**SUMÁRIO**

[2 Revisão da Literatura 4](#_Toc8953675)

[2.1 Seleção de Estudos 4](#_Toc8953676)

[2.2 Introdução 6](#_Toc8953677)

[2.3 Breve histórico da qualidade de produto de software e trabalhos relacionados 9](#_Toc8953678)

[2.4 Diferentes necessidades de informação e semelhantes necessidades de tomar decisões sobre o produto de software 13](#_Toc8953679)

[2.5 Ciclo de Vida de Desenvolvimento de Sistemas Contemporâneos 15](#_Toc8953680)

[Referências Bibliográficas 18](#_Toc8953681)

**ÍNDICE DE FIGURAS**

[Figura 1: Dimensões da Qualidade de Software 8](#_Toc8953252)

[Figura 2: Relacionamento entre as características de qualidade de produto da ISO 25010 12](#_Toc8953253)

[Figura 3: Pontos de observação da qualidade de versões de produtos de software 17](#_Toc8953254)

**ÍNDICE DE TABELAS**

[Tabela 1: Estudos secundários norteadores desta pesquisa 5](#_Toc8949413)

# Revisão da Literatura

Neste capítulo, apresentamos um panorama geral dos estudos da área de qualidade de produto de software ao longo do tempo, bem como, estudos relacionados a área de medição e métricas de software. Além disso, discutiremos brevemente como a observação da qualidade do produto de software é influenciada pela dinâmica do ciclo de produção/desenvolvimento do produto de software. Por fim, discutiremos alguns trabalhos que demostram a aplicabilidade do uso de tensores em diferentes áreas da ciência da computação.

## Seleção de Estudos

A seleção dos estudos que apoiaram a formulação do modelo proposto neste trabalho e que, nos forneceram uma visão geral dos estudos da qualidade do produto de software ao longo das 4 últimas décadas, começou com a formulação de *strings* de busca estruturadas além de, buscas realizadas de forma *ad-hoc*. Essas *strings* iniciais foram executadas na base digital *Scopus*, sem filtro de anos. Tínhamos uma percepção inicial do problema e, portanto, uma suspeita de que algumas áreas de conhecimento pudessem nos fornecer uma visão histórica e nos conduzisse na descoberta dos estudos mais contemporâneos sobre a qualidade do produto de software, de forma a nos apoiar em como endereçar o problema apresentado na subseção 1.2. Os resultados iniciais mostraram que a *string* de busca estava restritiva, retornando muito “ruído” e revelando poucos estudos que pudessem ser utilizados como artigos de controle.

Diante disso, optamos por uma estratégia de busca de estudos secundários para que, a partir destes, pudéssemos realizar *snowballing* (reverso e avante) a fim de encontrar os estudos que suportam e inspiram a idéia desta proposta.

Primeiramente iniciamos a busca por estudos que evidenciassem a avaliação da qualidade de produto de software. Aplicando *snowballing* reverso, encontramos os trabalhos seminais que se propuseram a identificar os fatores que descrevem o fenômeno da qualidade do produto de software. A partir daí, continuamos a utilizar essa estratégia e isso nos levou a identificar e selecionar os estudos secundários que nortearam a seleção dos principais estudos primários que fazem parte do arcabouço conceitual e teórico que apoia as ideias propostas nesta pesquisa. Apresentamos a lista desses estudos secundários na Tabela 1.

