MeasureSoftGram: um modelo de observação multidimensional da qualidade dO produto de software

Hilmer Rodrigues Neri

Tese de doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Guilherme Horta Travassos

Rio de Janeiro

XXXX de 2021

**SUMÁRIO**

[1 Introdução 5](#_Toc45315531)

[1.1 Introdução 5](#_Toc45315532)

[1.2 Delineamento do problema 7](#_Toc45315533)

[1.3 Motivação e Questão de Pesquisa 8](#_Toc45315534)

[1.4 Proposta de Solução 10](#_Toc45315535)

[1.5 Organização do documento 12](#_Toc45315536)

[2 Revisão da Literatura e Metodologia 14](#_Toc45315537)

[2.1 Modelos de Qualidade de Produto de Software 14](#_Toc45315538)

[2.2 Medição e Métricas de Software 14](#_Toc45315539)

[2.3 Aplicações Computacionais de Tensores 14](#_Toc45315540)

[2.4 Metodologia 14](#_Toc45315541)

[3 MeasureSoftGram: o modelo de medição multidimensional da qualidade de software 15](#_Toc45315542)

[3.1 Conceitos básicos sobre Tensores 16](#_Toc45315543)

[3.2 A Estrutura do Modelo de Qualidade 20](#_Toc45315544)

[3.2.1 Definição de métricas e medidas 24](#_Toc45315545)

[3.2.2 Funções de Interpretação e Valores de Referência 27](#_Toc45315546)

[3.2.3 Agregação e Ponderação 31](#_Toc45315547)

[3.2.4 A comparação entre configurações de qualidade de software 34](#_Toc45315548)

[3.2.5 Capturando e expondo as múltiplas relações entre as características de qualidade 38](#_Toc45315549)

[3.2.6 Operacionalizando o MeasureSoftGram: um estudo observacional 45](#_Toc45315550)

[3.2.7 Considerações Finais 48](#_Toc45315551)

[Referências Bibliográficas 49](#_Toc45315552)

**ÍNDICE DE FIGURAS**

[Figura 1: Esquema conceitual do *MeasureSoftGram* 11](#_Toc45285328)

[Figura 2 - Escalar, Vetor, Matriz e Tensor 16](#_Toc45285329)

[Figura 3: Fatias e Fibras em um tensor de 3ª. Ordem - figura adaptada de (CICHOCKI et al., 2009) 17](#_Toc45285330)

[Figura 4: Estrutura Geral do MeasureSoftGram 21](#_Toc45285331)

[Figura 5 : Exemplo da Descrição de um Tensor de Qualidade 22](#_Toc45285332)

[Figura 6: Exemplo da aplicação uma função de interpretação 27](#_Toc45285333)

[Figura 7: Percepção da qualidade ao longo do tempo 33](#_Toc45285334)

[Figura 8: Similaridade entre vetores medida pela diferença de cosseno 35](#_Toc45285335)

[Figura 9: Sinais do seno e cosseno no círculo trigonométrico 37](#_Toc45285336)

[Figura 10: Equalizador da qualidade de software 38](#_Toc45285337)

[Figura 11: Matriz de Balanceamento entre características da ISO 25010 40](#_Toc45285338)

[Figura 12: Exemplo da representação visual de uma escala VAS e uma escala de 5 pts. 41](#_Toc45285339)

[Figura 13: Transformação da escala VAS para escala Razão 42](#_Toc45285340)

[Figura 14: Matriz de correlação dos tensores de qualidade 42](#_Toc45285341)

[Figura 15: Comparação da qualidade planejada X realizada 43](#_Toc45285342)

[Figura 16: Operacionalização do MeasureSoftGram no projeto Parasite 45](#_Toc45285343)

**ÍNDICE DE TABELAS**

[Tabela 1: Estrutura Metodológica desta Pesquisa 12](#_Toc45285318)

[Tabela 2 - Notações matemáticas de espaços tensoriais 15](#_Toc45285319)

[Tabela 3 : notações matemáticas do MeasureSoftGram 20](#_Toc45285320)

[Tabela 4: Conflitos entre atributos de qualidade mitigados no MeasureSoftGram 39](#_Toc45285321)

[Tabela 5: Descrição da configuração de qualidade observada no projeto Parasite 45](#_Toc45285322)

# Introdução

## Introdução

As características pervasivas e ubíquas levam o software a "todo lugar" na sociedade contemporânea. O software deve funcionar em computadores, dispositivos móveis, sistemas embarcados e direciona para uma nova realidade onde não só mais o software se comunica com software, mas coisas se comunicam com coisas. Além disso, a expectativa pela disponibilidade antecipada do produto influenciou a evolução do processo de desenvolvimento no tocante a interatividade da construção e da parcialidade incremental das entregas até atingir a versão final do produto. Diante dessa nova realidade, a atividade de garantia da qualidade de sistemas de software, em especial a qualidade das versões de produtos de software liberadas ao longo do processo de desenvolvimento, tornou-se mais complexa, sendo vital para o sucesso e sobrevivência das organizações que desenvolvem produtos de software.

Ao mesmo tempo, informações geradas por usuários a respeito do uso do software e suas versões estão disponíveis em fóruns de discussão específicos, redes sociais, lojas virtuais e demais repositórios que disponibilizam produtos de software. Essa facilidade relacionada a oferta de informações sobre o produto de software em utilização, aliada as características interativas e incrementais do desenvolvimento de sistemas de software contemporâneos, enseja que engenheiros de software e gerentes de produtos sejam capazes de observar e entender a qualidade do produto de software e suas versões, de forma continuada ao longo do processo de desenvolvimento, acompanhando a sequência de entrega de novas versões do produto que, cada vez mais, são disponibilizadas em intervalos de tempo curtos e com mais frequência para os usuários finais (FITZGERALD; STOL, 2017).

A qualidade do produto de software (QPS) é um objeto de estudo na engenharia de software há aproximadamente quatro décadas. A partir dos modelos seminais de avaliação da qualidade do produto de software, definidos por Boehm et al. (1976) e McCall et al. (1977), os quais identificaram características, fatores, e subcaracterísticas de qualidade hierarquizados e inerentes ao software, os modelos posteriormente elaborados seguiram a inspiração das propostas iniciais de hierarquização destes atributos de qualidade. Essa organização viabilizou a observação isolada de aspectos de qualidade interna, externa e de utilização do produto de software e associados ao fenômeno da qualidade do produto de software, além de contribuir para que modelos mais robustos e recentes estabelecessem conexões lógicas entre as medidas e as descrições das características e subcaracterísticas (“teorias”) consideradas abstratas e, por conseguinte, de difícil utilização prática por parte da indústria (Mordal-Manet et al., 2009) (Wagner et al., 2012).

Em adição aos trabalhos pioneiros surgiu o modelo proposto por Dromey (1995). Trata-se do primeiro trabalho a estabelecer um mecanismo lógico e sistemático para mapear e relacionar as descrições conceituais de características, subcaracterísticas e fatores, com as definições de estruturas sintáticas presentes nas linguagens de programação. Dromey argumenta que o código-fonte e a atividade de implementação são fontes primárias de observação e percepção das propriedades “concretas” do produto de software, como por exemplo, variáveis e expressões. Orientado pelos trabalhos precursores, o modelo ISO/IEC 9126 (2001) e sua atualização ISO/IEC 25010 (2010) tornaram-se referência conceitual e normativa no que tange a observação da qualidade do produto de software. Adicionalmente, existem várias outras variações dos modelos iniciais, como por exemplo, aqueles que propõem novas características de qualidade, específicas ao contexto de software livre, como (Wasserman, 2006) e SQO-OSS (Samoladas, 2008), além de modelos adicionais, como aqueles encontrados nos resultados do estudo secundário realizado por Miguel et al. (2014).

O conhecimento acumulado ao longo dos tempos torna-se evidente através da observação desses diferentes modelos de avaliação da qualidade do produto: há um vasto conjunto de métricas e medidas identificadas como relacionadas a qualidade do produto de software (Kitchenham, 2010) (Elberzhager et al., 2012) (Tahir e MacDonell, 2012) (Mendonça et al., 2013); dos resultados de estudos sobre os relacionamentos entre diferentes atributos de qualidade (características e subcaracterísticas) que mostram a importância e necessidade de se entender as compensações (*tradeoffs)* associadas, tendo em vista os riscos financeiros com impacto direto no custo do desenvolvimento (Al-Daajeh et. al, 2012-b); da utilização de abordagens orientadas à análise de dados-ações, de forma a apoiar a tomada de decisão técnico-gerencial acerca da qualidade do produto de sistemas de software (Zhang et al., 2011) (Abdellatif et al. 2015); no uso de estratégias de experimentação em engenharia de software, em larga escala, para apoiar a escolha de versões de produto de software a serem disponibilizadas para os usuários finais, a partir do comportamento em uso do software (Kohavi et al. 2010) (Lindgren e Münch, 2015) (Schermann et al. 2017) (Fabijan et al., 2017); e por fim, nas diferentes necessidades de informação e especificidades de diferentes contextos, que fazem com que engenheiros de software, gerentes de produto, usuários finais e demais interessados nas informações sobre o produto tenham que decidir, ao longo do processo de desenvolvimento, em como garantir o alinhamento do desenvolvimento ou a evolução do produto à estratégia de negócio das organizações (Buse e Zimmermann, 2012) (Dybå et al., 2012).

Esse conhecimento, de certa forma, tem sido encapsulado em modelos de QPS recentes, tais como o SQuale (Mordal-Manet et al., 2009), Quamoco (Wagner et al., 2012) e, especialmente, nos modelos QATCH (Siavvas et al., 2017) e Q-Rapids (López et al., 2018), que também tomam por base a ISO/IEC 25010. Entretanto, analisar tão somente aspectos de qualidade interna, externa e de utilização do produto de software isoladamente não é suficiente para que gestores de produtos ou engenheiros de software consigam decidir adequadamente sobre a aceitação ou não de uma versão do produto, ao longo do processo de desenvolvimento ou do produto de software per si, em momento de entrega final.

## Delineamento do problema

A falta de entendimento acerca da qualidade apresentada pelo produto de software ao longo do ciclo de desenvolvimento e em momento de entrega da versão final é muitas vezes a causa de decisões inadequadas de projeto. Aliado a isso, a falta de qualificação dos dados (capturando medidas) previamente as avaliações da qualidade contribui sobremaneira a análises e interpretações equivocadas. Um desafio comum para os profissionais da área de software é a identificação e correção de decisões inadequadas antes que estas possam produzir efeitos indesejáveis no produto de software. Isso provavelmente ocorre porque é mais fácil julgar decisões em engenharia de software após o acontecimento do fato, do que monitorar e avaliar o produto de software, antes que o impacto das decisões na qualidade final do produto seja conhecido (Hoover et al., 2010).

Portanto, a carência de indicadores que permitam apoiar a compreensão ou clareza das características da qualidade do produto de software, principalmente a falta de conhecimento sobre suas relações e influências mútuas, contribui para a tomada de decisões não assertivas (inadequadas ou erradas), comprometendo:

1. a qualidade do produto de software em desenvolvimento ou em operação;
2. o comportamento do software em uso; e
3. a estratégia de negócios das organizações.

Particularmente, a observação da qualidade do produto de software de forma unidimensional (linear), esconde e, em vista disso, dificulta a percepção das relações entre suas características e subcaracterísticas de qualidade simultaneamente. Portanto, os efeitos positivos e colaterais das relações desses atributos não são conhecidos, representáveis, ou observáveis, afetando a percepção da qualidade do produto de software e influenciando, consequentemente, a percepção da qualidade do produto, o que pode afetar as tomadas de decisão ao longo do projeto. Encontrar um indicador multidimensional da qualidade do produto de software pode representar uma alternativa evolucionária na busca de uma solução para uma percepção fidedigna da qualidade do produto de software.

## Motivação e Questão de Pesquisa

Analisando os estudos dos modelos de qualidade do produto de software percebemos que: i) há pouca evidência experimental que apoie a análise das relações entre as características de qualidade (Henningsson e Wohlin, 2002) (Al-Daajeh et al., 2012-a); ii) existe dificuldade em agregar estudos sobre os relacionamentos entre as características de qualidade de produto (Svahnberg e Henningsson, 2009); iii) as evidências a respeito das relações entre características de qualidade, em sua maioria, são descritas em alto nível e foram obtidas por meio de *surveys* primários e secundários. Em alguns desses estudos a população é representada por profissionais da prática (Aldaajeh et al., 2012-b), em outros, por artigos (Haoues et al., 2017); iv) e, principalmente, a observação unidimensional da qualidade do produto de software, ou seja, observar uma ou duas características de qualidade em cada momento, sem considerar seus efeitos colaterais ou influências nas demais características de qualidade do produto. Essas lacunas nos levam a refletir sobre a confiança da representação dos indicadores unidimensionais frente a realidade da percepção da qualidade do produto de software pelos *stakeholders*.

Adicionalmente, estudos experimentais sobre medição e medidas de qualidade de software geralmente relatam a falta de rigor no processo de medição e validação das medidas (Kitchenham, 2010) (Meneely et al., 2013); problemas com escalas entre medidas (Juristo e Moreno, 2010); a dificuldade de estabelecer valores de referência para medidas de software (Lavazza e Morasca, 2016) (Lima et al., 2017); e, por último e não menos relevante, as características do contexto que dificultam, ou mesmo, impossibilitam a comparação de medidas entre diferentes produtos ou projetos, diante das especificidades dos diferentes contextos de projeto (Dybå et al., 2012) (Al-Daajeh et. al, 2012-b).

Ainda que os resultados obtidos sejam de grande valia para a evolução do conhecimento na área e o entendimento da qualidade do produto de software, os modelos sempre apoiam a observação da qualidade do produto, como apontado anteriormente, sob uma perspectiva unidimensional, focando preferencialmente em características diretamente mensuráveis e internas de qualidade. Entretanto, o referencial de qualidade, se tratado de forma unidimensional, impede uma visão holística da qualidade do produto, limitando a percepção da variação da qualidade relacionada a um atributo em particular, sem permitir a percepção fidedigna da qualidade final do produto, tendo em vista as expectativas dos *stakeholders* para diferentes aspectos da qualidade, conforme será discutido no Capítulo 2. Desta forma, argumentamos que as relações entre as características de qualidade internas, externas e de utilização devem ser simultaneamente consideradas e analisadas por meio de um modelo multidimensional de qualidade, direcionado por uma abordagem orientada à interpretação analítica de medidas, a fim de capturar e representar de forma fidedigna para os *stakeholders* o espectro da qualidade do produto de software. Essa é a principal motivação e diferencial do modelo multidimensional para observação da qualidade do produto de software a ser proposto nesta tese, se comparado aos modelos de qualidade atualmente disponíveis e descritos na literatura técnica.

Tendo em vista que: i) as limitações dos modelos de qualidade atuais afetam a avaliação da qualidade dos produtos de software ao longo do ciclo de desenvolvimento, e que ii) organizações que lidam com o desenvolvimento de sistemas de software contemporâneo necessitam compreender melhor os fatores que afetam a qualidade do produto de software, incluindo suas relações e influências mútuas, para perceber de forma fidedigna a qualidade do produto de software ao longo do ciclo de desenvolvimento interativo e incremental; a questão de pesquisa norteadora e a ser respondida nesta tese é:

*Como observar a qualidade do produto de software, de forma multidimensional, a partir da combinação das diferentes dimensões e relações da qualidade (interna, externa) do produto ao longo do seu ciclo de desenvolvimento?*

Acreditamos que a resposta a esta pergunta, através da proposição de um indicador multidimensional e evidenciado fidedigno a expectativa da qualidade do produto de software dos *stakeholders*, permita oportunidades de aplicação em diferentes cenários de desenvolvimento, incluindo sua utilização no processo de tomada de decisão de aceitação de versões, tanto pelas equipes de desenvolvimento quanto pelos gestores de desenvolvimento de produto. Para tanto, entendemos ser relevante atender o seguinte objetivo:

|  |  |
| --- | --- |
| **Analisar** | características e subcaracterísticas de qualidade de produto de software (internas, externas e uso) representados por atributos de qualidade (fatores, medidas e suas relações) |
| **com o propósito** | de caracterizar |
| **com respeito à** | Viabilidade e fidedignidade de um indicador multidimensional de qualidade do produto de software combinando diferentes atributos de qualidade |
| **do ponto de vista de** | gerentes de projeto e engenheiros de software |
| **no contexto de** | projetos de software em ambientes de software-livre e organizações públicas federais brasileiras. |

## Proposta de Solução

Considerando a questão de pesquisa e os objetivos previamente descritos, apresentamos os conceitos iniciais que subsidiam a proposição de um modelo multidimensional para observação da qualidade do produto de software, o qual denominaremos *MeasureSoftGram*. O nome do modelo intenciona capturar essencialmente a metáfora associada ao eletrocardiograma, onde diferentes indicadores biométricos cardíacos são combinados para representar a situação clinica (qualidade) do coração (Feldman e Goldwasser, 2004).

