EPS – Engenharia de Produto de Software: Introdução à Qualidade de Software

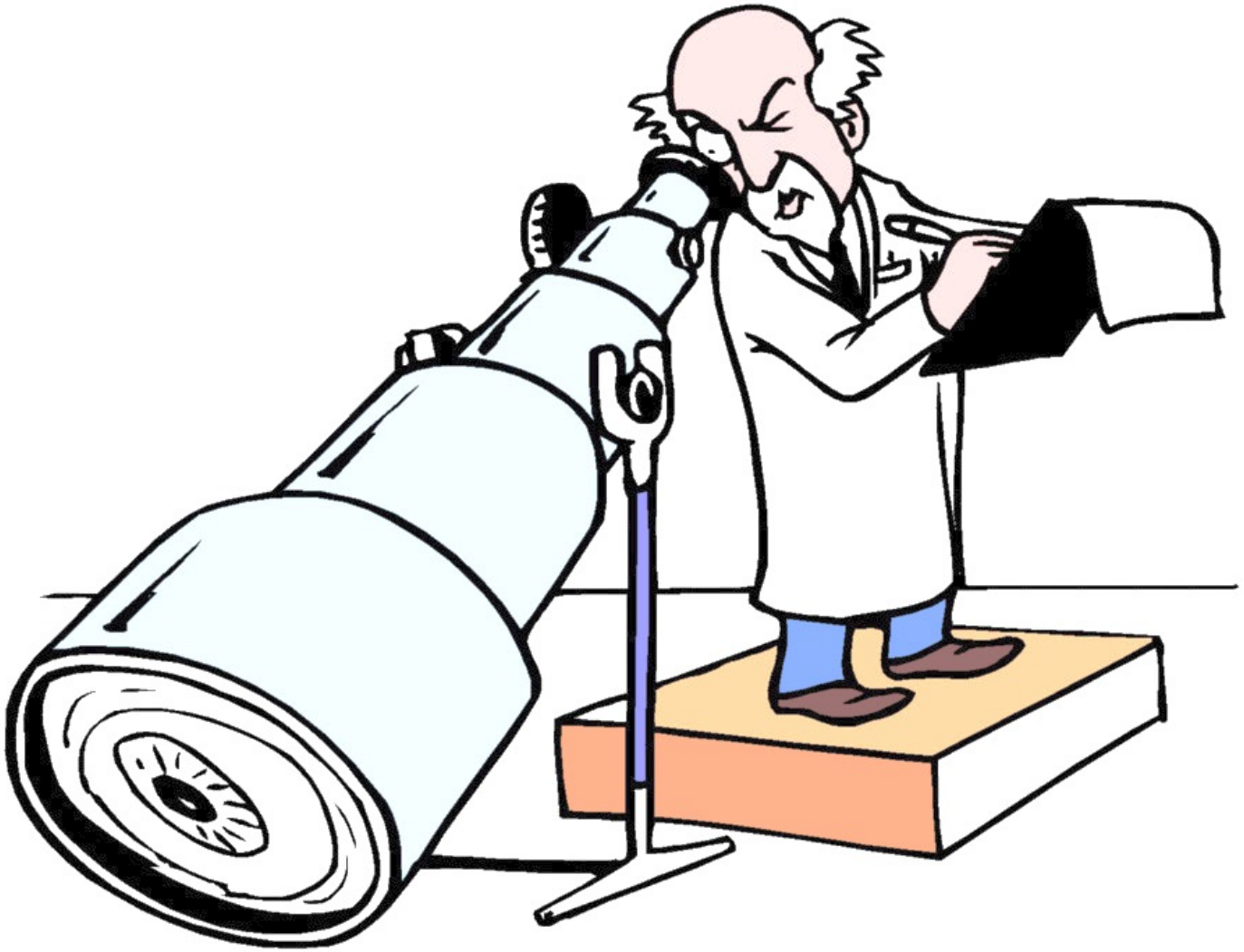
Qualidade de Software ao longo do tempo

(medição, modelos, medidas e ferramentas)

Prof. Hilmer R. Neri

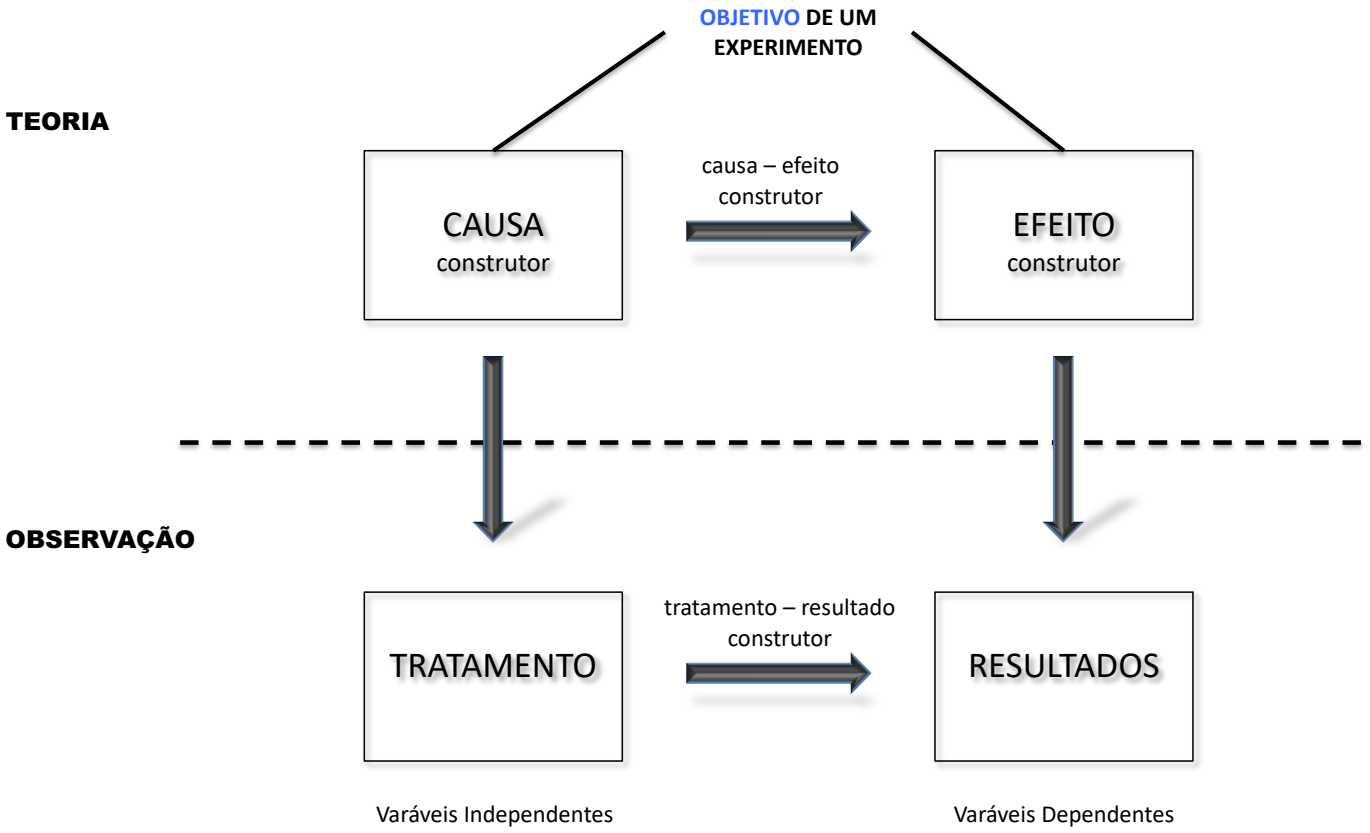
UnB

Observação Científica

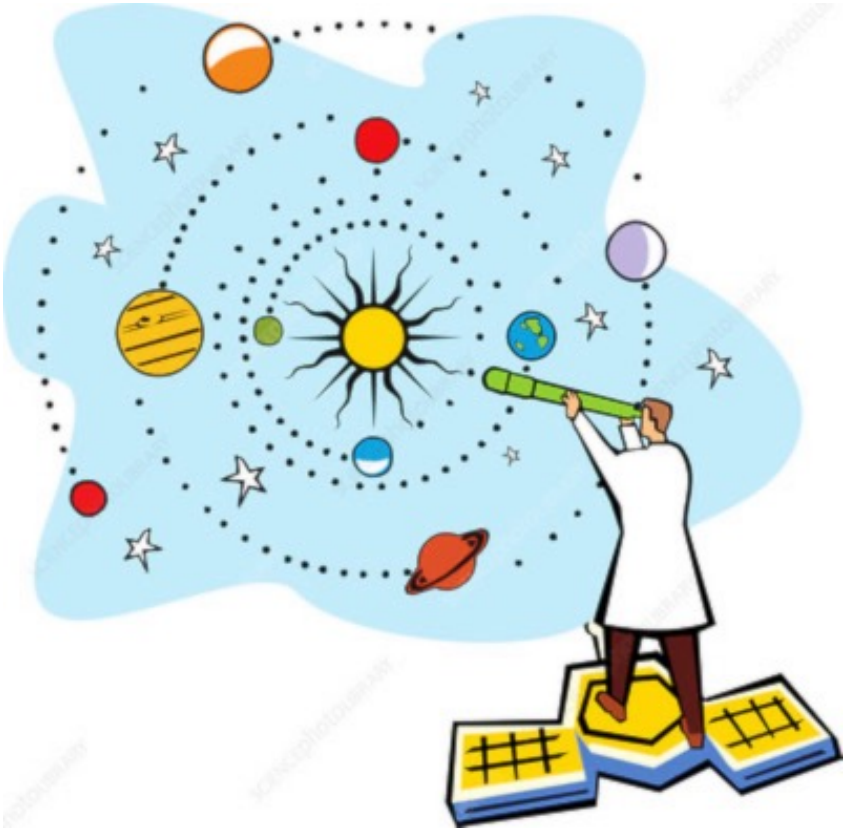


Observação Científica

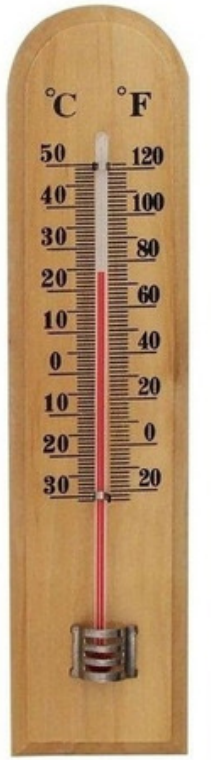
Motivação

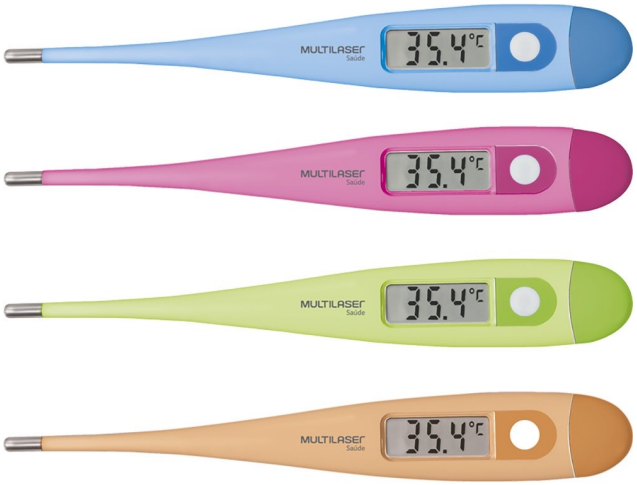


Observação Científica



Por que medir?





Fonte imagens:

- https://milebehind.wordpress.com/2013/11/24/why-measurement-systems-including-the-metric-system-are-important/
- https://www.shrm.org/resourcesandtools/hr-topics/benefits/pages/measure-financial-wellness-success.aspx
- https://www.integritymcg.com/blog/archives/01-2019

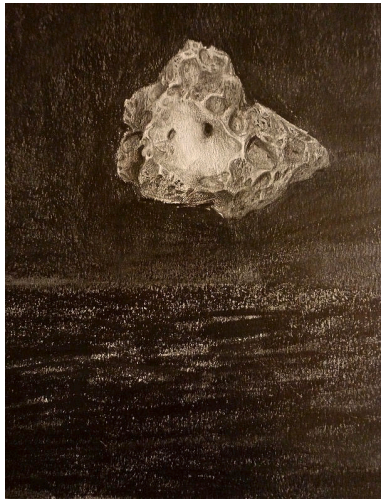
Por que medir?

"Não se consegue controlar o que não se consegue medir"

Tom De Marco – Controlling Software Projects, Yourdon Press 1992

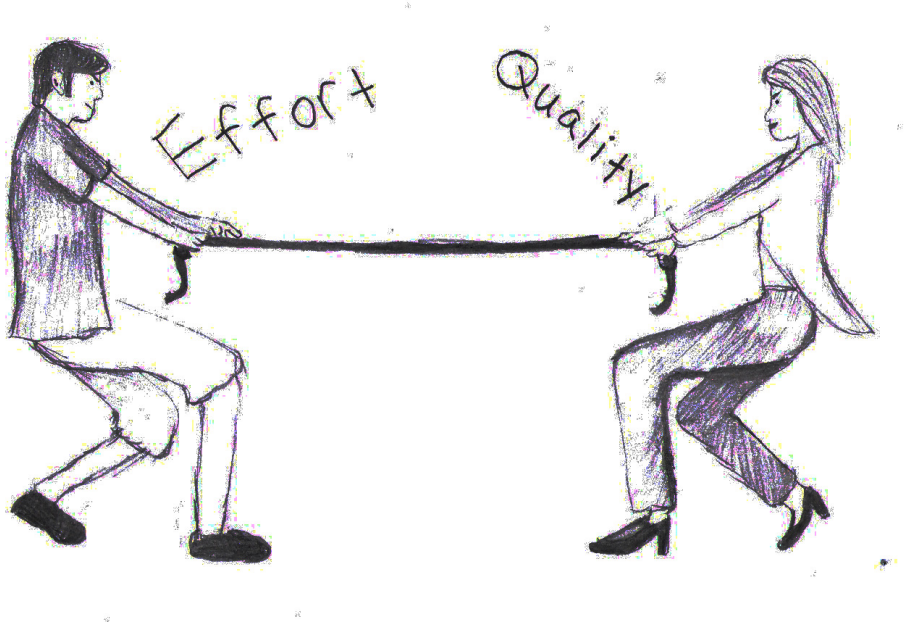
"Não se consegue melhorar o que não se consegue medir"

Fenton & Pfleeger, Software Metrics: a rigorous and practical approach, PWS Publishing Company, 1997




Quality

A qualidade...



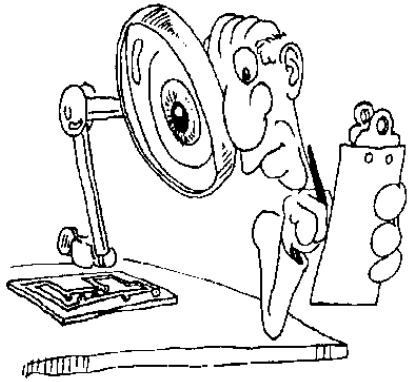
Medição de Software

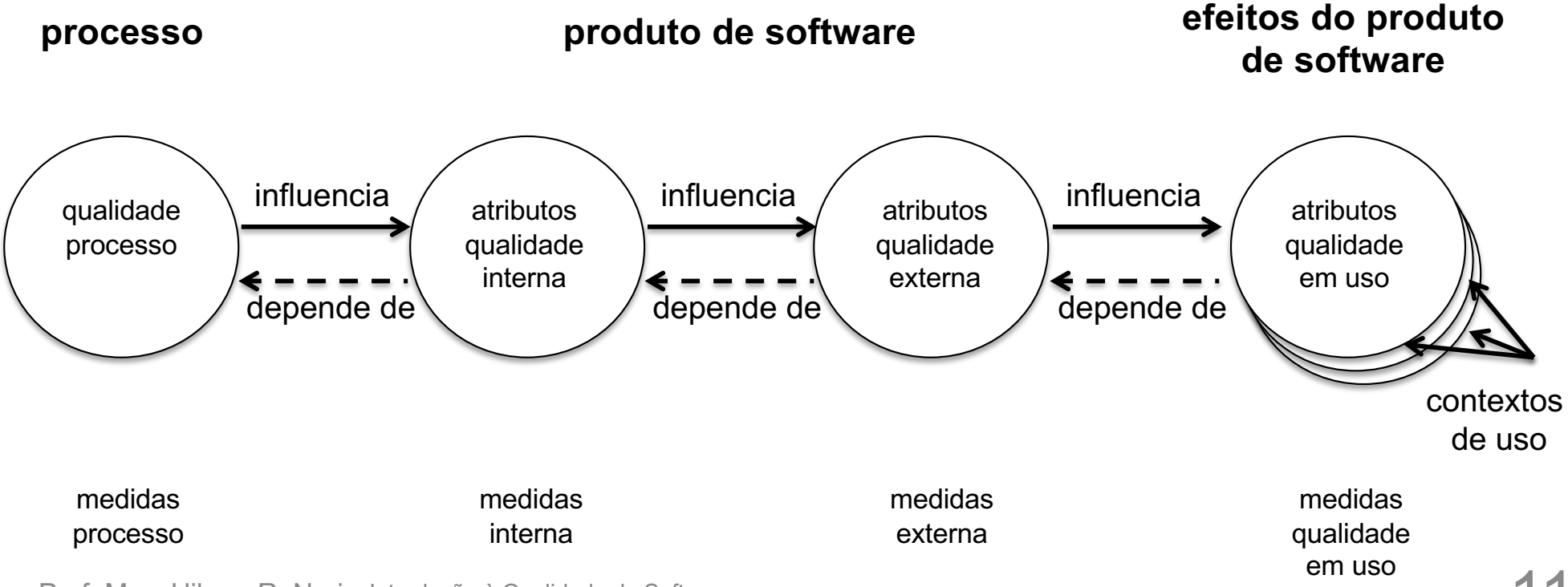
Medição de software é uma avaliação quantitativa ou qualitativa de qualquer aspecto dos processos e produtos de software, que permite seu melhor entendimento e, com isso, auxilia o planejamento, controle e melhoria do que se produz e de como é produzido.

BASS,L., BELADY,L.BROWN,A., FREEMAN,P. ISENSEE,S., KAZMAN,R., KRANSNER,H., MUSA,J. PFEEGER,S, VREDENBURG,K., WASSERMAN,T., 1999, Constructing Superior Software, Software Quality Institute Series, Macmillan Technical Publishing.

- Peso, Altura, temperatura...?

Dimensões da Qualidade de Software





Processos e Normas

Desenvolvimento de produtos e Gerenciamento de projetos de software

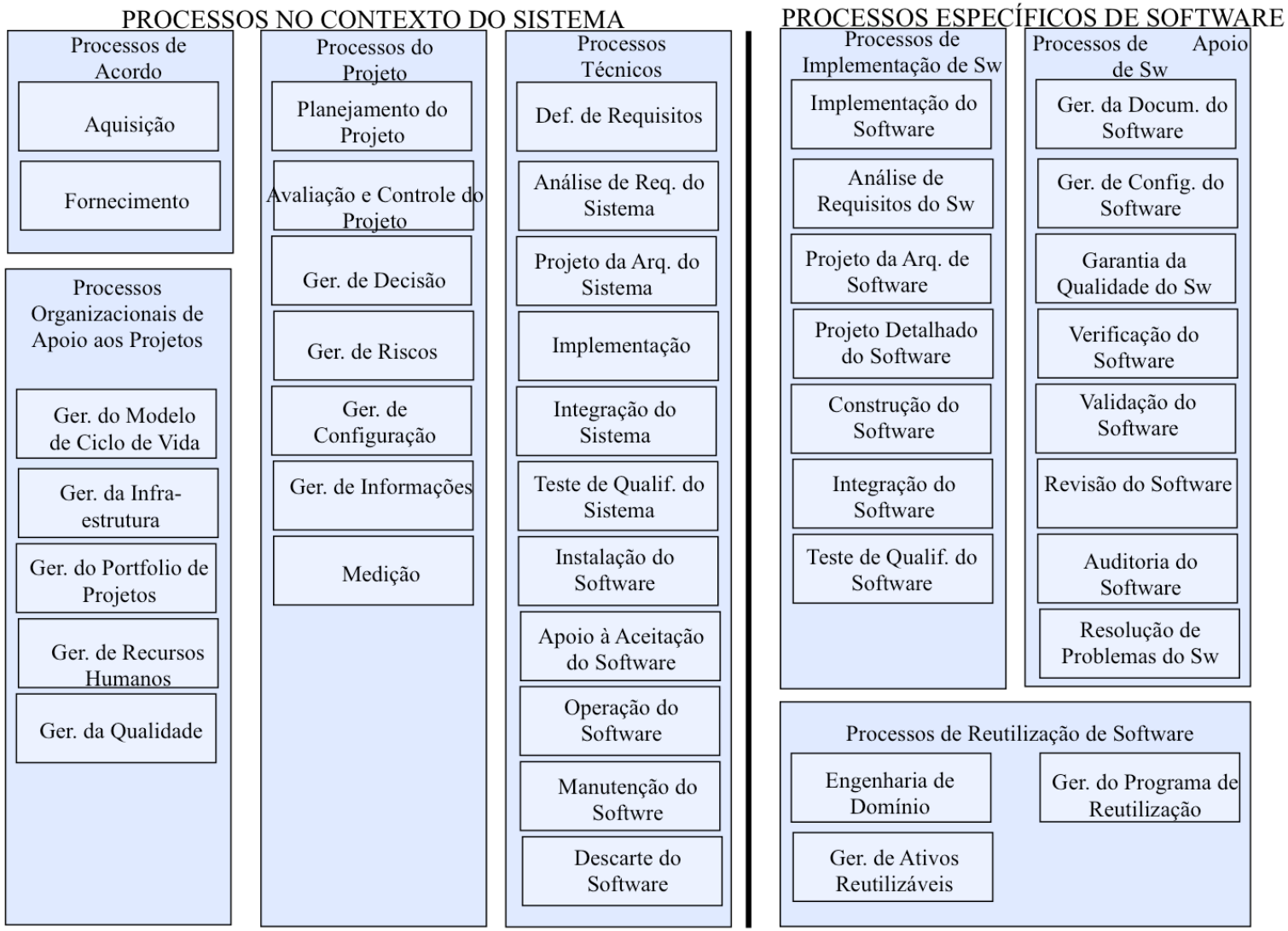
- monitoramento da execução e garantia da qualidade

Verificação e Validação

Qualidade de Software

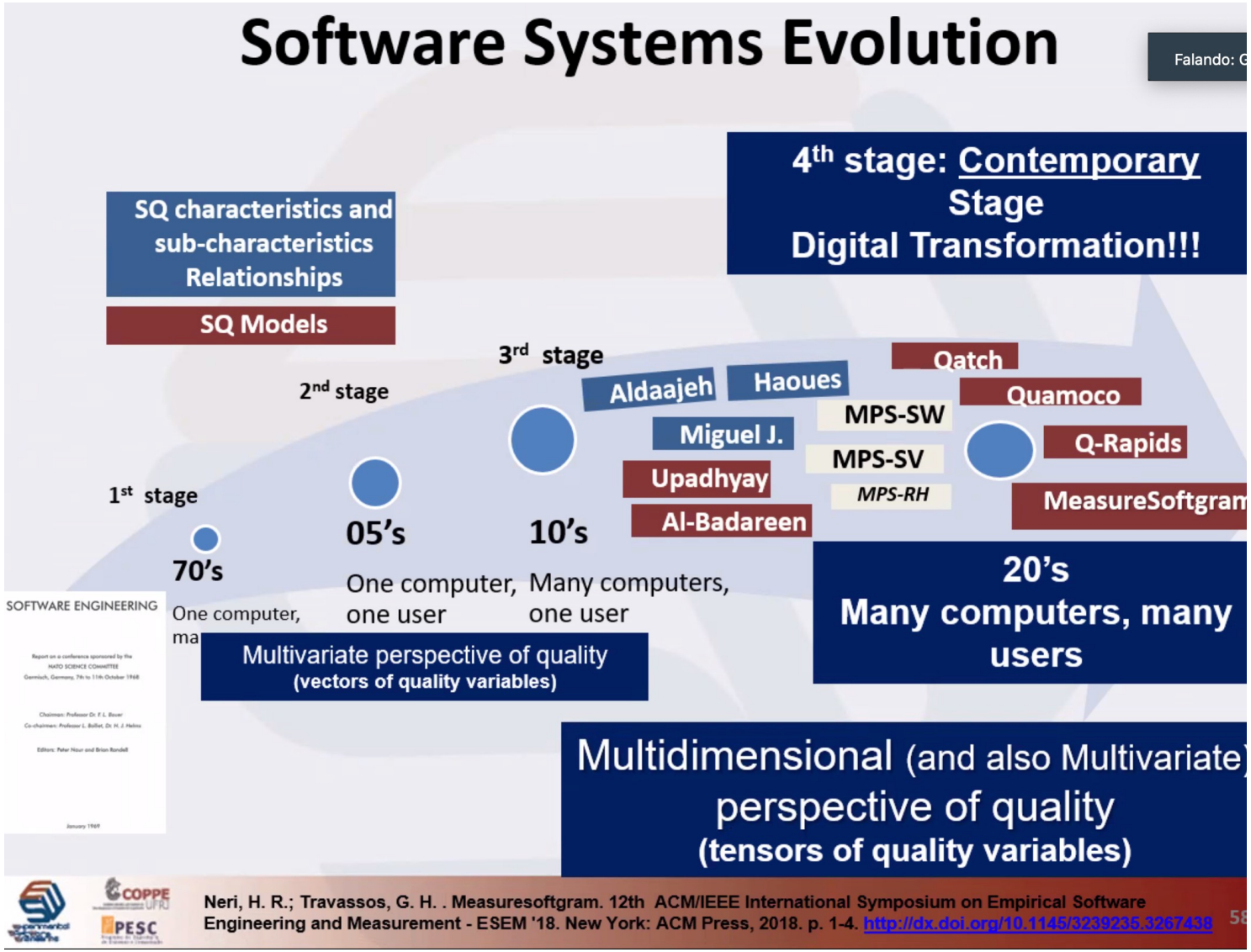
- Produto
  - Família ISO 25000/ (*Software Product Quality Requeriments and Evaluation*)
- Processo
  - ISO 15939 (*Systems and Software Engineering — Measurement Process*)
  - ISO 12207 (*Software Life Cycle Process*)
  - ISO 15504 (Information Technology – Process Assessment)

