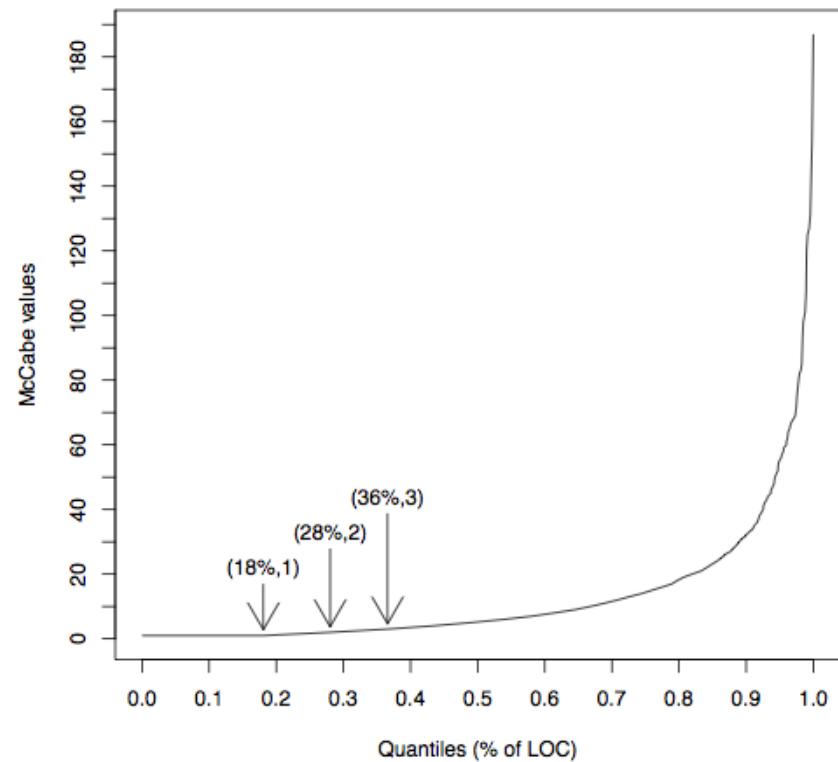
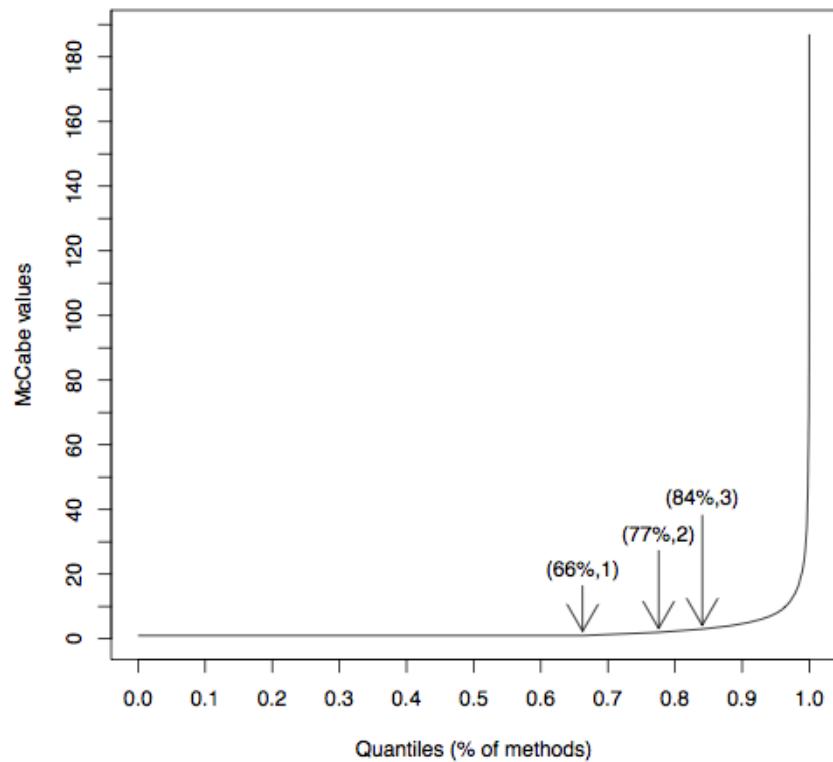
Derivation of risk thresholds: Weight by size