Desta forma, as contribuições esperadas dessa proposta de tese de doutorado são:

1. proposição de um modelo matemático que permite modelar a qualidade de software de forma multivariada e multiespacial, por meio do uso de tensores comparáveis no espaço. Por hora, tensores representam a generalização dos conceitos de vetores e escalares. Logo, um tensor é um *array* multidimensional;
2. proposição de um conjunto de regras de transformações matemáticas (normalização, ponderação e agregação), em espaços multidimensionais, construído a partir da aplicação de conceitos da álgebra multilinear e aplicável em para qualquer quantidade de métricas ou medidas;
3. um mecanismo, que chamamos de “equalizador da qualidade”, para expor e balancear (*tradeoff*) os relacionamentos entre características e subcaracterísticas da qualidade, além de associar a necessidade de informação a objetivos de medição a serem priorizados;
4. um modelo de análise da qualidade construído a partir da aplicação de conceitos trigonométricos, que auxilia a interpretação das transformações muldimensionais e viabiliza a comparação entre os tensores de qualidade, no espaço e a percepção da qualidade no plano;
5. Um conjunto de facilidades computacionais e visualizadores gráficos que permitem observar a variação dos indicadores ao longo de versões sucessivas do produto de software.

Na Figura 1 apresentamos a abstração conceitual do MeasureSoftGram. As demais propriedades e detalhes do modelo serão tratados no capítulo 3.



Figura 1: Esquema conceitual do *MeasureSoftGram*

Para a construção do *MeasureSoftGram* seguiremos uma estratégia de experimentação baseada no trabalho de Wohlin e Aurum (2015), que propuseram uma estrutura conceitual para auxiliar pesquisadores a estruturarem o delineamento da pesquisa (*research design*) de seus estudos. Essa estrutura é composta por um conjunto de pontos de decisões organizados em três fases: estratégica; tática e operacional. Na Tabela 1 apresentamos a síntese do delineamento da pesquisa desta proposta de tese, que será detalhada na subseção 3.1 – Metodologia, parte integrante do Capítulo 3.

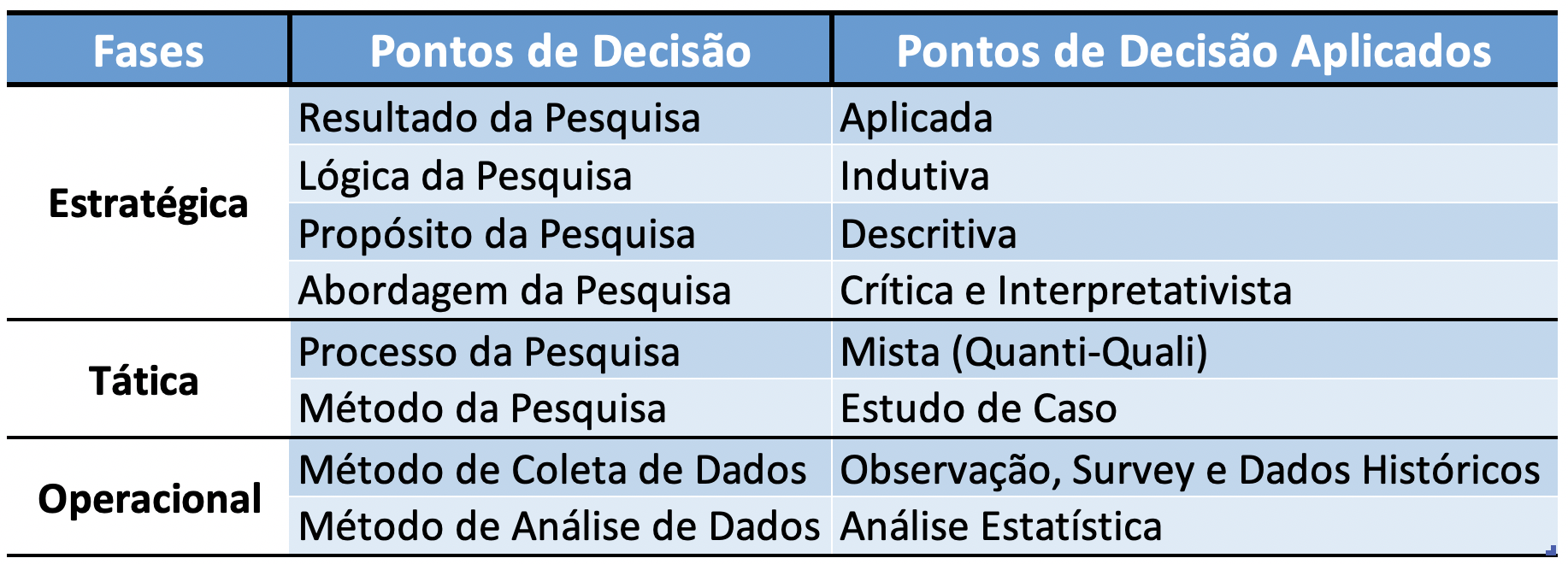


Tabela 1: Estrutura Metodológica desta Pesquisa

A disponibilização de um indicador muldimensional da qualidade do produto de software, com evidencia de viabilidade e fidedignidade quanto a percepção da qualidade fornecida aos *stakeholders* definem o escopo desta tese. Entretanto, acreditamos que sua disponibilidade permitirá apoiar diferentes atividades do processo de desenvolvimento, em destaque: engenharia de software contínua (Fitzgerald e Stol, 2017); experimentação contínua em engenharia de software (Kohavi et al., 2013) (Fabijan et al., 2017) ; e, o uso de uma abordagem analítica para interpretar medidas de qualidade de software que apoiem a tomada de decisão sobre o aceite de versões de produto de software (Zhang et al., 2011) (Bird et al., 2015). Investigar a viabilidade de uso do indicador nestes cenários está fora do escopo deste trabalho, porém compõem relevantes cenários de investigação que merecem atenção em futuro próximo.

## Organização do documento

Este trabalho está organizado em cinco capítulos. O primeiro, compreende esta introdução.

No segundo capítulo, apresentamos a revisão bibliográfica das áreas de conhecimento que sustentam as ideias associadas a esta proposta de tese. Essa revisão aborda os estudos dos modelos de qualidade de produto desenvolvidos ao longo do tempo na engenharia de software, bem como estudos que se propuseram a estudar as relações entre os fatores que governam a qualidade do produto de software. Em seguida, são introduzidos conceitos sobre uma abordagem analítica de interpretação de medidas de software. Também são introduzidos conceitos sobre o processo de desenvolvimento de sistemas contemporâneos e como a qualidade do produto de software vem sendo tratada segundo essa nova realidade. Por fim, são apresentados alguns conceitos matemáticos sobre espaços tensoriais.

Apresentamos no terceiro capítulo a proposta do modelo, onde são detalhados os pontos de decisão referentes à metodologia desta pesquisa. Além disso, são descritos os componentes da solução proposta nesta proposta de tese. Em destaque, a concepção de um modelo matemático e o uso de tensores para modelar e analisar a qualidade do produto de software. São abordadas diferentes questões a serem observadas ao lidar com abstrações no campo vetorial, além do tratamento de dados, como: tratamento de escalas e agregação de medidas; observação das mútuas e múltiplas relações entre variáveis que descrevem um sistema de qualidade de produto de software; ponderação da relevância entre medidas; espaços e subespaços vetoriais usados para representar os fatores da qualidade de produto de software; comparações vetoriais para abstrair e viabilizar a aferição da qualidade de versões de produto de software.

No quarto capítulo, apresentamos os resultados de uma prova de conceito utilizada para indicar a viabilidade inicial do MeasureSoftGram por meio de um protótipo operacional.

Por fim, o quinto capítulo contém as considerações finais, próximos passos planejados e contribuições esperadas desta pesquisa.

# Revisão da Literatura e Metodologia

## Modelos de Qualidade de Produto de Software

## Medição e Métricas de Software

## Aplicações Computacionais de Tensores

## Metodologia

# MeasureSoftGram: o modelo de medição multidimensional da qualidade de software

Inicialmente serão apresentados alguns conceitos preliminares, além de notações sobre o espaço tensorial, necessárias à compreensão do modelo matemático proposto. Depois, serão apresentados a descrição do modelo, seguido de um estudo preliminar, que serviu de prova de conceito sobre a viabilidade de aplicação da teoria proposta.

Para auxiliar a leitura, na Tabela 2 são descritas algumas notações matemáticas utilizadas nas próximas seções deste documento. Seguiremos a notação apresentada no estudo secundário realizado por Kolda e Bader (2009) que provê uma visão geral sobre a aplicação de tensores em diferentes áreas. Essa notação também é utilizada em outro estudo secundário, voltado para o uso de tensores em aplicações computacionais, conduzido por Papalexakis, Falouttsos e Sidiropoulos (2016).

|  |  |
| --- | --- |
| **Símbolo** | **Definição** |
|  | Escalar, vetor, matriz, tensor |
|  | Conjunto dos números reais |
| • | Produto interno |
|  | Produto externo |
|  | Produto Kronecker |
|  | Produto Khatri-Rao |
|  | Produto modo- |
|  | Matricização modo- do tensor |
|  | Norma de Frobenius do tensor |
|  | i-ésima posição de (o mesmo se aplica para matrizes e tensores de ordem maior) |
|  | Abrange toda a i-*ésima* coluna de (o mesmo se aplica em tensores de ordem maior) |
|  | Abrange toda a i-*ésima* linha de (o mesmo se aplica em tensores de ordem maior) |
| vec() | Operador de vetorização |

Tabela 2 - Notações matemáticas de espaços tensoriais

## Conceitos básicos sobre Tensores

Um dos interesses de estudo da álgebra multilinear são os espaços vetoriais -dimensionais com aplicações -lineares. Nesse sentido, a noção de tensores pode ser entendida como a generalização dos conceitos de escalares, vetores e matrizes, para arrays de ordens maiores, ou seja, igual ou maior a 3 dimensões. Logo, um tensor é um array multidimensional e usualmente suas dimensões são conhecidas como modos. Assim, a dimensionalidade de um tensor é definida pela sua ordem, representada por seus modos. Portanto, um vetor é de um caso especial de um tensor de ordem 0, assim como, um tensor de terceira ordem possui modo 3, conforme é possível observar por meio da Figura 2

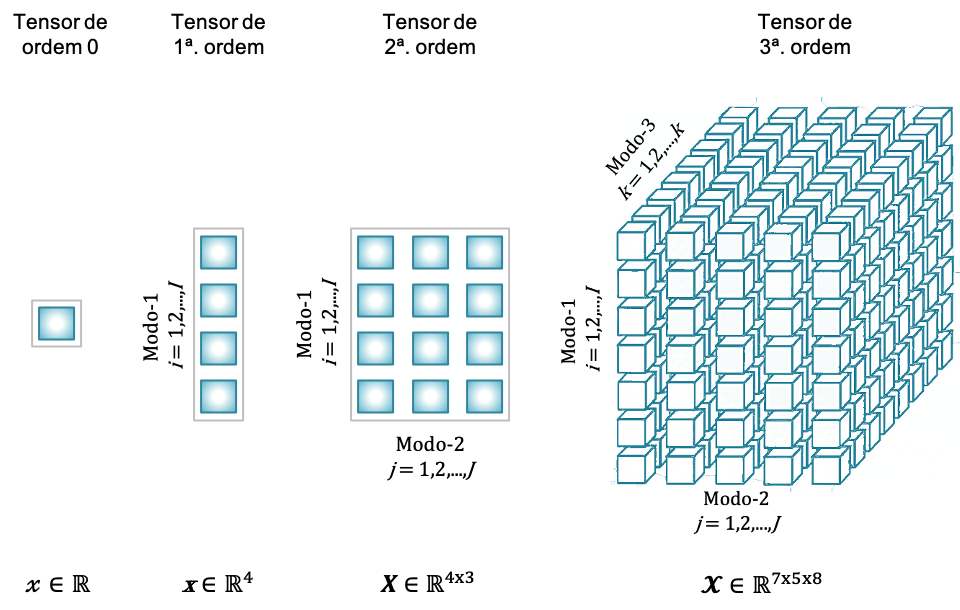


Figura 2 - Escalar, Vetor, Matriz e Tensor

Tem aplicação clássica em campos da física e matemática como, por exemplo, estudos sobre tensão mecânica e gravidade, em virtude de sua expressividade em modelar fenômenos observados de forma multidimensional. Na área de computação, especialmente nesta década, a aplicação do uso de tensores se mostrou muito eficiente, denotando a maturidade da sua aplicação em áreas como, por exemplo: processamento de sinais e visão computacional (NION et al., 2010); processamento de linguagem natural (CHANG; YIH; MEEK, 2013); cidades inteligentes (WANG; ZHENG; XUE, 2014); análise de dados em redes em redes sociais (JIANG et al., 2014); recuperação de informação na web por meio de mecanismos de buscas (AGRAWAL; GOLSHAN; PAPALEXAKIS, 2015); e sistemas de recomendação (PANTRAKI; KOTROPOULOS, 2015). Apoiado por essas ferramentas computacionais, há também aplicação na área de medicina, como por exemplo, em análises de exames de sinais cerebrais (ACAR et al., 2007) (PAPALEXAKIS et al., 2014).

Uma importante operação realizada com tensores são as decomposições. Essas operações favorecem análises multidimensionais oferecendo tratamento, por exemplo, para lidar com a esparcidade (ausência de medidas) e revelar relações latentes entre os entre variáveis (múltiplas relações), além de padrões de comportamento dos dados. As duas decomposições mais conhecidas e utilizadas são a *Canonical Polyadic* (CP) e a decomposição de Tucker (Papalexis et al. 2016).

Como resultado, a decomposição de tensores, tipicamente utilizada para tratar sua ordem, e por consequência lidar com alta dimensionalidade, se mostrou satisfatoriamente apropriada nesses contextos de aplicação, uma vez que auxiliou revelar relações latentes entre as variáveis, além de padrões de comportamento dos dados, de forma a apoiar "*insights*". (PAPALEXAKIS; FALOUTSOS; SIDIROPOULOS, 2016). Além disso, existem eficientes operações para lidar com a esparcidade e otimização de dados como os apresentados no *survey* conduzido por (KOLDA; BADER, 2009). Essa gama de aplicação e uso de tensores se deve sua habilidade em representar e modelar dados multimodais ou multiaspectos.

Seguem algumas definições clássicas sobre tensores, suas propriedades e aplicações, que são importantes para compreensão do modelo proposto:

**Tensor:** um tensor é formalmente definido por: .

**Índices em Tensores:** dado um tensor , podemos obter submatrizes a partir da fixação de um subconjunto de seus índices por meio das operações de fibras(*slices*) e fatias. Uma fibra é obtida quando todos os índices, exceto um, são fixados. Já as fatias, são obtidas quando todos os índices, exceto dois, são fixados. Um exemplo de fatias e fibras de um tensor de ordem 3 é apresentado na Figura 3

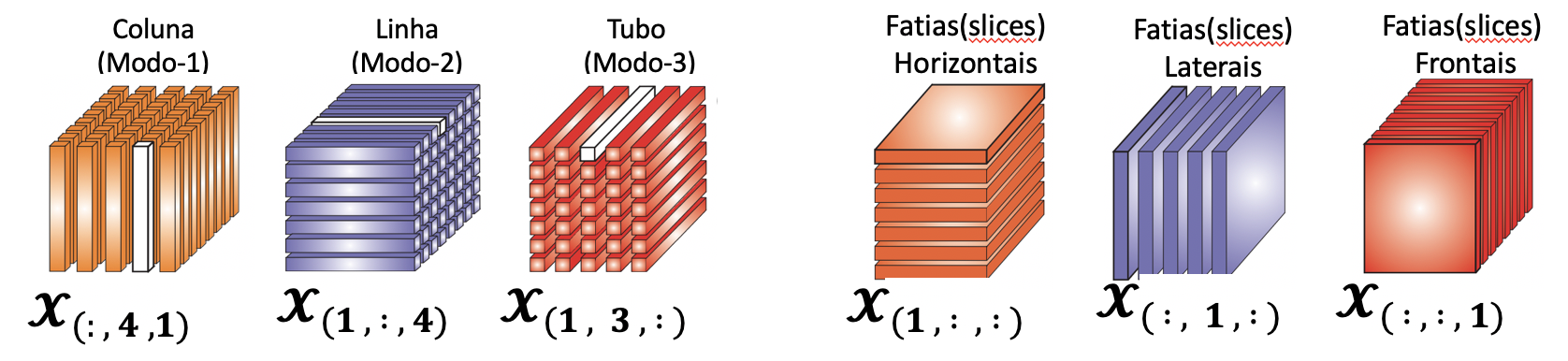


Figura 3: Fatias e Fibras em um tensor de 3ª. Ordem - figura adaptada de (CICHOCKI et al., 2009)

**Produto Interno e Externo:** O produto tensorial conforme apresentado e representado por Lima (1965) permite que façamos operações matemáticas de soma e produto em tensores e é assim definido, , onde

é um espaço vetorial e :é uma aplicação bilinear de par em

dim = dimdim

() gera , isto é, todo elemento de pode ser obtido como combinação linear e, portanto, soma de elementos ()

Substituindo () por , temos:

O caso linear pode ser estendido para *p*-linear, generalizando sua aplicação. Existe uma aplicação bilinear canônica expressa por . Essa aplicação é chamada de produto entre tensores.

O produto interno de dois tensores de mesma ordem é obtido pela soma dos produtos de seus elementos e gera um resultado escalar:

Já o produto externo de dois tensores de mesma ordem é obtido pelo produto de seus elementos e, diferentemente do produto interno, gera uma matriz . Generalizando,

**Tensores de Grau-1(rank-1)**: dizemos que é um tensor de classificação-1, quando este pode ser decomposto no produto externo de vetores. Uma matriz de classificação-1 pode, portanto, ser escrita como . Já um tensor de ordem 3 e classificação-1 é representado como . A generalização é descrita pela com a generalização do produto externo.