Áreas do Processo de Desenvolvimento/Produção



Notas de aula da disciplina Controle da Qualidade de Software,profa. Dra. Ana Regina, COPPE-PESC-2016.2

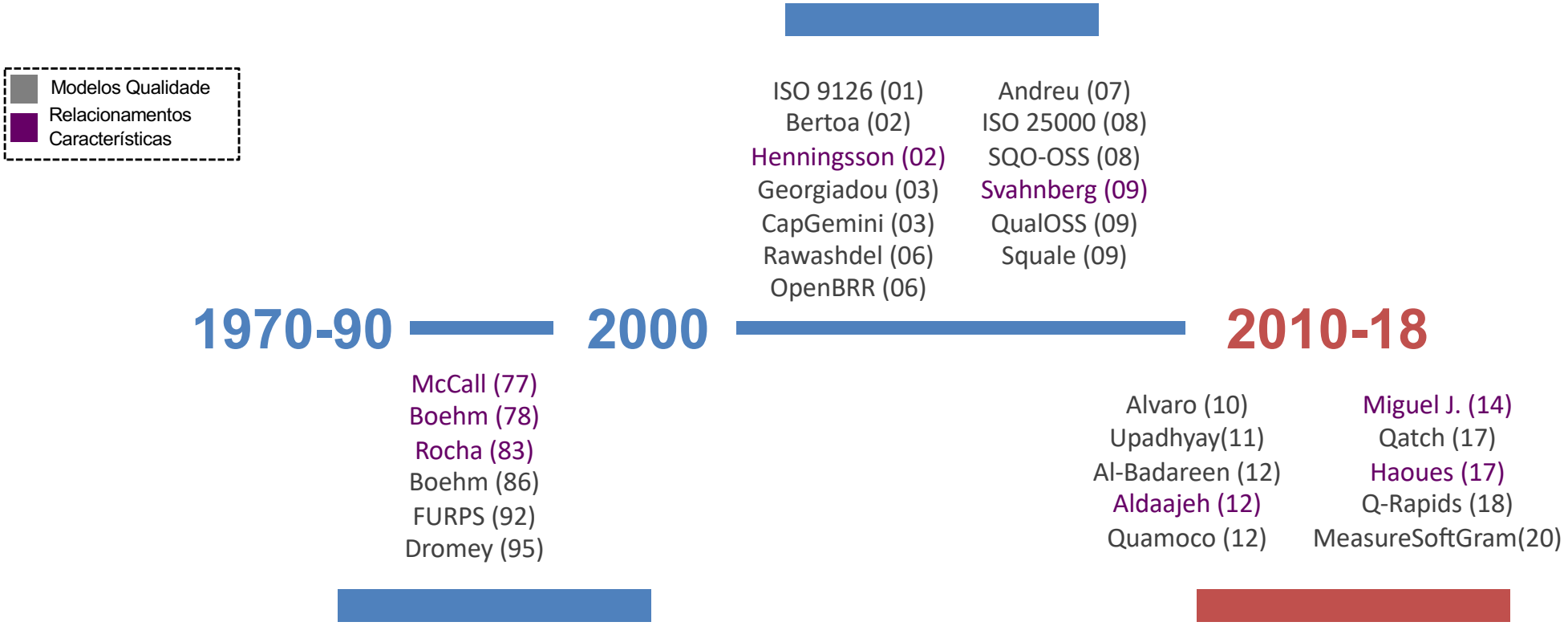
Slide emprestado do GHT



Key Note: **Guilherme H. Travassos, SBQS-2022**

Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Fatores da Qualidade de Produto

Characteristic	McCall	Boehm	FURPS	Dromey	ISO-9126	ISO-25010
Accuracy					X	X
Adaptability			X			X
Analyzability					X	X
Attractiveness					X	X
Changeability					X	X
Correctness	X					X
Efficiency	X	X		X	X	X
Flexibility	X					
Functionality			X	X	X	X
Human Engineering		X				
Installability					X	X
Integrity	X					X
Interoperability	X					X
Maintainability	X			X	X	X
Maturity					X	X
Modifiability						X
Operability					X	X
Performance			X		X	X
Portability	X	X		X	X	X
Reliability	X	X	X	X	X	X
Resource utilization					X	X
Reusability	X			X		X
Stability					X	X
Suitability					X	X
Supportability			X		X	X
Testability	X	X			X	X
Transferability						X
Understandability		X			X	X
Usability	X		X	X	X	X

Miguel JP, Mauricio D, Rodríguez G (2014) A review of software quality models for the evaluation of software products. IJSEA ppl 5(6):31–53

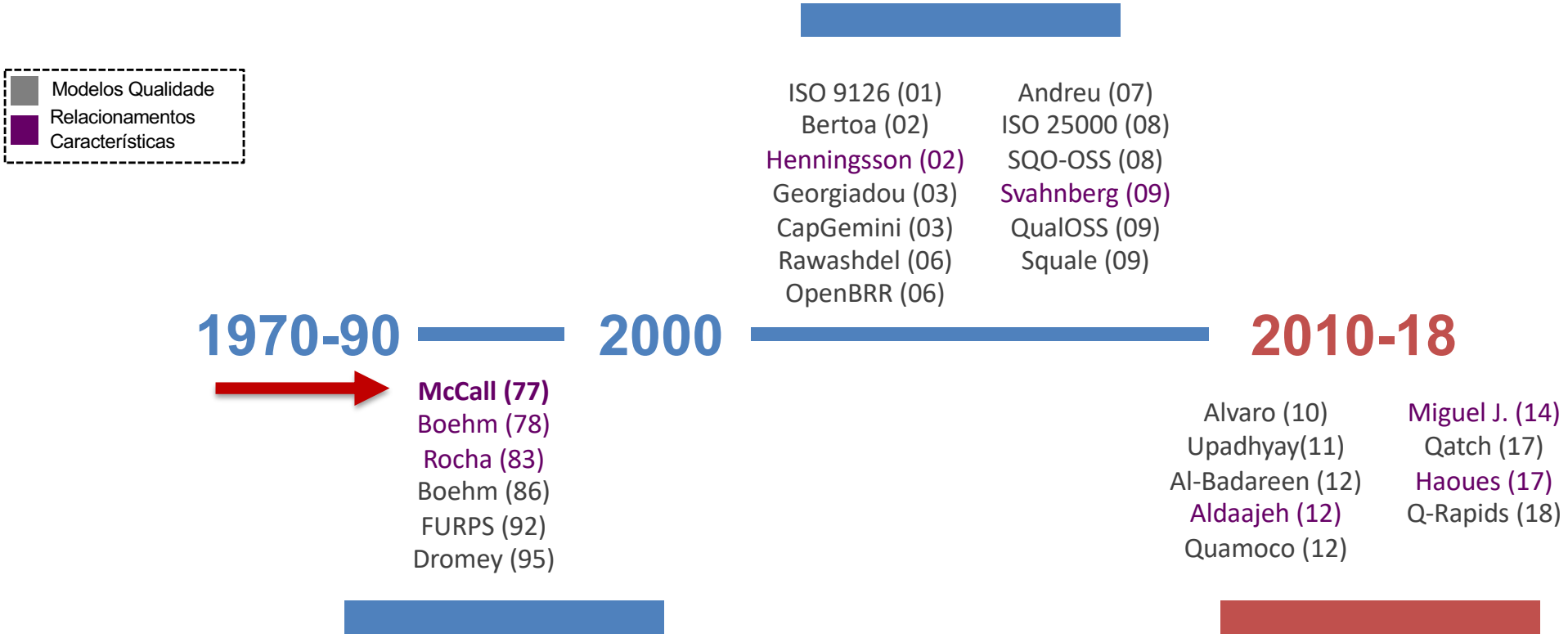
A Engenharia de Software conhece:

- os fatores / características que descrevem o fenômeno da qualidade do software;
- o comportamento multivariado de qualidade

- Sabe como:
  - descobrir limites e valores de referência para métricas;
  - interpretar e julgar valores de uma medida;
  - normalizar as variáveis nos modelos;
  - ponderar características/fatores ;
  - agregar as variáveis em indicadores;
  - julgar as preferências de especialistas em qualidade

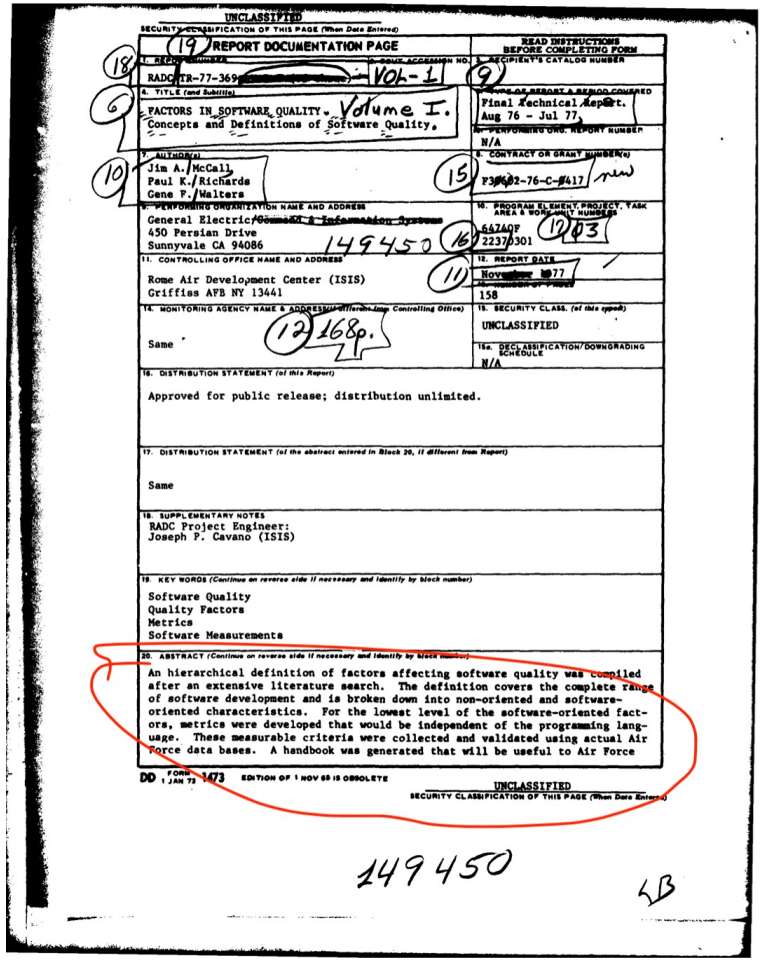
Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality



UNCLASSIFIED

REPORT DOCUMENTATION PAGE

RADC-TR-77-369 Vol-1

FACTORS IN SOFTWARE QUALITY. Volume I. Concepts and Definitions of Software Quality.

Final Technical Report. Aug 76 - Jul 77

N/A

Jim A. McCall
Paul K. Richards
Gene F. Walters

F30602-76-C-0417

General Electric
450 Persian Drive
Sunnyvale CA 94086

64740F
2237
0301

Rome Air Development Center (ISIS)
Griffiss AFB NY 13441

November 1977
158

Same

168p.

UNCLASSIFIED

N/A

Approved for public release; distribution unlimited.

Same

RADC Project Engineer:
Joseph P. Cavano (ISIS)

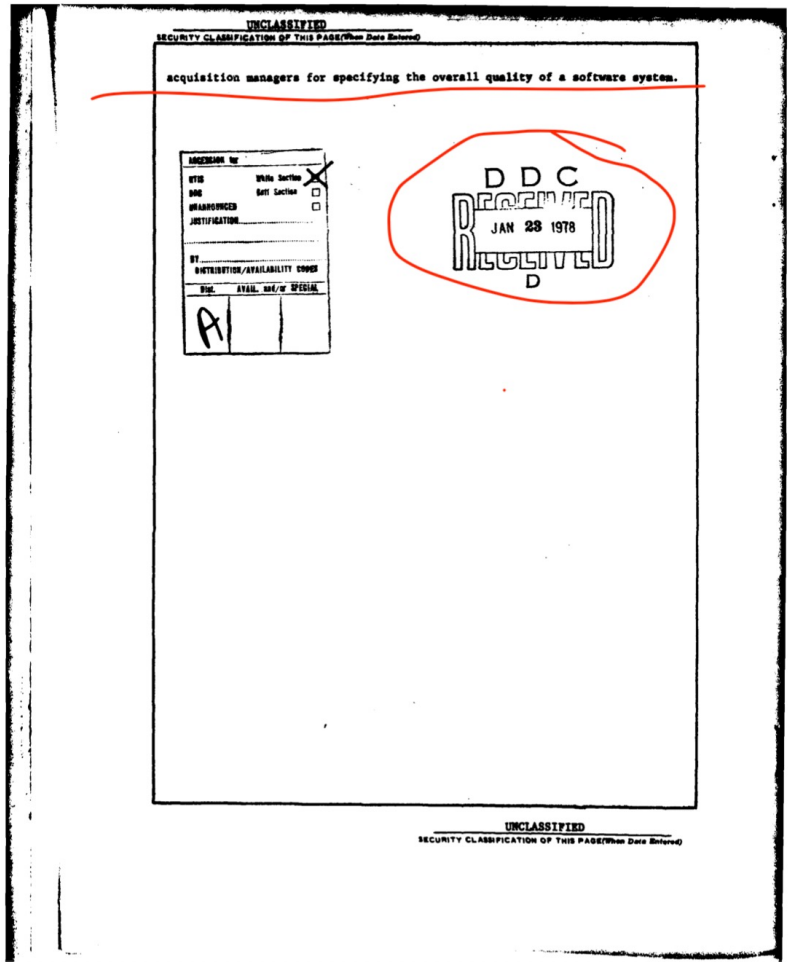
Software Quality
Quality Factors
Metrics
Software Measurements

An hierarchical definition of factors affecting software quality was compiled after an extensive literature search. The definition covers the complete range of software development and is broken down into non-oriented and software-oriented characteristics. For the lowest level of the software-oriented factors, metrics were developed that would be independent of the programming language. These measurable criteria were collected and validated using actual Air Force data bases. A handbook was generated that will be useful to Air Force

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

149450



UNCLASSIFIED

acquisition managers for specifying the overall quality of a software system.

DDC RECEIVED JAN 23 1978 D

UNCLASSIFIED

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

1.1 TASK OVERVIEW

The Factors in Software Quality task was conducted in support of the U.S. Air Force Electronic Systems Division's (ESD) and Rome Air Development Center's (RADC) mission to provide standards and technical guidance to software acquisition managers. ESD sponsored the task and RADC provided technical project management.

In the acquisition of a new software system, a major problem facing a System Program Office (SPO) is to specify the requirements to the software developer, and then to determine whether those requirements are being satisfied as the software system evolves. The parameters of the specification center about the technical definition of the application and the software role within the overall system. Following this, a realistic schedule and costs are negotiated.

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

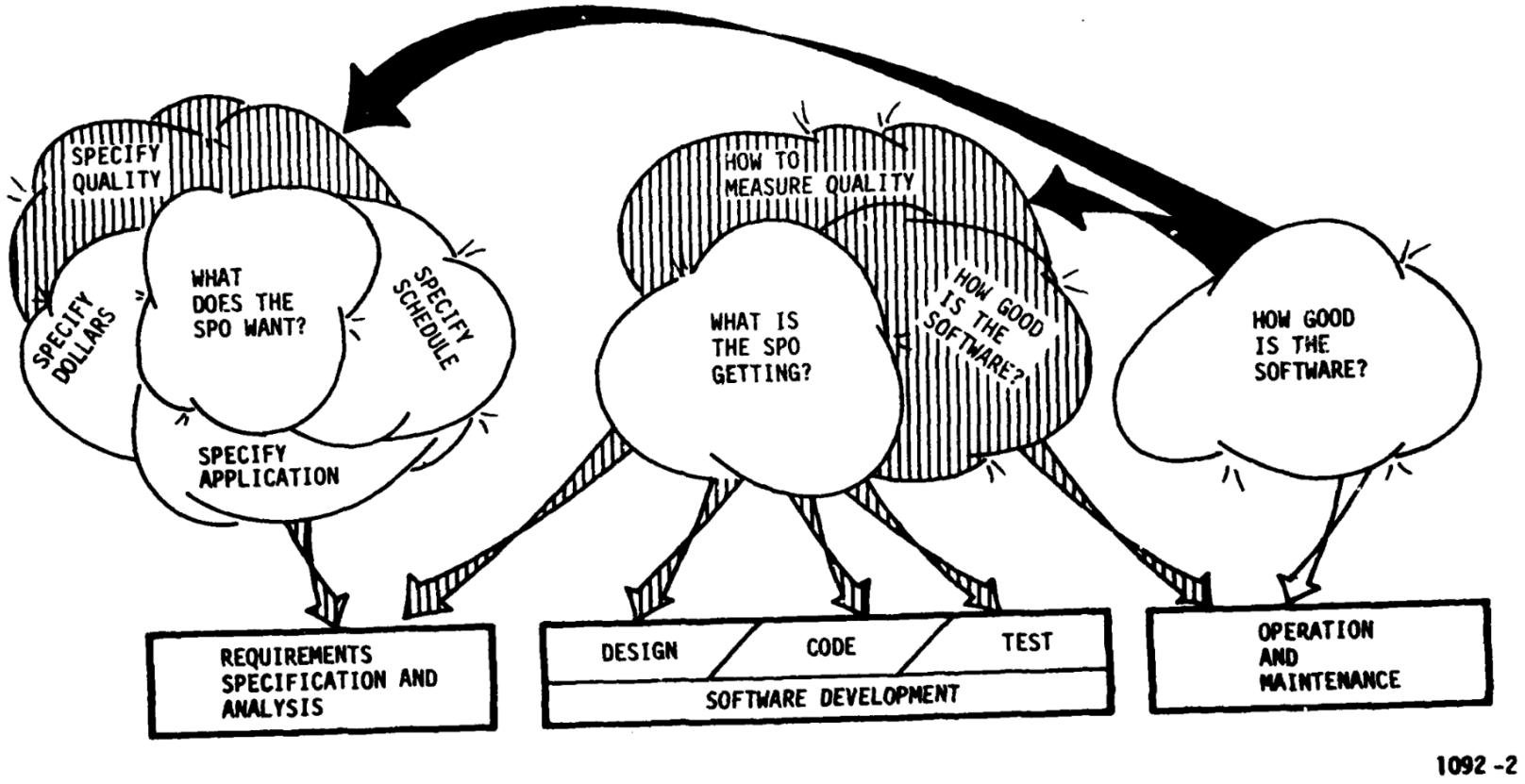


Figure 1.2-1 Specifying and Measuring Quality Software

1-3

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

2.1 DEFINITION OF TERMS

To be consistent in our determination of factors, criteria, and metrics, we first established a set of working definitions. This was done in order to provide a framework from which to more objectively judge candidate quality factors. The working definitions are as follows:

- Software: the programs and documentation associated with and resulting from the software development process.
- Quality: a general term applicable to any trait or characteristic, whether individual or generic, a distinguishing attribute which indicates a degree of excellence or identifies the basic nature of something.
- Factor: a condition or characteristic which actively contributes to the quality of the software. For standardization purposes, all factors will be related to a normalized cost to either perform the activity characterized by the factor or to operate with that degree of quality. For example, maintainability is the effort required to locate and fix an error in an operational program. This effort required may be expressed in units such as time, dollars, or manpower. The following rules were used to determine the prime set of quality factors:
  - a condition or characteristic which contributes to software quality,
  - a user-related characteristic,
  - related to cost either to perform the activity characterized by the function or to operate with that degree of quality,
  - relative characteristic between software products.

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

- Criteria: attributes of the software or software production process by which the factors can be judged and defined. The following rules were applied to the determination of criteria:
  - attributes of the software or software products of the development process; i.e., criteria are software oriented while factors are user oriented,
  - may display a hierarchical relationship with subcriteria,
  - may affect more than one factor.
- Metrics: measures of the criteria or subcriteria related to the quality factors. The measures may be objective or subjective. The units of the metrics are chosen as the ratio of actual occurrences to the possible number of occurrences. Metrics will be discussed further in Section 6.

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

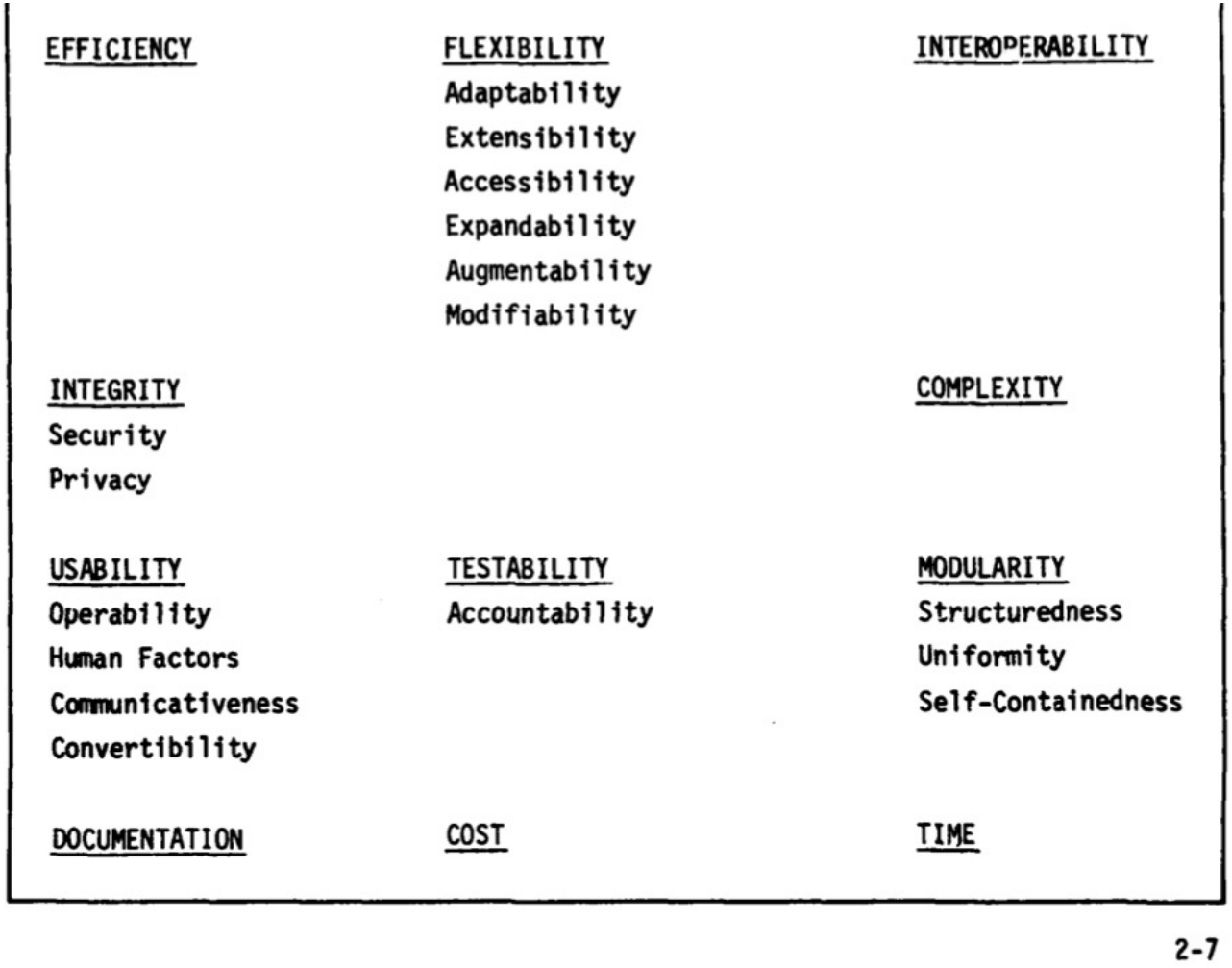
Table 2.4-1 Grouping of Software Quality Factors to Achieve Unambiguous Set

CORRECTNESS	UNDERSTANDABILITY	PORTABILITY
Acceptability	Clarity	Transferability
Completeness	Legibility	Compatability
Consistency	Self-Descriptiveness	
Expression		
Validity		
Performance		
RELIABILITY	**MAINTAINABILITY**	**REUSABILITY**
Availability	Stability	Generality
Accuracy	Manageability	Utility
Robustness	Conciseness	
Precision	Repairability	
Tolerance	Serviceability	

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality



EFFICIENCY	FLEXIBILITY	INTEROPERABILITY
	Adaptability	
	Extensibility	
	Accessibility	
	Expandability	
	Augmentability	
	Modifiability	
INTEGRITY		COMPLEXITY
Security		
Privacy		
USABILITY	TESTABILITY	MODULARITY
Operability	Accountability	Structuredness
Human Factors		Uniformity
Communicativeness		Self-Containedness
Convertibility		
DOCUMENTATION	COST	TIME

2-7

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

Table 3.1-1 Definition of Software Quality Factors

CORRECTNESS	Extent to which a program satisfies its specifications and fulfills the user's mission objectives.
RELIABILITY	Extent to which a program can be expected to perform its intended function with required precision.
EFFICIENCY	The amount of computing resources and code required by a program to perform a function.
INTEGRITY	Extent to which access to software or data by unauthorized persons can be controlled.
USABILITY	Effort required to learn, operate, prepare input, and interpret output of a program.
MAINTAINABILITY	Effort required to locate and fix an error in an operational program.
TESTABILITY	Effort required to test a program to insure it performs its intended function.
FLEXIBILITY	Effort required to modify an operational program.
PORTABILITY	Effort required to transfer a program from one hardware configuration and/or software system environment to another.
REUSABILITY	Extent to which a program can be used in other applications - related to the packaging and scope of the functions that programs perform.
INTEROPERABILITY	Effort required to couple one system with another.