Software Improvement Group



15 | 41

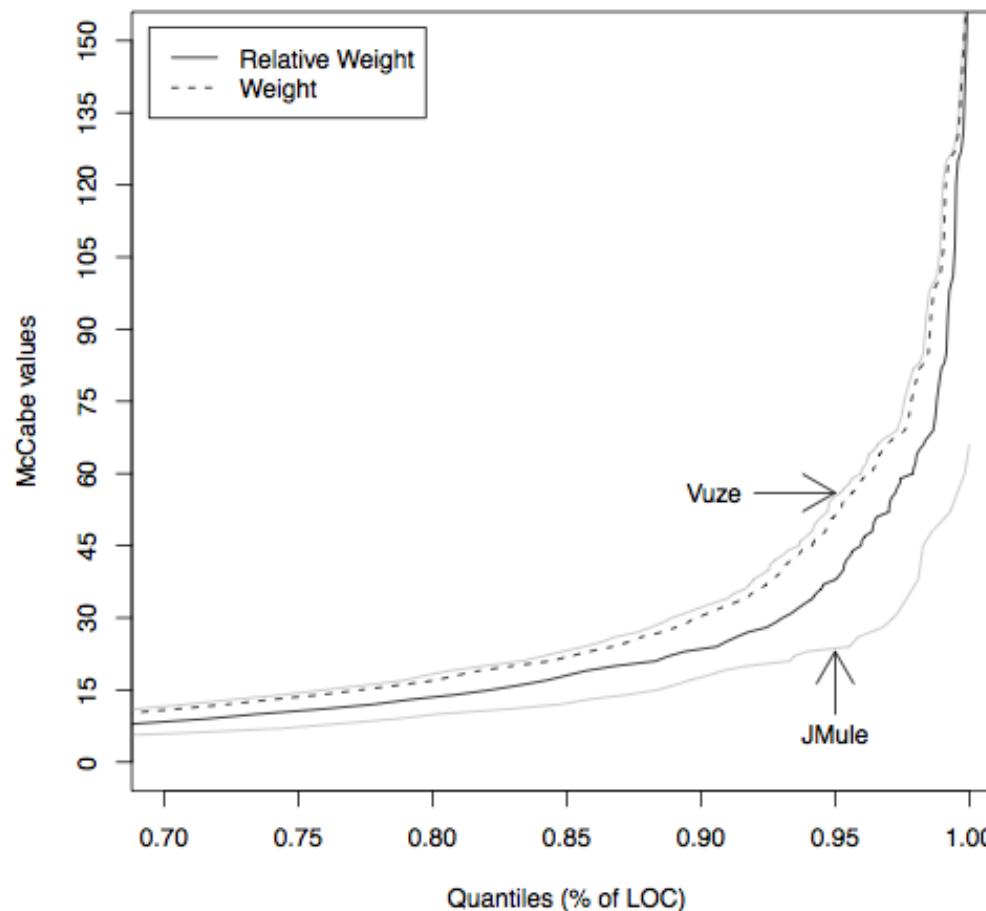


Derivation of risk thresholds

Summarizing a metric distribution



16 | 41

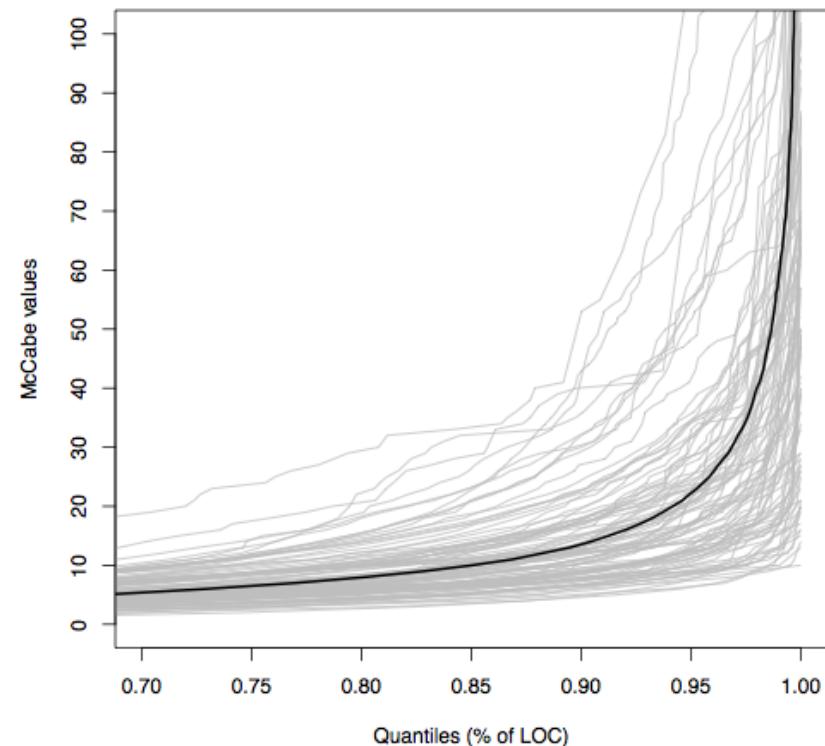
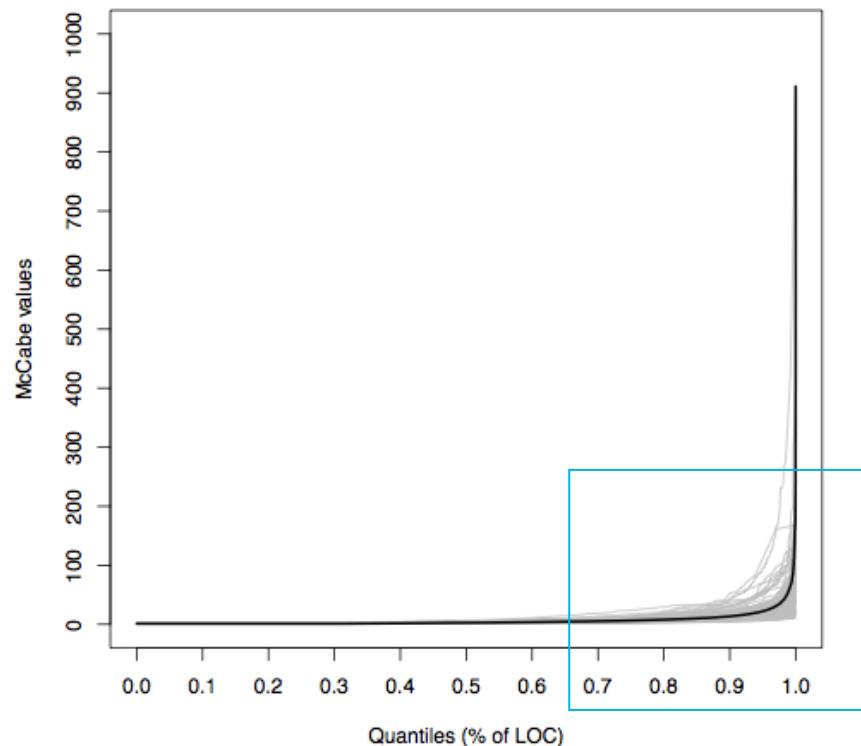