**Classificação de Tensores:** a classificação de um tensor é descrita como o número mínimo de tensores de grau-1 necessários para gerar como sua soma, e é assim generalizada:

**Simetria entre Tensores:** um tensor é chamado de cúbico se possuir todos os modos do mesmo tamanho. Um tensor de ordem 3 é simétrico se, e somente se:

, para todo

Um tensor pode ser parcialmente simétrico. Por exemplo, dizemos que um tensor é simétrico nos modos um e dois se todas as suas fatias frontais são simétricas.

, para todo

**Vetorização de tensores:** um tensor pode ser transformado em vetor reordenando seus elementos verticalmente.

**Matricização de tensores:** analogamente a vetorização, podemos reordenar os elementos de um tensor e transformá-lo em uma matriz. A matricização modo-*n* ou desdobramento(*unfolding*), de um tensor denota-se que, Na matricização as fibras modo-*n* de são reordenadas em colunas de .

Seja um elemento do tensor e um elemento de um tensor seja um elemento do tensor desdobrado. A matricização modo-*n* é assim definida:

De acordo com o exemplo apresentado por Kolda e Bader (2009) , seja um tensor de ordem 3 com as seguintes fatias frontais:

|  |  |
| --- | --- |
|  |  |

Então as três matrizes modo-n geradas a partir da matricização de são:

**Produto Kronecker:** o produto Kronecker de duas matrizes é representado por . O resultado será uma matriz de tamanho assim definida:

**Produto Khatri-Rao:** o produto Khatri-Rao de duas matrizes é representado por . O resultado será uma matriz de tamanho , assim definido:

**Norma Khatri-Rao:** a norma de Frobenius de um tensor é definida como:

## A Estrutura do Modelo de Qualidade

Conforme apresentado no Capítulo 2, pudemos perceber que o conhecimento acumulado a partir das pesquisas relacionadas à qualidade de produto de software propiciou a identificação de diferentes fatores e subfatores que permitem descrevermos e observarmos esse fenômeno da qualidade. Esses fatores e subfatores foram organizados e agrupados por diferentes modelos de qualidade propostos ao longo do tempo, estabelecendo com isso, a definição de uma estrutura hierárquica de características e subcaracterísticas que descrevem as propriedades observadas do fenômeno da qualidade de produto. Também pudemos constatar que os modelos de qualidade de produto propostos mais recentemente endereçaram a operacionalização entre essas características e subcaracterísticas, ditas como conceituais e abstratas, a um conjunto de métricas e medidas de forma a tornar as análises mais concretas e objetivas. Ademais, também pudemos perceber que historicamente a qualidade de produto de software foi observada da forma unidimensional.

O modelo de qualidade aqui proposto obedece a estrutura hierárquica de características e subcaracterísticas dos modelos de referência da literatura. Além disso, se apoia no mecanismo de operacionalização entre essas características e suas respectivas medidas e métricas, utilizado nos modelos recentemente propostos. Contudo, propomos o uso de tensores, como ferramenta matemática, para modelar e analisar o fenômeno da qualidade de produto de software de forma multidimensional. Na Tabela 2 apresentamos algumas definições iniciais do modelo proposto que serão necessárias para a compreensão das próximas seções:

|  |  |
| --- | --- |
|  | i-ésima configuração de qualidade de software. Representa a modelagem das características de qualidade de software definidas nos modelos de referência, segundo a literatura, modeladas em um tensor de qualidade. Corresponde às medidas observadas em uma versão de produto de software. |
|  | i-ésimo tensor de qualidade. Representa uma determinada característica de qualidade definida em . Captura o conjunto de medidas de todas as subcaracterísticas que descrevem uma característica observada. |
|  | i-ésima dimensão de um tensor de qualidade. Representa o conjunto de medidas de uma subcaracterística de qualidade. Essas medidas são chamadas de medidas avaliadas. |
|  | i-ésima medida avaliada. Trata-se da medida escalar que é a representação numérica, tratada e interpretada de um conjunto de métricas de software. Captura o julgamento de quão melhor ou pior foi a aferição da qualidade. |
|  | i-ésima função de interpretação. Trata-se da fórmula definida que fornece significado a uma . É utilizada para normalizar e interpretar um conjunto de métricas de software. |
|  | i-ésima métrica de qualidade de software. Dado não tratado, extraído de uma fonte de informação. (ex: LOC, CBO, issues abertas, defeitos críticos, uso de uma funcionalidade, etc) |

Tabela 3 : notações matemáticas do MeasureSoftGram

No MeasureSoftGram, uma configuração da qualidade, , corresponde a um ponto de medição e avaliação de uma determinada versão do software, em um determinado período de tempo do seu ciclo de vida. Cada , é composta por um conjunto de tensores, chamados de tensor de característica de qualidade, , que representa uma determinada característica de qualidade, definida em um modelo de referência descrito na literatura. Nesses tensores, cada dimensão ou modo, representa uma determinada subcaracterística de qualidade relacionada a uma característica de qualidade específica. As subcaracterísticas, por sua vez, são representadas numericamente por um conjunto de medidas escalares, chamadas de medidas avaliadas, .

Cada medida avaliada descreve uma função, chamada de interpretação. Uma função de interpretação representa a aplicação de uma função matemática, , que mapeia como uma determinada medida avaliada é definida, . Ao se definir uma função de interpretação, um conjunto de métricas de software é selecionado, de forma a prover um significado de interpretação, ou seja, a regra de interpretação de uma medida avaliada. Além disso, as medidas avaliadas são padronizadas estatisticamente, o que significa que as métricas coletadas em diferentes escalas e em diferentes fontes de informação são padronizadas para uma mesma escala, no caso, escala razão. Essa padronização viabiliza a comparação proporcional de medidas em produtos de software de diferentes tamanhos. Na Figura 4 podemos observar a estrutura conceitual do MeasureSoftGram.

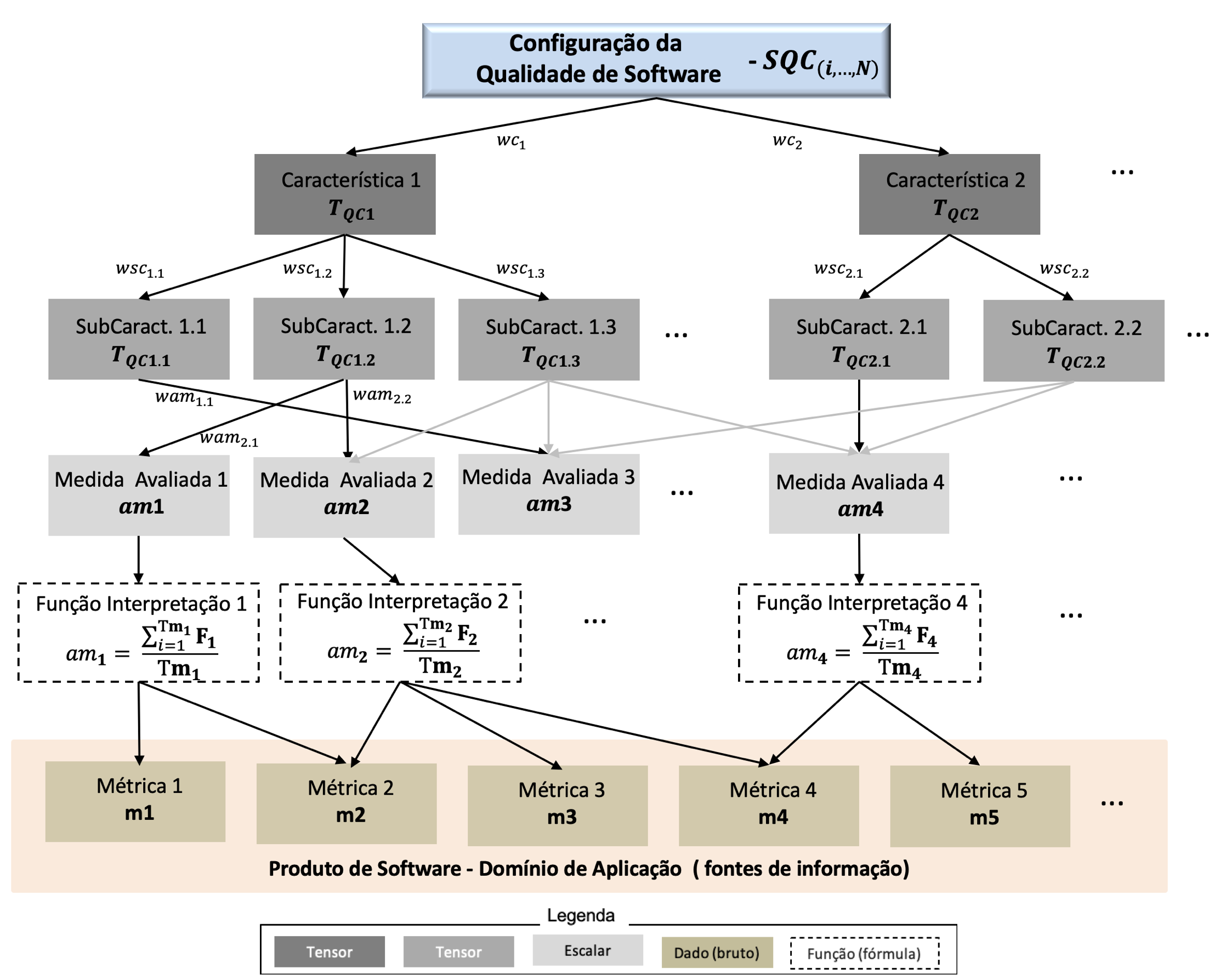


Figura 4: Estrutura Geral do MeasureSoftGram

Tomemos como exemplo a definição de uma configuração de qualidade, descrita pelo tensor de qualidade de Facilidade de Manutenção. Este tensor por sua vez, é de ordem 3, ou seja, possui três dimensões que representam as subcaracterísticas que descrevem, ou, ajuda a perceber, observar, a característica de Facilidade de Manutenção, quer seja: . Sendo que:

representa a subcaracterística de Modificabilidade;

representa a subcaracterística de Testabilidade e;

representa a subcaracterística de Dívida Técnica

A característica de qualidade Facilidade de Manutenção e as subcaracterísticas Modificabilidade e Testabilidade foram extraídas da norma ISO 25010 (ISO/IEC, 2010). Já a subcaracterística Dívida Técnica, foi extraída do modelo de qualidade SQUALE (MORDAL-MANET et al., 2009). Nesta configuração de qualidade a dívida técnica também é percebida como um *surrogate* para aspectos que interferem na modificabilidade do software. Neste exemplo podemos observar a flexibilidade do modelo proposto, pois podemos definir uma configuração de qualidade de acordo com a necessidade de medição de cada organização ou projeto. A Figura 5 ilustra um exemplo da descrição do tensor de qualidade **.**

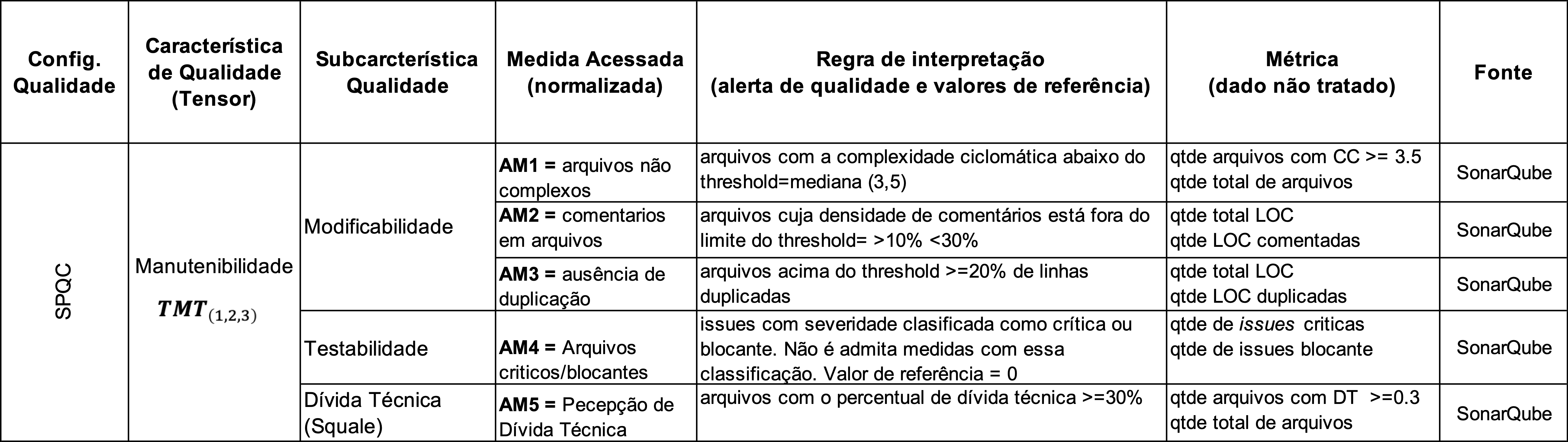


Figura 5 : Exemplo da Descrição de um Tensor de Qualidade

Ainda de acordo com o apresentado na Figura 4, há a descrição das medidas avaliadas, , bem como suas respectivas regras de interpretação, e métricas, **,** que auxiliam capturar o comportamento e dar significado à subcaracterística observada. Portanto, as métricas são extraídas de uma fonte de informação, no exemplo, a ferramenta SonarCube. Em seguida, passam por um processo de tratamento e transformação onde as funções de interpretação são calculadas segundo suas regras de interpretação, de forma a gerar as medidas avaliadas. Neste exemplo, a função de interpretação , que define a medida avaliada é assim definida: . Onde: qtde arquivos cuja complexidade ciclomática, CC, seja maior ou igual à 3,5 e; qtde total de arquivos. Dessa maneira, ao obtermos temos uma medida que captura a proporcionalidade, ou seja, a importância relativa de frente a **,** que representa o total observado dessa métrica. Portanto, uma medida avaliada é uma medida indireta, normalizada, em escala razão, que assume qualquer valor possível entre 0 e 1. Com esse tratamento, as medidas operadas pelo modelo estarão na mesma escala e, do ponto de vista da teoria de medição, passam a ser passíveis de comparação. A partir desse momento os dados estão prontos para serem carregados e modelados nos tensores, de forma que as operações sobre os dados tratados ocorrerão no espaço tensorial.

Esse mecanismo de operacionalização das métricas e medidas avaliadas que, tipicamente caracterizam as etapas de Extração, Transformação e Carga-ETL, não é proposta nesta pesquisa de doutorado. O MeasureSoftGram captura a essência conceitual e é inspirado no metamodelo Quamoco. Portanto, o mecanismo aqui utilizado é equivalente ao definido e validado experimentalmente em (WAGNER et al., 2012) e também utilizado na definição e validação do modelo Q-Rapids (LÓPEZ et al., 2018) e (CHORAS et al., 2020). Logo, o mecanismo utilizado no MeasureSoftGram, se apoia em evidências dos modelos de qualidade recentemente propostos e validados experimentalmente.

O que nos diferencia fundamentalmente desses modelos recentes e dos clássicos, estudados e utilizados na análise de produtos de software, segundo a literatura, é que, em nossa modelagem, análises e operações se dão no espaço tensorial, portanto, multidimensional.

### Definição de métricas e medidas

Comumente a 1ª pergunta que vem à mente em gestores e engenheiros de software é: Quais são as métricas que devemos monitorar ao longo do ciclo de vida de um produto de software? E possivelmente, a resposta mais apropriada é: depende! E sim, de fato depende de diferentes questões que perpassam desde dos objetivos estratégicos de uma organização(BUSE; ZIMMERMANN, 2012), aos objetivos de medição(BASILI; CALDIERA; ROMBACH, 1994), ao paradigma e linguagem de programação (LANZA; MARINESCU, 2010) ou tecnologias utilizadas. Portanto, ao se definir um plano de medição, as diferentes informações de contexto devem ser tratadas e observadas de acordo com a necessidade de cada interessado na informação (SJØBERG; CRUZES, 2012).

Para iniciar a operacionalização do MeasureSoftGram é necessário que sejam definidas quais as métricas de qualidade de software serão extraídas e, principalmente, o porque tais métricas serão escolhidas. Toda medição deve estar orientada a um objetivo (BASILI; CALDIERA; ROMBACH, 1994) além, de possuir uma clara necessidade de informação (BUSE; ZIMMERMANN, 2012).

As métricas são um meio para que disponhamos de informações do software, viabilizando com isso, análises e avaliações de suas características (LIMA et al., 2018). Trata-se do valor numérico, um dado não tratado, extraído de uma fonte de informação. Já uma medida, pode ser entendida como número ou um símbolo atribuído a uma entidade a fim de caracterizar um atributo (FENTON; BIEMAN, 2014). Portanto, a medida fornece significado de interpretação a uma métrica. Embora não haja uma definição única (MENEELY; SMITH; WILLIAMS, 2013), métricas e medidas estão frequentemente associadas à avaliação de características do processo de desenvolvimento ou do produto de software em si, seja observando aspectos qualitativos ou quantitativos, como forma de perceber e entender o fenômeno do software. Não faz sentido observar uma métrica, sem que seu contexto, a forma de interpretar, a validação teórica ou experimental, sejam minimamente conhecidas. Neste modelo métricas e medidas são assim definidas e representadas:

**Definição:** uma métrica de qualidade de software é um dado não tratado, extraído de uma fonte de informação e que qualifica uma medida avaliada. Ex: (ex: LOC, CBO, Complexidade Ciclomática, issues abertas, número de defeitos críticos, tempo de uso de uma funcionalidade, quantidade de cliques para realizar uma ação, etc).