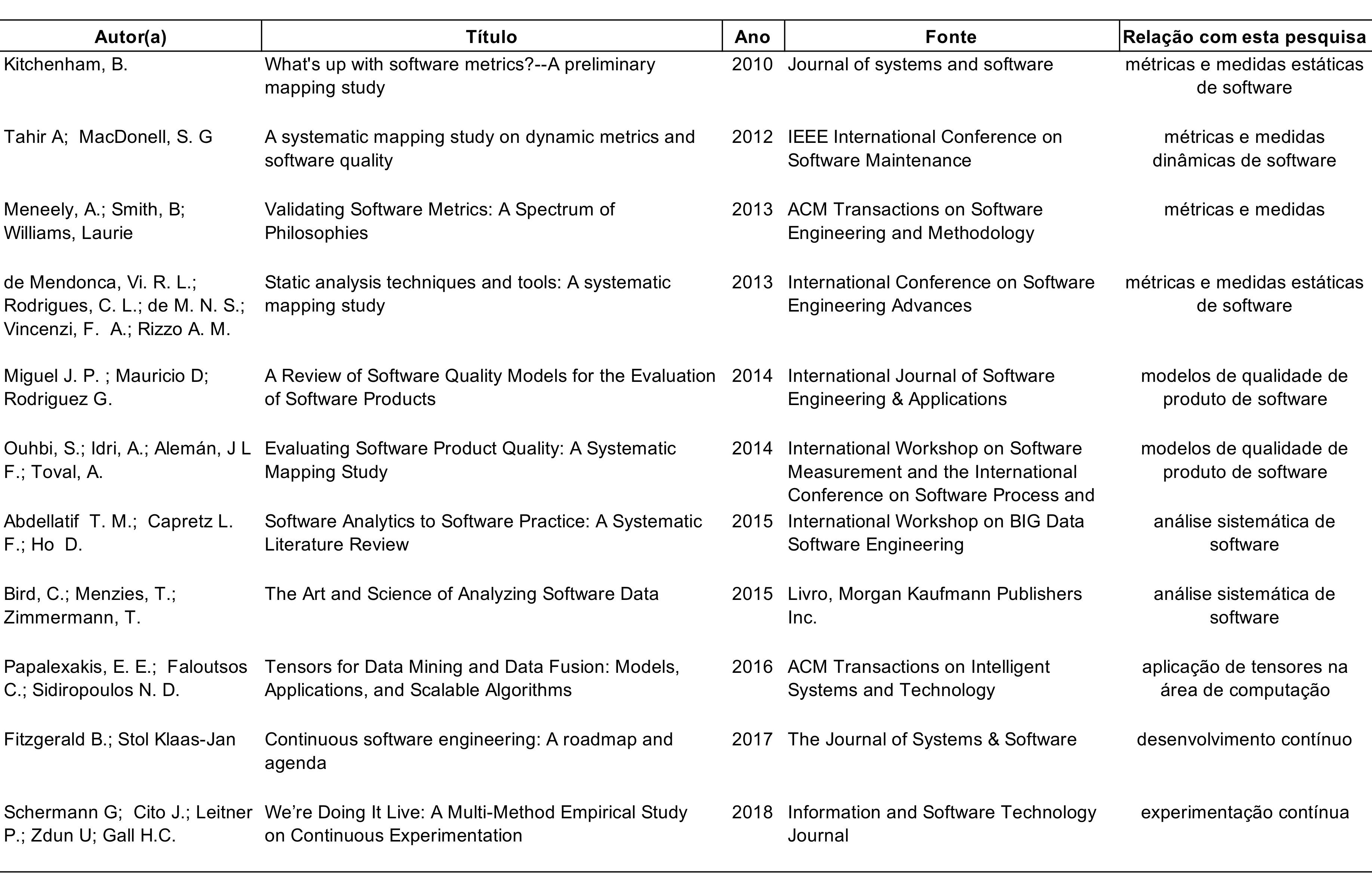


Tabela 1: Estudos secundários selecionados nesta pesquisa

Ainda que não tenhamos elaborado um protocolo sistemático e estruturado de revisão da literatura, entendemos que a estratégia utilizada nos forneceu boa cobertura sobre o estado da arte e da prática das diferentes áreas de conhecimento de apoio a esta proposta. Também, há de se considerar a experiência dos pesquisadores que, possuem atuação tanto na academia, quanto na indústria de software.