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

Table 6.2-1 Software Quality Metrics

FACTOR(S): CORRECTNESS

CRITERION/ SUBCRITERION	METRIC	REQMTS		DESIGN		IMPLEMENTATION	
		YES/NO 1 OR Ø	VALUE	YES/NO 1 OR Ø	VALUE	YES/NO 1 OR Ø	VALUE
TRACEABILITY	TR. 1 Cross reference relating modules to requirements. $\left(\frac{\text{\# itemized requirements traced}}{\text{total \# requirements}}\right)$				☐		☐
	SYSTEM METRIC VALUE: Same as above line				☐		☐
COMPLETENESS	CP. 1 COMPLETENESS CHECKLIST:						
	(1) Unambiguous references (input, function, output).	☐		☐		☐	
	(2) All data references defined, computed, or obtained from an external source.	☐		☐		☐	
	(3) All defined functions used.	☐		☐		☐	
	(4) All referenced functions defined.	☐		☐		☐	
	(5) All conditions and processing defined for each decision point.	☐		☐		☐	
	(6) All defined and referenced calling sequence parameters agree.			☐		☐	
	(7) All problem reports resolved.	☐		☐		☐	
	(8) Design agrees with requirements.			☐			
	(9) Code agrees with design.					☐	
	SYSTEM METRIC VALUE = $\frac{\sum_{i}^{9} \text{Score for element } i}{9}$		☐		☐		☐

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

Table 6.2-1 Software Quality Metrics

FACTOR(S): CORRECTNESS

CRITERION/ SUBCRITERION	METRIC	REQMTS		DESIGN		IMPLEMENTATION	
		YES/NO 1 OR Ø	VALUE	YES/NO 1 OR Ø	VALUE	YES/NO 1 OR Ø	VALUE
TRACEABILITY	TR. 1 Cross reference relating modules to requirements. $\left(\frac{\text{\# itemized requirements traced}}{\text{total \# requirements}}\right)$				☐		☐
	SYSTEM METRIC VALUE: Same as above line				☐		☐
COMPLETENESS	CP. 1 COMPLETENESS CHECKLIST:						
	(1) Unambiguous references (input, function, output).	☐		☐		☐	
	(2) All data references defined, computed, or obtained from an external source.	☐		☐		☐	
	(3) All defined functions used.	☐		☐		☐	
	(4) All referenced functions defined.	☐		☐		☐	
	(5) All conditions and processing defined for each decision point.	☐		☐		☐	
	(6) All defined and referenced calling sequence parameters agree.			☐		☐	
	(7) All problem reports resolved.	☐		☐		☐	
	(8) Design agrees with requirements.			☐			
	(9) Code agrees with design.					☐	
	SYSTEM METRIC VALUE = $\frac{\sum_{i}^{9} \text{Score for element } i}{9}$		☐		☐		☐

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

Factors in Software Quality

Table 6.2-1 Software Quality Metrics

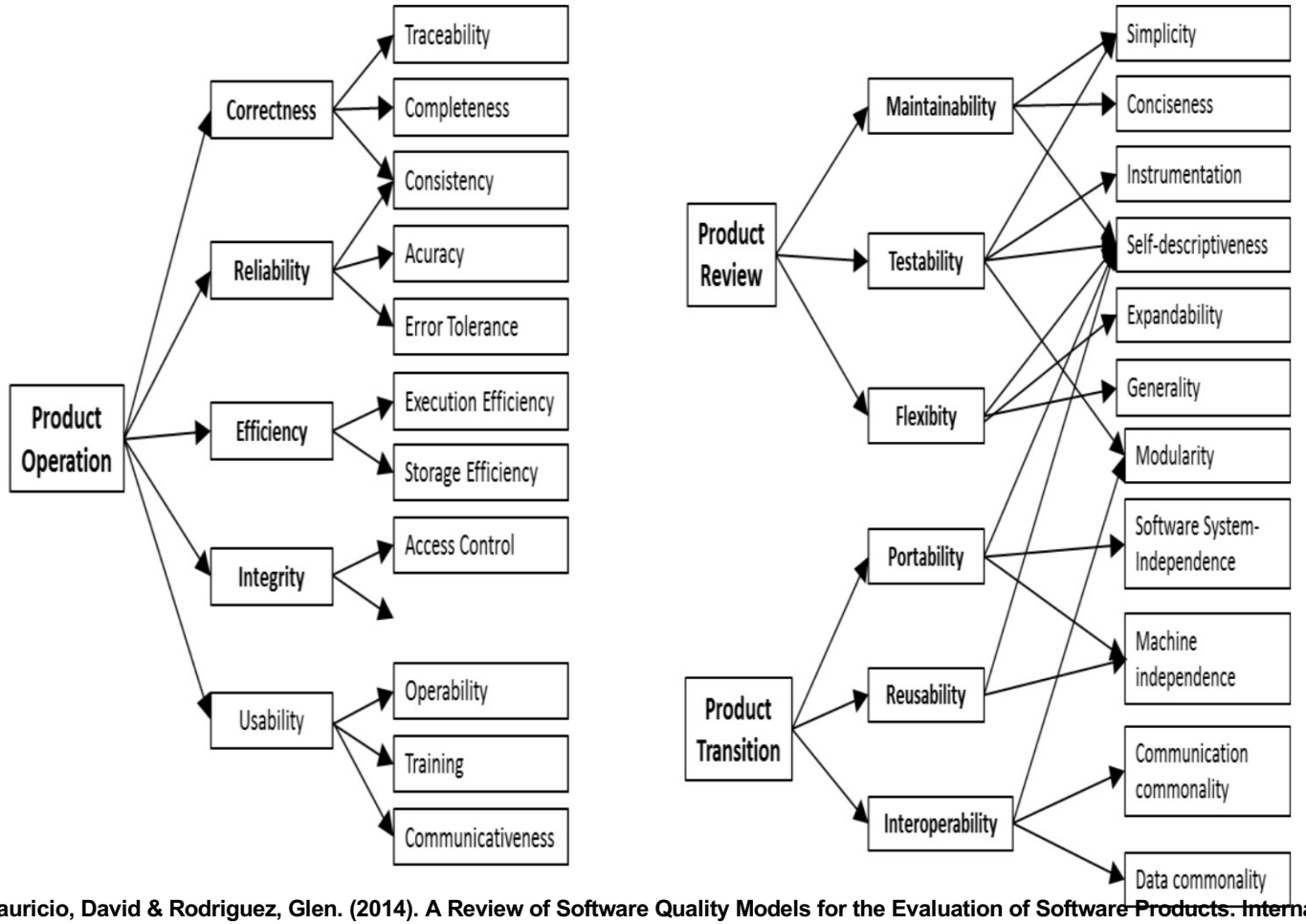
FACTOR(S): CORRECTNESS

CRITERION/ SUBCRITERION	METRIC	REQMTS		DESIGN		IMPLEMENTATION	
		YES/NO 1 OR Ø	VALUE	YES/NO 1 OR Ø	VALUE	YES/NO 1 OR Ø	VALUE
TRACEABILITY	TR. 1 Cross reference relating modules to requirements. $\left(\frac{\text{\# itemized requirements traced}}{\text{total \# requirements}}\right)$				☐		☐
	SYSTEM METRIC VALUE: Same as above line				☐		☐
COMPLETENESS	CP. 1 COMPLETENESS CHECKLIST:						
	(1) Unambiguous references (input, function, output).	☐		☐		☐	
	(2) All data references defined, computed, or obtained from an external source.	☐		☐		☐	
	(3) All defined functions used.	☐		☐		☐	
	(4) All referenced functions defined.	☐		☐		☐	
	(5) All conditions and processing defined for each decision point.	☐		☐		☐	
	(6) All defined and referenced calling sequence parameters agree.			☐		☐	
	(7) All problem reports resolved.	☐		☐		☐	
	(8) Design agrees with requirements.			☐			
	(9) Code agrees with design.					☐	
	SYSTEM METRIC VALUE = $\frac{\sum_{1}^{9} \text{Score for element } i}{9}$		☐		☐		☐

Jim McCall, Paul Richards, and Gene Walters. 1977. Factors in Software Quality. Volume I, II and III. US Rome Air Devel. Center Reports, US Dept. of Comm. (1977)

Modelos de Qualidade ao Longo do Tempo

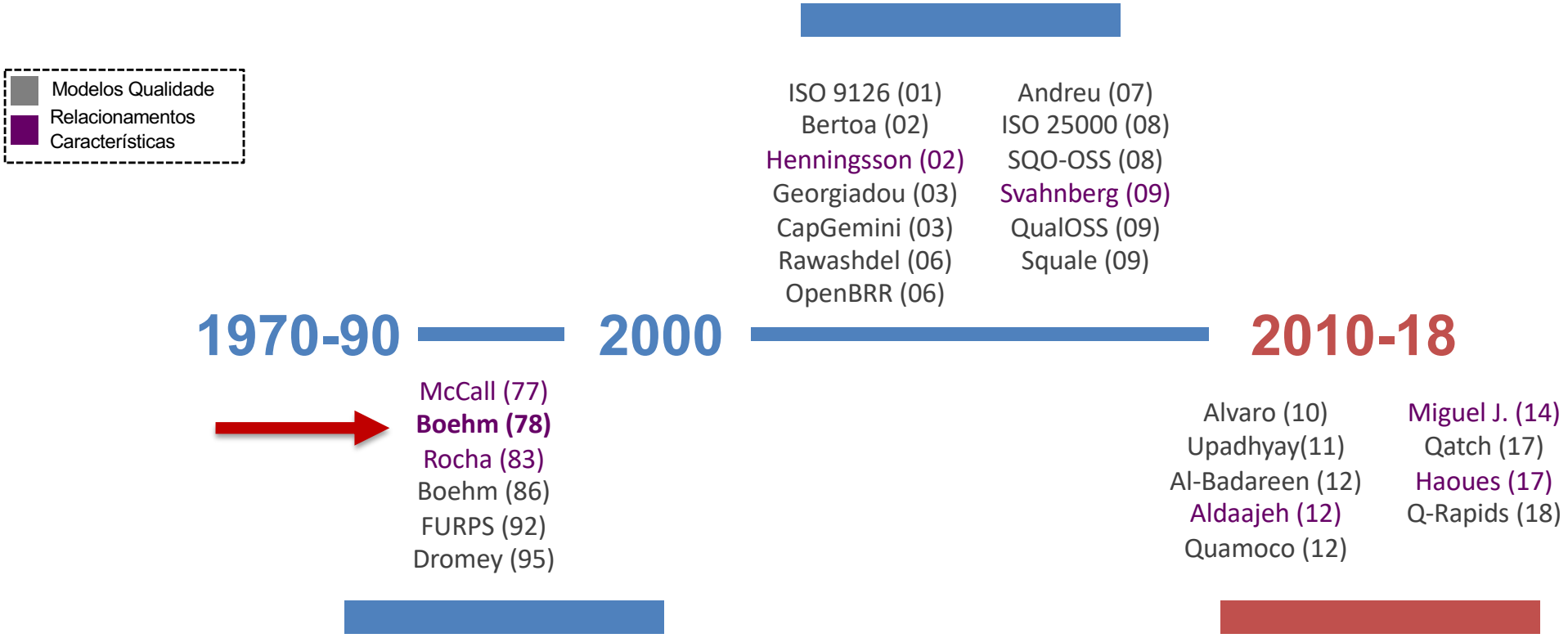
McCall Model



Miguel, Jose & Mauricio, David & Rodriguez, Glen. (2014). A Review of Software Quality Models for the Evaluation of Software Products. International journal of Software Engineering & Applications. 5. 31-54. 10.5121/ijsea.2014.5603.

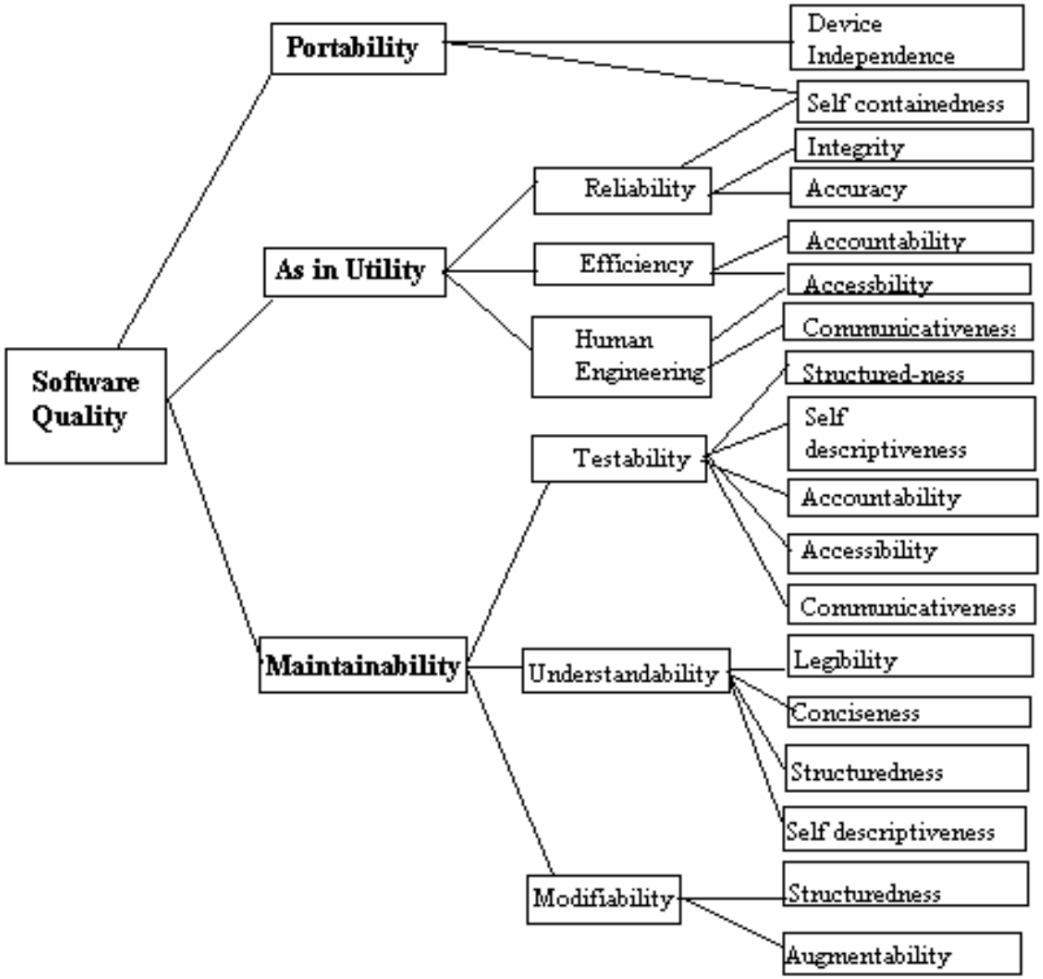
Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Modelos de Qualidade ao Longo do Tempo

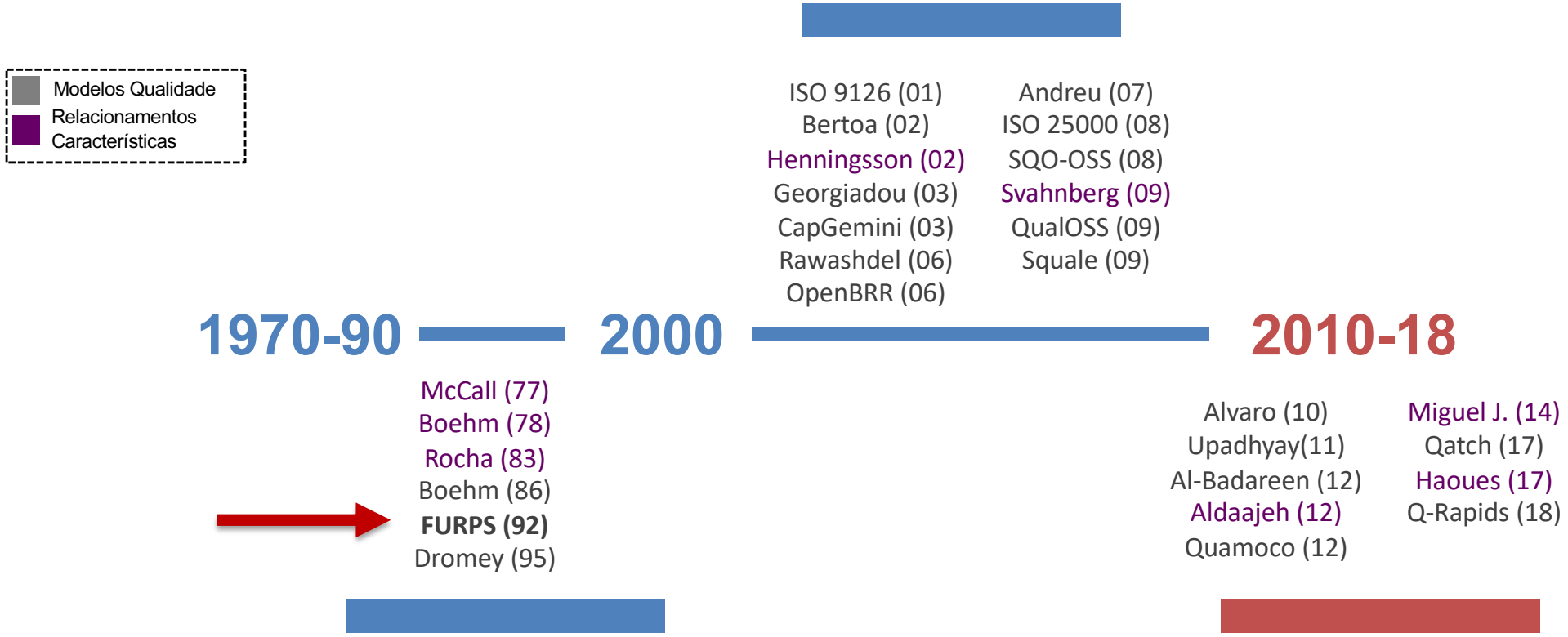
Boehm Model



Miguel, Jose & Mauricio, David & Rodriguez, Glen. (2014). A Review of Software Quality Models for the Evaluation of Software Products. International journal of Software Engineering & Applications. 5. 31-54. 10.5121/ijsea.2014.5603.

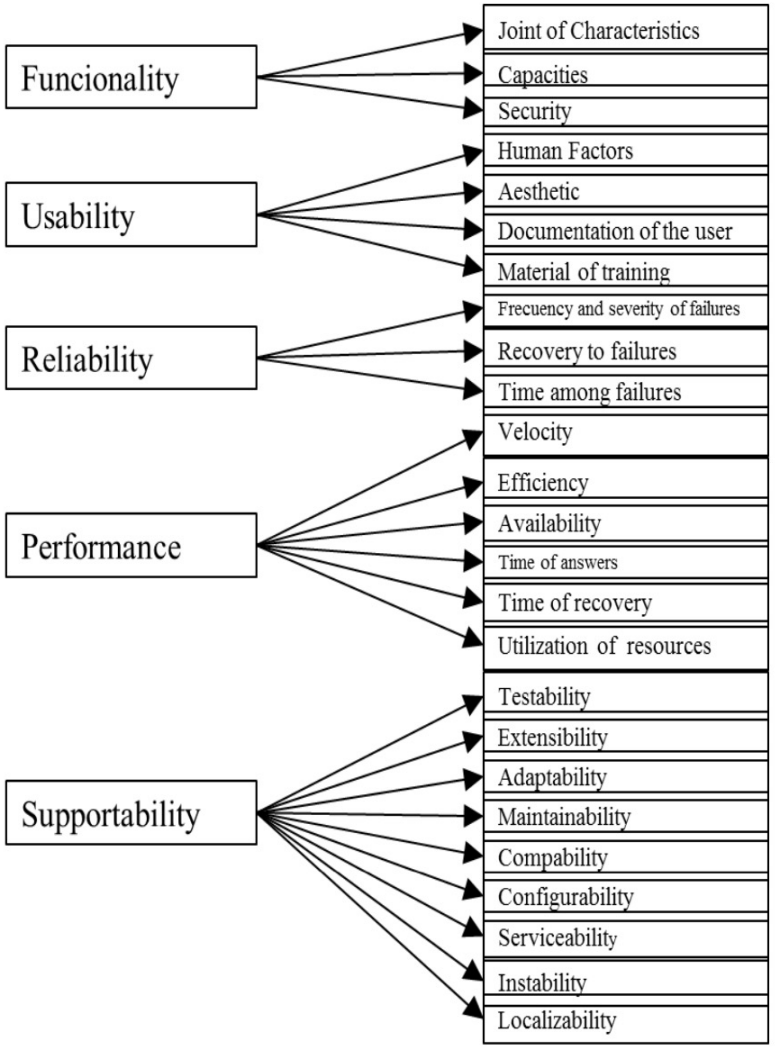
Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Modelos de Qualidade ao Longo do Tempo

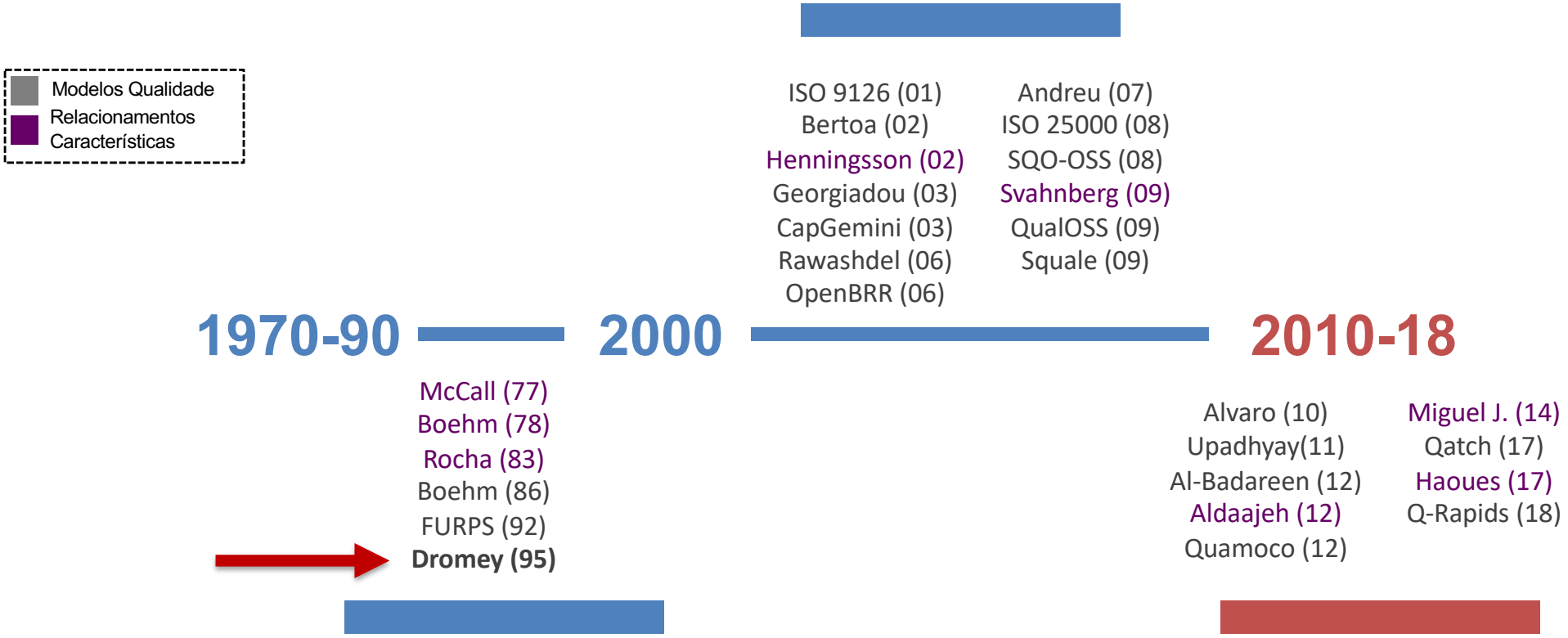
FURPS Model



Miguel, Jose & Mauricio, David & Rodriguez, Glen. (2014). A Review of Software Quality Models for the Evaluation of Software Products. International journal of Software Engineering & Applications. 5. 31-54. 10.5121/ijsea.2014.5603.