**Definição:** uma medida avaliada é o dado tratado e interpretado. Essa medida é a representação numérica de uma determinada subcaracterística de qualidade de software. É uma medida indireta, normalizada e definida em escala razão. Essa medida fornece significado, ou seja, uma interpretação a um conjunto de métricas.

e

onde:

= número de ocorrências da amostra (ex: arquivos, classes, funções, issues no repositório, uso de uma funcionalidade, defeitos, etc.)

= total absoluto da métrica de qualidade de software observada na amostra.

= função de interpretação e qualificação de uma medida (definida em 3.2.2)

Pudemos observar nas diferentes discussões sobre métricas e medição apresentadas na seção 2.2, que a definição e escolha de métricas e medidas para representar características associadas ao software vem sendo estudadas ao longo das últimas 5 décadas. Diferentes estudos secundários procuraram mapear as métricas mais estudadas e utilizadas na indústria (KITCHENHAM, 2010) (TAHIR; MACDONELL, 2012) (ELBERZHAGER; MÜNCH; NHA, 2012). Já outros estudos consolidaram compêndios que também mapearam uma diversidade de métricas (LANZA; MARINESCU, 2010) (FENTON; BIEMAN, 2014).

No metamodelo Quamoco, a partir de projetos de software livre, foram analisados 194 fatores relacionados a 526 medidas, predominantemente extraídas de ferramentas de análise estática de código-fonte, em um estudo conjunto entre academia e indústria. Cada uma dessas medidas foi revisada por dois especialistas que decidiram sobre a normalização e interpretação mais apropriada. Esses fatores e medidas foram associados a caraterísticas da norma ISO 25010. Os resultados obtidos pelo modelo foram comparados com a opinião de profissionais da prática, especialistas em qualidade, de forma que houve uma percepção que o modelo contribuiu e auxiliou à observação da qualidade de produto. Trata-se de um estudo conjunto entre a academia e a indústria (WAGNER et al., 2012). O Q-Rapdis, baseado no Quamoco, também apresenta um conjunto de métricas e de medidas utilizadas no monitoramento e avaliação de características de qualidade de produto e processo em ciclos de desenvolvimento contínuo (LÓPEZ et al., 2018) (CHORAS et al., 2020)

O amadurecimento das discussões e uso de práticas de desenvolvimento ágil, de software livre e da mentalidade do movimento Lean, contribuiu para o estabelecimento de uma visão convergente sobre cadência do ciclo de desenvolvimento. A atividades do processo de desenvolvimento passam a ser realizadas de forma contínua, com o objetivo da liberação de versões do produto de software em intervalos de tempo curtos e frequentes (FITZGERALD; STOL, 2017). Considerando essa visão, aproximadamente a partir da última década, pudemos observar o florescer dos estudos do uso da abordagem de experimentação contínua para apoiar a tomada de decisão acerca de versões de produto de software. Interessante notar que essa estratégia é focada na observação de características a partir do uso do software. Consequentemente, as medidas e métricas capturadas são predominantemente relacionadas a aspectos da qualidade em uso (KOHAVI et al., 2013) (LINDGREN; MÜNCH, 2015) (KEVIC et al., 2017) (FABIJAN et al., 2018).

Existem também associações compostas por representantes da indústria e academia que reúnem esforços e colaboram entre si, como, por exemplo, a Common Weakness Enumeration – CWE[[1]](#footnote-3), que desde 2006, publicam anualmente uma lista com características relacionadas à segurança. São apresentadas as vulnerabilidades de software mais comuns observadas por esse consórcio, bem como as métricas utilizadas para quantificá-las.

Existem também ferramentas de apoio como por exemplo, o SonarQube[[2]](#footnote-4), com ampla aceitação e uso por parte da indústria e também utilizada em diferentes estudos científicos (MORDAL-MANET et al., 2009) (LÓPEZ et al., 2018). Por meio dessa ferramenta de análise estática de código-fonte e de resultados da execução de testes de software é possível extrair e minerar métricas relacionadas a diferentes características e subcaracterísticas de atributos de qualidade interna e externa como por exemplo, a implementação do modelo SQUALE, que fornece métricas quantitativas para capturar a percepção de dívida técnica. Essa ferramenta é distribuída sob licença de software livre, multiplataforma e fornece métricas para aproximadamente 30 linguagens de programação diferentes.

Por fim, há um vasto corpo de conhecimento teórico e experimental sobre medição, métricas e medidas, acumulado ao longo do tempo. Portanto, nesta pesquisa de tese de doutorado nos apoiaremos nas evidências observadas quando da definição de um conjunto de métricas e as características do produto que elas quantificam. Por conseguinte, não é escopo deste trabalho definir uma suíte de métricas a serem monitoradas. Tanto a escolha das métricas, como a forma de interpretá-las, dependem essencialmente da participação de especialistas em conjunto com o time de desenvolvedores. Trata-se de uma atividade de intensa de participação humana!

### Funções de Interpretação e Valores de Referência

A partir do momento que as métricas foram criteriosamente escolhidas é necessário que haja um mecanismo capaz de quantificar e fornecer o julgamento, em essência subjetivo, de quão bom ou ruim podem ser interpretados os resultados observados nas métricas. Esse comportamento é provido por uma função de interpretação. Ela pode ser definida a partir de opinião de especialistas e/ou de forma automática (LÓPEZ et al., 2018).

**Definição:** uma função de interpretação define e normaliza uma medida avaliada. Ela faz uso da função de interpolação linear entre o valor calculados a partir de uma métrica Os valores assumidos por uma função de interpretação correspondem a um intervalo aberto entre 0 e 1.

onde:

= resultado obtido através da interpolação linear entre os valores de referência de uma métrica e o valor calculado de uma medida avaliada a partir a partir de sua regra de interpretação,

0 = ausência de qualidade (pior)

1 = qualidade máxima (melhor)

**Consideremos** o exemplo de uma versão de produto onde a noção da complexidade ciclomática (CC) é observada por meio de uma medida avaliada chamada *arquivos não complexos*, assim definida:

onde:

total de arquivos do produto analisados

CC de um arquivo

quantidade de funções de um arquivo

A regra de interpretação de é definida como aqueles arquivos que possuam CC . Esse valor de referência representa a mediana de e foi definido após a análise da distribuição. Na Figura 6 podemos observar um exemplo de aplicação da função de interpretação. Os valores de FI são calculados a partir da interpolação linear de em .

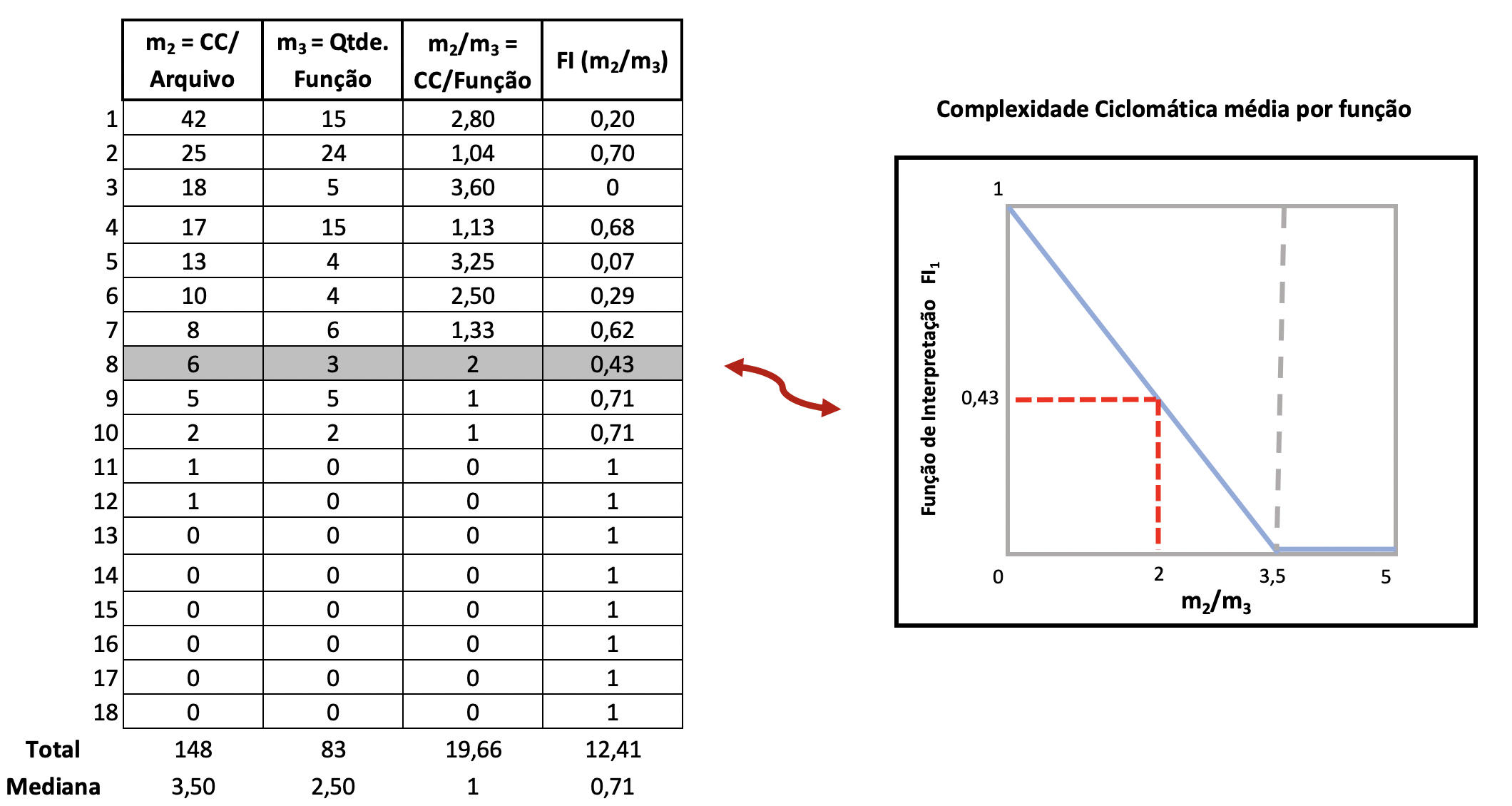


Figura 6: Exemplo da aplicação uma função de interpretação

Logo:

Uma questão determinante na interpretação de uma função de interpretação é a definição dos valores de referência das métricas e medidas, sem os quais, essa atividade se torna inexequível. Trata-se de uma questão muito sensível e não trivial na engenharia de software, uma vez que a natureza lógica e intangível do produto de software impossibilita que sejam utilizados parâmetros definidos e apoiados pelas leis da física, como é comum, por exemplo, nas áreas de engenharia e medicina.

Analisando as estratégias de definição de valores de referência utilizadas pelos modelos de qualidade recentemente propostos, percebemos a aplicação de duas abordagens de tratamento para definição desses parâmetros: i) baseada na opinião de especialistas em qualidade e, ii) obtidas de forma automática a partir da realização de *benchmarking* em repositórios de produtos de software livre. Nos estudos que analisamos, essas abordagens foram utilizadas sozinhas ou combinadas.

Os métodos para derivação automática de valores de referência propostos na literatura consideram a análise da frequência de valores observados na distribuição estatística das métricas e medidas. Alguns métodos inicialmente propostos para observar métricas no paradigma orientado(OO) a objetos sugerem que os valores de referência sejam obtidos analisando a frequência média de valores observados em uma distribuição. De acordo com Erni e Lewerentz (1996), a média-μ e o desvio padrão-σ devem considerados como referência de valores mínimo e máximo onde, a μ - σ é utilizada com valor mínimo de referência e a μ + σ como valor de referência máximo. Já Marinescu e Ratiu (2004) e Lanza e Marinescu (2010) propuseram uma estratégia de detecção de anomalias em código-fonte(*bad smells*) e também utilizaram como base os valores da média e desvio padrão, onde μ - σ é utilizada como valor de referência baixo, μ + σ como o alto e 1,5 (μ + σ) como o valor de referência “muito alto”. Esses métodos assumem a premissa de que as distribuições de métricas de código-fonte assumem um comportamento da distribuição normal (LAVAZZA; MORASCA, 2016). No entanto, as distribuições de métricas de software, em especial as de código-fonte, dificilmente assumem comportamento da distribuição normal e, portanto, outros pontos de corte e ajuste da distribuição devem ser considerados.

Ainda levando em conta as estratégias de derivação automática, alguns estudos evidenciam que a análise de quartis e percentis apresentam maior acurácia em comparação ao uso da média e desvio padrão e, portanto, são mais adequados para estratégias de *benchmarking* (WAGNER et al., 2012) (LOCHMANN, 2012) (MEIRELLES, 2013) (VALE; FIGUEIREDO, 2015) (SIAVVAS; CHATZIDIMITRIOU; SYMEONIDIS, 2017) (LIMA et al., 2018).

Wagner et al. (2012) propuseram o uso de uma abordagem interquartil, fazendo um corte na distribuição entre os percentis 25% e 75%, ou seja, descartando os *outliers* e cortes inferiores e superiores ao intervalo. Com isso, cada medida passou a ter dois valores de referência, assim definidos:

onde:

= -percentil

= intervalo interquartil

Lochhann (2012) refinou a proposta Wagner et al. (2012) e sugeriu a inclusão da mediana do intervalo de . Nesse estudo ele observa que o tamanho do produto de software não influencia nos resultados, não apresentando variância significativa em comparação a produtos maiores. Por outro lado, o tamanho da base utilizada para o *benchmarking* influencia. Quanto maior é a base, menor é a variância dos resultados. Assim, passou a considerar três valores de referência para uma métrica ou medida. Esse método também foi utilizado e validado por Siavvas, Chatzidimitriou e Symeonidis (2017):

Embora a abordagem de análise interquartil (25% à 75%) tenha se mostrado mais adequada para definição de valores de referência do que o uso da média e desvio padrão, ela pode não capturar valores representativos para uma métrica. Meirelles (2013) analisou métricas OO observando os percentis (5%, 10%, 25%, 50%, 75%, 90%, 95%, 99%) e propôs a definição de intervalos de valores de referência de acordo com a frequência da distribuição, como: muito frequente, frequente, pouco frequente e não frequente. A conclusão desse estudo mostra que algumas métricas OO só começam apresentar comportamento observável partir do percentil 50% e em alguns casos a partir do percentil 90%, inclusive apresentando um crescimento abrupto. Lima et al. (2018) adaptou a proposta de Meirelles (2013) e observou métricas de anotação em código-fonte considerando os percentis (5%, 10%, 25%, 50%, 75%, 90%, 95%, 99%). O valor muito frequente corresponde à média obtida dos valores observados no percentil 90%. O valor frequente diz respeito à média obtida dos valores observados no percentil 95% e, por fim, o valor menos frequente compreende à média obtida da coluna percentil 99. Nesse estudo o comportamento observado nas distribuições das métricas de anotação em código-fonte foi semelhante ao observado por Meirelles (2013).

Ainda que a abordagem automática de análise dos percentis se mostre adequada para observar métricas em diferentes domínios ela não pode ser generalizada, principalmente se as características de contexto e os objetivos de medição não estiverem claramente definidos. Existem muitos fatores de confusão desconhecidos ao se realizar em *benchmarking* em repositórios de produtos de software.

Wagner et al. (2012) e Siavvas, Chatzidimitriou e Symeonidis (2017) fizeram uso de uma abordagem mista, onde os valores de referência obtidos de forma automática foram submetidos ao julgamento e análise de especialistas. Já López et al. (2018) e Choras et al. (2020) se basearam na opinião de especialistas para essas definições e os instrumentos mais comuns utilizados para capturar e qualificar os valores nessa abordagem mais analítica foram entrevistas, questionários e workshops, envolvendo especialistas em qualidade representantes da indústria, além de pesquisadores.

Por fim, o que concluímos dessas discussões é que o uso combinado de abordagens automáticas e baseada em opinião de especialistas seja o cenário ideal para lidar com a definição de valores de referência para métricas e medidas de produto de software. Caso a organização possua base histórica de dados, esta seria preferível ao uso de repositórios de software livre, uma vez que as características de contexto serão mais semelhantes. Alternativamente, pode-se usar uma base de projetos de software livre na estratégia de *benchmarking.* Porém, a escolha dos projetos deve ser criteriosamente planejada, considerando fatores como, por exemplo: características de contexto; tamanho da base do *benchmarking*; a percepção da qualidade observada a partir métricas; objetivos de medição; além, da diversidade dos projetos (NAGAPPAN; ZIMMERMANN; BIRD, 2013).

O passo seguinte é agregar as medidas avaliadas em suas respectivas subcaracterísticas de qualidade.

### Agregação e Ponderação

A agregação do modelo é precedida da ponderação das diferentes características e subcaracterísticas observadas. Os pesos representam a importância relativa de cada medida avaliada em sua respectiva subcaracterística, bem como das subcaracterísticas em relação à sua respectiva característica, além da característica frente ao sistema como todo.

Diferentemente da definição dos valores de referência, os pesos devem ser definidos por especialistas. Isso se deve ao fato de que a percepção de qualidade é muito subjetiva e também muito dependente do contexto. A definição de como uma medida ou característica influencia a qualidade geral do sistema depende de como engenheiros de software, gestores e usuários percebem a noção de qualidade. Por consequência, a definição dos pesos também carrega esse aspecto subjetivo. (SIAVVAS; CHATZIDIMITRIOU; SYMEONIDIS, 2017). Assim sendo, essa etapa de calibração é de suma importância e deve ser executada meticulosamente.