## Introdução

O gerenciamento de projetos é uma atividade de fundamental importância a qualquer organização que desenvolve produtos. O ciclo de vida de um produto tem seu início a partir do ciclo de vida do projeto e se estende até que o produto seja descontinuado (PMI, 2014). As atividades de gerenciamento proporcionam ao gerente, e também a equipe, uma visão ampla do andamento do ciclo de vida dos produtos em desenvolvimento ou em evolução. Atividades de gerenciamento portanto, procuram conciliar as diferentes necessidades de informação, uma vez que, tratam tanto das informações estratégicas, de maior interesse daqueles que administram as organizações, quanto das informações no nível tático e operacional, de maior interesse daqueles envolvidos no ciclo de desenvolvimento de produtos. Os métodos de gerenciamento e desenvolvimento de produto mais contemporâneos procuram aproximar os níveis gerenciais e técnicos, por entenderem que as decisões sobre o desenvolvimento de produtos de software devam ser tomadas conjuntamente entre os níveis gerenciais e técnicos, de forma colaborativa e contínua, com intuito de alinhar os incrementos das versões parciais até a versão final do produto. (Poppendieck e Poppendieck, 2006) (Dybä e Dingsøyr, 2015) e Safe

Um dos processos relacionados ao gerenciamento de projetos refere-se ao monitoramento e controle dos parâmetros oriundos da execução dos projetos e envolve diferentes áreas de conhecimento. Esses parâmetros são observados por meio de medidas, métricas e indicadores, e comumente as áreas de custo, tempo, escopo e qualidade, representam as principais medidas monitoradas.

A norma ISO/IEC 15939 (2017) estabelece as principais atividades em um processo de medição:

1. Estabelecer e sustentar o compromisso de medição

2. Planejar o processo de medição

3. Realizar o processo de medição

4. Avaliar a medição

Esse processo de medição é fundamental para a atividade de monitoramento e controle do ciclo de vida dos produtos, fornecendo informações importantes e necessárias para apoiar a tomada de decisões técnicas e gerenciais.

Os processos da engenharia de software se apresentam como um rico contexto para a observação de parâmetros de desempenho e qualidade. Há de se observar as diferentes fontes de informação relacionadas ao produto de software, como por exemplo: repositórios de código-fonte, resultados de execução de testes, bases de registro de defeitos, registros de log de operação e de transação. No que se refere à extração e tratamento de dados oriundos dessas fontes de informação, destaca-se o alto potencial de automação. Uma vez instrumentalizadas essas etapas, o processo torna-se capaz de disponibilizar informações tempestivas e automáticas sobre o comportamento do produto de software em desenvolvimento ou em utilização. Além disso, também provê informações gerenciais sobre parâmetros oriundos do próprio sistema de gestão do produto. Então, na medida em que, as informações sejam melhores qualificadas, com devidos tratamentos matemáticos, estatísticos e computacionais, além de, baseados em evidência, melhor é a capacidade de observação e compreensão do comportamento do produto de software. Em última instância, essa melhor compreensão ou clareza tende a nos levar a tomar melhores decisões sejam de ordem técnica, seja gerencial.

As medidas dão significado e interpretação às métricas. Essas por sua vez, possuem escalas que consistem um conjunto ordenado de valores contínuos ou discretos, ou um conjunto de categorias no qual um atributo é mapeado (ISO 15939, 2002). De acordo com a definição apresentada nessa norma e também por Wohlin et al., (2012), essas escalas podem ser do tipo: nominal, ordinal, intervalar ou razão. É importante destacar o papel da escala, uma vez que, resultados de estudos são invalidados, ou enviesados, por não se observar devidamente a escala das métricas ao se realizar as análises.