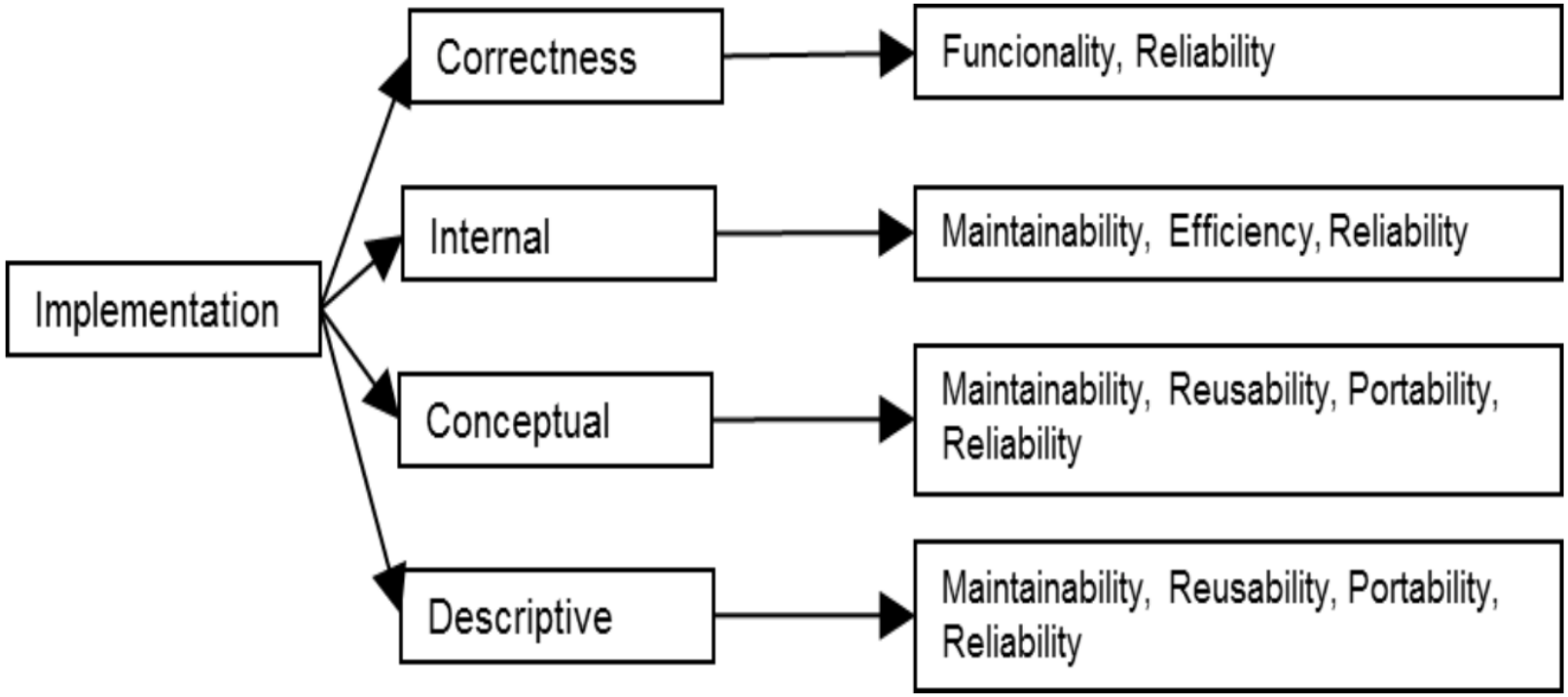
Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Modelos de Qualidade ao Longo do Tempo

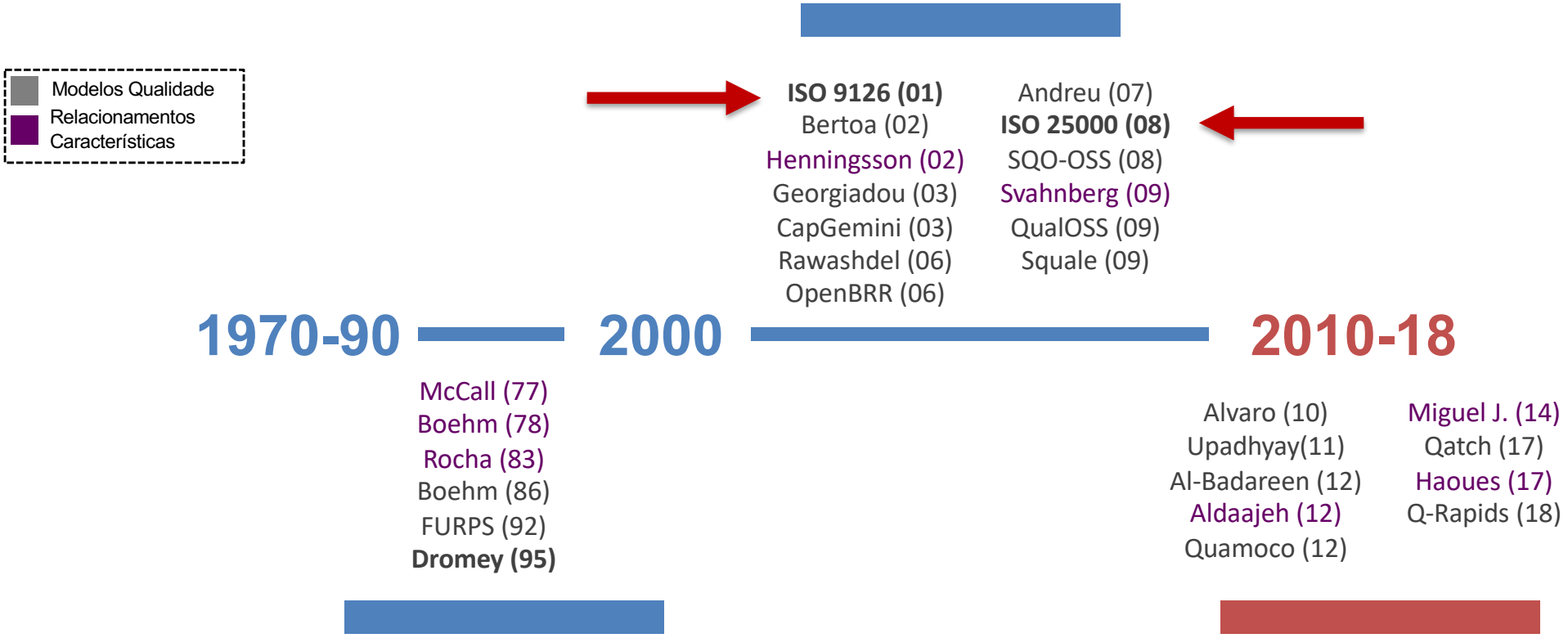
Dromey Model



Miguel, Jose & Mauricio, David & Rodriguez, Glen. (2014). A Review of Software Quality Models for the Evaluation of Software Products. International journal of Software Engineering & Applications. 5. 31-54. 10.5121/ijsea.2014.5603.

Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



ISO 25010

- A ISO 9126 e sua atualização, a ISO 25010, tem sido o modelo de qualidade de produto mais estudado ao longo do tempo (Ouhb et al, 2014)

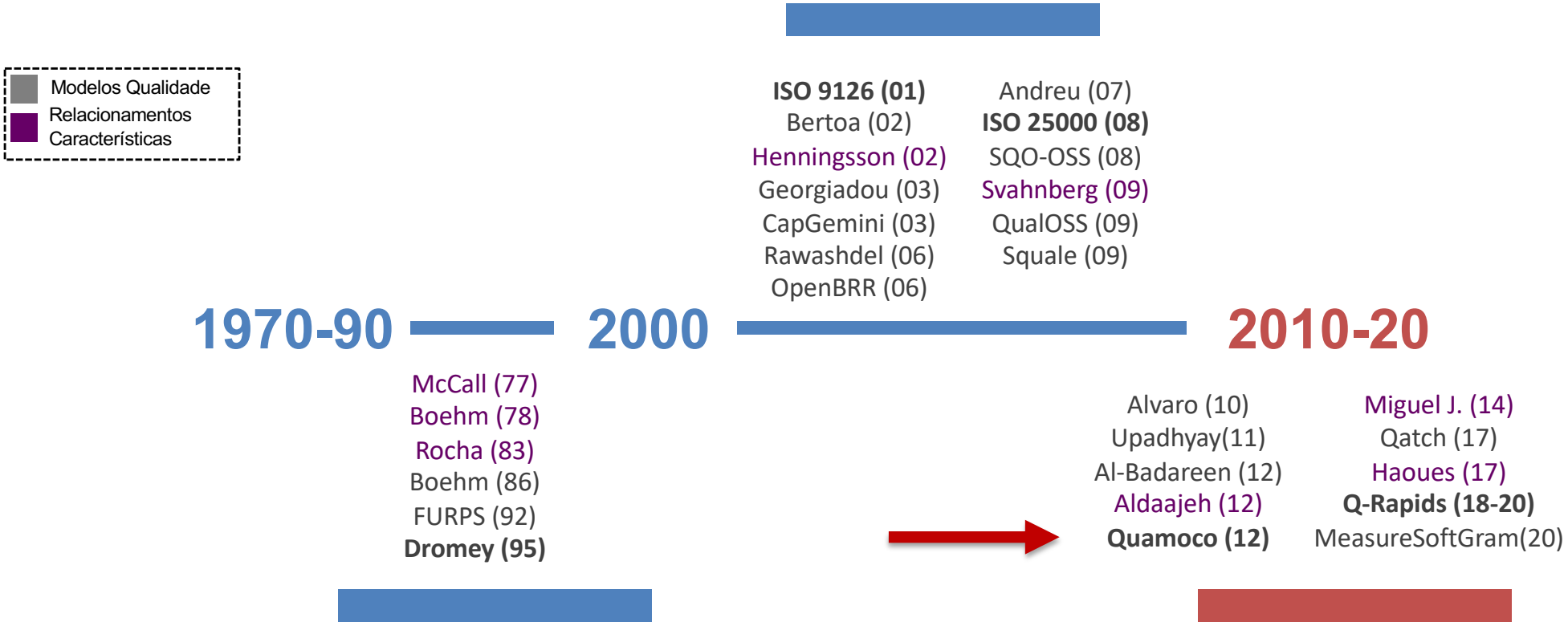
Qualidade De Produto							
Adequação Funcional	**Eficiência de Desempenho**	**Compatibilidade**	**Usabilidade**	**Confiabilidade**	**Segurança**	**Facilidade de Manutenção**	**Portabilidade**
Integridade Funcional	Comportamento Temporal	Coexistência	Adequação no Reconhecimento	Maturidade	Confidencialidade	Modularidade	Adaptabilidade
Corretude Funcional	Utilização de Recursos	Interoperabilidade	Capacidade de Aprendizagem	Disponibilidade	Integridade	Reutilização	Facilidade de Instalação
Pertinência Funcional	Capacidade		Proteção Contra Erros do Uso	Telerância a Falhas	Não Repudiação	Facilidade de Análise	Capacidade Substituição
			Operacionalidade	Capacidade de Recuperação	Responsabilização	Mutabilidade	
			Acessibilidade		Autenticidade	Testabilidade	
			Estética de Interface				

Qualidade em Uso				
Eficiência	**Eficácia**	**Satisfação**	**Liberdade do risco**	**Cobertura do Contexto**
Eficiência	Eficácia	Facilidade de Uso	Mitigação de risco econômico	completude do contexto
		Confiança	Mitigação de riscos à saúde e segurança	Flexibilidade
		Prazer	Mitigação de riscos de ambiente	
		Conforto		

Características e Sub-Características Qualidade de Sistemas (Produto e em Uso)

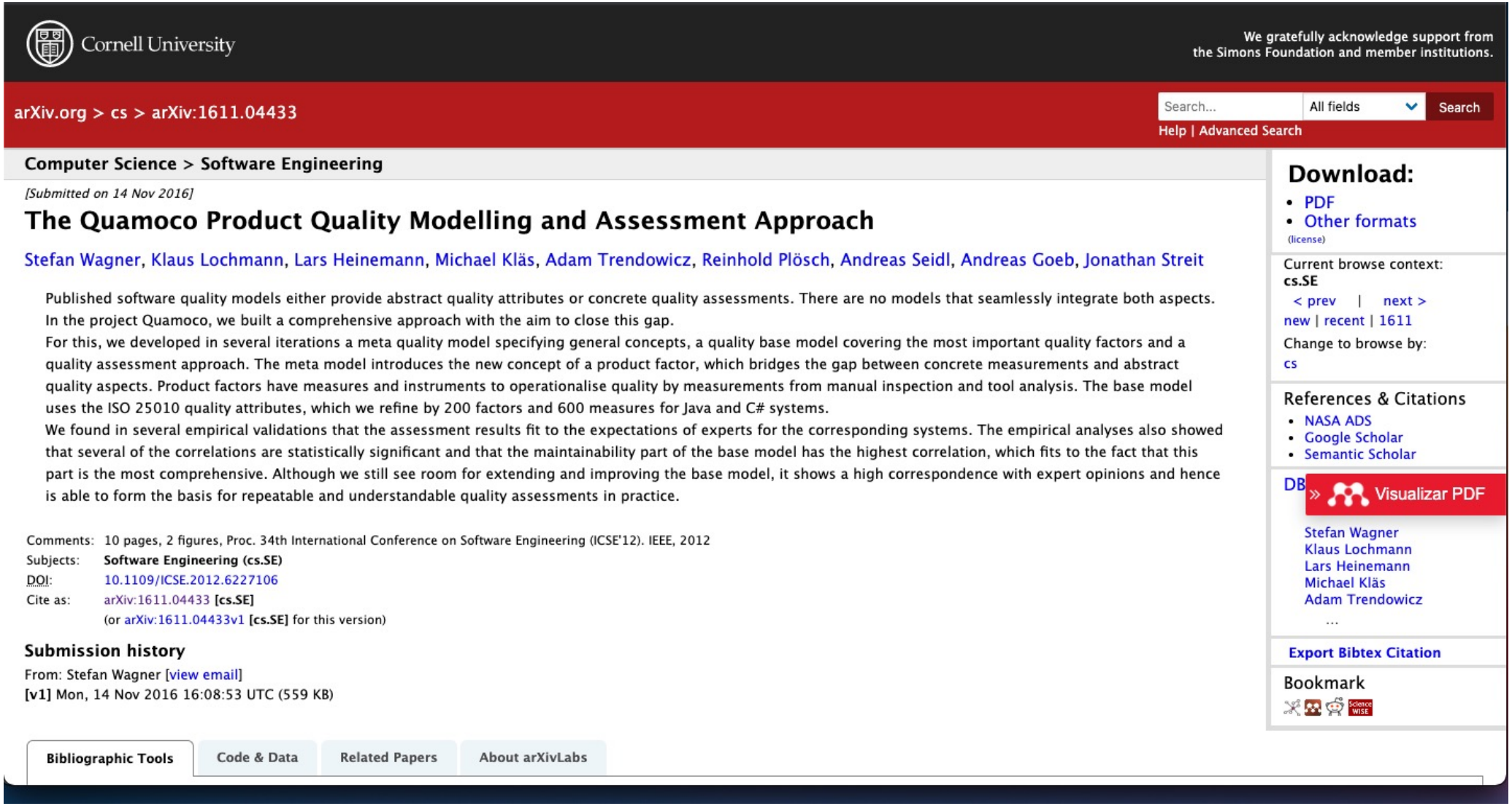
Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Modelos de Qualidade ao Longo do Tempo

Quamoco



Cornell University

We gratefully acknowledge support from the Simons Foundation and member institutions.

arXiv.org > cs > arXiv:1611.04433

Search... All fields Search

Help | Advanced Search

Computer Science > Software Engineering

[Submitted on 14 Nov 2016]

The Quamoco Product Quality Modelling and Assessment Approach

Stefan Wagner, Klaus Lochmann, Lars Heinemann, Michael Kläs, Adam Trendowicz, Reinhold Plösch, Andreas Seidl, Andreas Goeb, Jonathan Streit

Published software quality models either provide abstract quality attributes or concrete quality assessments. There are no models that seamlessly integrate both aspects. In the project Quamoco, we built a comprehensive approach with the aim to close this gap.
For this, we developed in several iterations a meta quality model specifying general concepts, a quality base model covering the most important quality factors and a quality assessment approach. The meta model introduces the new concept of a product factor, which bridges the gap between concrete measurements and abstract quality aspects. Product factors have measures and instruments to operationalise quality by measurements from manual inspection and tool analysis. The base model uses the ISO 25010 quality attributes, which we refine by 200 factors and 600 measures for Java and C# systems.
We found in several empirical validations that the assessment results fit to the expectations of experts for the corresponding systems. The empirical analyses also showed that several of the correlations are statistically significant and that the maintainability part of the base model has the highest correlation, which fits to the fact that this part is the most comprehensive. Although we still see room for extending and improving the base model, it shows a high correspondence with expert opinions and hence is able to form the basis for repeatable and understandable quality assessments in practice.

Comments: 10 pages, 2 figures, Proc. 34th International Conference on Software Engineering (ICSE'12). IEEE, 2012
Subjects: **Software Engineering (cs.SE)**
DOI: 10.1109/ICSE.2012.6227106
Cite as: arXiv:1611.04433 **[cs.SE]**
(or arXiv:1611.04433v1 **[cs.SE]** for this version)

Submission history

From: Stefan Wagner [view email]
[v1] Mon, 14 Nov 2016 16:08:53 UTC (559 KB)

Bibliographic Tools | Code & Data | Related Papers | About arXivLabs

Download:

- PDF
- Other formats

(license)

Current browse context: **cs.SE**

< prev | next >

new | recent | 1611

Change to browse by: cs

References & Citations

- NASA ADS
- Google Scholar
- Semantic Scholar

DB

Visualizar PDF

Stefan Wagner
Klaus Lochmann
Lars Heinemann
Michael Kläs
Adam Trendowicz
...

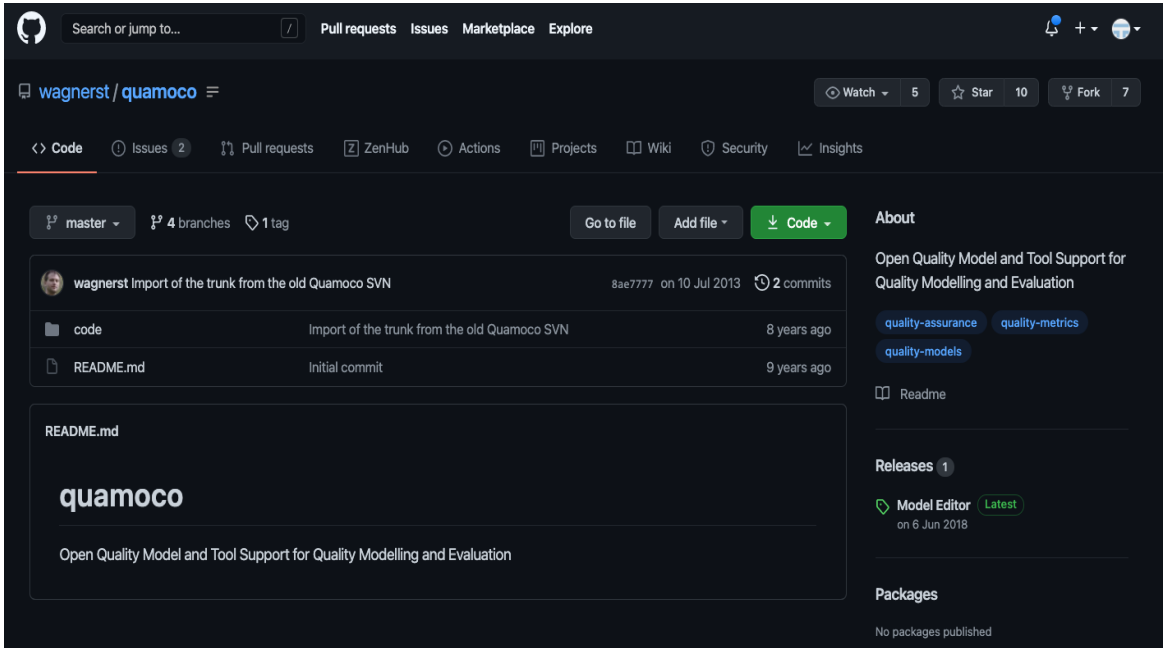
Export Bibtex Citation

Bookmark

WAGNER, S. et al. The Quamoco Product Quality Modelling and Assessment Approach. Proceedings of the 34th International Conference on Software Engineering. Anais...: ICSE '12.Piscataway, NJ, USA: IEEE Press, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2337223.2337372>

Quamoco

- O metamodelo introduz o novo conceito de fator de produto, que preenche a lacuna entre medidas concretas e aspectos abstratos de qualidade.
- Fatores de produto têm medidas e instrumentos para operacionalizar a qualidade por meio de medições de inspeção manual e análise de ferramentas.
- O modelo básico usa os atributos de qualidade ISO 25010, que foram refinamos em:
  - 200 fatores
  - 600 medidas
- Sistemas Java e C #.



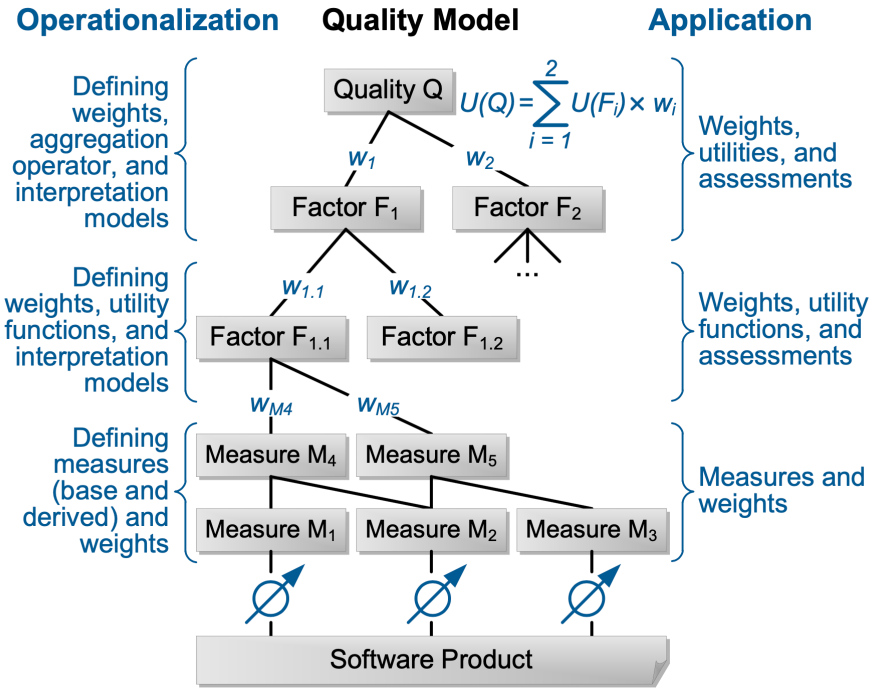
WAGNER, S. et al. The Quamoco Product Quality Modelling and Assessment Approach. Proceedings of the 34th International Conference on Software Engineering. Anais...: ICSE '12.Piscataway, NJ, USA: IEEE Press, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2337223.2337372>

Quamoco

- Especialistas em qualidade de software da academia e da indústria na Alemanha uniram forças no projeto de pesquisa Quamoco.

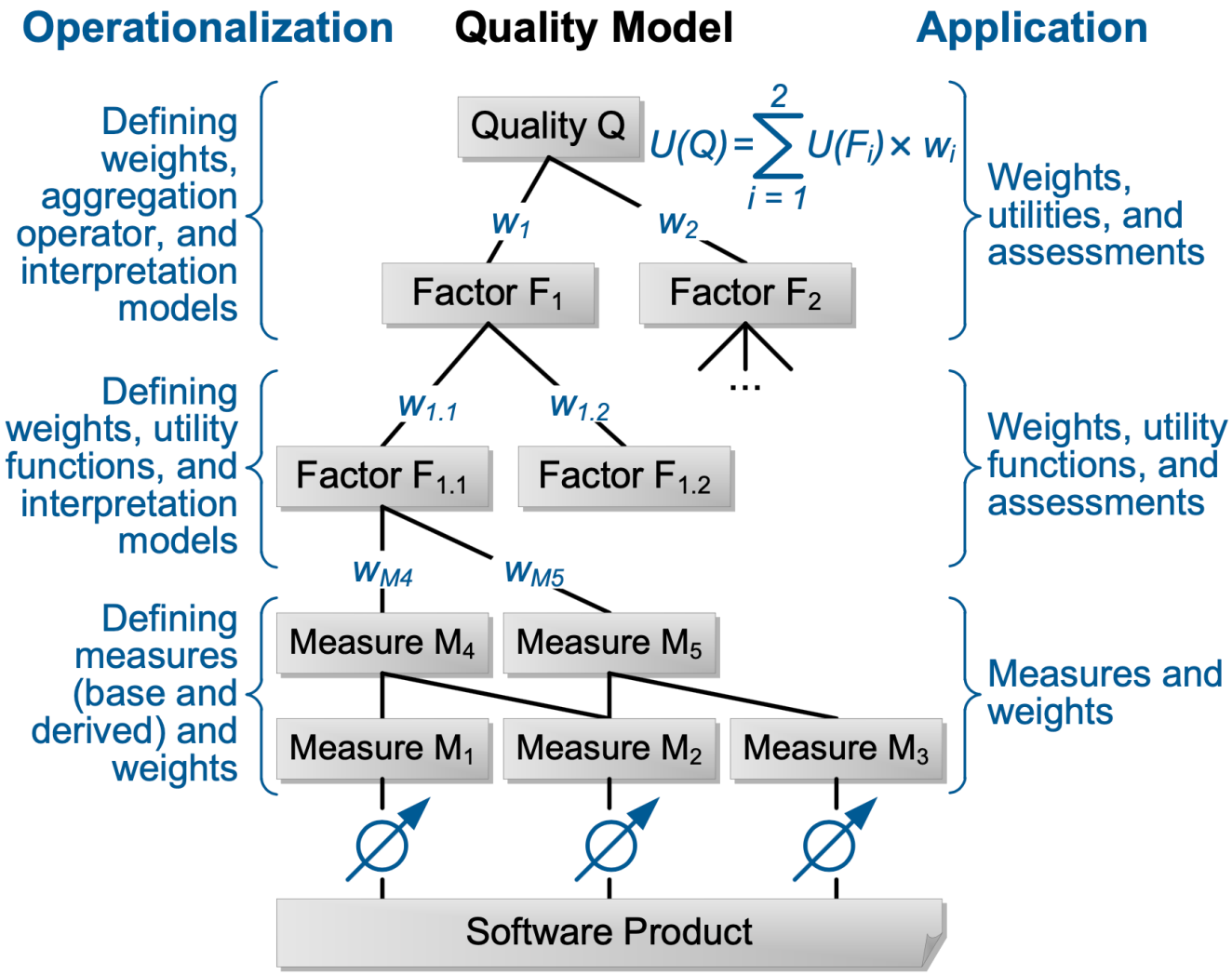
- O consórcio do projeto foi composto por:
  - Technische Universitat Munchen
  - SAP
  - Siemens
  - Capgemini
  - Fraunhofer IESE
  - Itestra.