**Definição:** uma configuração da qualidade de software é representada por tensor, que modela e representa um conjunto de tensores associados a características que descrevem a qualidade do produto de software. Cada dimensão ou eixo do tensor representa uma determinada característica estabelecida em modelos de referência. É definida como:

onde:

= número de dimensões de uma configuração de qualidade. Cada define uma dimensão do tensor. Logo, corresponde à i-ésima dimensão de uma configuração de qualidade. Retrata o conjunto de medidas observadas em uma característica de qualidade.

**Definição:** um tensor qualidade, , modela e representa uma característica de qualidade do produto de software estabelecida em modelos de referência. Cada dimensão ou eixo do tensor representa uma determinada subcaracterística e é assim definido:

onde:

= número de dimensões de um tensor qualidade. Cada define uma dimensão do tensor. Logo, corresponde à i-ésima dimensão de um tensor de qualidade. Simboliza o conjunto de medidas observadas em uma subcaracterística de qualidade.

**Definição:** sejam, , e os pesos respectivamente associados a uma medida avaliada , a uma dimensão de um tensor de qualidade e a uma dimensão de uma configuração de qualidade . No nível das medidas é procedida a multiplicação direta entre a medida e seu peso. No nível das subcaracterísticas, por meio da operação de fatia (*slice*), cada uma das dimensões de um tensor é recuperada e então, é realizado o produto Kronecker por seu respectivo peso. Por fim, no nível das características, cada dimensão de uma configuração de qualidade é recuperada realizando uma fatia e em seguida, operado o produto Kronecker por seu respectivo peso

onde:

Uma vez que os elementos do modelo foram devidamente ponderados então os resultados podem ser agregados e analisados de forma sumariada. Como estamos lidando com uma modelagem multidimensional, escolhemos a função da norma de um tensor como mecanismo de agregação. Essa estratégia difere dos modelos de qualidade propostos até o presente momento, que realizam a agregação com a aplicação de uma média ponderada.

**Definição:** a agregação dos elementos modelo é definida pela norma Frobenius. Deste modo, podemos obter um valor agregado em qualquer nível da hierarquia do modelo por meio da aplicação da função norma. A norma Frobenius de uma configuração de qualidade e de um tensor de qualidade são assim definidas:

Os valores agregados per si, não são o aspecto mais relevante do modelo. O mais importante é que ele capture e represente a noção subjetiva da qualidade do software no espaço multidimensional, de forma a perceber as variações ocorridas ao longo do tempo do ciclo de vida do produto. Apresentamos uma abstração da visão geral do comportamento do modelo na Figura 7.

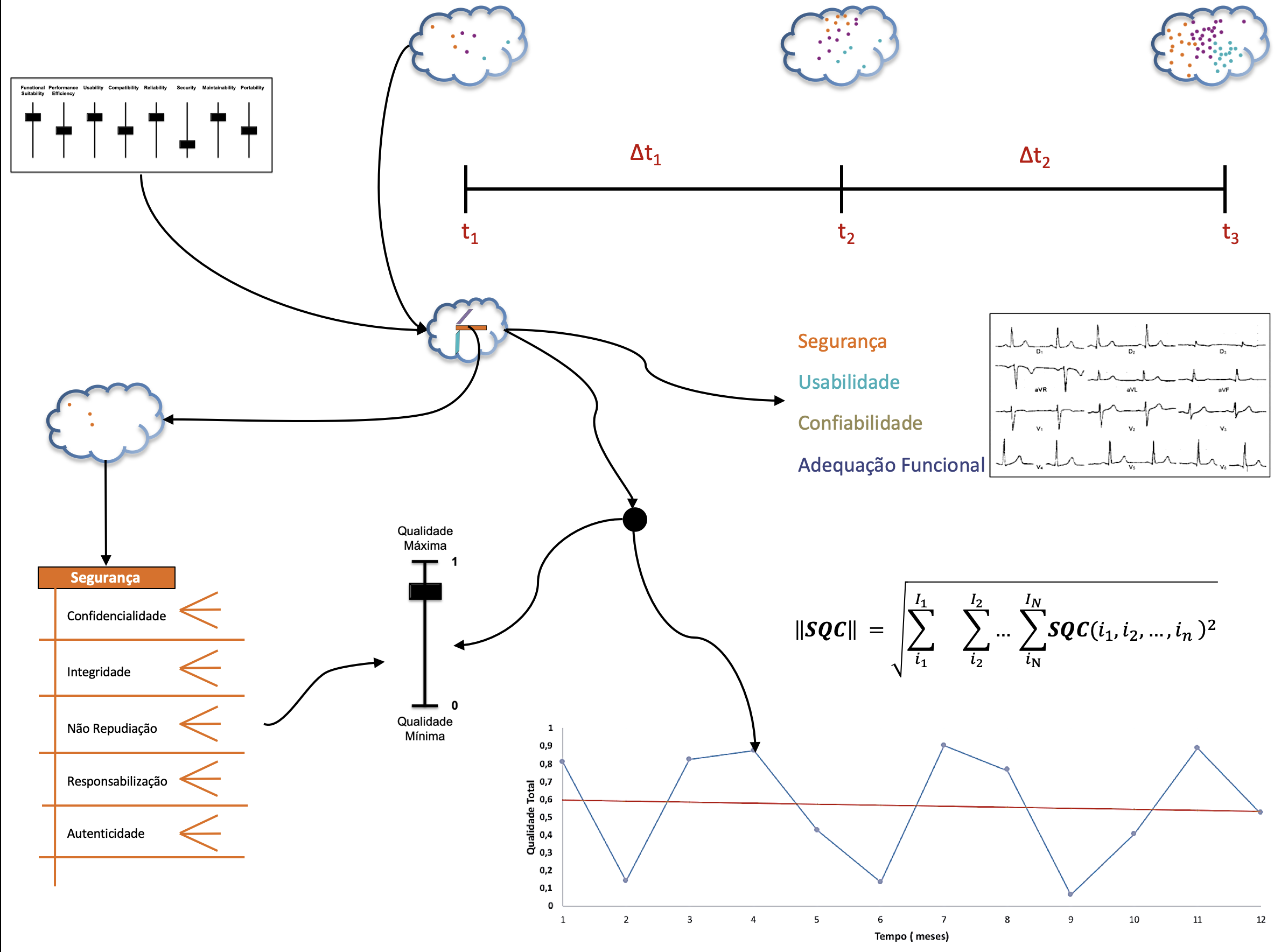


Figura 7: Percepção da qualidade ao longo do tempo

### A comparação entre configurações de qualidade de software

A atividade de comparação entre diferentes versões de produto de software faz parte do dia a dia do desenvolvimento. Ela ocorre em diferentes granularidades, como por exemplo: a) na comparação de versões para resolução de conflitos de *merge*; b) comparação de versões desenvolvidas no passado que permitam observar o comportamento da qualidade ao longo do tempo, de forma a identificar tendências e viabilizar predições e projeções para versões futuras; c) apoiar decisões de aceitação de versões de produto.

Em especial em ambientes de desenvolvimento e experimentação contínua, a decisão e escolha entre uma versão candidata A e uma versão candidata B se dá pelos resultados obtidos da execução de experimentos controlados. Logo, é imprescindível que engenheiros de software e gestores tenham mecanismos que os auxiliem a fazer comparações e apoiarem a tomada de decisão orientada a dados e baseada em evidência.

Em se tratando de um modelo multidimensional, onde as medidas da qualidade medidas são representadas em espaços tensoriais, sabemos que estes podem ser transformados em vetores, ou seja, serem reduzidos a um tensor de 1ª. ordem. Podemos então, representar esses tensores muldimensionais do espaço em vetores bidimensionais, por conseguinte, no plano Euclidiano[[3]](#footnote-5), no espaço . A partir dessa representação, podemos utilizar aplicações trigonométricas de forma a caracterizar e quantificar de forma objetiva a atividade de comparação de diferentes versões de produto de software. É importante ressaltar que estamos tratando de comparações realizadas ao longo do ciclo de vida de desenvolvimento e evolução de um mesmo produto de software.

Sendo muito comum haver diferenças entre a versão do produto produzida e os objetivos de qualidade planejados, e que essas diferenças, por vezes, podem até mesmo inviabilizar a comparação, faz-se necessário que o modelo preveja e trate tal comportamento. Portanto, no MeasureSoftGram dois tipos de comparações são possíveis, a parcial e a total. Sejam e dois tensores que definem duas configurações de qualidade e que representam duas versões de produtos de software distintas .

**Definição:** dois tensores de qualidade e são parcialmente comparáveis se e somente se:

1. existe pelo menos uma

**Definição:** dois tensores de qualidade e são totalmente comparáveis se e somente se:

1. para qualquer
2. houver a mesma quantidade de dimensões

**Pressuposto:** uma vez que uma característica de qualidade passou a ser medida, esta não poderá mais ser excluída do modelo, em prejuízo às comparações de diferentes versões de um produto de software.

Uma medida de similaridade entre vetores destacada na literatura e muito utilizada em áreas como por exemplo, busca e recuperação de informação, indexação, é a diferença de cosseno entre dois vetores. Nessas áreas de pesquisa, o modelo vetorial é utilizado para representar a relação entre os termos utilizados em uma busca e os documentos que melhor satisfazem os critérios da busca. Nesse contexto, os termos utilizados em uma busca são representados em um vetor e, os documentos que satisfazem a busca são representados por um vetor . Esses documentos são indexados por ordem de relevância de forma que a similaridade entre os vetores é quantificada pelo cosseno da diferença (BAEZA-YATES; RIBEIRO-NETO, 1999).

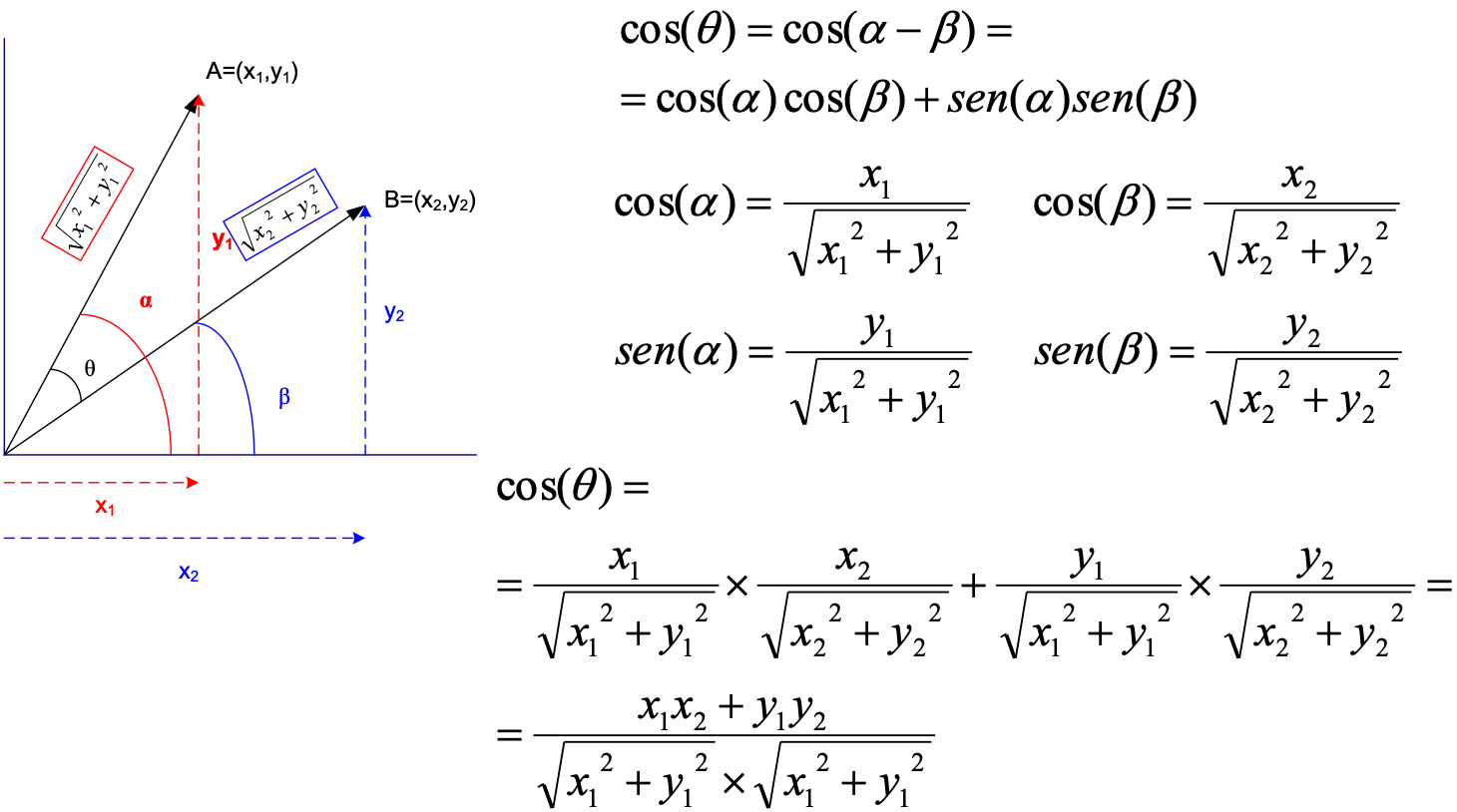


Figura 8: Similaridade entre vetores medida pela diferença de cosseno [[4]](#footnote-6)

Logo:

onde: total de termos indexados

Com isso, a interpretação compreendia é que quanto mais próximo de zero for o cosseno do ângulo mais há similaridade entre os vetores.

No MeasureSoftGram utilizamos a medida de similaridade entre vetores para quantificarmos a comparação entre duas versões distintas de software produzidas. Consequentemente, dizemos que duas versões de produto são iguais se os vetores que representam estas versões possuírem similaridade igual a um.

**Definição:** sejam e dois vetores de qualidade e , definimos se somente se:

Para que essa condição seja alcançada é necessário que os tensores, transformados em vetores, tenham os mesmos tipos de dimensões e que se refiram ao mesmo tipo de elemento, ou seja, representem as mesmas características e subcaracterísticas de qualidade. Significa dizer que os vetores obtidos a partir da transformação dos tensores devem possuir orientações iguais, ou seja, mesma direção, sentido ou módulo, além de não serem ortogonais.

Dizemos que dois vetores e são ortogonais quando o produto interno entre eles é igual a zero, logo: • . O ângulo formado entre esses tipos vetores sempre será , e consequentemente, nunca haverá convergência entre eles. Essa situação se dará quando os tensores não tiverem nenhuma característica ou subcaracterística de qualidade em comum. Assim, não faz sentido e não devem ser comparados.

Como a condição de comparação entre dois tensores definida neste modelo exige que haja ao menos uma dimensão em comum entre eles, logo, parcialmente comparáveis, dizemos então, que o MeasureSoftGram não computa tensores minimamente comparáveis. É dessa maneira que o modelo trata o comportamento da ortogonalidade.

Outro comportamento importante que precisa ser tratado diz respeito à orientação dos orientação dos vetores. Vetores com direção ou sentido opostos não possuem ponto de convergência, ainda que tenham o mesmo módulo. Para tratar esse tipo de comportamento faz-se necessário analisar os sinais das coordenadas. Utilizamos a análise dos sinais de seno e cosseno para observarmos esse comportamento.

De acordo com o quadrante no sistema de coordenadas tridimensional, os valores do seno e cosseno variam e eles podem assumir valores negativos, conforme exemplificado na Figura 9.

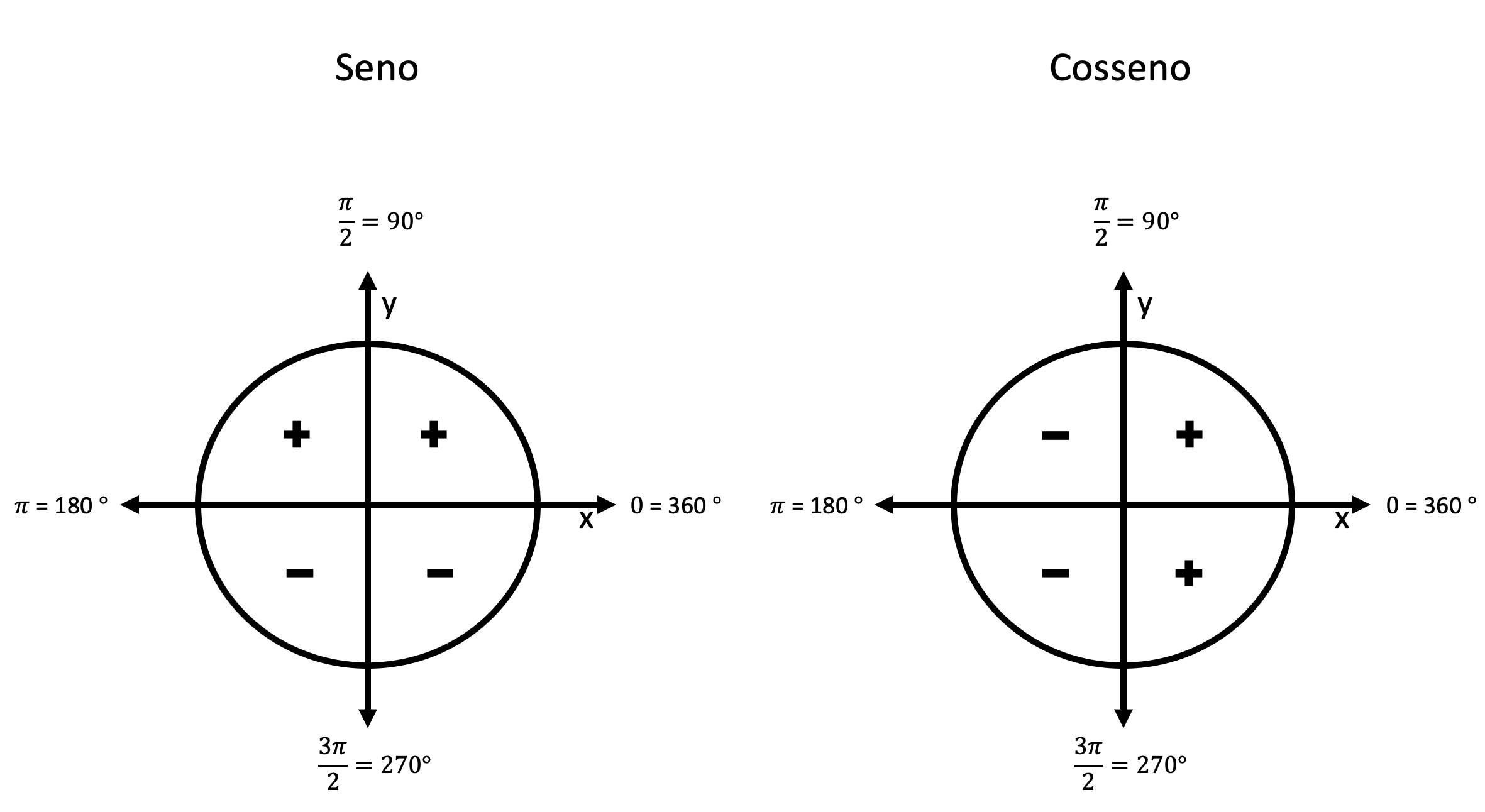


Figura 9: Sinais do seno e cosseno no círculo trigonométrico

Como no MeasureSoftGram os valores assumidos pela qualidade são sempre positivos, entre zero e um, então, os valores do seno e cosseno sempre serão positivos. Isso reduz nosso espaço de análise para vetores projetados no 1º quadrante, o que, simplifica nossas análises e previne essa anomalia no modelo.