Fenton e Bieman (2014), mostram que medidas e métricas de software estão associadas a muitas atividades, as quais estão envolvidas um certo grau de medição, como por exemplo: estimativas de custo, esforço e qualidade. Nesse contexto a norma ISO/IEC 25010 (2010) categoriza as medidas de acordo com os diferentes objetivos de medição:

* Medidas de qualidade de processo: Utilizadas durante a execução dos processos de software. Oferecem medidas sobre a execução das atividades do processo de desenvolvimento e auxiliam a monitorar a organização do trabalho durante o desenvolvimento.
* Medidas de qualidade interna: Aplicadas em um produto de software não executável, como o código-fonte. Oferecem aos usuários, desenvolvedores ou avaliadores, o benefício de poder avaliar a qualidade do produto antes que ele seja executado;
* Medidas de qualidade externa: Aplicadas a um produto de software executável, medindo o comportamento do sistema, no qual o software é uma parte, por meio da execução de testes, operação ou mesmo observação. Oferecem aos usuários, desenvolvedores ou avaliadores, o benefício de poder avaliar a qualidade do produto durante seu processo de teste ou operação.
* Medida de qualidade em uso: Aplicadas para medir o quanto um produto atende as necessidades de um usuário para que sejam atingidas metas especificadas como eficácia, segurança e satisfação.

A Figura 1 ilustra como as medidas se influenciam mutuamente nos contextos em que elas estão envolvidas, seja em relação ao software propriamente dito (tanto internamente quanto externamente) ou ao efeito produzido pelo uso de software:

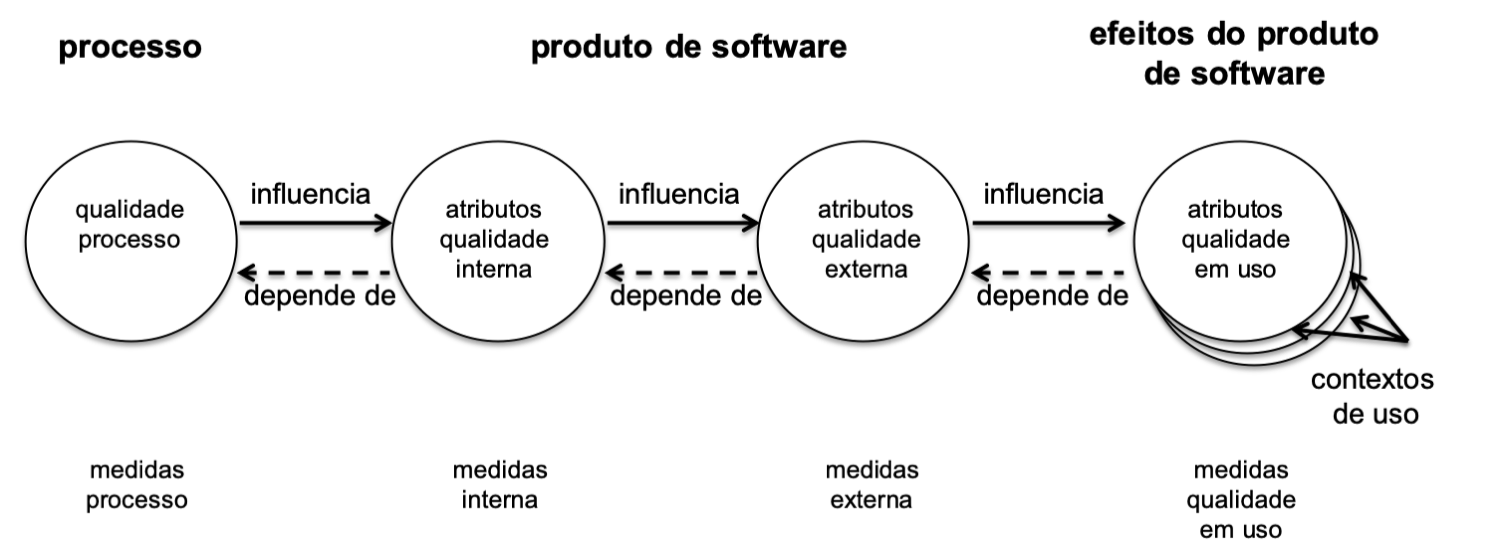


Figura 1: Dimensões da Qualidade de Software (traduzido da ISO 25023)