- No total, foram envolvidas 558 pessoas-mês no projeto.



WAGNER, S. et al. The Quamoco Product Quality Modelling and Assessment Approach. Proceedings of the 34th International Conference on Software Engineering. Anais...: ICSE '12.Piscataway, NJ, USA: IEEE Press, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2337223.2337372>

Quamoco - Meta Modelo



WAGNER, S. et al. The Quamoco Product Quality Modelling and Assessment Approach. Proceedings of the 34th International Conference on Software Engineering. Anais...: ICSE '12.Piscataway, NJ, USA: IEEE Press, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2337223.2337372>

Quamoco

- **Robusta validação experimental em relação a sua eficácia e eficiência!**

- Os pesquisadores validaram as seguintes questões de pesquisa:

RQ 1: Can the base model be used to detect quality differences between different systems or subsystems?
RQ 2: Can the base model be used to detect quality improvements over time in a software system?

$$H_{1_A} : r(\text{rankingBM}, \text{rankingLSV}) > 0$$
$$H_{1_0} : r(\text{rankingBM}, \text{rankingLSV}) \leq 0$$
$$H_{2_A} : r(\text{rankingBM}, \text{rankingExp}) > 0$$
$$H_{2_0} : r(\text{rankingBM}, \text{rankingExp}) \leq 0$$

WAGNER, S. et al. The Quamoco Product Quality Modelling and Assessment Approach. Proceedings of the 34th International Conference on Software Engineering. Anais...: ICSE '12.Piscataway, NJ, USA: IEEE Press, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2337223.2337372>

Quamoco

- **Robusta validação experimental em relação a sua eficácia e eficiência!**

COMPARISON OF THE ASSESSMENT RESULTS OF FIVE SUBSYSTEMS AND AN EXPERT'S OPINION

Subsystem	Rank Exp Quality	Rank Exp Maint.	Rank BM Quality	Rank BM Maint.
Subsystem A	2	2	5	2
Subsystem B	2	3	3	5
Subsystem C	4	4	2	3
Subsystem D	1	1	1	1
Subsystem E	4	3	4	4

Version	Quality grade
1.9.0	4.15
2.0.0	3.34
2.0.1	3.63
2.0.2	3.42
2.1.0	3.27
2.2.1	3.17

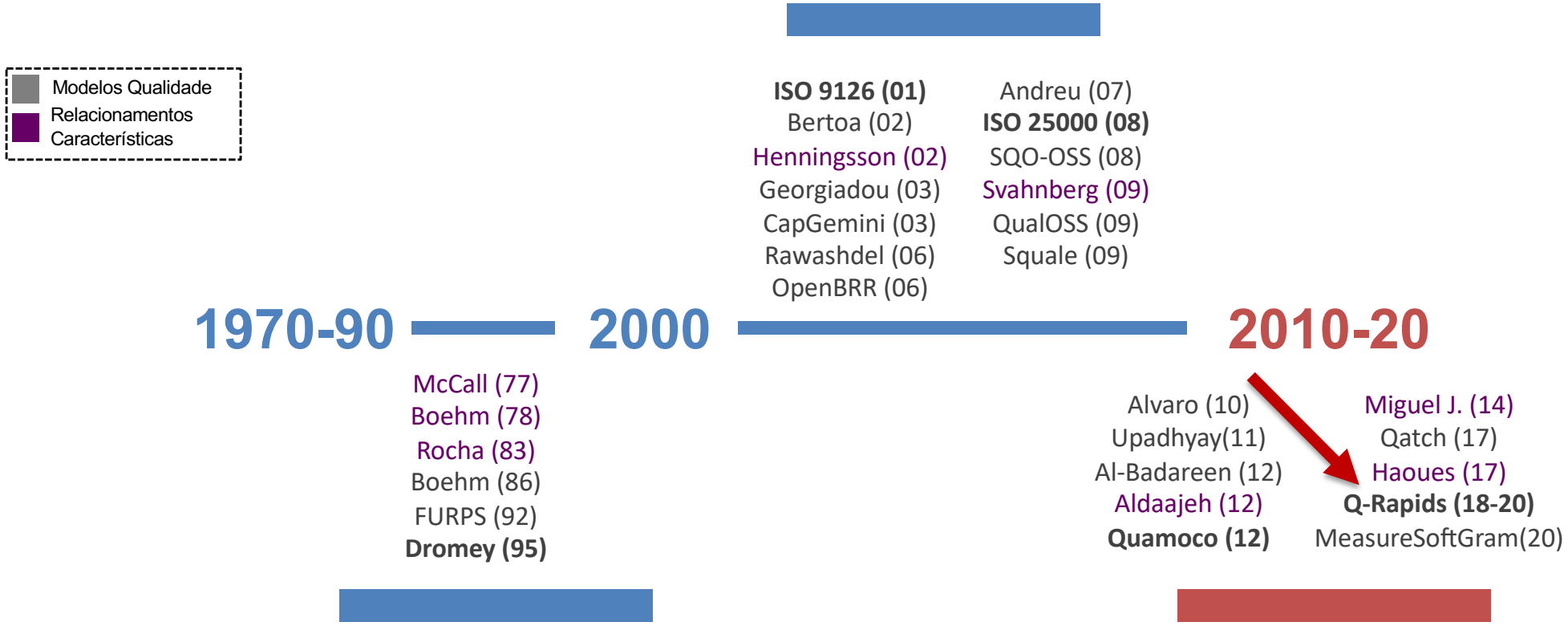
COMPARISON OF THE ASSESSMENT RESULTS AND THE RESULTS OF "LINZER SOFTWARE-VERKOSTUNG"

Product	LOC	Result BM	Rank BM	Rank LSV
Checkstyle	57,213	1 (1.87)	1	1
RSSOwl	82,258	3 (3.14)	2	3
Log4j	30,676	3 (3.36)	2	2
TV-Browser	125,874	4 (4.02)	4	4
JabRef	96,749	5 (5.47)	5	5

WAGNER, S. et al. The Quamoco Product Quality Modelling and Assessment Approach. Proceedings of the 34th International Conference on Software Engineering. Anais...: ICSE '12.Piscataway, NJ, USA: IEEE Press, 2012Disponível em: <http://dl.acm.org/citation.cfm?id=2337223.2337372>

Modelos de Qualidade ao Longo do Tempo

e os estudos sobre os relacionamentos entre caracteríticas e subcaracteríticas



Modelos de Qualidade ao Longo do Tempo

Q-Rapids

LÓPEZ, L. et al. Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development. CAiSE Forum. Anais...: Lecture Notes in Business Information Processing.Springer, 2018

Modelos de Qualidade ao Longo do Tempo

Q-Rapids

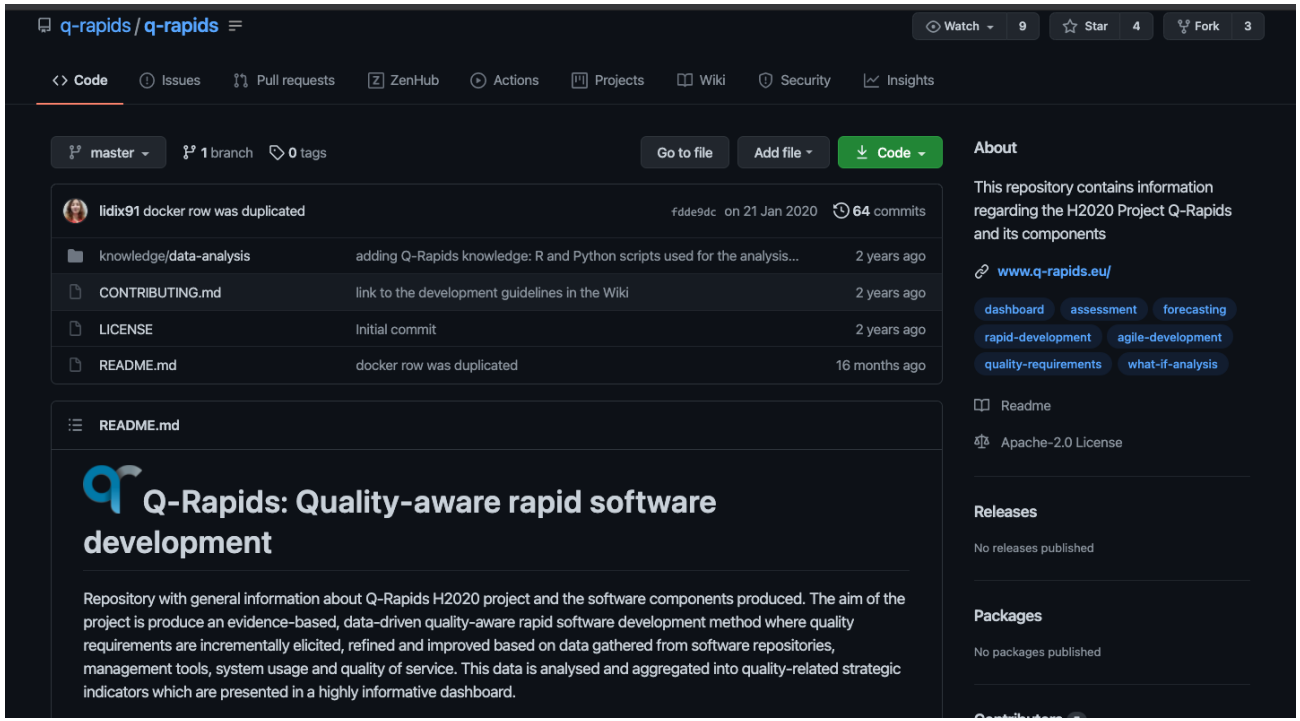
LÓPEZ, L. et al. Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development. CAiSE Forum. Anais...: Lecture Notes in Business Information Processing.Springer, 2018

Modelos de Qualidade ao Longo do Tempo

Q-Rapids

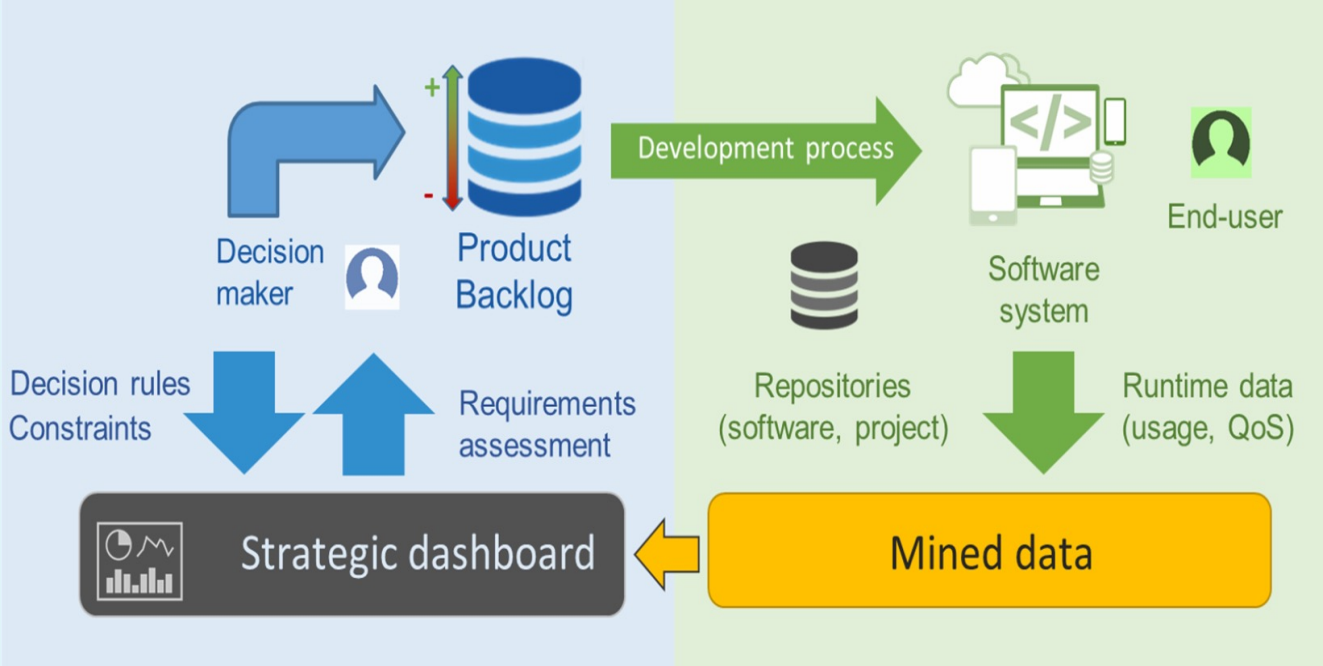
- Baseado no metamodelo Quamoco
  Participação da indústria e academia em um projeto de P&D envolvendo diferentes países.
- **Validação experimental! Evidência sobre sua a eficácia e eficiência.**

- Versão de produto disponibilizada em software de código-fonte aberto
- Projetado para planejar e monitorar a qualidade em ambientes de desenvolvimento contínuo



LÓPEZ, L. et al. Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development. CAiSE Forum. Anais...: Lecture Notes in Business Information Processing.Springer, 2018

Q-Rapids



Consortium

Q-Rapids consortium consists of seven organisations from five countries.

- Universidad Politecnica de Catalunya (UPC)
- University of Oulu (UoO)
- Fraunhofer IESE
- Bittium
- Softeam
- iTTi
- Nokia

www.q-rapids.eu/

dashboard · assessment · forecasting · rapid-development · agile-development · quality-requirements · what-if-analysis

Readme

Apache-2.0 License

Releases

No releases published

Packages

No packages published

Contributors 5

Languages

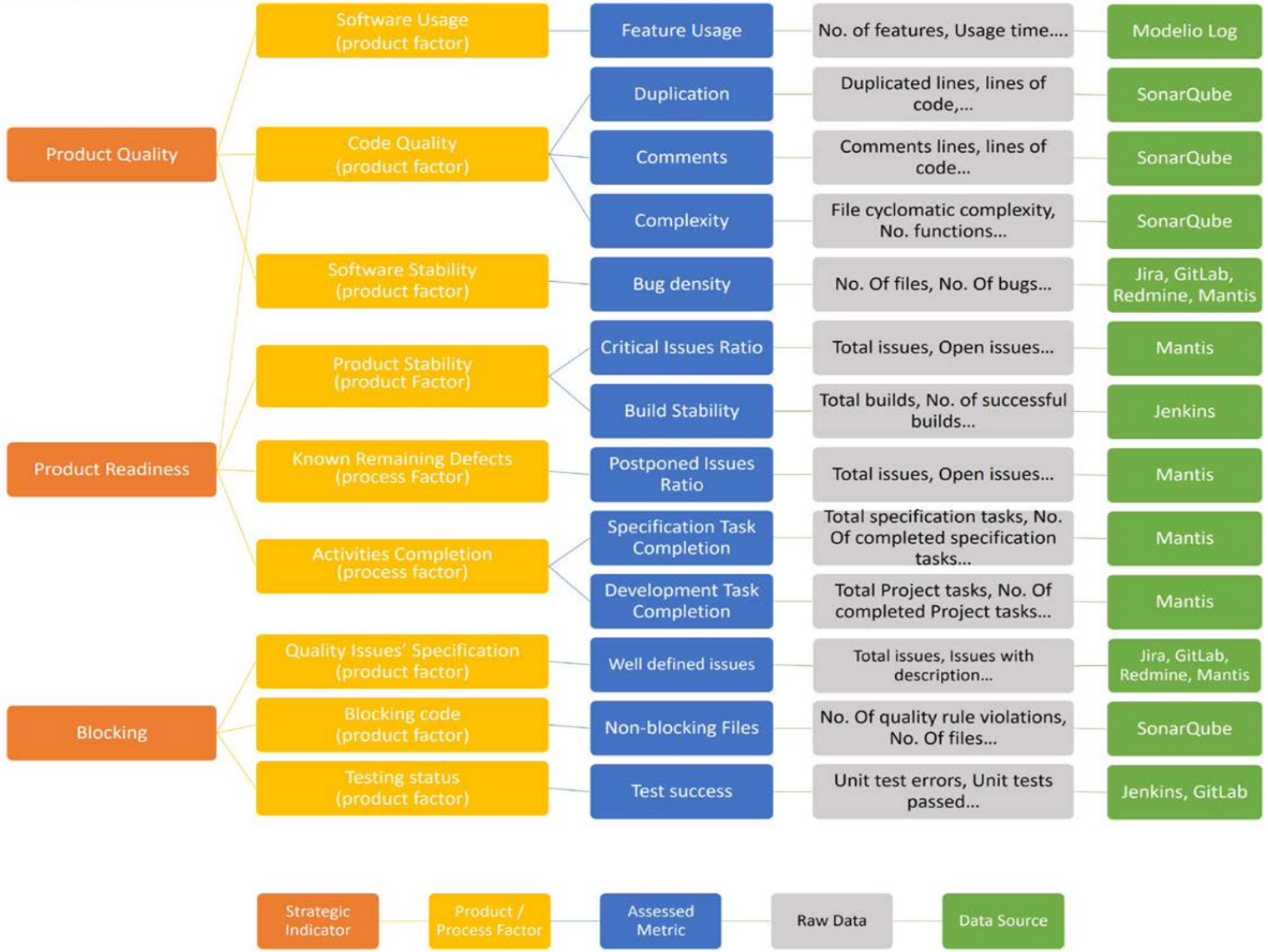
Python 77.0% R 23.0%

LÓPEZ, L. et al. Q-Rapids Tool Prototype: Supporting Decision-Makers in Managing Quality in Rapid Software Development. CAiSE Forum. Anais...: Lecture Notes in Business Information Processing.Springer, 2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732253.

Q-Rapids

Graphical representation of the quality model



Modelos de Qualidade ao Longo do Tempo

Q-Rapids

TABLE I. Q-RAPIDS QUALITY MODEL

QA[a]	Factor [c]	Assessed Metric [c]	Description	Raw Data	Data Source
Maintainability	Code Quality	Non-complex files [b]	Files below the threshold of cyclomatic complexity (10 by default)	*Cyclomatic complexity* per *function* of each file, *total number of files*	SonarQube
		Commented files [b]	Files whose comment density is outside the defined thresholds (by default 10%-30%)	*Density of comment lines* and *lines of code* per each file	SonarQube
		Absence of duplications [b]	Files below the threshold of duplicated lines percentage	*Duplicated lines* and *lines of code* per file	SonarQube
	Blocking Code	Fulfillment of critical/blocker quality rules [b]	Files without critical or blocker quality rule violations	Number of *quality rule violations* per file and their *severity* (blocker, critical, major, minor, info) and *type* (code smell, bug, vulnerability)	SonarQube, Coverity, CodeSonar
		Highly changed files	Unstable files that have been highly changed in the last commits	For each *commit: files changed*, *lines of code added/modified/deleted*, *author*, and *revision*	SVN, git, Gerrit
Reliability	Testing Status	Passed tests [b]	Unit test success density	Number of *unit test errors*, *failures*, *skipped*, and *total*	Jenkins, GitLab
		Fast test builds [b]	Test builds below the duration threshold	*Duration of unit test execution*, *tests* conforming to a *pipeline*	Jenkins, GitLab
		Test coverage	Tests with appropriate coverage	*Condition coverage* and *line coverage* per *unit test*	Jenkins (JaCoCo plugin), SonarQube
	Software Stability	Non-bug density [b]	Ratio of open issues of the type bug with respect to the total number of issues within a customized timeframe	Total number of *issues* (a.k.a. tasks) per *status* (e.g., open, done), *type* (e.g., bug, maintenance, feature), and *timeframe* (e.g., current/last month or current/last sprint)	Jira, GitLab, Redmine, Mantis
		Errors at runtime	Occurrence of critical errors at runtime at the end user site	All tracks of logs, including *type of error* (fatal, error, warning, info, debug, trace), *file* and *line* where it occurred, and *error message*	Logs
		Availability uptime	Percentage of time that the product is accessible	Timestamp at which the system is not available and derived metrics, e.g., *availability uptime*, *mean time between failures*	Zabbix, Nagios
Functional Suitability	Software Usage	Feature usage	Appropriateness of the features included in the software product regarding their usage	For each *functionality (or feature)*: number of *times used*, *average usage time*, *customer feedback on the feature* (if available)	Logs, monitoring plugin
Productivity	Issues' Velocity	Resolved issues assigned to a date	Resolved issues assigned to a date (simple date, iteration, or release)	For each *issue* (a.k.a. task): *time created*, *status*, *time updated*, *iteration(s)*, *release(s)*.	Jira, GitLab, Redmine, Mantis
		Issues completely specified [b]	Density of incomplete issues within a timeframe	*Fields of each issue* (e.g., description, due date, assignee, estimated time)	Jira, GitLab, Redmine, Mantis

ISO 25010

- As características e subcaracterísticas se influenciam mutuamente, produzindo diferentes efeitos.

ISO 25010	Functional Suitability	Performance Efficiency	Usability	Compatibility	Reliability	Security	Maintainability	Portability
Functional suitability	❖	o	+	o	+	–	+	o
Performance efficiency	o	❖	–	–	o	–	–	–
Usability	+	–	❖	o	+	–	–	o
Compatibility	o	–	o	❖	o	–	+	+
Reliability	+	o	+	o	❖	+	+	o
Security	–	–	–	–	+	❖	±	o
Maintainability	+	–	–	+	+	±	❖	+
Portability	o	–	o	+	o	o	+	❖

Haoues, Mariem & Sellami, Asma & Ben-Abdallah, Hanêne & Cheikhi, Laila. (2016). A guideline for software architecture selection based on ISO 25010 quality related characteristics. International Journal of System Assurance Engineering and Management. 8. 10.1007/s13198-016-0546-8.

Medição de Software

Medidas e Métricas

"O elemento básico da medição são as **medidas**. Medidas caracterizam, em termos quantitativos ou qualitativos, uma **propriedade** de um objeto e fornecem informações capazes de **apoiar tomadas de decisão** técnicas e de negócios."

Rocha, A. & Santos, Gleison & Barcellos, Monalessa. (2012). Medição de Software e Controle Estatístico de Processos.

Medição de Software

Métricas

Não há uma definição absoluta.

Uma **métrica** relaciona um conceito a uma medida (um símbolo que quantifica este conceito para o determinado objeto de estudo) e representa um **valor numérico** dos **dados não tratados**, extraídos de uma fonte de informação.

Medição de Software

Métricas

Por outro lado, uma **medida** é um **número ou símbolo** atribuído a uma entidade para caracterizar um **atributo**.

Norman Fenton and James Bieman. 2014. Software Metrics-A Rigorous and Practi- cal Approach, Third Edition (third ed.).

Portanto, a medida fornece significado e interpretação para uma métrica.

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » nominal;
  - » ordinal;
  - » intervalar;
  - » razão.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. 2000. Experimentation in software engineering: an introduction. Kluwer Academic Publishers, USA.

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Nominal:
    - ❖ Emprega expressões semânticas (nomes) para representar objetos para fins de identificação.
    - ❖ A ordem não possui qualquer significado na interpretação dos valores nesta escala

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2000. Experimentation in software engineering: an introduction. Kluwer Academic Publishers, USA.

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Nominal:
    - ❖ Exemplos em Engenharia de Software:
      - Linguagens de programação
    - ❖ A escala não nos permite dizer, por exemplo, que linhas de código escritas em Python seja menor que em C++

Notas de aula do prof. Guilherme Horta Travassos, PESC-COPPE/UFRJ, 2016 – cps820_aula6

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Ordinal:
    - ❖ Seus valores podem ser comparados em ordem, onde e´ possível agrupar valores em categorias que também podem ser ordenadas.
    - ❖ Porém, esta escala não oferece informações sobre a magnitude da diferença entre os elementos, portanto não há interpretação numérica.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. 2000. Experimentation in software engineering: an introduction. Kluwer Academic Publishers, USA.