Essas propriedades intrínsecas a espaços vetoriais, se não forem observadas e devidamente tratadas, podem levar a anomalias e comportamentos imprevisíveis na percepção dos gestores e engenheiros de software e isso pode fazer com que mudanças radicais e intangíveis sejam erroneamente decididas no ciclo do desenvolvimento do produto de software.

### Capturando e expondo as múltiplas relações entre as características de qualidade

Um ciclo de produção/desenvolvimento deve ser planejado de forma que os objetivos e requisitos desejados sejam definidos ante o balanceamento das necessidades dos usuários do produto e as expectativas de gestores e equipe de desenvolvimento. Ao final do ciclo, o que se deseja é que a qualidade obtida na versão do produto de software produzida esteja em conformidade com os requisitos planejados. Assim, existe a necessidade de se confrontar o resultado gerado ao final do ciclo, em particular a versão de software produzida, frente aos requisitos e objetivos de medição de qualidade planejados para tal.

Porém, as influências mútuas entre as características de qualidade restringem que estabeleçamos objetivos de medição de qualidade máxima para todas características. Em outras palavras, é impossível definir uma configuração de qualidade em que seja considerada a totalidade absoluta(100%) de todas as características de qualidade simultaneamente

Conforme pudemos perceber por meio da revisão da literatura apresentada no capítulo 2, há evidências, em variados estudos, de que as diferentes características de qualidade se influenciam mutuamente. (HENNINGSSON; WOHLIN, 2002) (SVAHNBERG; HENNINGSSON, 2009) (HAOUES et al., 2017). Assim, planejar os objetivos de medição da qualidade do produto sem considerar as mútuas relações existentes entre os diferentes atributos de qualidade pode levar a situações intangíveis durante o desenvolvimento. Essas situações acarretam impactos de esforço, financeiro e de tempo sobre versão produzida.

Numa situação hipotética ideal, o desejado é que a qualidade aferida na versão do produto produzida ao final de um ciclo de desenvolvimento seja igual aos objetivos da qualidade planejada no início do ciclo, embora no mundo real isso seja muito difícil de acontecer. Mas o fato é que, para realizar essa aferição, é necessário comparar a versão de produto produzida com os objetivos de qualidade planejados.

Sejam duas configurações de qualidade e . Seja ainda, o intervalo de tempo compreendido entre a data de início e fim do ciclo de produção de uma versão de produto de software.

Para capturar a expectativa dos requisitos de qualidade planejados de forma quantitativa, além de representar as múltiplas relações entre os atributos de qualidade, utilizamos um mecanismo que chamamos de equalizador de qualidade. A ideia é inspirada na metáfora de um equalizador de frequência sonora, conforme exemplificado na Figura 10.

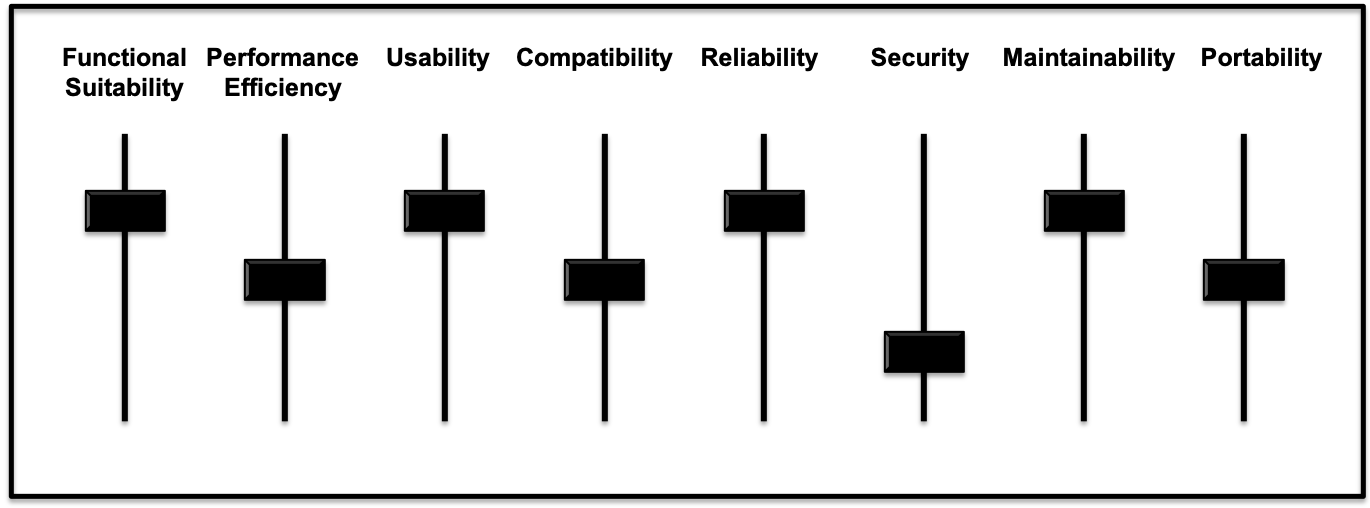


Figura 10: Equalizador da qualidade de software

Para prover esse mecanismo de equalização é necessário que conheçamos as relações entre as características de qualidade. A partir de nossa revisão bibliográfica percebemos que atributos de qualidade podem ter sinergias ou conflitos. Além disso, não encontramos evidências sobre o balanceamento dessas relações mútuas obtidas e observadas a partir da análise de dados de versões de produtos de software. Embora o estudo de Shatnawi (2017) investigue as relações entre atributos de qualidade de forma quantitativa a partir de versões de produtos de software, o contexto foi muito específico, quantificando as relações para cada produto de software analisado, não propondo uma matriz generalizada. Mesmo nos estudos qualitativos, pudemos perceber que há diferenças nas relações entre os atributos de qualidade. De acordo com Berander et al. (2005) isso se deve ao fato de tais relações serem influenciadas por muitos fatores além de suas descrições semânticas serem muito abstratas. As conclusões desse estudo mostram que é viável a construção desse tipo de matriz, porém a generalização dos resultados é limitada. Recomenda-se que a construção desse tipo de matriz considere as características específicas do contexto, ou seja, dentro de uma mesma organização ou em um domínio específico, onde as interpretações e estratégias de medição para atingir os níveis de atributos de qualidade de mais alta abstração sejam semelhantes.

A maior parte dos estudos sobre o balanceamento entre atributos de qualidade utiliza a ISO/IEC 9126 (2001) como referência e isso possivelmente se deve ao fato de a sua atualização, a ISO/IEC 25010 (2010), ser mais recente. O estudo de Haoues et al. (2017) propõe uma matriz baseada na ISO 25010. Após analisarmos os conflitos evidenciados nesse estudo, procuramos resolvê-los ao confrontá-los com os resultados evidenciados por Zulzalin et al. (2008). Com isso, conseguimos mitigar 4 conflitos dos 10 conflitos apontados, conforme apresentado na Tabela 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **(HAOUES et al., 2017)** | | **(ZULZALIL et al., 2008)** | **Mitigação**  **proposta** | **Característica de Qualidade** |
| ¢ | − | − | **−** | Functional Suitability X Performance Efficiency |
| ± | + | + | **+** | Maintainability X Reliability |
| ± | ¢ | ¢ | **¢** | Maintainability X Usability |
| ± | − | − | **−** | Usability X Performance Efficiency |

Tabela 4: Conflitos entre atributos de qualidade mitigados no MeasureSoftGram

Após a mitigação desses 4 conflitos, geramos uma nova matriz que será a referência de calibração dos balanceamentos entre as características da ISO 25010 adotadas por nosso modelo, conforme apresentado na Figura 11. As demais relações seguem a proposta apresentada por Haoues et al. (2017).

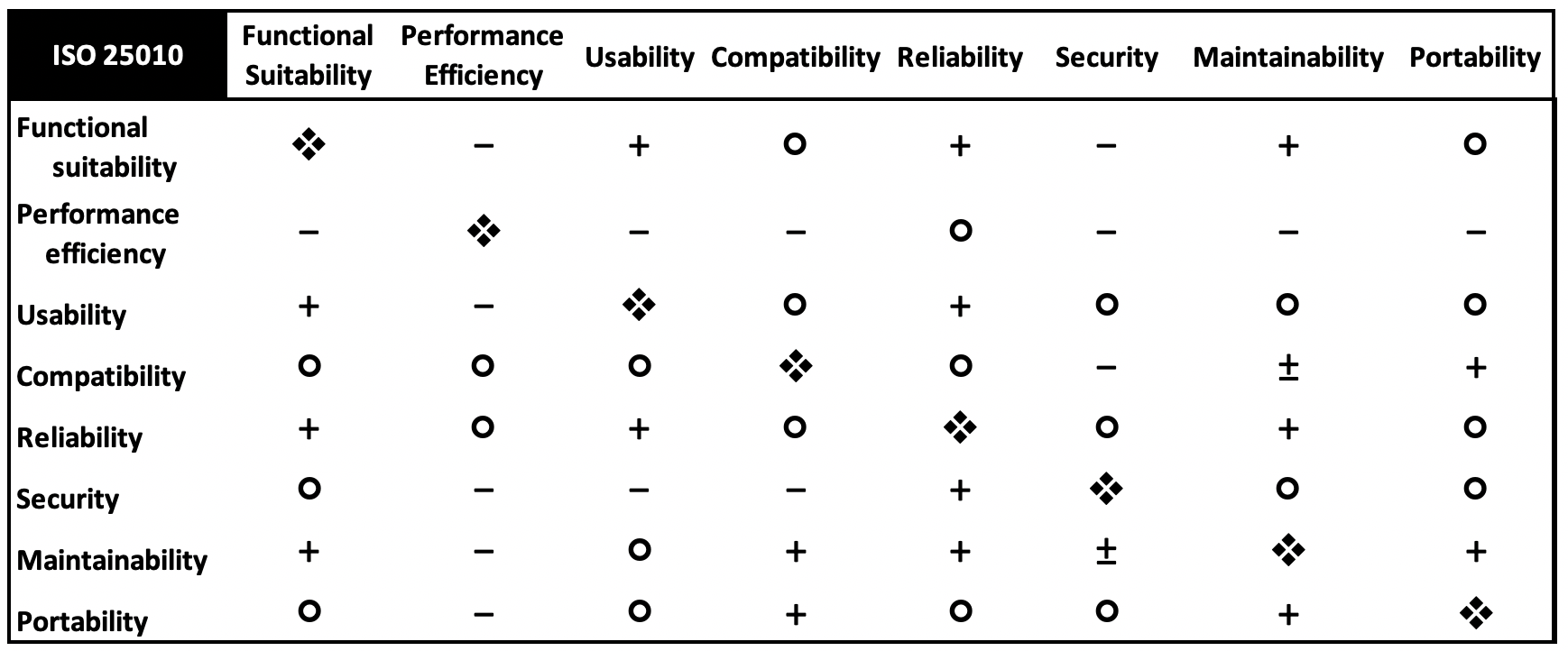


Figura 11: Matriz de Balanceamento entre características da ISO 25010

Legenda:

**°** independente, ou seja, não há influencia entre as características.

+ positivo, ou seja, um bom valor para a característica A significa em um bom valor para a característica B, portanto, uma relação direta de influência.

− negativo, ou seja, um bom valor para um atributo A significa em um valor ruim para o atributo B, portanto, uma relação inversa de influência.

± positivo e negativo, ou seja, algumas subcaracterísticas possuem influência positiva e outras negativas

Conforme é possível perceber, essa matriz não é simétrica. Segundo Wiegers e Beatty (2013) esse tipo de matriz possui essa idiossincrasia porque o efeito de aumento que um atributo A provoca em um atributo B, não é necessariamente o mesmo que o efeito de aumento que um atributo B provocará em um atributo A.

Portanto, existem conflitos e eles precisam ser resolvidos no momento do planejamento do ciclo de produção do produto, considerando a opinião de especialistas, de gestores, do time de desenvolvimento e da prioridade do usuário, a fim de, estabelecerem em conjunto e em acordo, os objetivos de medição planejados e possíveis para uma versão do produto de software. Uma vez definidos esses objetivos, o MeasureSoftGram fornece um mecanismo de quantificá-los e representá-los em um tensor de qualidade, possibilitando com isso, a comparação entre os requisitos de qualidade planejados e os resultados alcançados na versão produzida. Trata-se da clássica situação em projetos de comparar o planejamento frente à execução. Em outras palavras, uma configuração de software planejada *versus* a configuração desenvolvida.

Para capturar a percepção subjetiva expressa nos requisitos de qualidade planejados e transformá-los em uma representação numérica que seja possível comparar de forma quantitativa ao final do ciclo de produção, utilizamos a representação das características do equalizador de qualidade em uma escala analógica visual (VAS).

VAS são escalas de classificação contínua. A vantagem desse tipo de escala em relação a uma escala ordinal, de classificação discreta, como por exemplo, a escala de 5 pontos, é que as respostas não são restritas a um conjunto discreto de pontos. Com isso, nuances muito finas podem ser medidas. Além disso, podemos utilizar mais testes estatísticos para analisar os dados coletados a partir das variáveis representadas na escala VAS, fazendo com que os ajustes realizados na distribuição sejam mais confiáveis (FUNKE; REIPS, 2012). Esse tipo de escala tem se mostrado muito útil para capturar percepções subjetivas e relativas, em respostas de questionários on-line. Um exemplo é apresentado na Figura 12.

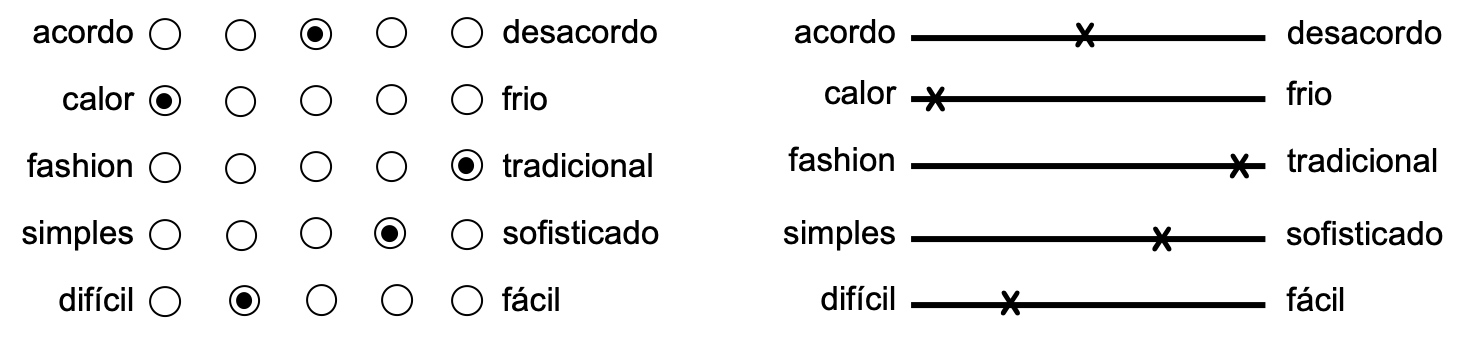


Figura 12: Exemplo da representação visual de uma escala VAS e uma escala de 5 pts.

Em uma escala VAS, cada posição escolhida corresponde a um ponto da escala, por conseguinte, um valor possível assumido pela variável. A quantidade de pontos presentes é definida pelo pesquisador.

No MeasureSoftGram utilizamos um ajuste de 100 pontos. Deste modo, ao capturar a posição definida na escala realizamos uma transformação da escala VAS para escala razão, conforme exemplificado na Figura 13.