Derivation of risk thresholds

Relative weighting



17 | 41



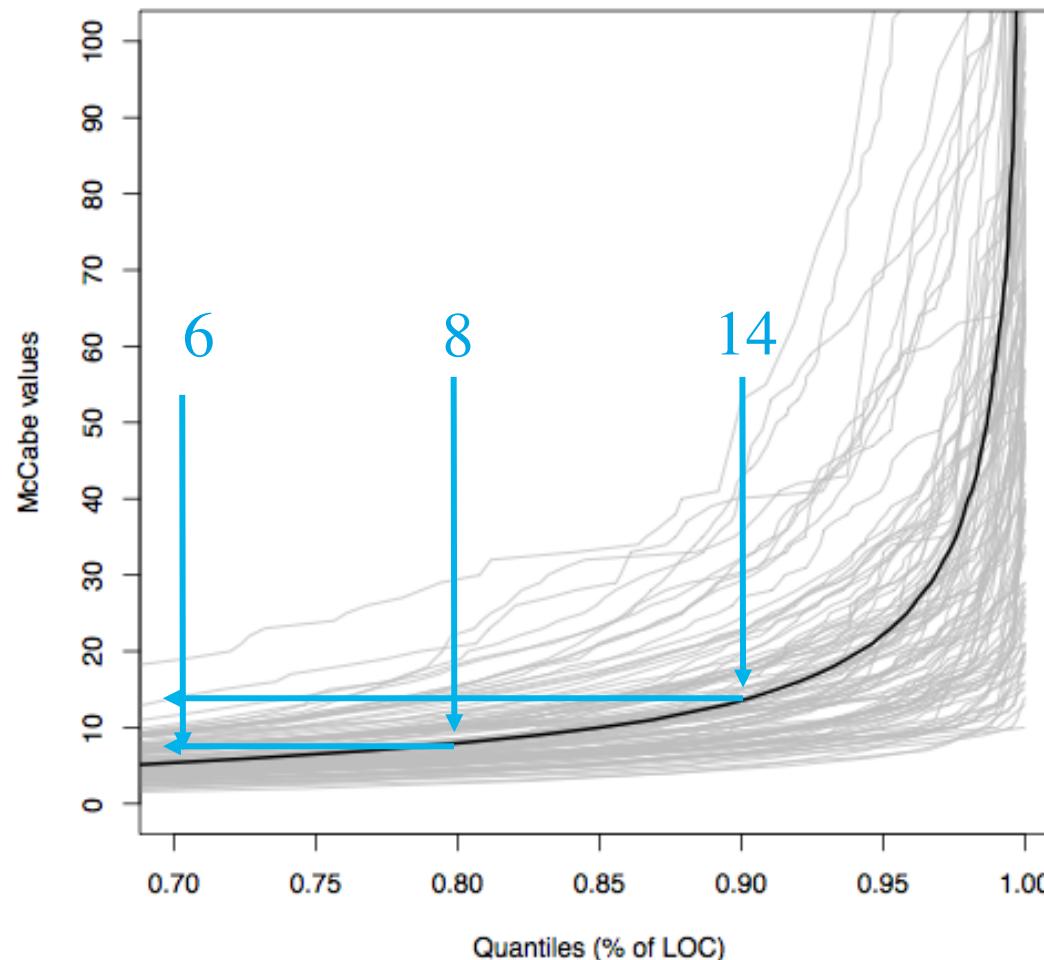
Derivation of risk thresholds: values



Software Improvement Group



18 | 41



Risk Thresholds for the SIG Quality Model



Software Improvement Group



19 | 41

Metric / Quantiles	70%	80%	90%
Unit complexity	6	8	14
Unit size	30	44	74
Module inward coupling	10	22	56
Module interface size	29	42	73
Metric / Quantiles	80%	90%	95%
Unit interfacing	2	3	4