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Ordinal:
    - ❖ Exemplos em Engenharia de Software:
      - Diferentes níveis de classificação SonarQube(A,...,E), CodeClimate(A,...,F), CMMI(1,...,5) ...
    - ❖ A escala permite dizer, por exemplo, que o CodeClimate nível A é maior que o nível B, mas não permite dizer que a diferença de qualidade entre projetos CodeClimate nível A e B é igual a projetos com CodeClimate nível B e C.

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Intervalar:
    - ❖ É utilizada quando a diferença entre medidas é representativa.
    - ❖ A ordem dos resultados é importante, assim como o tamanho dos intervalos que separam os pontos mas. Já a razão entre valores não é necessariamente válida, pois não tem significado.
    - ❖ Em geral, o valor zero é atribuído por conveniência e não representa ausência da propriedade representada.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2000. Experimentation in software engineering: an introduction. Kluwer Academic Publishers, USA.

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Intervalar:
    - ❖ A comparação de valores é possível porque toda escala de intervalo apresenta um ponto zero arbitrário (no caso de datas, o ano 0)
    - ❖ Exemplo: embora possamos dizer que 2020 represente um ano depois de 2019 e um ano antes de 2021 não há sentido em calcular a razão entre 2020 e 2019.

Notas de aula do prof. Guilherme Horta Travassos, PESC-COPPE/UFRJ, 2016 – cps820_aula6

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Intervalar: Likert (5pts)
    - ❖ Usando uma escala Likert podemos definir diferentes nomes para representar, em geral, a intensidade de uma propriedade que não pode ser medida diretamente.
    - ❖ Exemplo em Engenharia de Software:
      - podemos construir uma escala Likert para avaliar o grau de satisfação de um usuário usando os seguintes valores: totalmente insatisfeito (0), parcialmente insatisfeito (1), indiferente (2), parcialmente satisfeito (3), totalmente satisfeito (4).

Medição de Software

Escalas das Medidas

- Principais tipos de escalas:
  - » Razão:
    - ❖ Possui definição semelhante a escala de intervalar. A diferença está no fato da razão entre os valores é preservada.
    - ❖ Os valores podem ser ordenados, a distância entre valores consecutivos tem o mesmo significado e a razão entre os valores pode ser interpretada.
    - ❖ Uma escala razão possui valor zero absoluto, ou seja, uma medida com valor zero indica ausência da propriedade representada

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2000. Experimentation in software engineering: an introduction. Kluwer Academic Publishers, USA.

Medição de Software

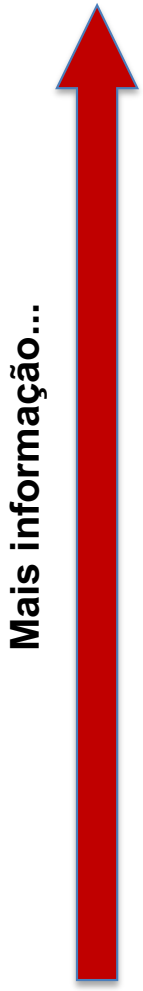
Escalas das Medidas

- Principais tipos de escalas:
  - » Razão:
    - ❖ Exemplos em engenharia de software incluem tamanho, esforço e tempo de execução do projeto.
      - A escala razão nos permite dizer, por exemplo, que um produto de software com X linhas de código é duas vezes menor que um software com 2X linhas de código (considerando a mesma linguagem de programação e procedimento de medição)

Notas de aula do prof. Guilherme Horta Travassos, PESC-COPPE/UFRJ, 2016 – cps820_aula6

Medição de Software

Escalas das Medidas



Razão

Os valores podem ser contados, ordenados, a distância entre os valores pode ser interpretada, e é possível realizar a razão entre valores.

Intervalo

Os valores podem ser contados, ordenados e a distância entre os valores pode ser interpretada

Ordinal

Valores podem ser contados e ordenados

Nominal

Valores podem ser contados

Medição de Software

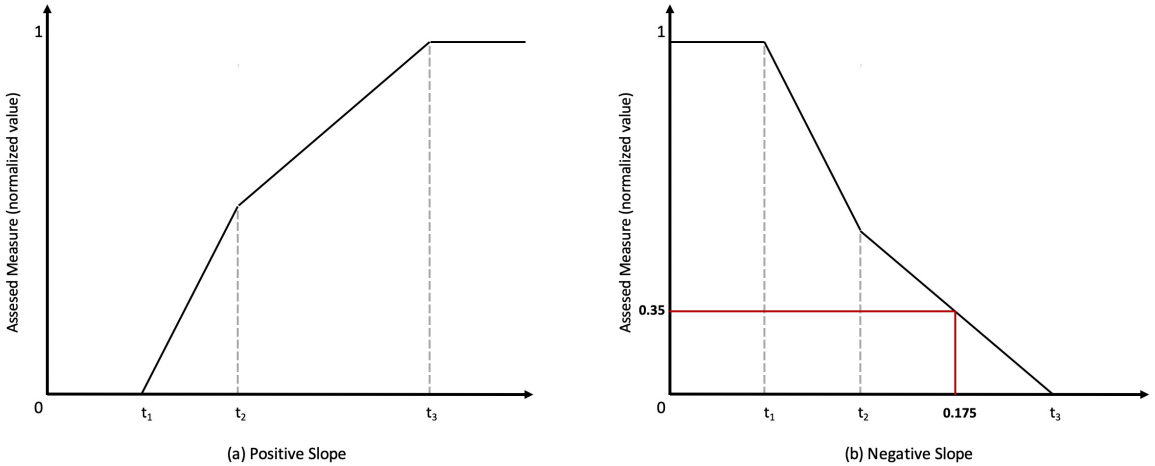
Escalas das Medidas

- De acordo com os tipos de escala, podemos realizar diferentes operações com seus valores.

Escala/Operações	Nominal	Ordinal	Intervalar	Razão
Contar Valores	X	X	X	X
Ordenar Valores		X	X	X
Intervalos equidistantes			X	X
Adição e Subtração de valores			X	X
Divisão entre valores				X

Medição de Software

Valores de Referência



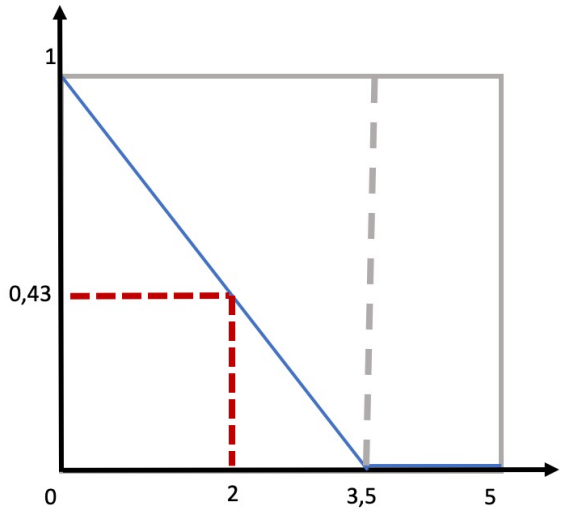
$$t_1 = min\ (\{x: x\ \geq\ Q_{25\%}(x_1, \ldots, x_n) - 1.5\ * IRQ(x_1, \ldots, x_n)\})$$
$$t_2 = median(x_1, \ldots, x_n)$$
$$t_3 = max\ (\{x: x\ \geq\ Q_{75\%}(x_1, \ldots, x_n) + 1.5\ * IRQ(x_1, \ldots, x_n)\})$$

whereby:

Q_p= p-percentile

$IRQ(x_1, \ldots, x_n)$ = inter-quartile-range $Q_{75\%}(x_1, \ldots, x_n) - Q_{25\%}(x_1, \ldots, x_n)$

Derivação automática



Definido por especialistas em Qualidade

ANL metric observing the percentiles (5%, 10%, 25%, 50%, 75%, 90%, 95%, 99% and 100%) - Ref. [21]

Projects	X5.	X10.	X25.	X50.	X75.	X90.	X95.	X99.	X100.	mean	std
Agilefant	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	2.00	0.02	0.16
ANTLR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Apache_Derby	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Apache_Isis	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.03
Apache_Tapestry	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.02
Apache_Tomcat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.05
ArgoUML	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Checkstyle	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Dependometer	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ElasticSearch	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hibernate_commons	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hibernate_core	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	4.00	0.03	0.23
JChemPaint	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Jenkins	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
JGit	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
JMock	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Junit	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.01	0.08
Lombok	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.01	0.08
Megamek	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Metric_Miner	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OpenCMS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.03
Oval	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.01	0.09
Spring	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.05
VoltDB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
VRaptor	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Derivação automática

Referências: [19, 20, 21, 23, 24, 25, 26, 27, 36]

Medição de Software

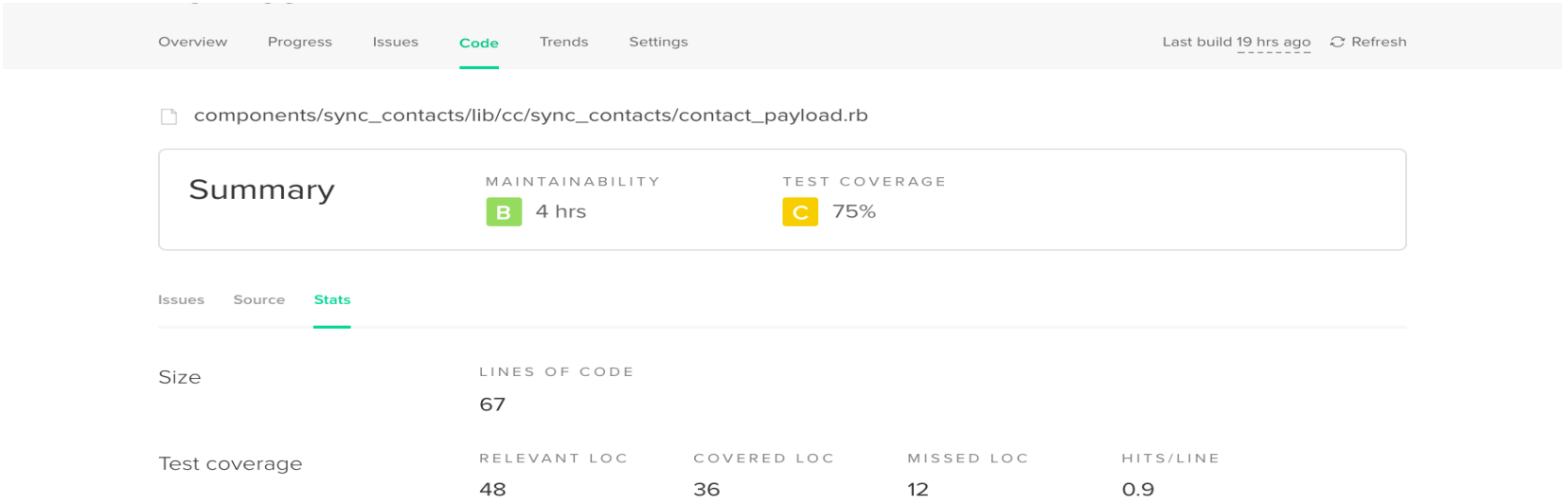
Métricas

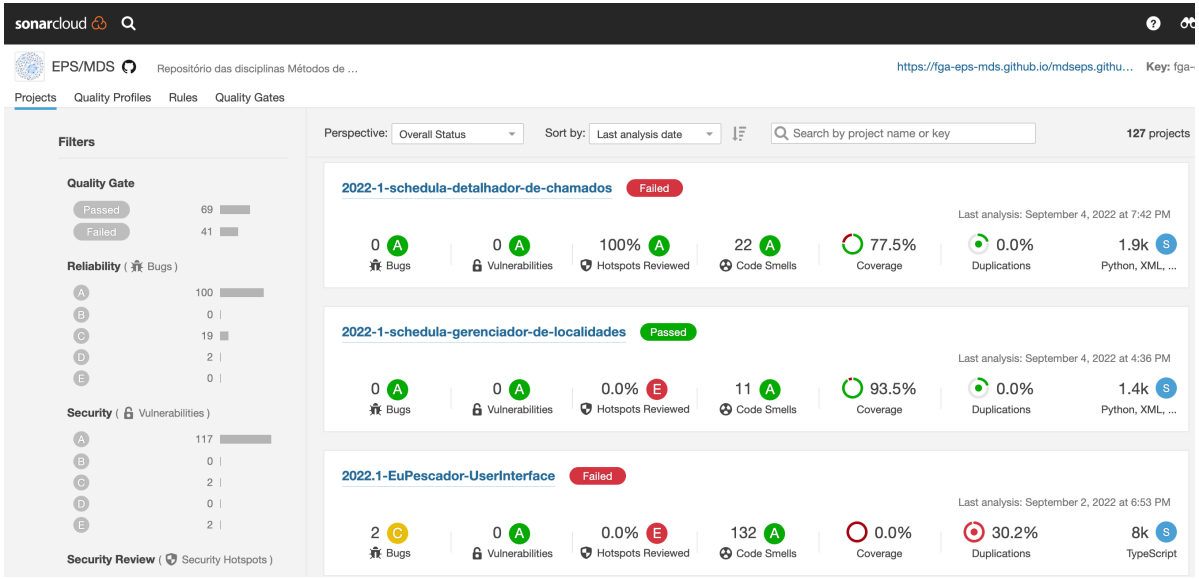
Ok, mas ...

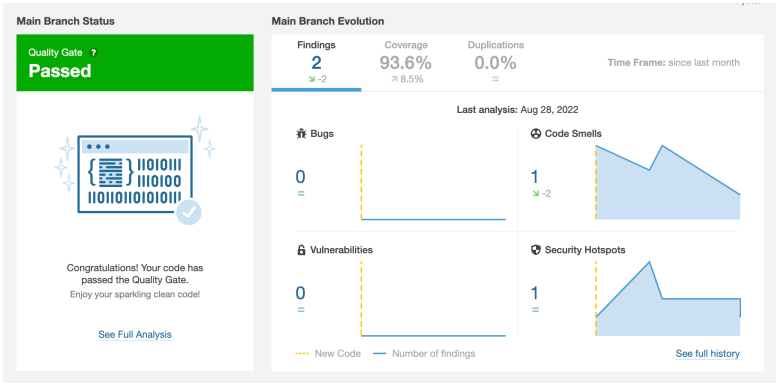
Quais são as métricas e medidas que devem ser usadas para monitorar a qualidade de um produto de software ao longo de seu ciclo de vida?

Medição de Software

A escolha das medidas e métricas







Medição de Software

Métricas

E possivelmente a melhor resposta é...

Depende!

Medição de Software

Métricas

- Diferentes questões podem influenciar a resposta:
  - » os objetivos estratégicos de uma organização;
  - » objetivos de medição;
  - » linguagem de programação e paradigma
  - » ou as tecnologias de desenvolvimento utilizadas.
- Portanto, durante a atividade de planejamento, ao se definir os objetivos de medição, as informações de contexto devem ser tratadas e observadas de acordo com a necessidade de interesse de cada interessado na informação.

Medição de Software

Métricas de Acoplamento (propriedade)

Measure	Type of coupling	Strength of coupling	Import or export coupling	Indirect coupling	Inheritance	C2	C4	C5
CBO	method invocation, attribute reference	#coupled classes	both (indifferent)	no	both			x
CBO´					non-inh.-based			x
RFC_{α}	method invocation	#methods invoked	import	depends	both	x		x
RFC				no		x		x
RFC´				yes		x		x
MPC	method invocation	#method invocations	import	no	both			
DAC	type of attribute	#attributes	import	no	both	x		
DAC´		# distinct types	import	no		x		x
COF	method invocation, attribute reference	#coupled classes	both (distinguished)	no	non-inh.-based		x	x
ICP	method invocation	#method invocations, #parameters passed	import	no	both			
IH-ICP					inh.-based			
NIH-ICP					non-inh.-based			
IFCAIC	type of attribute	#attributes	import	no	non-inh.-based			
ACAIC			import	no	inh.-based			
OCAIC			import	no	non-inh.-based			
FCAEC			export	no	non-inh.-based			
DCAEC			export	no	inh.-based			
OCAEC			export	no	non-inh.-based			
IFCMIC	type of parameter	#of parameters	import	no	non-inh.-based			
ACMIC			import	no	inh.-based			
OCMIC			import	no	non-inh.-based			
FCMEC			export	no	non-inh.-based			
DCMEC			export	no	inh.-based			
OCMEC			export	no	non-inh.-based			
OMMIC	method invocation, passing of pointer to method	#method invocations, #pointers passed to a method	import	no	non-inh.-based			
IFMMIC			import	no	non-inh.-based			
AMMEC			import	no	inh.-based			
OMMEC			export	no	non-inh.-based			
FMMEC			export	no	non-inh.-based			
DMMEC			export	no	inh.-based			

J. Daly, L. Briand and J. WÃ¼st, "A Unified Framework for Coupling Measurement in Object-Oriented Systems" in *IEEE Transactions on Software Engineering*, vol. 25, no. 01, pp. 91-121, 1999. doi: 10.1109/32.748920

Qualidade de Software

Das métricas às caracteríscas

- Os conceitos das características de qualidade definidos nas normas e modelos de referência são vistos como abstratos (vagos).
- As características nos permitem observar diferentes propriedades da qualidade do produto.
- As métricas precisam interpretadas por meio das medidas.

Qualidade de Software

Das métricas às caracteríscas

- As métricas e medidas devem fazer sentido a um contexto específico.
- Diferentes medidas podem representar uma mesma característica ou subcaracterística.
- Uma mesma métrica ou medida pode ser utilizada de forma indireta para calcular diferentes medidas.
- Medidas precisam ser normalizadas: ex: “tamanho”, “total de defeitos”; total de histórias de usuário”; “total

Qualidade de Software

Modelo de qualidade Q-Rapids

QA[a]	Factor [c]	Assessed Metric [c]	Description	Raw Data	Data Source
Maintainability	Code Quality	Non-complex files [b]	Files below the threshold of cyclomatic complexity (10 by default)	*Cyclomatic complexity* per *function* of each file, *total number of files*	SonarQube
		Commented files [b]	Files whose comment density is outside the defined thresholds (by default 10%-30%)	*Density of comment lines* and *lines of code* per each file	SonarQube
		Absence of duplications [b]	Files below the threshold of duplicated lines percentage	*Duplicated lines* and *lines of code* per file	SonarQube
	Blocking Code	Fulfillment of critical/blocker quality rules [b]	Files without critical or blocker quality rule violations	Number of *quality rule violations* per file and their *severity* (blocker, critical, major, minor, info) and *type* (code smell, bug, vulnerability)	SonarQube, Coverity, CodeSonar
		Highly changed files	Unstable files that have been highly changed in the last commits	For each *commit: files changed*, *lines of code added/modified/deleted*, *author*, and *revision*	SVN, git, Gerrit
Reliability	Testing Status	Passed tests [b]	Unit test success density	Number of *unit test errors*, *failures*, *skipped*, and *total*	Jenkins, GitLab
		Fast test builds [b]	Test builds below the duration threshold	*Duration of unit test execution*, *tests* conforming to a *pipeline*	Jenkins, GitLab
		Test coverage	Tests with appropriate coverage	*Condition coverage* and *line coverage* per *unit test*	Jenkins (JaCoCo plugin), SonarQube
	Software Stability	Non-bug density [b]	Ratio of open issues of the type bug with respect to the total number of issues within a customized timeframe	Total number of *issues* (a.k.a. tasks) per *status* (e.g., open, done), *type* (e.g., bug, maintenance, feature), and *timeframe* (e.g., current/last month or current/last sprint)	Jira, GitLab, Redmine, Mantis
		Errors at runtime	Occurrence of critical errors at runtime at the end user site	All tracks of logs, including *type of error* (fatal, error, warning, info, debug, trace), *file* and *line* where it occurred, and *error message*	Logs
		Availability uptime	Percentage of time that the product is accessible	Timestamp at which the system is not available and derived metrics, e.g., *availability uptime, mean time between failures*	Zabbix, Nagios
Functional Suitability	Software Usage	Feature usage	Appropriateness of the features included in the software product regarding their usage	For each *functionality (or feature)*: number of *times used*, *average usage time, customer feedback on the feature* (if available)	Logs, monitoring plugin
Productivity	Issues' Velocity	Resolved issues assigned to a date	Resolved issues assigned to a date (simple date, iteration, or release)	For each *issue* (a.k.a. task): *time created, status, time updated, iteration(s), release(s)*.	Jira, GitLab, Redmine, Mantis
		Issues completely specified [b]	Density of incomplete issues within a timeframe	*Fields of each issue* (e.g., description, due date, assignee, estimated time)	Jira, GitLab, Redmine, Mantis

S. Martínez-Fernández, A. Jedlitschka, L. Guzmán and A. M. Vollmer, "A Quality Model for Actionable Analytics in Rapid Software Development," 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, 2018, pp. 370-377, doi: 10.1109/SEAA.2018.00067.

Qualidade de Software

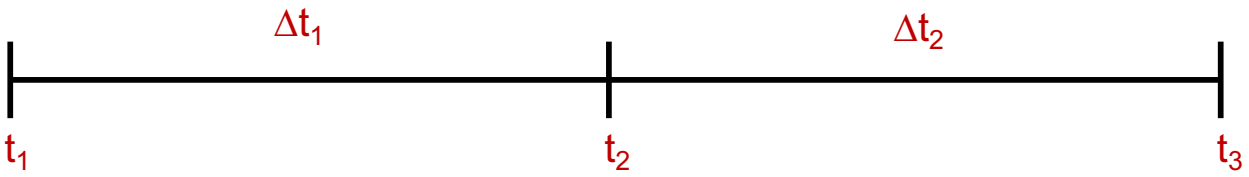
Modelo de qualidade Q-Rapids

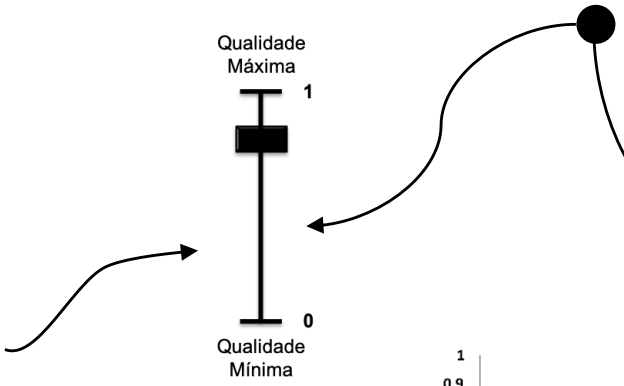
- Métrica LOC/arquivo utilizada de forma indireta para o cálculo das medidas Arquivos não-complexos e arquivos comentados.

QA[a]	Factor [c]	Assessed Metric [c]	Description	Raw Data	Data Source
Maintainability	Code Quality	Non-complex files [b]	Files below the threshold of cyclomatic complexity (10 by default)	*Cyclomatic complexity* per *function* of each file, *total number of files*	SonarQube
		Commented files [b]	Files whose comment density is outside the defined thresholds (by default 10%-30%)	*Density of comment lines* and *lines of code* per each file	SonarQube
		Absence of duplications [b]	Files below the threshold of duplicated lines percentage	*Duplicated lines* and *lines of code* per file	SonarQube
	Blocking Code	Fulfillment of critical/blocker quality rules [b]	Files without critical or blocker quality rule violations	Number of *quality rule violations* per file and their *severity* (blocker, critical, major, minor, info) and *type* (code smell, bug, vulnerability)	SonarQube, Coverity, CodeSonar
		Highly changed files	Unstable files that have been highly changed in the last commits	For each *commit: files changed*, *lines of code added/modified/deleted*, *author*, and *revision*	SVN, git, Gerrit

S. Martínez-Fernández, A. Jedlitschka, L. Guzmán and A. M. Vollmer, "A Quality Model for Actionable Analytics in Rapid Software Development," 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, 2018, pp. 370-377, doi: 10.1109/SEAA.2018.00067.