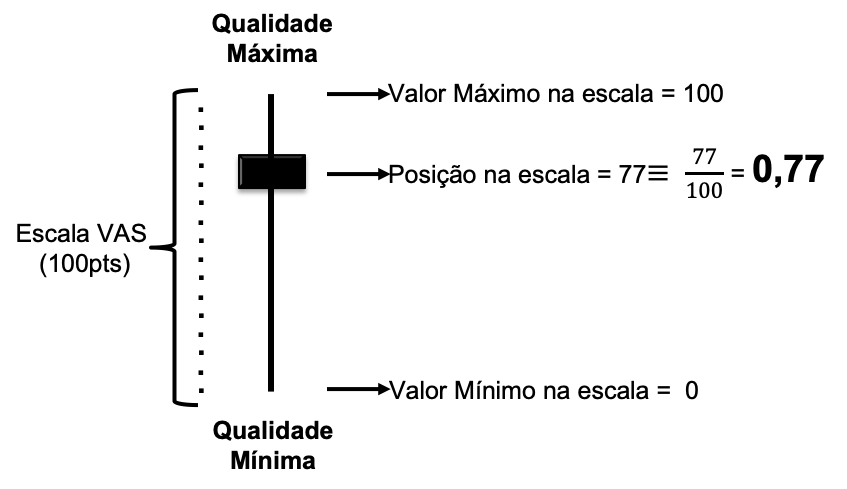


Figura 13: Transformação da escala VAS para escala Razão

Após essa transformação, a percepção de priorização dos requisitos de qualidade planejados é quantificada e com isso, conseguimos representá-la em uma de qualidade . Ao final do ciclo de produção, a partir da versão do produto desenvolvida, a qualidade é calculada, agregada e os resultados são representados em . Desta maneira, pode-se aplicar as regras de comparação definidas.

Como definida, a agregação do modelo é realizada a partir do cálculo da norma dos tensores de qualidade. Utilizamos essa representação agregada da qualidade para analisar as mútuas relações entre as características de forma quantitativa e a partir dos dados obtidos na versão desenvolvida. Para analisar os efeitos das múltiplas relações entre as características de qualidade, calculamos a matriz de correlação entre os tensores considerando a norma de cada um, conforme exemplificado na Figura 14.

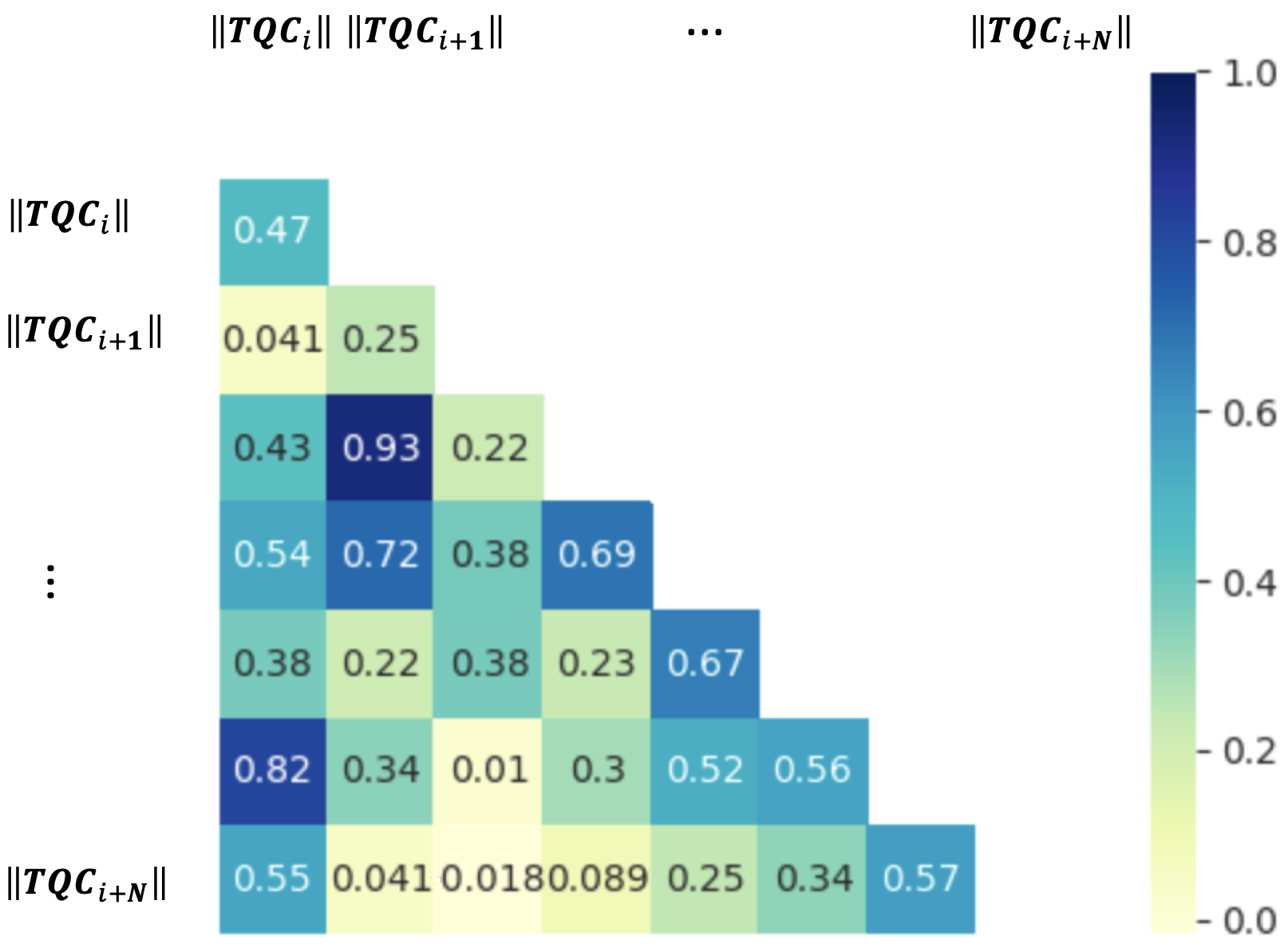


Figura 14: Matriz de correlação dos tensores de qualidade

A interpretação da matriz nos fornece uma percepção do grau de impacto de cada característica em relação a outra. Quando a correlação observada é positiva, temos uma indicação de que a relação entre as características ocorre de forma direta, ou seja, quando os valores de uma característica A aumentam então, os valores de uma característica B também aumentam. Quando tivermos uma correlação negativa, temos uma indicação de que a relação entre as características ocorre de forma inversa, ou seja, quando os valores de uma característica A aumenta, então B diminui. Já quando a correlação for nula, temos uma indicação de que as duas características não se influenciam, ou seja, são independentes.

A idéia é apresentar graficamente essa comparação e “esconder” a complexidade do modelo, de forma que os interessados na informação disponham desses dados e análises de maneira simples e, de certa forma, até lúdica, conforme exemplificado na Figura 15.

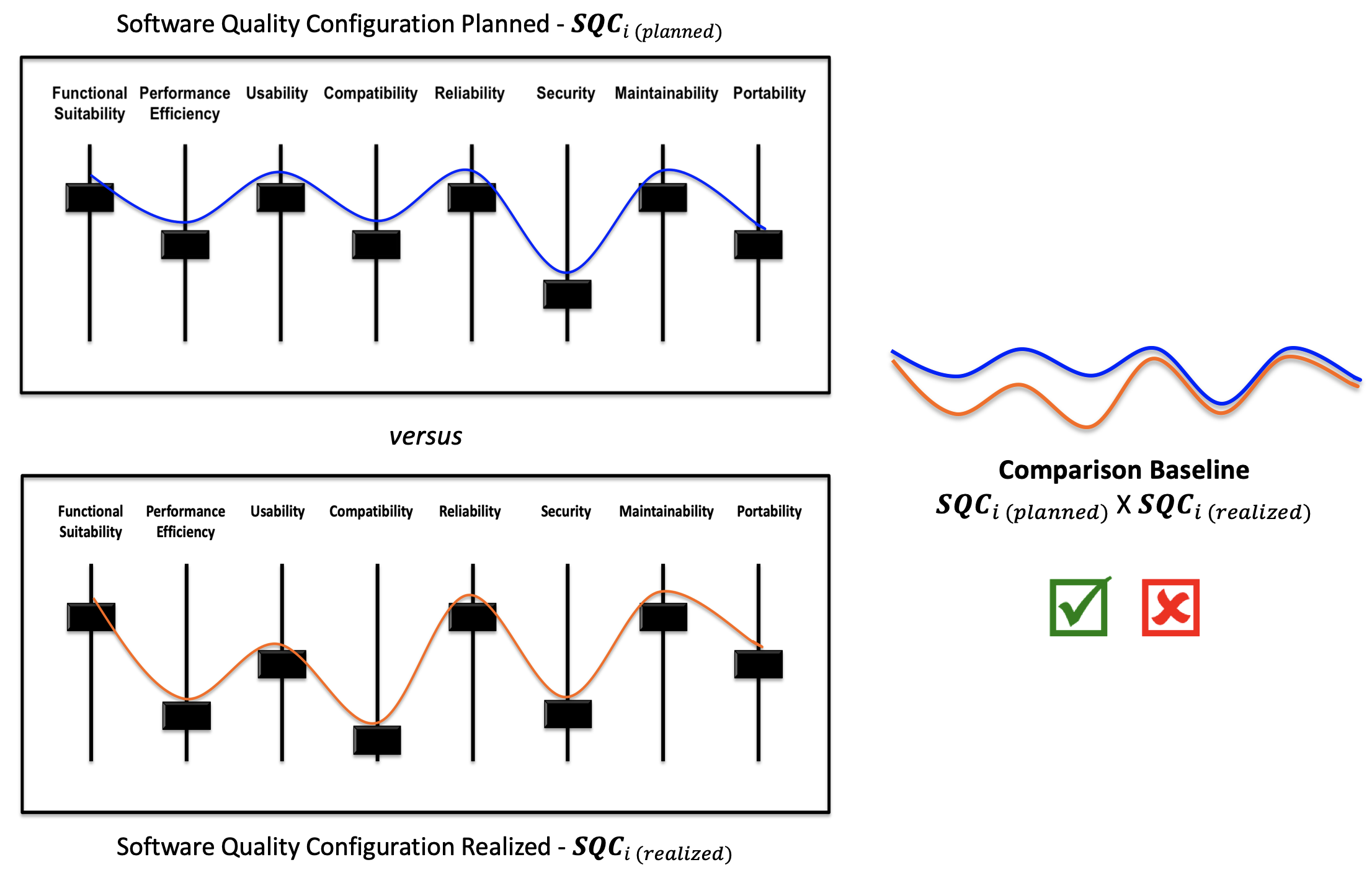


Figura 15: Comparação da qualidade planejada X realizada

A partir da utilização do MeasureSfotGram, o time de desenvolvimento e gestores passam a tomar decisões baseadas nas evidências obtidas a partir das versões produzidas, fazendo com que o conhecimento acerca do produto seja continuamente construído, desde o ciclo de vida do projeto. Essa é a metáfora do projeto aprender com o próprio projeto.

### Operacionalizando o MeasureSoftGram: um estudo observacional

Para investigar a viabilidade da operacionalização do modelo realizamos um estudo observacional, exploratório, onde o principal o objetivo foi averiguar a operacionalização e representação da qualidade de forma multidimensional.

O estudo foi executado em um produto pequeno, chamado Parasite. Trata-se de um sistema de software que tem o objetivo de apoiar o diagnóstico de doenças tropicais negligenciadas. Esse projeto foi realizado por meio da parceria entre a COPPE/UFRJ e a Fundação Oswaldo Crus-Fiocruz. A equipe de desenvolvimento foi composta por alunos da graduação da URFJ, além de alunos e professores do mestrado e doutorado de programas de pós-graduação da COPPE/UFRJ.

A versão analisada foi a 1ª release. O projeto nessa versão continha 1.081 LOC, 18 arquivos, 28 classes, 9 diretórios, 120 commits e foi escrito na linguagem Python.

Para a instrumentação do estudo utilizamos:

* 1 Máquina virtual (VM) – Linux Debian( stretch) AMD64, 4GB RAM, 2 núcleos de processadores, 1GB armazenamento. Nessa VM estão instaladas as ferramentas SonarQube 7.6, Gitlab 11.19, o repositório do projeto Parasite. Essa VM possibilita a execução da etapa de extração das métricas estáticas do repositório do projeto.
* Bibliotecas para manipulação de tensores, além de tratamentos matemáticos e estatísticos que apoiam a análise quantitativa e qualitativa dos dados:
  + - TensorLy 0.4.3 - <http://tensorly.org/stable/home.html>
    - Scipy 1.2.1 - <https://www.scipy.org/>
    - NumPy 1.16.2 - <http://www.numpy.org/>
    - Pandas 0.23.4 - <http://pandas.pydata.org/>
    - Python 3.7.0 - <https://www.python.org/downloads/release/python-370/>
    - Basic Linear Algebra Subprograms(BLAS) - <http://www.netlib.org/blas/>
    - Planilhas eletrônicas
    - Arquivos do tipo .cvs (datasets)

Foram observadas 7 medidas avaliadas que envolveram 10 métricas. Essas As medidas avaliadas foram representadas em dois tensores de qualidade e estes definiram a configuração de qualidade. A descrição desses elementos pode ser vista na Tabela 5.

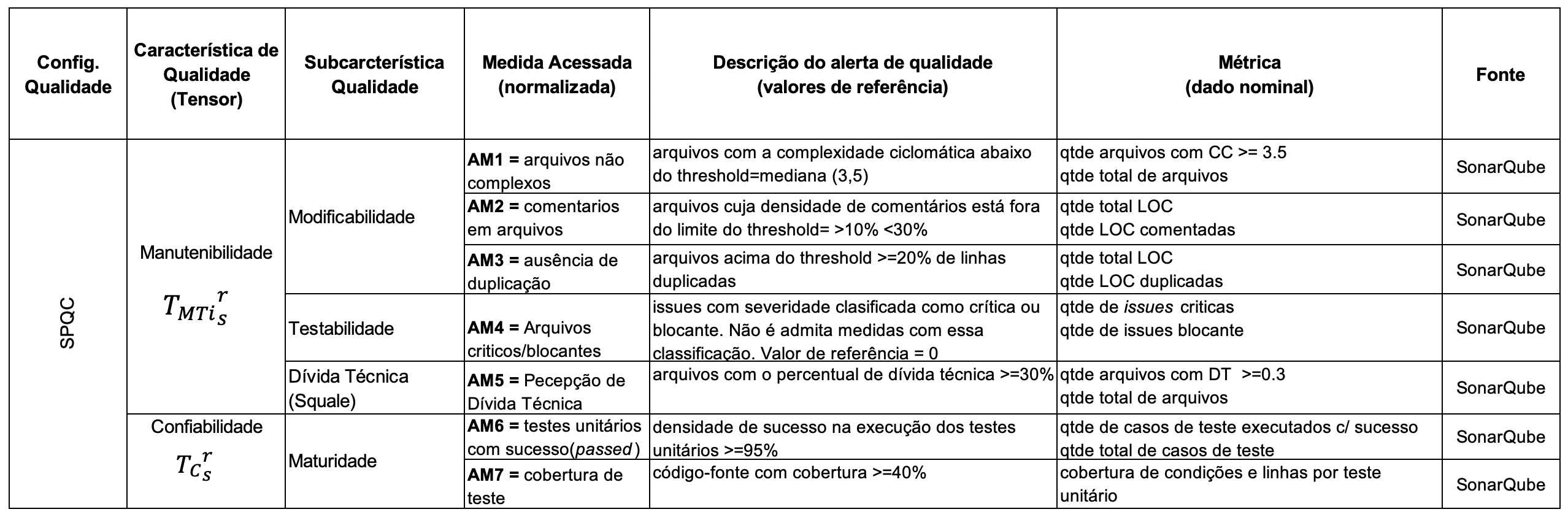


Tabela 5: Descrição da configuração de qualidade observada no projeto Parasite

A medida avaliada foi definida no modelo SQUALE e as demais no modelo Q-Rapids (MORDAL-MANET et al., 2009) (LÓPEZ et al., 2018). Na Figura 16 apresentamos uma visão ilustrativa da operacionalização do MeasureSoftGram no projeto Parasite.