Using thresholds to create risk profiles

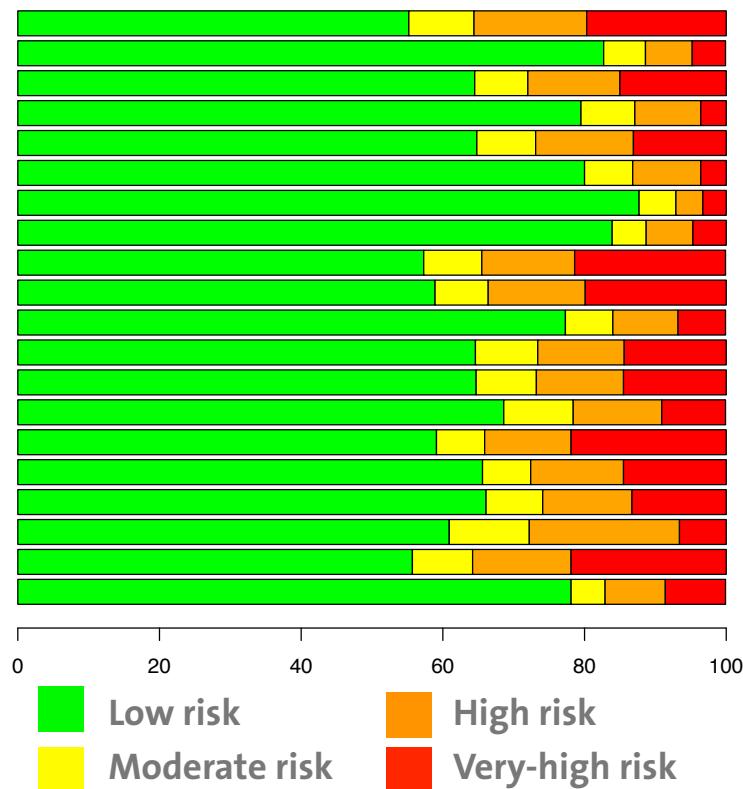


Software Improvement Group



Unit complexity risk profiles for 20 random systems

20 | 41



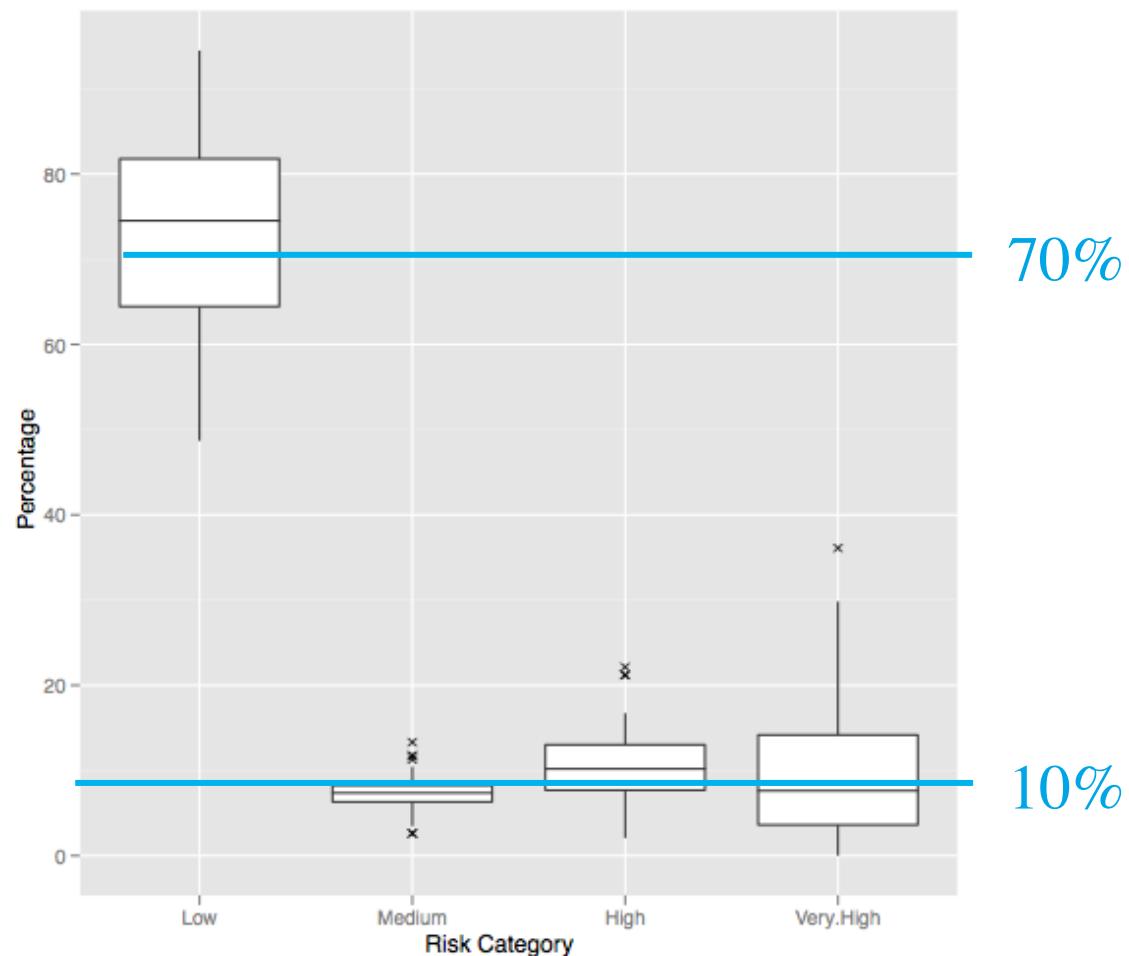
Analysis of risk thresholds



Software Improvement Group



21 | 41



Derivation of risk thresholds

Concluding remarks



22 | 41

Novel methodology to derive metric thresholds

- Solid methodology based on benchmark (depart from expert opinion)
- Agrees with expert opinion (thresholds are sensible)

“The” Lessons

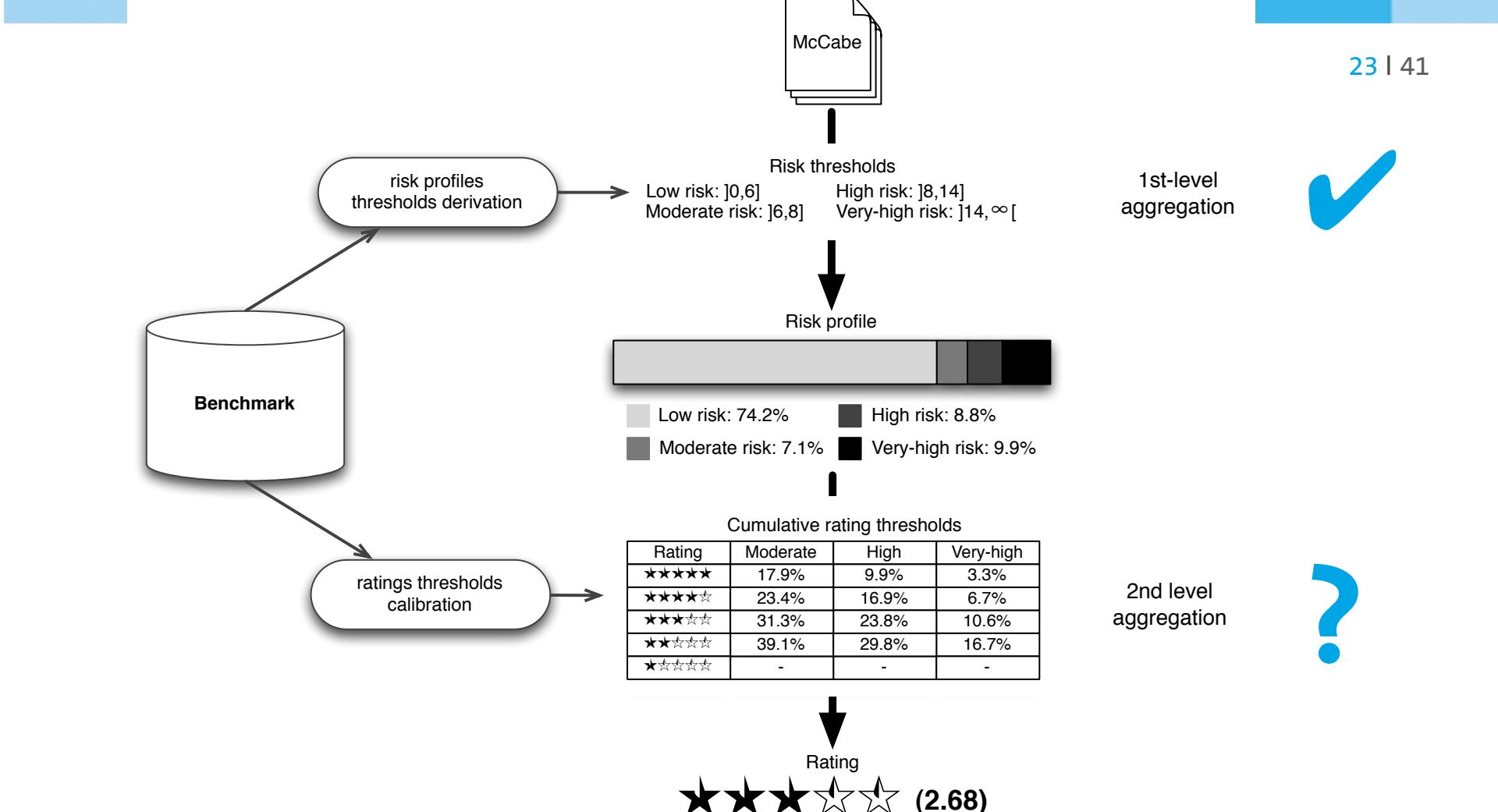
- We can attribute meaning to thresholds
- Benchmarks are of extreme importance