Qualidade de Softwate ao longo do tempo



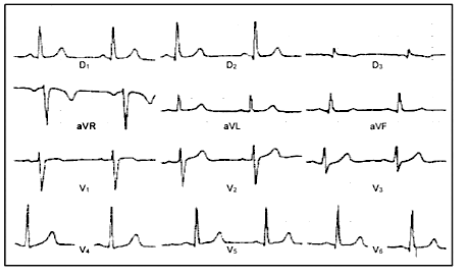


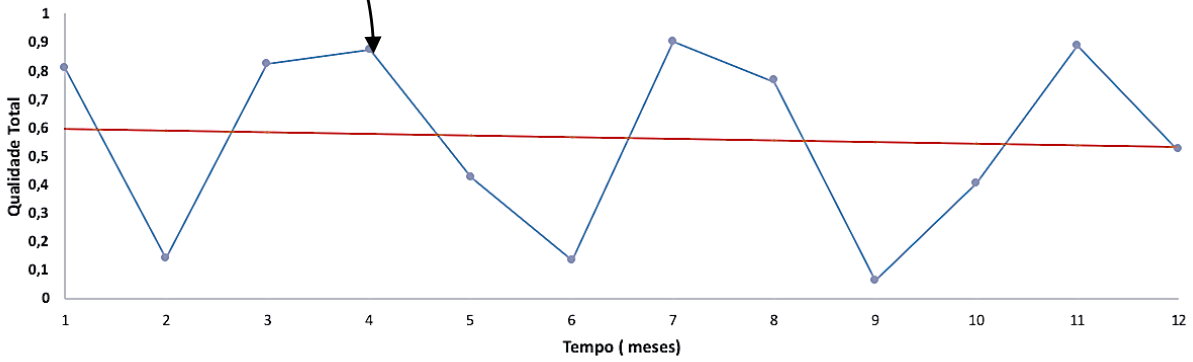
Segurança

Usabilidade

Confiabilidade

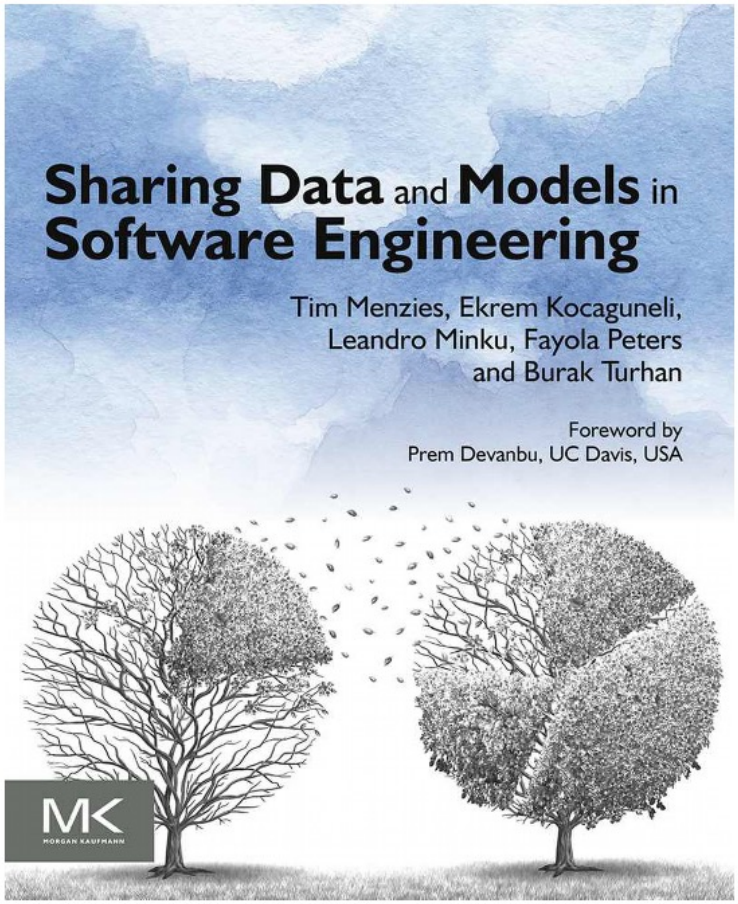
Adequação Funcional

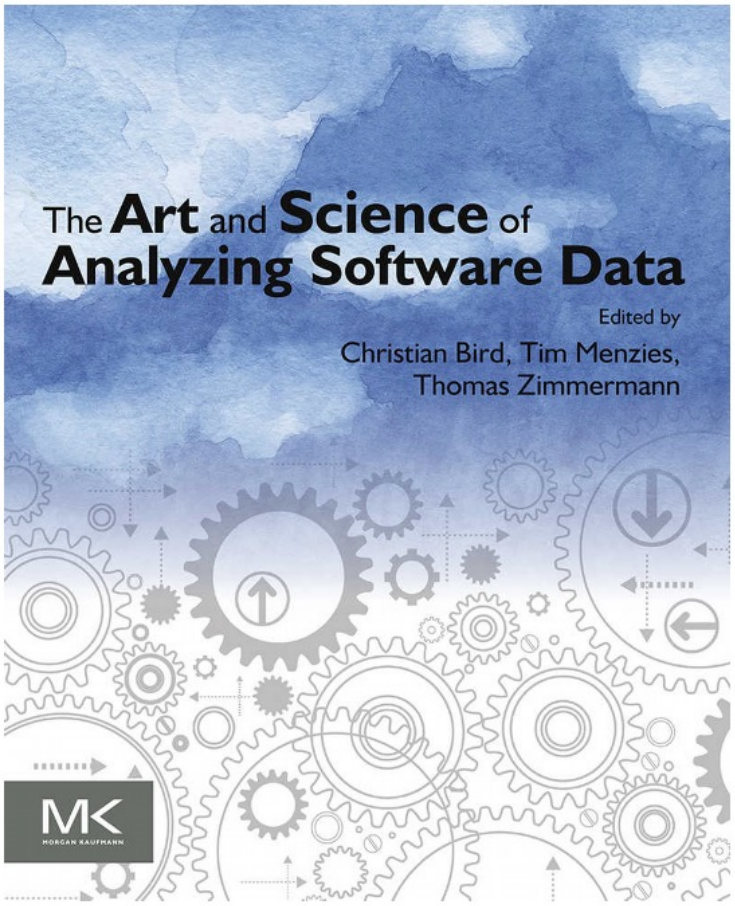




Software Analytics

Software Analytics





Software Analytics

2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering

Software Analytics to Software Practice: A Systematic Literature Review

Tamer Mohamed Abdellatif, Luiz Fernando Capretz, and Danny Ho
Department of Electrical & Computer Engineering
Western University
London, Ontario, Canada
tmohame7@uwo.ca, lcapretz@uwo.ca, danny@nfa-estimation.com

Software Analytics

O que é?

Hassan A, Xie T. Software intelligence: the future of mining software engineering data. FoSER 2010: 161-166.	[Software Intelligence] offers software practitioners (not just developers) up-to-date and pertinent information to support their daily decision-making processes.
Buse RPL, Zimmermann T. Analytics for software development. FoSER 2010:77-90.	The idea of analytics is to leverage potentially large amounts of data into real and actionable insights.
Zhang D, Dang Y, Lou J-G, Han S, Zhang H, Xie T. Software analytics as a learning case in practice: approaches and	Software analytics is to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data driven tasks around software and services (and software practitioners typically include software developers,

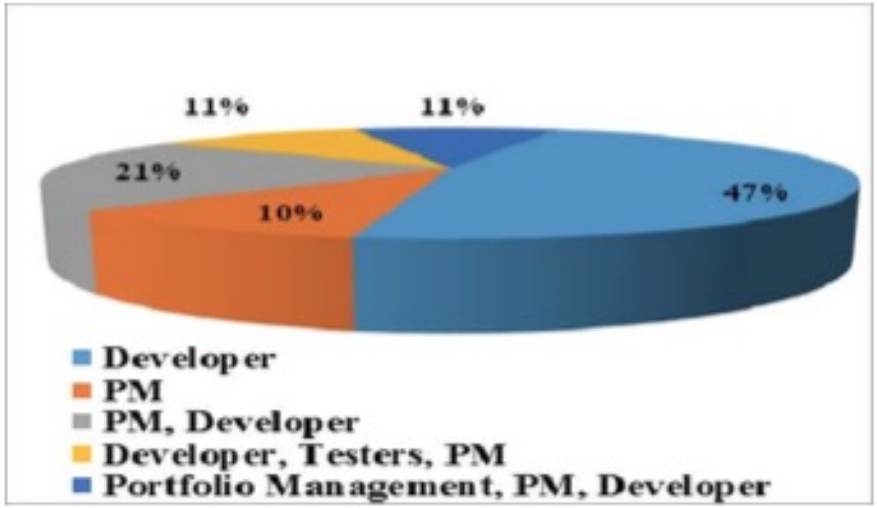
Software Analytics

O que é?

Buse RPL, Zimmermann T. Information needs for software development analytics. ICSE 2012:987-996.	Software development analytics … empower(s) software development teams to independently gain and share insight from their data without relying on a separate entity.
Menzies T, Zimmermann T. Software analytics: so what? IEEE Softw 2013;30(4):31-7.	Software analytics is analytics on software data for managers and software engineers with the aim of empowering software development individuals and teams to gain and share insight from their data to make better decisions.
Zhang D, Han S, Dang Y, Lou J-G, Zhang H, Xie T. Software analytics in practice. IEEE Softw 2013;30(5):30-7.	With software analytics, software practitioners explore and analyze data to obtain insightful, actionable information for tasks regarding software development, systems, and users.

Software Analytics

Onde se aplica?

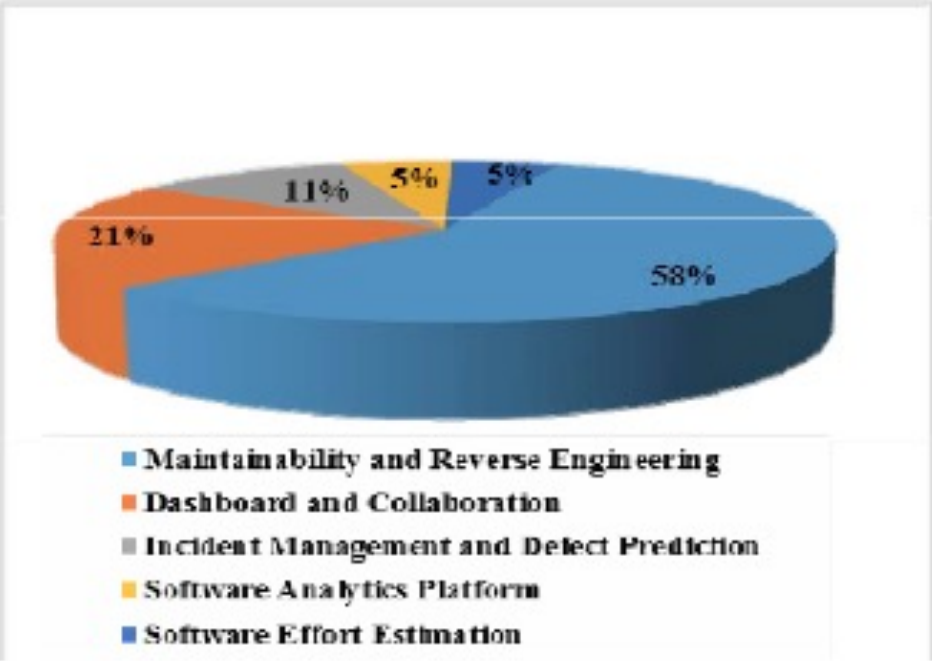


Practitioner	Supporting Studies
Developer	S1, S4, S5, S6, S7, S8, S9, S10, S11, S13, S14, S15, S16, S17, S19
Tester	S2, S13
Project Manager	S2, S3, S4, S8, S10, S11, S12, S13, S18, S19
Portfolio Manager	S10, S19

Tamer Mohamed Abdellatif, Luiz Fernando Capretz, and Danny Ho. 2015. Software analytics to software practice: a systematic literature review. In Proceedings of the First International Workshop on BIG Data Software Engineering (BIGDSE '15). IEEE Press, 30–36.c

Software Analytics

Onde se aplica?

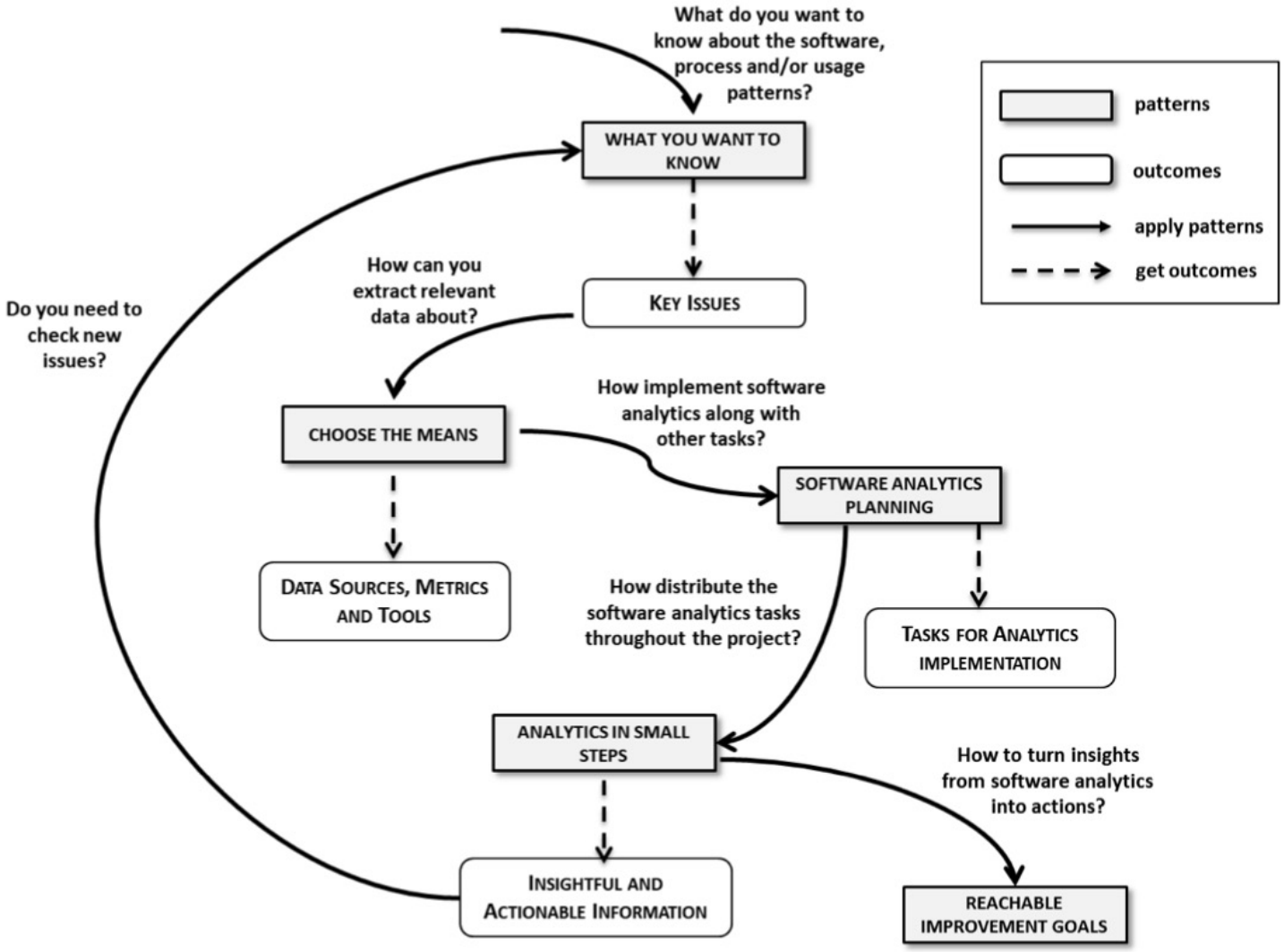


Domain	Studies
Maintainability and Reverse Engineering	S1, S2, S4, S5, S7, S12, S13, S14, S15, S16, S17
Team Collaboration and Dashboard	S3, S9, S10, S18
Incident Management and Defect Prediction	S6, S8
Software Analytics Platform	S11
Software Effort Estimation	S19

Tamer Mohamed Abdellatif, Luiz Fernando Capretz, and Danny Ho. 2015. Software analytics to software practice: a systematic literature review. In Proceedings of the First International Workshop on BIG Data Software Engineering (BIGDSE '15). IEEE Press, 30–36.c

Software Analytics

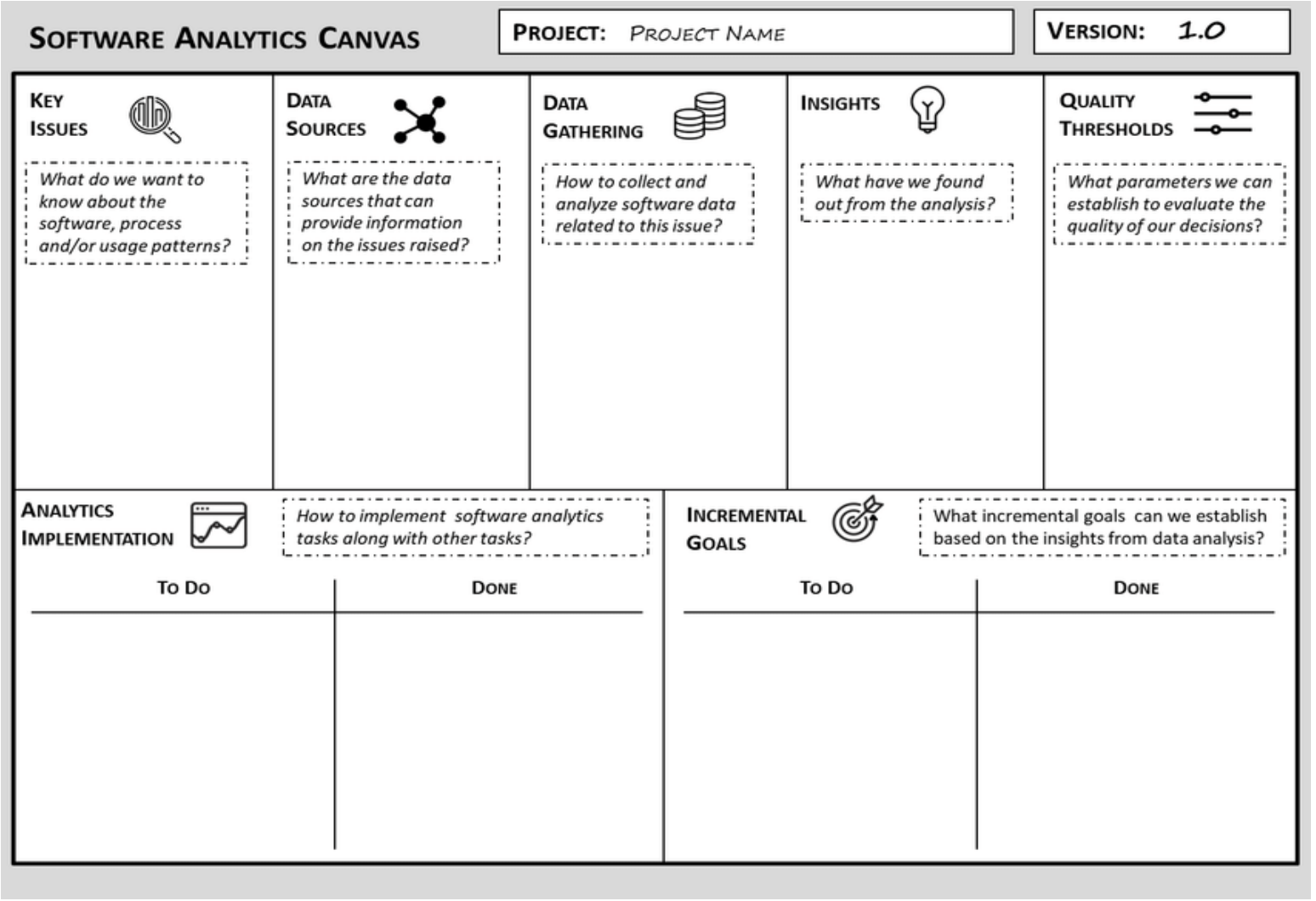
Padrões para estruturar dados e apoiar a tomada de decisão



Joelma Choma, Eduardo Martins Guerra, and Tiago Silva Da Silva. 2017. Patterns for implementing software analytics in development teams. In <i>Proceedings of the 24th Conference on Pattern Languages of Programs (PLoP '17). The Hillside Group, USA, Article 22, 1–12.

Software Analytics

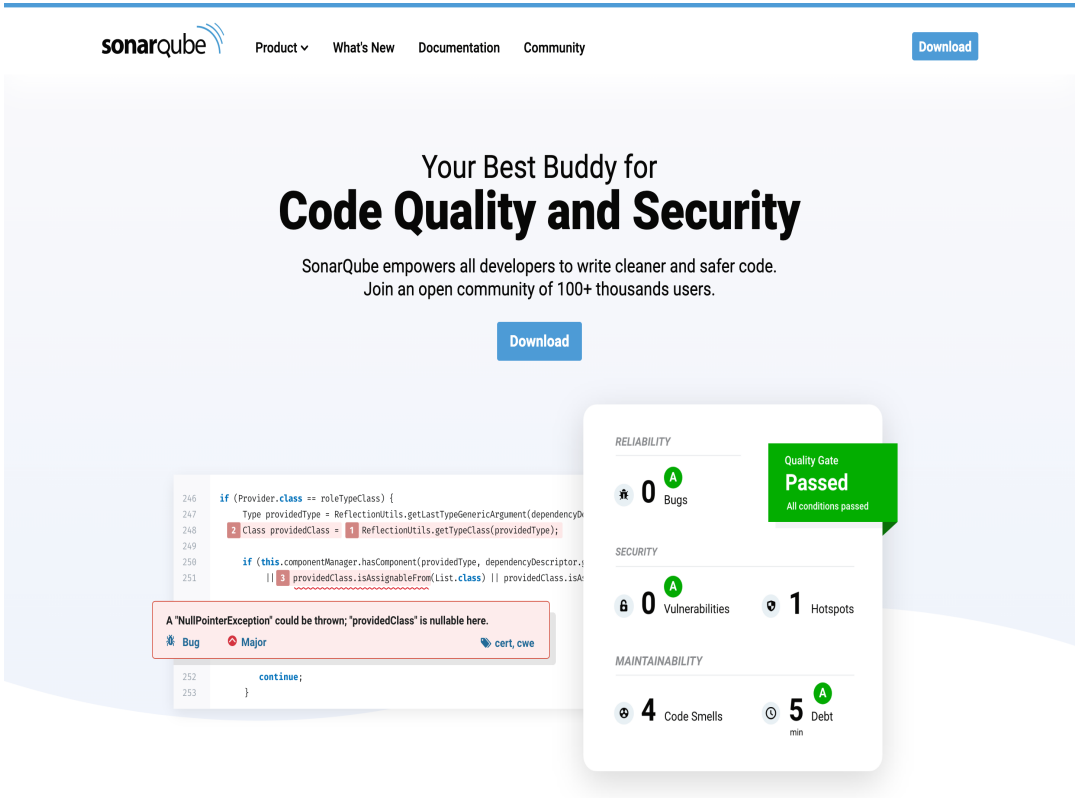
Padrões para estruturar dados e apoiar a tomada de decisão



Choma, Joelma & Guerra, Eduardo & Da Silva, Tiago & Zaina, Luciana & Correia, Filipe. (2019). Towards an artifact to support agile teams in software analytics activities. 88-93. 10.18293/SEKE2019-146.

Ferramenta - SonarQube

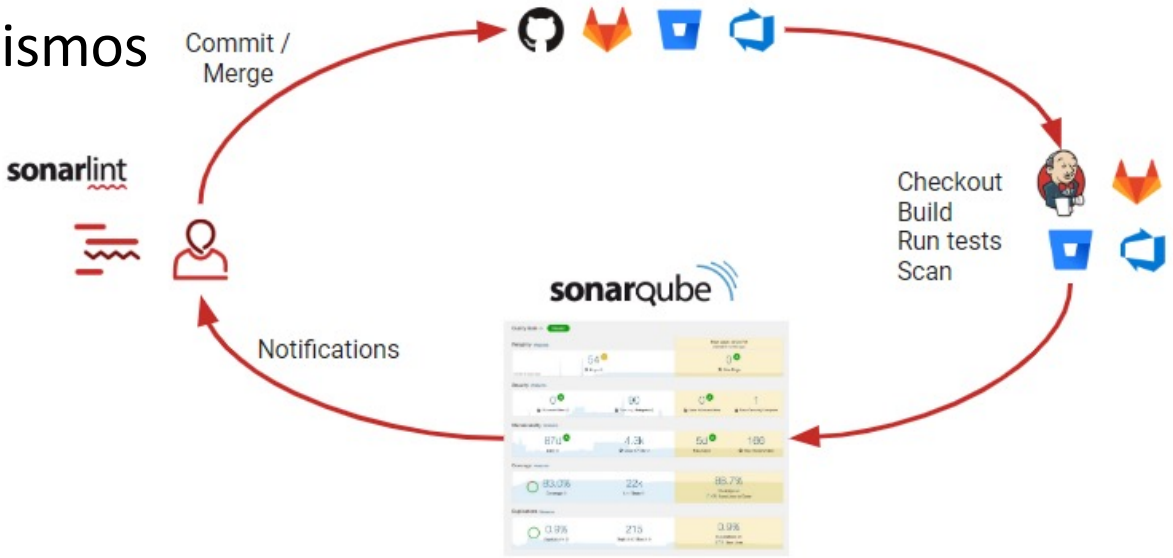
- Análise estática de código-fonte
- Distribuída sob a licença LGPL v3(community-edition)
- +27 Linguagens de Programação
- Possivelmente +popular



Ferramenta - SonarQube

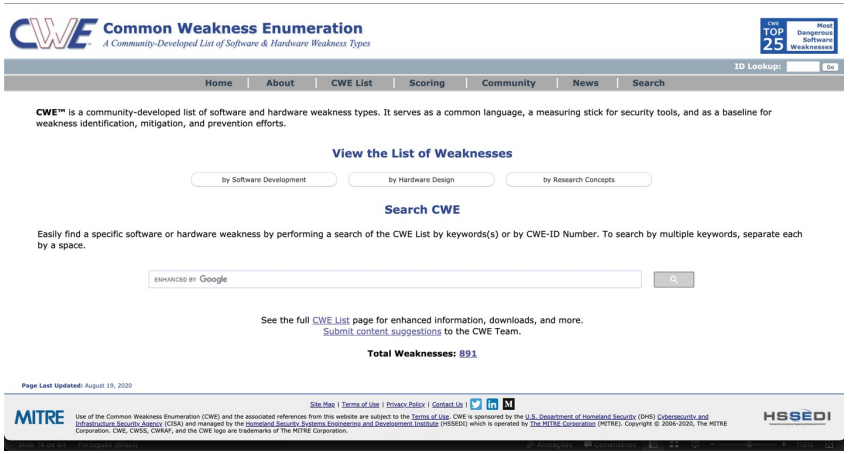
- Atualmente na versão 8.4
- +180 releases
- +100 contribuidores
- Histórico de métricas de qualidade de código-fonte
- Detecta defeitos e vulnerabilidades
- Possui integração com mecanismos de CI/CD

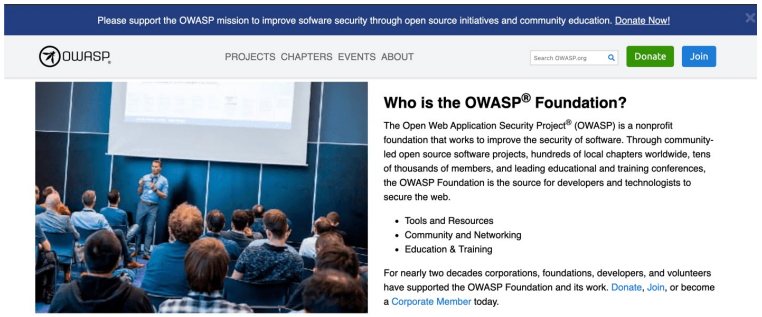
https://github.com/SonarSource/sonarqube

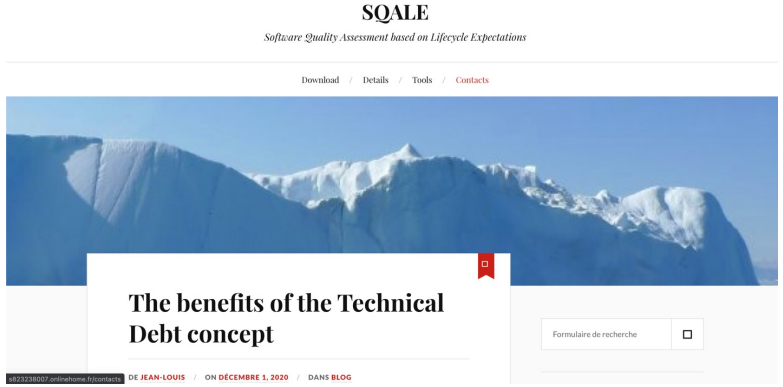


Ferramenta - SonarQube

- Atualmente é uma das ferramentas de análise estática de repositórios de código-fonte mais adotadas na indústria e em estudos acadêmicos.
  - Mais utilizada para observar e monitorar Dívida Técnica. Usa o modelo Sqale como modelo base.
  - Utiliza o catálogo do CWE e OWASP para observar e monitorar a característica de segurança.
  - ...