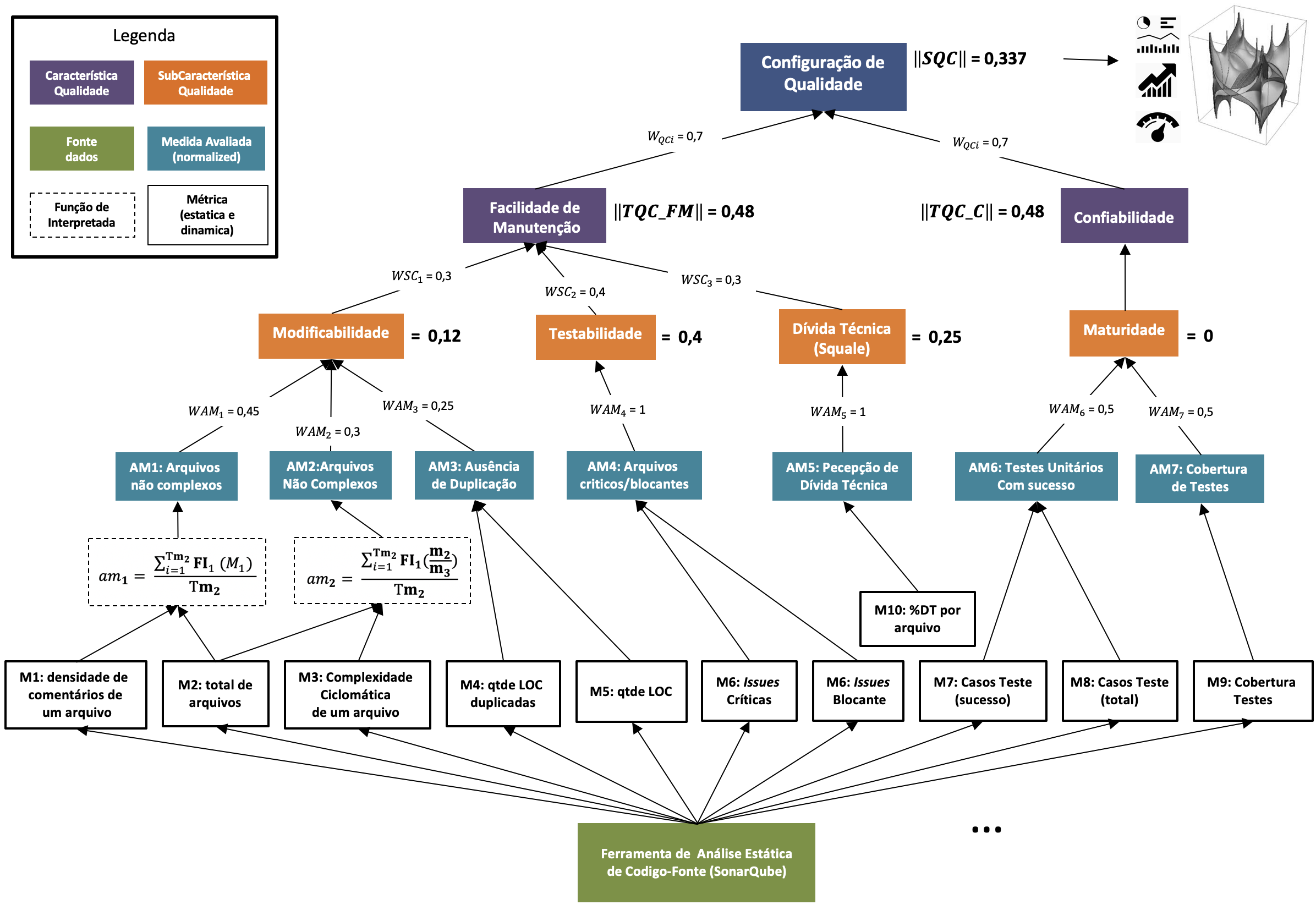


Figura 16: Operacionalização do MeasureSoftGram no projeto Parasite

Certas ameaças à validade foram tratadas para mitigar alguns fatores de confusão associados a este estudo e são apresentadas abaixo:

1. *Constructo:* a validade de constructo está associada a capacidade da teoria descrita em representar o fenômeno observado. Ainda não podemos fazer nenhuma consideração a respeito. Por outro lado, no intuito de mitigar essa ameaça, desenvolvemos um protótipo funcional utilizando dados da qualidade de produto extraídos de um projeto real, em desenvolvimento. Outras ameaças à validade de constructo, discutidas por Wohlin et al. (2012), são mitigadas neste estudo:

* *mono method bias* - diferentes tipos de métricas são utilizadas neste estudo
* *mono operation bias* – várias classes e métodos são analisados

1. Interna: como o participante deste estudo é um produto de software, onde “versões” já produzidas serão analisadas, muitas das ameaças à validade descritas por Wohlin et al. (2012), ainda que em ambiente mais controlado, são observadas como estratégia de mitigação, como por exemplo ameaças sociais e ameaças de múltiplos grupos. Por outro lado, deve-se ter especial atenção a algumas ameaças que dizem respeito ao uso de diferentes soluções computacionais. Essa questão está relacionada à ameaça de um grupo simples, principalmente a ambiguidade da influência na direção da causa. Nesse sentido, algumas dessas ameaçadas identificadas foram mitigadas:
   * A estrutura de dados representando tensores (arrays multidimensionais) são os mesmos para representar a qualidade de produto, a qualidade em uso e as respectivas configurações de qualidade
   * Fatores de confusão relacionados a comparação e interpretação das configurações de qualidade procuram ser mitigados por meio de conceitos e aplicações algébricas e trigonométricas
   * A biblioteca TensorLy fornece um conjunto de testes automatizados que podem ser reproduzidos e auditados. <https://github.com/tensorly/tensorly/tree/master/tensorly/tests>
   * As bibliotecas de tratamento algébrico e estatístico, disponibilizadas na linguagem Python têm sido utilizadas em diferentes pesquisas, inclusive em áreas de conhecimento diferentes da ciência da computação e engenharia de software. [http://www.bookmetrix.com/detail/book/c96d4bed-b58c-4fc0-9397-d8254be19de8#citations](http://www.bookmetrix.com/detail/book/c96d4bed-b58c-4fc0-9397-d8254be19de8)
   * Por fim, todas as soluções computacionais utilizadas neste estudo estão sob licença de software livre. Logo, como em todo projeto de software livre o código-fonte é aberto, todas as soluções utilizadas neste estudo são passíveis de auditoria a qualquer tempo.
2. Externa: está relacionada à capacidade de generalização. Por se tratar de um estudo observacional de caracterização, há claramente a presença da ameaça de interação entre os tratamentos e o pesquisador, uma vez que, a amostra do estudo não é representativa. Ainda não se pode fazer nenhum tipo de consideração nesse sentido. Por outro lado, a ameaça de interação entre tratamento e a configuração do estudo é garantida uma vez que o participante é um produto de software que está sendo desenvolvido e que passou por sucessivas evoluções de melhoria. Além disso, por se tratar da análise de “versões” já produzidas do produto, a ameaça de interação entre o histórico e tratamento é mitigada
3. Conclusão: está associada à robustez dos testes estatísticos realizados no estudo. Para mitigar esse tipo de ameaça foram definidas:
   * um modelo matemático com aplicações algébricas e trigonométricas que indicam indícios de propriedade canônica. O conjunto de regras de transformação é aplicado em qualquer hierarquia do modelo.

### Considerações Finais

Neste capítulo descrevemos o modelo matemático que fornece a base para a representação da qualidade de software de forma multidimensional. Apresentamos as descrições das propriedades do modelo e os passos necessários para que ele seja instanciado e operacionalizado.

Apresentamos também os resultados de um estudo preliminar, exploratório, que teve o objetivo de investigar a viabilidade da operacionalização do modelo base proposto. Obtivemos indícios positivos da viabilidade após configurarmos o modelo e observarmos dados de um projeto real. Contudo, por se tratar de um estudo exploratório e realizado em ambiente controlado, não conseguimos obter nenhum indicativo de que o modelo realmente consegue capturar as nuances do fenômeno da qualidade de produto de software.

Por outro lado, os resultados nos encorajaram e deram perspectiva para a continuidade da pesquisa. Assim, no capítulo seguinte apresentamos as estratégias e discutimos os resultados dos estudos experimentais que apoiam a validação da teoria proposta.

# Referências Bibliográficas

ACAR, E. et al. **Multiway analysis of epilepsy tensors**. Proceedings 15th International Conference on Intelligent Systems for Molecular Biology {(ISMB)} & 6th European Conference on Computational Biology (ECCB), Vienna, Austria, July 21-25, 2007. **Anais**...2007Disponível em: <https://doi.org/10.1093/bioinformatics/btm210>

AGRAWAL, R.; GOLSHAN, B.; PAPALEXAKIS, E. **A Study of Distinctiveness in Web Results of Two Search Engines**. Proceedings of the 24th International Conference on World Wide Web. **Anais**...: WWW ’15 Companion.New York, NY, USA: ACM, 2015Disponível em: <http://doi.acm.org/10.1145/2740908.2743060>

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern information retrieval**. [s.l.] ACM press New York, 1999. v. 463

BASILI, V. R.; CALDIERA, G.; ROMBACH, D. H. The Goal Question Metric Approach. In: **Encyclopedia of Software Engineering**. [s.l.] John Wiley & Sons, 1994. v. I.

BERANDER, P. et al. **Software quality attributes and trade-offs**. 2005

BUSE, R. P. L.; ZIMMERMANN, T. **Information needs for software development analytics**. Proceedings - International Conference on Software Engineering. **Anais**...2012Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84864194945&partnerID=40&md5=c3501a80b7d9cdd917a665867484ee78>

CHANG, K.-W.; YIH, W.; MEEK, C. **Multi-Relational Latent Semantic Analysis**. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. **Anais**...Seattle, Washington, USA: Association for Computational Linguistics, out. 2013Disponível em: <https://www.aclweb.org/anthology/D13-1167>

CHORAS, M. et al. Measuring and Improving Agile Processes in a Small-size Software Development Company. **IEEE Access**, v. PP, p. 1, 2020.

CICHOCKI, A. et al. **Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation**. [s.l.] Wiley Publishing, 2009.

ELBERZHAGER, F.; MÜNCH, J.; NHA, V. T. N. A systematic mapping study on the combination of static and dynamic quality assurance techniques. **Information and software technology**, v. 54, n. 1, p. 1–15, 2012.

ERNI, K.; LEWERENTZ, C. **Applying design-metrics to object-oriented frameworks**. Proceedings of the 3rd International Software Metrics Symposium. **Anais**...1996

FABIJAN, A. et al. **Online Controlled Experimentation at Scale: An Empirical Survey on the Current State of A/B Testing**. 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). **Anais**...2018

FENTON, N.; BIEMAN, J. **Software Metrics-A Rigorous and Practical Approach, Third Edition**. Third ed. [s.l: s.n.].

FITZGERALD, B.; STOL, K.-J. Continuous software engineering: A roadmap and agenda. **The Journal of Systems & Software**, v. 123, p. 176–189, 2017.

FUNKE, F.; REIPS, U.-D. Why Semantic Differentials in Web-Based Research Should Be Made from Visual Analogue Scales and Not from 5-Point Scales. **Field Methods**, v. 24, n. 3, p. 310–327, 2012.

HAOUES, M. et al. A guideline for software architecture selection based on ISO 25010 quality related characteristics. **International Journal of System Assurance Engineering and Management**, v. 8, n. 2, p. 886–909, 2017.

HENNINGSSON, K.; WOHLIN, C. **Understanding the Relations Between Software Quality Attributes - A Survey Approach**. 12:th International Conference on Software Quality. **Anais**...2002

ISO/IEC. **ISO/IEC 9126. Software engineering -- Product quality**. [s.l.] ISO/IEC, 2001.

ISO/IEC. **ISO/IEC 25010 System and software quality models**. [s.l.] ISO/IEC, 2010.

JIANG, M. et al. FEMA: Flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, 2014.

KEVIC, K. et al. **Characterizing Experimentation in Continuous Deployment: A Case Study on Bing**. Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track. **Anais**...: ICSE-SEIP ’17.Piscataway, NJ, USA: IEEE Press, 2017Disponível em: <https://doi.org/10.1109/ICSE-SEIP.2017.19>

KITCHENHAM, B. What’s up with software metrics?--A preliminary mapping study. **Journal of systems and software**, v. 83, n. 1, p. 37–51, 2010.

KOHAVI, R. et al. **Online Controlled Experiments at Large Scale**. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais**...: KDD ’13.New York, NY, USA: ACM, 2013Disponível em: <http://doi.acm.org/10.1145/2487575.2488217>

KOLDA, T. G.; BADER, B. W. Tensor Decompositions and Applications. **SIAM Review**, v. 51, p. 455–500, 2009.

LANZA, M.; MARINESCU, R. **Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems**. 1st. ed. [s.l.] Springer Publishing Company, Incorporated, 2010.

LAVAZZA, L.; MORASCA, S. **An Empirical Evaluation of Distribution-based Thresholds for Internal Software Measures**. Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering. **Anais**...: PROMISE 2016.New York, NY, USA: ACM, 2016Disponível em: <http://doi.acm.org/10.1145/2972958.2972965>

LIMA, E. L. **Cálculo tensorial**. [s.l.] Instituto de Matemática Pura e Aplicada do Conselho Nacional de Pesquisas, 1965.

LIMA, P. et al. A Metrics Suite for code annotation assessment. **Journal of Systems and Software**, v. 137, p. 163–183, 2018.

LINDGREN, E.; MÜNCH, J. **Software Development as an Experiment System: A Qualitative Survey on the State of the Practice**. Lecture Notes in Business Information Processing. **Anais**...2015

LOCHMANN, K. A Benchmarking-Inspired Approach to Determine Threshold Values for Metrics. **SIGSOFT Softw. Eng. Notes**, v. 37, n. 6, p. 1–8, nov. 2012.

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

MARINESCU, R.; RATIU, D. **Quantifying the quality of object-oriented design: The factor-strategy model**. Proceedings - Working Conference on Reverse Engineering, WCRE. **Anais**...2004

MEIRELLES, P. R. M. **Monitoring Source Code Metrics in Free Software Projects**. [s.l.] Institute of Mathematics and Statistics of University of São Paulo, 2013.

MENEELY, A.; SMITH, B.; WILLIAMS, L. Validating Software Metrics: A Spectrum of Philosophies. **ACM Trans. Softw. Eng. Methodol.**, v. 21, n. 4, p. 24:1--24:28, fev. 2013.

MORDAL-MANET, K. et al. **The squale model - {A} practice-based industrial quality model**. ICSM. **Anais**...{IEEE} Computer Society, 2009

NAGAPPAN, M.; ZIMMERMANN, T.; BIRD, C. **Diversity in Software Engineering Research**. Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. **Anais**...: ESEC/FSE 2013.New York, NY, USA: Association for Computing Machinery, 2013Disponível em: <https://doi.org/10.1145/2491411.2491415>

NION, D. et al. Batch and Adaptive PARAFAC-based Blind Separation of Convolutive Speech Mixtures. **IEEE Trans. on Audio, Speech and Language Proc.**, v. 18, n. 6, p. 1193–1207, ago. 2010.

PANTRAKI, E.; KOTROPOULOS, C. **Automatic image tagging and recommendation via PARAFAC2**. 25th {IEEE} International Workshop on Machine Learning for Signal Processing, {MLSP} 2015, Boston, MA, USA, September 17-20, 2015. **Anais**...2015Disponível em: <https://doi.org/10.1109/MLSP.2015.7324363>

PAPALEXAKIS, E. E. et al. **Turbo-smt: Accelerating coupled sparse matrix-tensor factorizations by 200x**. Proceedings of the 2014 SIAM International Conference on Data Mining. **Anais**...Society for Industrial and Applied Mathematics, 2014

PAPALEXAKIS, E. E.; FALOUTSOS, C.; SIDIROPOULOS, N. D. Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms. **ACM Trans. Intell. Syst. Technol.**, v. 8, n. 2, p. 16:1--16:44, out. 2016.

SHATNAWI, R. Synergies and conflicts among software quality attributes and bug fixes. **International Journal of Information Systems and Change Management**, v. 9, p. 3, 2017.

SIAVVAS, M. G.; CHATZIDIMITRIOU, K. C.; SYMEONIDIS, A. L. {QATCH} - An adaptive framework for software product quality assessment. **Expert Syst. Appl.**, v. 86, p. 350–366, 2017.

SJØBERG, T. D. AND D. I. K.; CRUZES, D. S. **What works for whom, where, when, and why?: on the role of context in empirical software engineering**. 2012 {ACM-IEEE} International Symposium on Empirical Software Engineering and Measurement, {ESEM} ’12, Lund, Sweden - September 19 - 20, 2012. **Anais**...2012Disponível em: <http://doi.acm.org/10.1145/2372251.2372256>

SVAHNBERG, M.; HENNINGSSON, K. **Consolidating different views of quality attribute relationships**. WoSQ@ICSE. **Anais**...DBLP:conf/icse/SvahnbergH09: {IEEE} Computer Society, 2009

TAHIR, A.; MACDONELL, S. G. **A systematic mapping study on dynamic metrics and software quality**. Software Maintenance (ICSM), 2012 28th IEEE International Conference on. **Anais**...2012

VALE, G. A. DO; FIGUEIREDO, E. M. L. **A Method to Derive Metric Thresholds for Software Product Lines**. 29th Brazilian Symposium on Software Engineering, {SBES} 2015, Belo Horizonte, MG, Brazil, September 21-26, 2015. **Anais**...{IEEE} Computer Society, 2015Disponível em: <https://doi.org/10.1109/SBES.2015.9>

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>

WANG, Y.; ZHENG, Y.; XUE, Y. **Travel Time Estimation of a Path Using Sparse Trajectories**. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais**...: KDD ’14.New York, NY, USA: ACM, 2014Disponível em: <http://doi.acm.org/10.1145/2623330.2623656>

WIEGERS, K. E.; BEATTY, J. **Software Requirements 3**. USA: Microsoft Press, 2013.

WOHLIN, C. et al. **Experimentation in Software Engineering**. [s.l.] Springer Publishing Company, Incorporated, 2012.

ZULZALIL, H. et al. A Case Study to Identify Quality Attributes Relationships for Web based Applications. 2008.

1. <http://cwe.mitre.org/data/index.html> [↑](#footnote-ref-3)
2. <https://www.sonarqube.org/> [↑](#footnote-ref-4)
3. <https://mathworld.wolfram.com/EuclideanSpace.html> [↑](#footnote-ref-5)
4. A figura 10 foi extraída do material de aula do prof. Geraldo Xexeo COPPE/UFRJ, COS738 2016.1 [↑](#footnote-ref-6)