Benchmarking metrics to ratings



Software Improvement Group

23 | 41





Software Improvement Group



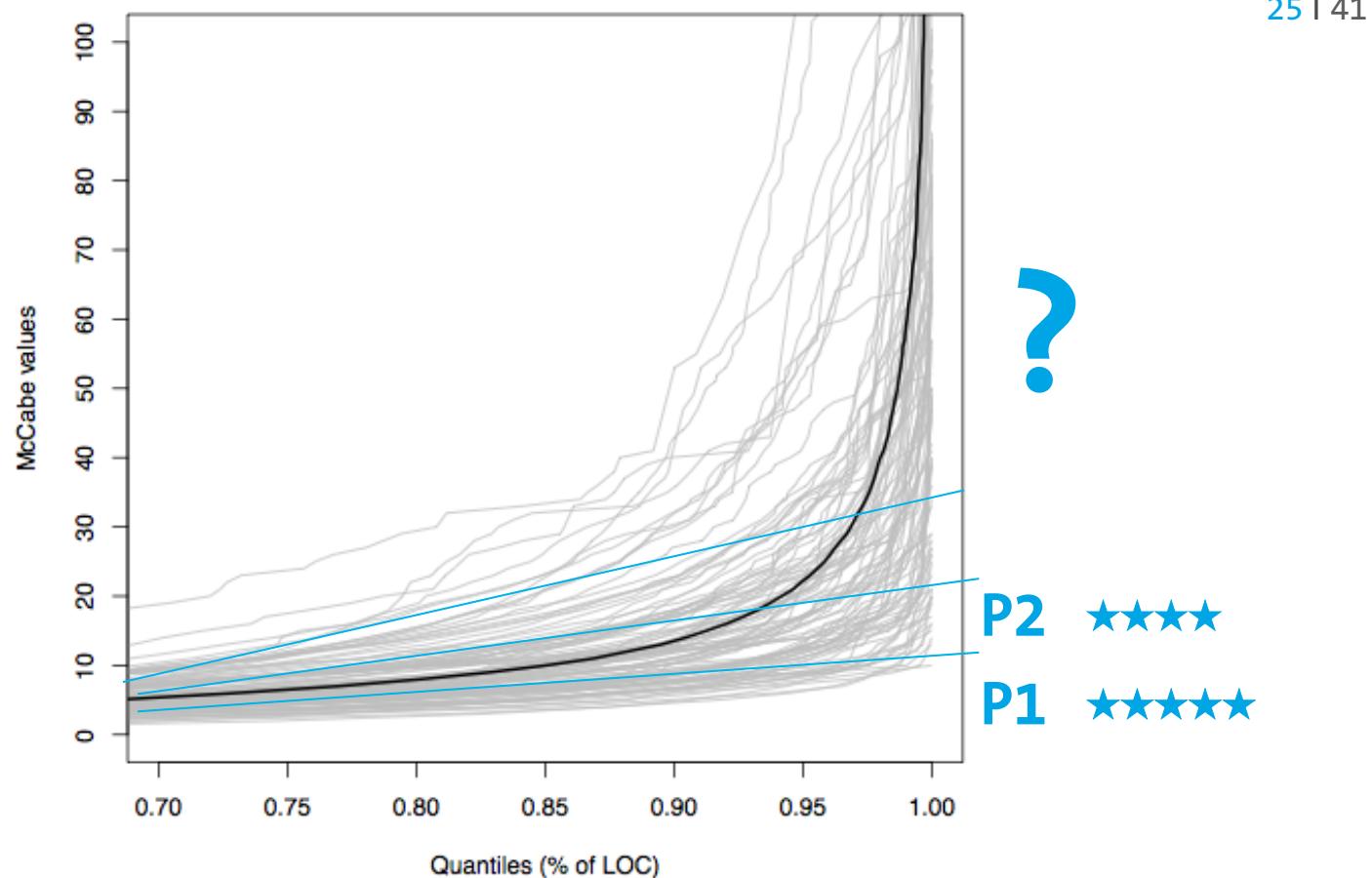
24 | 41

Part II

Calibration of rating thresholds

Benchmark partitioning

The problem



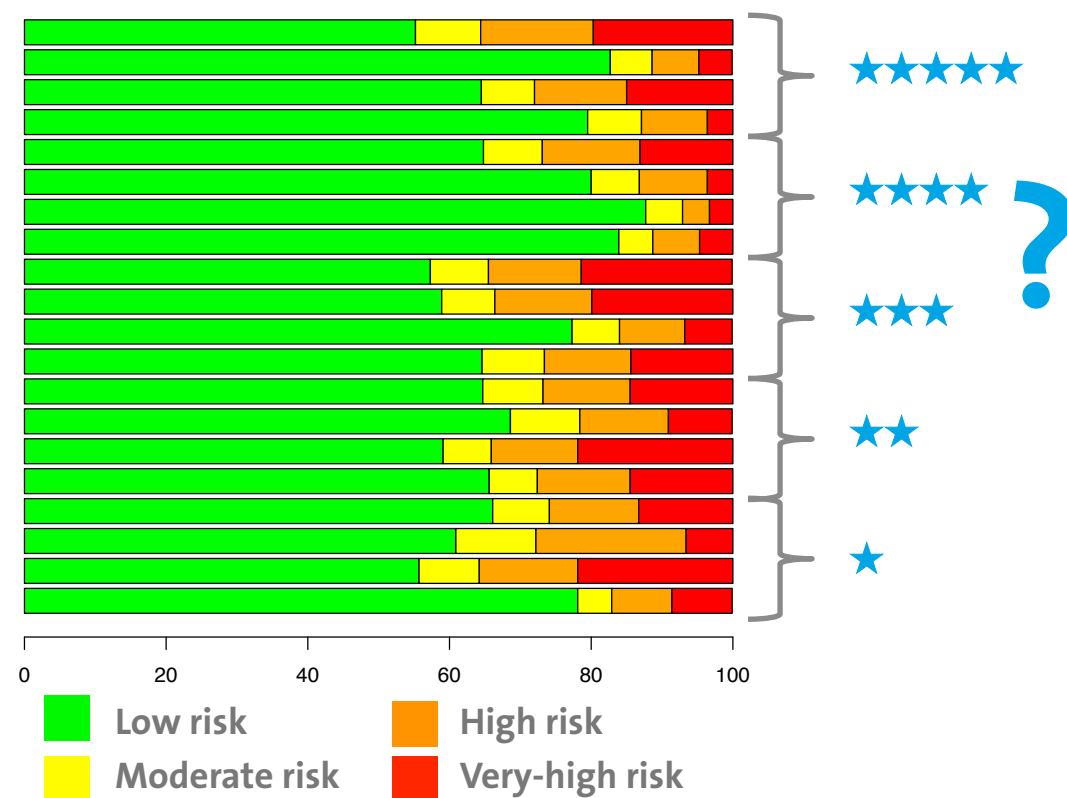
Benchmark partitioning

The problem #2



Unit complexity risk profiles for 20 random systems

26 | 41



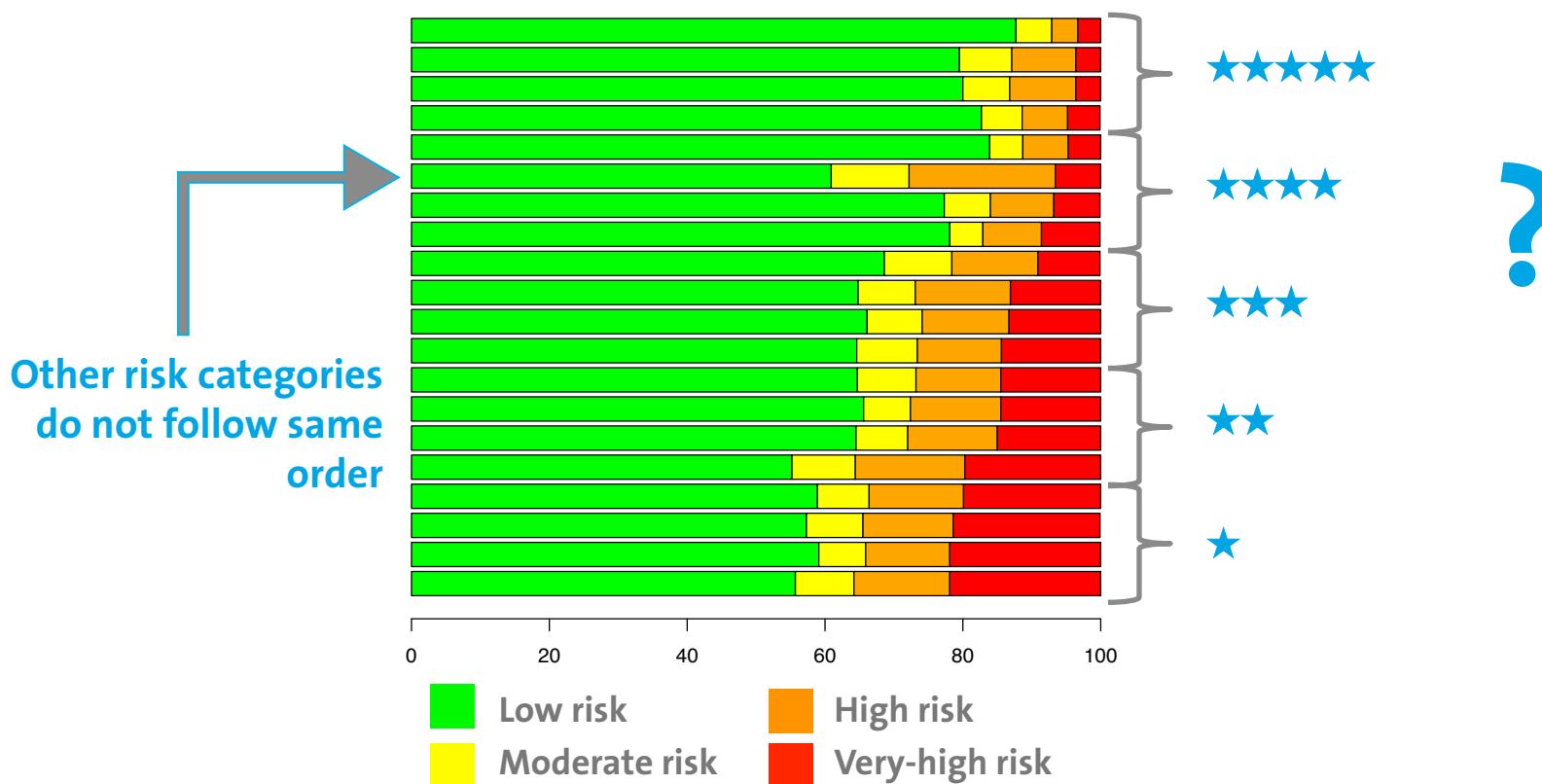
Benchmark partitioning

Order by Very-high risk category



Unit complexity risk profiles for 20 random systems

27 | 41



Ratings calibration algorithm



Software Improvement Group



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

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

28 | 41

Calibrated rating thresholds

Unit complexity (McCabe)



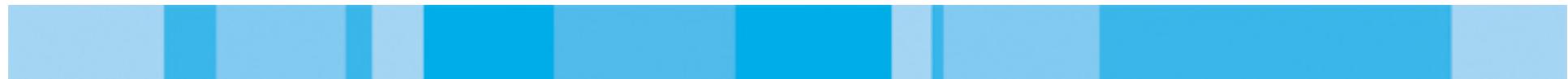
29 | 41

Cumulative rating thresholds calibrated from 100 system benchmark

Rating	Low]0,6]	Moderate]6,8]	High]8,14]	Very-High > 14
★★★★★	-	17.9%	9.9%	3.3%
★★★★	-	23.4%	16.9%	6.7%
★★★	-	31.3%	23.8%	10.6%
★★	-	39.1%	29.8%	16.7%
★	-	-	-	-