P. C. Avgeriou et al., "An Overview and Comparison of Technical Debt Measurement Tools," in IEEE Software, doi: 10.1109/MS.2020.3024958.

Ferramenta - SonarQube

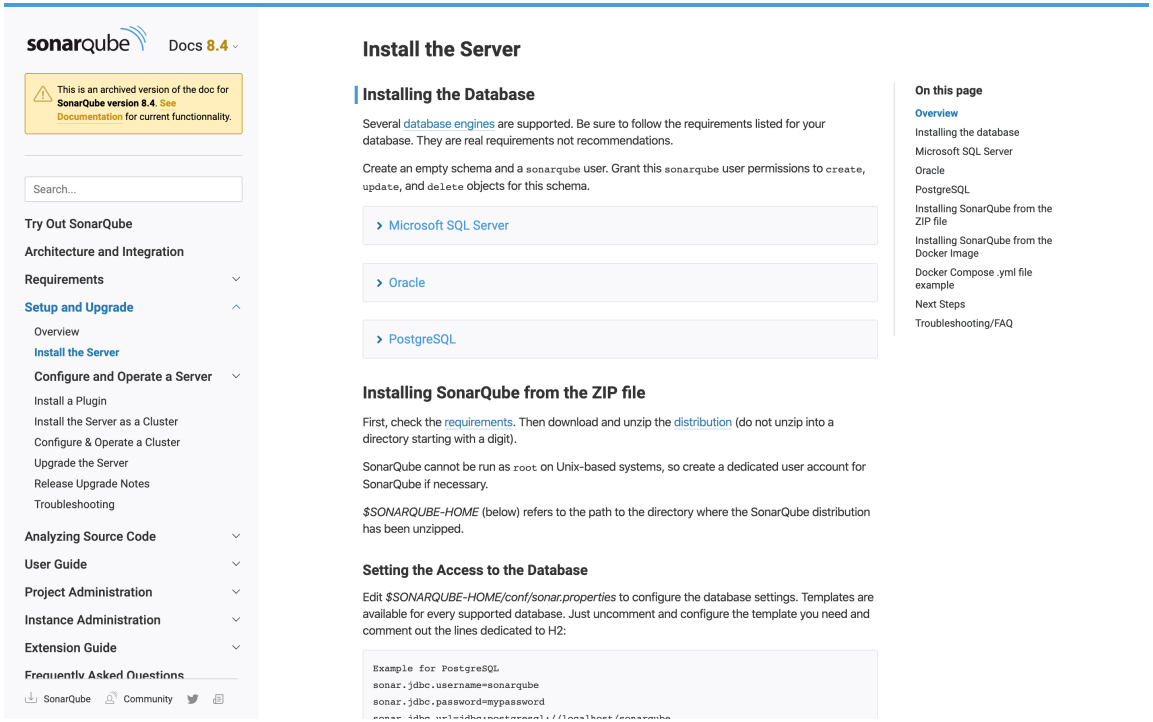
Download e Instalação

https://www.sonarqube.org/downloads/

Ferramenta - SonarQube

Download e Instalação

- Configurar o banco de dados
- Adicionar o drive JDBC
- Configurar o path para o Elasticsearch
- Iniciar o servidor Web
- Ajustar a versão do Java
- Editar os arquivos:
  - **sonar.properties**
  - **wrapper.conf**

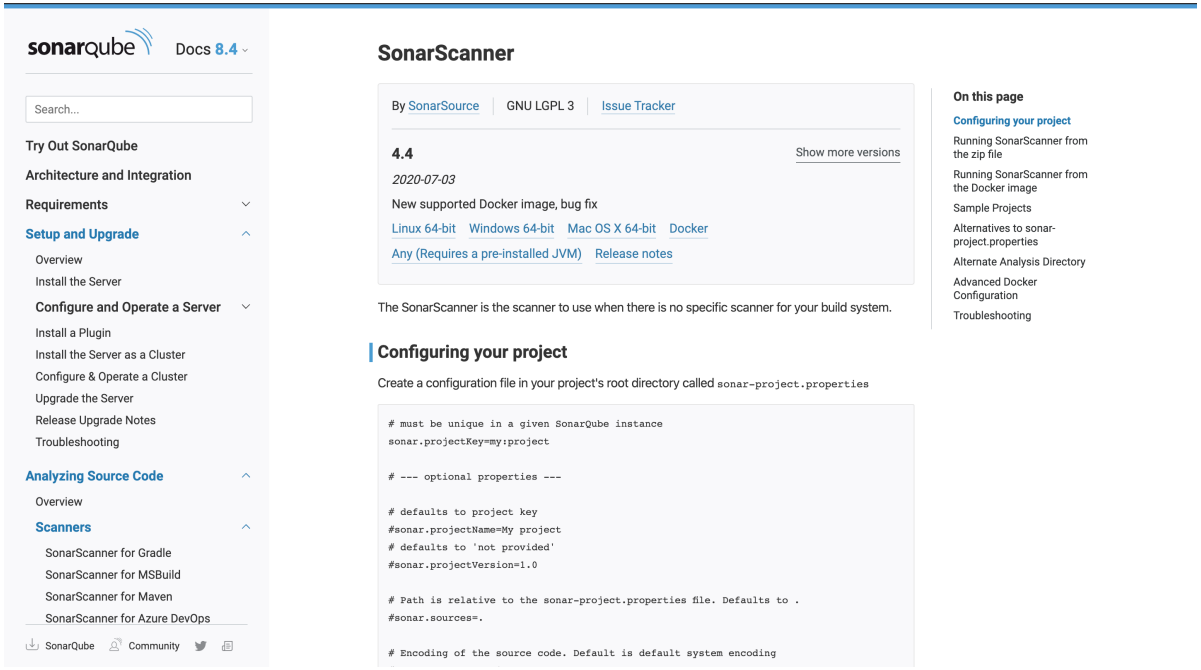


https://docs.sonarqube.org/latest/setup/install-server/

Ferramenta - SonarQube

Download e Instalação

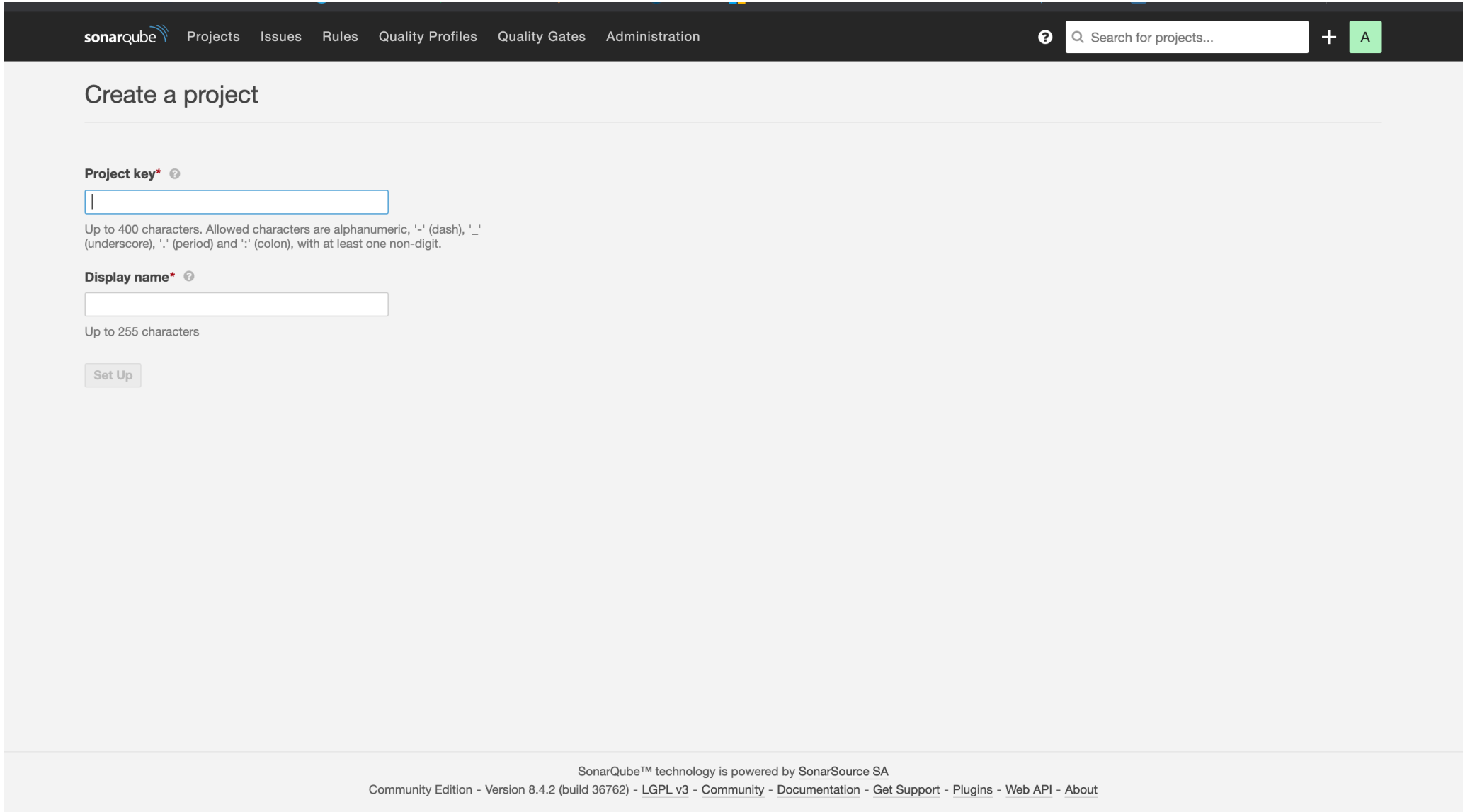
- Fazer o download e instalar o SonarScanner
- Criar o arquivo
  - **sonar-project.properties**
- Executar o SonarScanner



https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/

Ferramenta - SonarQube

Criando um Projeto

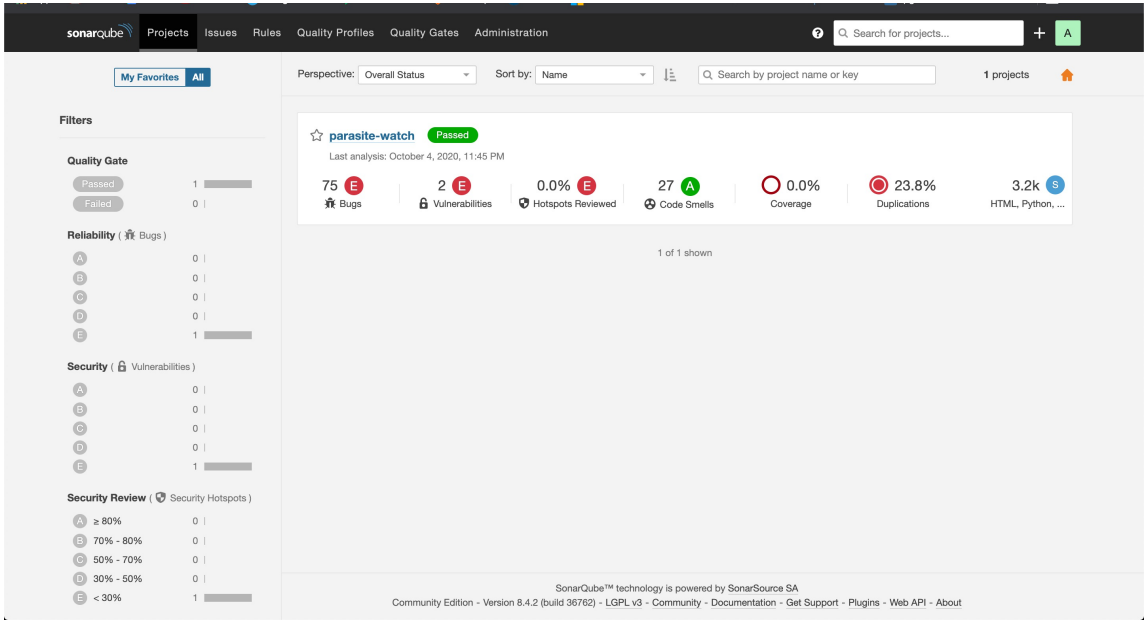


Ferramenta - SonarQube

Analizando um projeto

- Informe o token de usuário
- No diretório do seu projeto(código-fonte), execute os comandos sugeridos pelo sonar-scanner.
  Exemplo:

```
sonar-scanner \
  -Dsonar.projectKey=parasite-watch \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=XXXXXXXXXX
```



Ferramenta - SonarQube

Analisando um Projeto

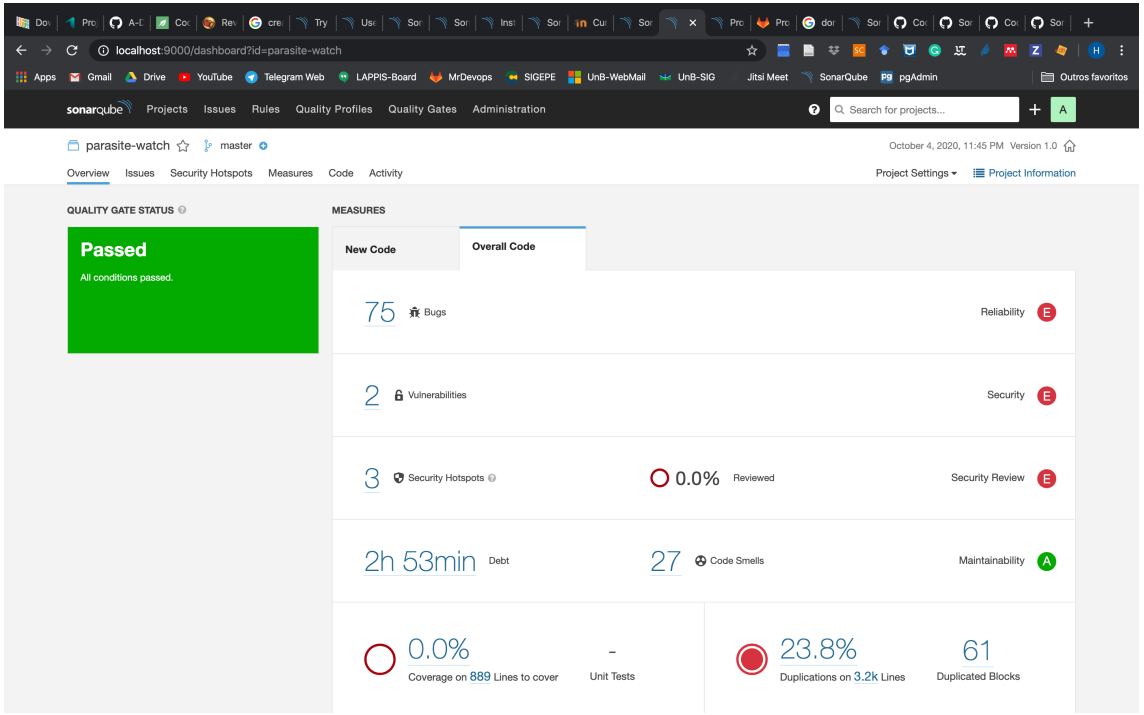
- Tipos de *issues:*

  - Bug, Vulnerability, Code Smell, Coverage or Duplication

- Tipos de Severidades:

  - blocker, critical, major, minor e info

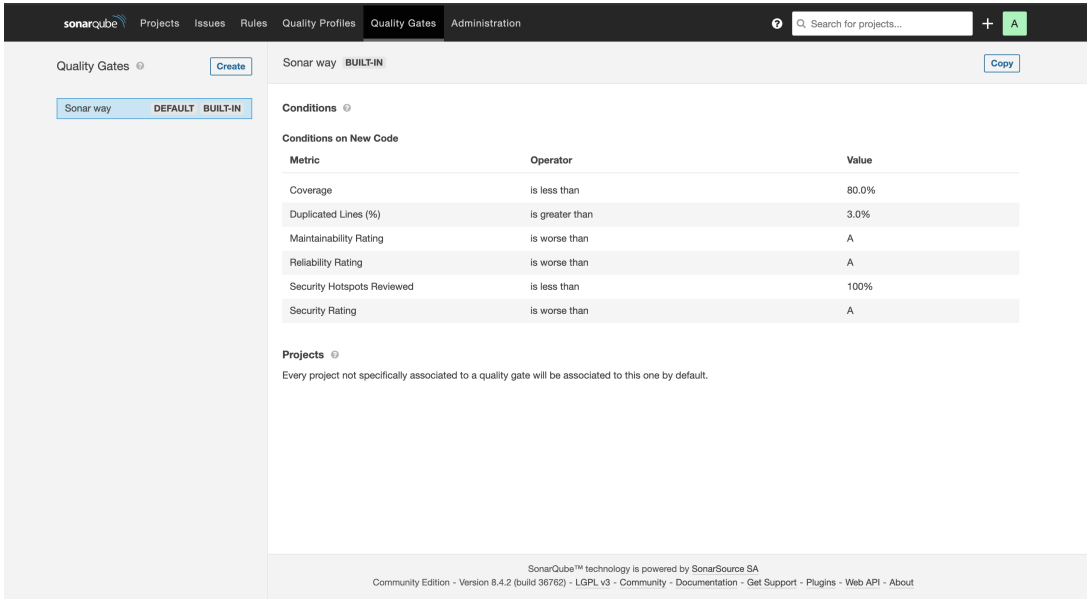
- Passou (**verde**)?



Ferramenta - SonarQube

Analisando um Projeto – *Quality Gate*

- Um *quality gate* representa um conjunto de condições que o projeto deve atender antes de se qualificar para a liberação de produção.

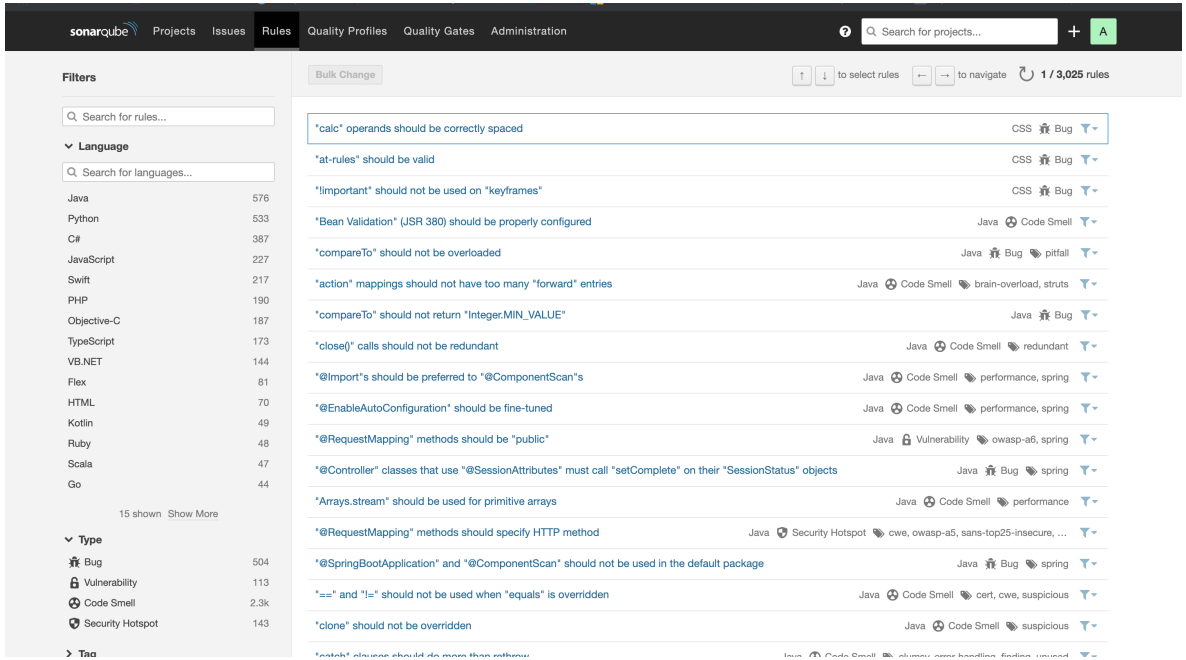


Posso enviar meu código para produção em seu estado atual ou não?

Ferramenta - SonarQube

Analisando um Projeto – *Rules*

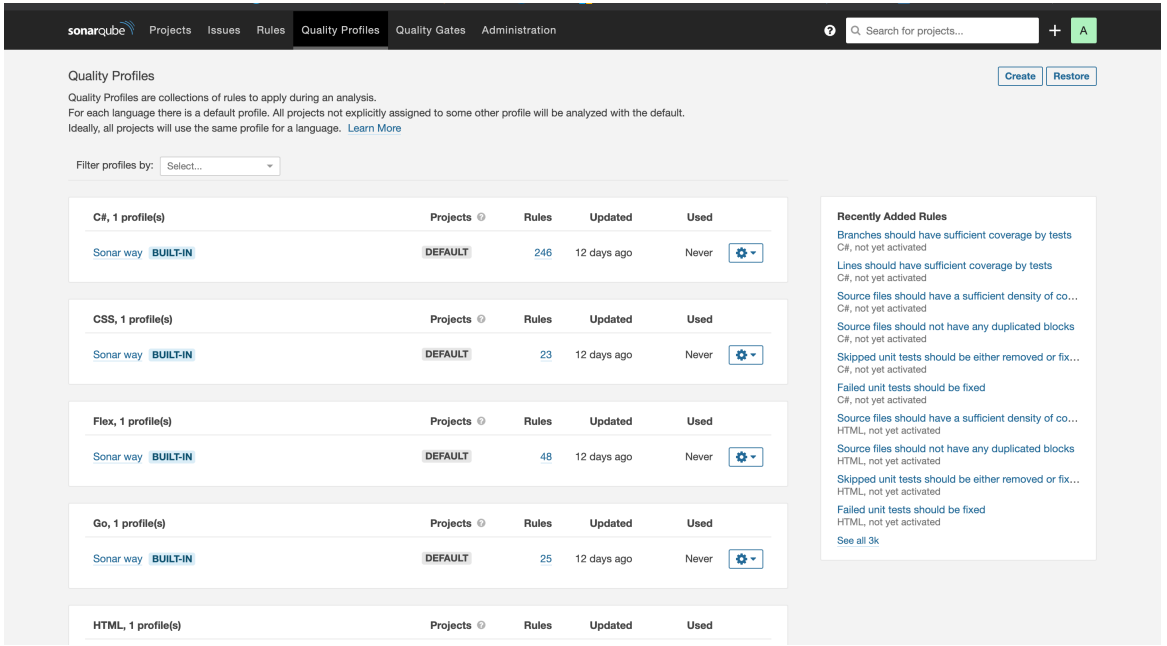
- As regras representam um conjunto de restrições que são aferidas durante a análise
  - estilos
  - boas práticas de programação (code-smells - https://github.com/lee-dohm/code-smells)
  - Percepção de Dívida Técnica (http://www.sqale.org/)
  - ...



Ferramenta - SonarQube

Analisando um Projeto – *Quality Profile*

- São coleções de regras a serem aplicadas durante uma análise.
- Há um perfil padrão para cada linguagem de programação.
- Todos os projetos não atribuídos explicitamente a algum outro perfil serão analisados com o padrão.



Valeu!!