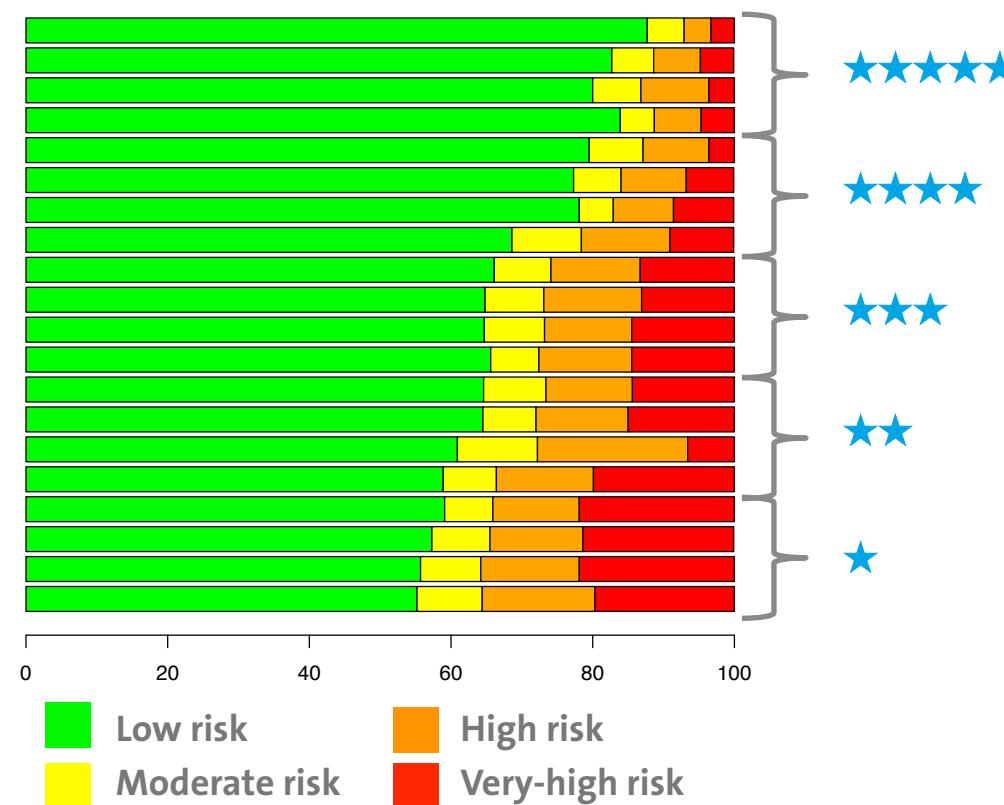
Benchmark partitioning

Calibration algorithm result



Unit complexity risk profiles for 20 random systems

30 | 41



Calibration of rating thresholds

Concluding remarks



31 | 41

Novel methodology to aggregate metric to ratings

- Support of N-point scale (used 5-point star rating)
- Solid methodology based on benchmark
- Enables traceability using thresholds and risk profiles

Plans for the future

- Step for building quality models

“The” Lessons

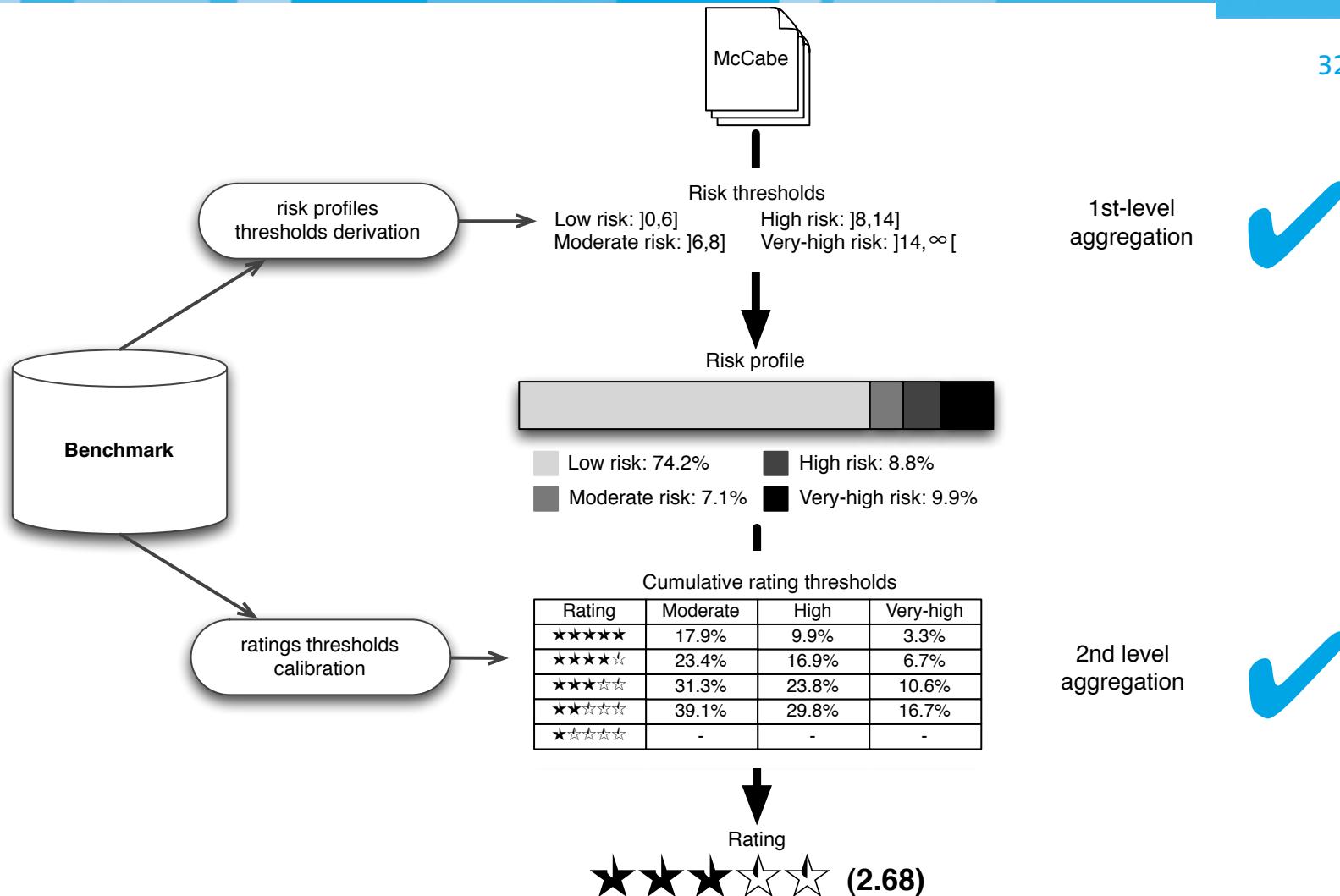
- We can attribute meaning to ratings
- Ratings can be used to rank and evaluate software systems

Benchmarking metrics to ratings



Software Improvement Group

32 | 41





Part III

Using ratings for quality evaluation

Simulators for Space Domain



34 | 41

EuroSim

- Commercial simulator
- Consortium: Dutch Space, NLR, TASK24
- Supports hardware-in-the-loop and man-in-the-loop
- Hard real-time
- Has non-space applications (e.g. aircraft simulation)

ESA/ESOC SimSat

- ESA owned (free)
- Mainly used for spacecraft telecommunication simulation
- Real-time

Technical Analysis on EuroSim

Research Motivation



35 | 41

Comparison of the technical quality between systems of the same domain

- Dutch Space EuroSim mk4.1.5
- ESA SimSat v4.0.1 issue 2

Research question

- How does the technical quality of EuroSim compare to SimSat?
(how to use ratings to evaluate and compare quality)

Scope of the analysis



36 | 41

Analyzed programming languages

- C/C++
- Java

Considerations

- Production and test code analyzed separately
- Excluded documentation code and examples
- Excluded generated code (Icon files generated by XMP)
- Excluded open-source libraries
- Excluded drivers supplied by hardware suppliers
 - device drivers developed by EuroSim Consortium were included in the analysis
- Excluded code not in use: PerflGS, LibCadese

Unit Complexity definition

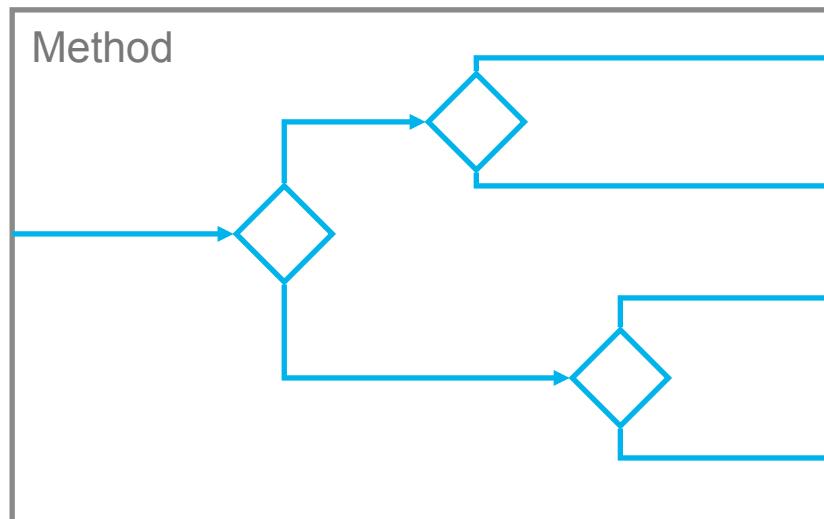


Software Improvement Group



37 | 41

- McCabe, *IEEE Transactions on Software Engineering*, 1976
- Cyclomatic Complexity = Number of decision points per unit (method/function)
- Widely accepted measurement for code complexity
- Should be as small as possible



McCabe: 4

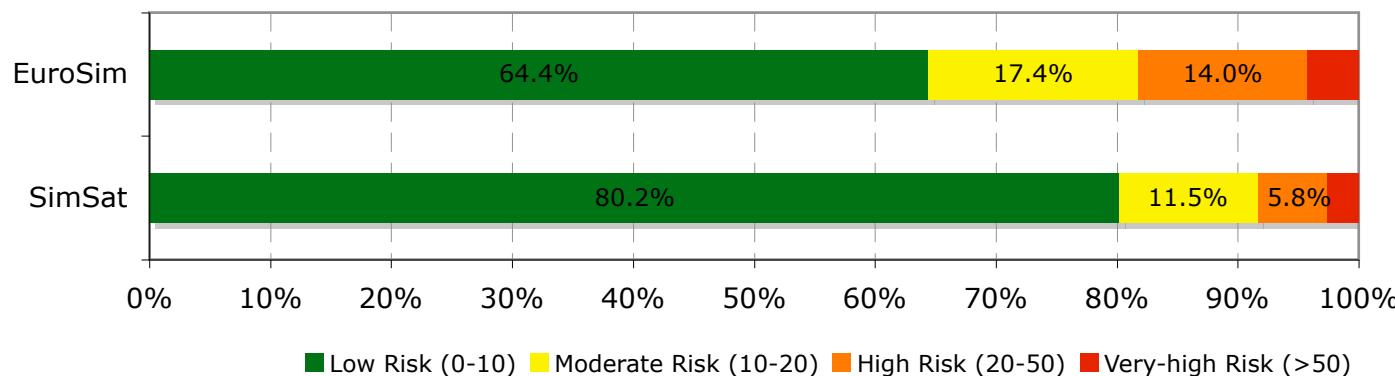
Unit Complexity comparison



Software Improvement Group



38 | 41



Discussion

- Both EuroSim and SimSat rank two stars
- Very-high complexity was found: 4.2% for EuroSim and 2.6% SimSat
- For EuroSim, very-high complexity is localized in 16 methods.
- Taking the last three risk categories, EuroSim has almost twice the risk of SimSat
- For EuroSim, comparing versions reveals slow decrease of the quality

ISO 9126 Maintainability

Comparison between EuroSim and SimSat



	Volume	Duplication	Unit size	Complexity	Unit interfacing	Test quality	
	Score	★★★★	★★★	★★	★★	★★	Score
Dutch Space EuroSim	Analysability	X	X	X			★★★
	Changeability		X		X	X	★★
	Stability				X	X	★★
	Testability			X	X		★★
ESA SimSat	Score	★★★	★★	★★	★★	★★	Score
	Analysability	X	X	X			★★★
	Changeability		X		X	X	★★
	Stability				X	X	★★★
	Testability			X	X		★★

Dutch Space Eurosim Maintainability: ★★★

ESA SimSat Maintainability: ★★

Conclusion



40 | 41

Contributions

- Generic methodology to use metrics for quality evaluation
- Demonstration of the methodology using space-domain software

Benchmark-based approach benefits

- Meaningful (relation with industrial systems)
- Operational (thresholds can be obtained automatically)

Future work

- Use the methodology to validate metrics external characteristics
- Finalize PhD thesis!!!

More info? Feel free to contact...



41 | 41

Tiago L. Alves

E: t.alves@sig.eu
W: www.sig.eu
T: +31 20 3140950