Hoje em dia é comum em grandes empresas como Google®, Facebook® e Microsoft® a aplicação diária de métodos de análise de dados em seus projetos e produtos. Na Microsoft®, por exemplo, foi desenvolvida uma plataforma chamada CODEMINE, que apoia os times de desenvolvimento de produto (Czerwonka et al., 20113). Por meio dessa plataforma vários times utilizam os dados coletados dos projetos para melhorar seus processos como:

* Avaliação de riscos e mudança em ferramentas de análise de impacto;
* Otimização da estrutura de ramos dos repositórios de código;
* Análise sociotécnica de dados;
* Busca customizada de defeitos e logs;
* Acompanhamento de tendências e relatórios do status de desenvolvimento.

Portanto, para analisar e tomar decisões eficazes em relação a um produto de software é imprescindível que sejam consideradas diferentes tipos de dimensões de qualidade, de forma a capturar as diferentes facetas do produto. É importante ressaltar a necessidade de se analisar os aspectos relacionados a qualidade em uso em conjunto com aspectos da qualidade do produto, bem como suas relações e influências mútuas, afim de proporcionar a análise sistêmica da qualidade de software. Neste trabalho, trataremos a análise da qualidade de produto considerando essa visão mais holística e sistêmica.

## Breve histórico da qualidade de produto de software e trabalhos relacionados

### Modelos de Qualidade

O interesse por parte da academia sobre o fenômeno da qualidade do produto de software é tão longevo quanto os estudos que tratam da construção e desenvolvimento do software. Os modelos inicialmente propostos por McCall et al. (1977) e Boehm et al. (1976) foram os pioneiros a descrever e categorizar fatores associados a qualidade do produto de software. Nesse período havia uma comum percepção de que o desenvolvimento de software era uma atividade propensa a erros e, portanto, a observação da qualidade passava pela percepção da capacidade do software não apresentar defeitos durante sua utilização (Miguel et al., 2014).

McCall et al. (1977) descreveram e analisaram 39 fatores do produto de software que foram organizados em níveis hierárquicos de fatores e subfatores. Estes por sua vez, foram distribuídos nas perspectivas de revisão, operação e transição do produto de software. Os fatores descritos na perspectiva de operação do produto foram: corretude; robustez; eficiência; integridade e usabilidade. Já os fatores facilidade de manutenção, testabilidade e flexibilidade foram descritos na perspectiva de revisão do produto; e, na perspectiva de transição do produto, os fatores portabilidade, reuso e interoperabilidade. Outra contribuição importante desse trabalho foi a definição de critérios, que funcionou como um mecanismo de percepção da presença (sim) ou ausência (não) de um determinado fator. Esse julgamento era realizado por um especialista da área de qualidade que indicava sua percepção por meio de um questionário, denotando forte caráter de interpretação subjetiva. Para tanto, os autores estabeleceram e observaram as relações entre os fatores e seus subfatores com suas respectivas métricas associadas. Esse estudo serviu de base para proposição dos subsequentes.

Já o trabalho de Boehm et al. (1978) propõe uma diferente organização hierárquica que possui em seu nível mais alto os fatores utilidade, portabilidade e facilidade de manutenção. Novos fatores e subfatores foram propostos. Destaca-se àqueles relacionados aos usuários finais, agrupados como subfatores do fator utilidade. Trata-se do primeiro estudo a se preocupar em observar fatores da qualidade do produto de software que produzem efeitos associados a facilidade de uso. Os autores redefinem o conceito de facilidade de manutenção agrupando os fatores de testabilidade, compreensibilidade e modificabilidade de forma a caracterizar o grau de facilidade de se realizar atividades de manutenção ou evolução do software. Outra conclusão importante apresentada nesse estudo é que a atenção em esforços e atividades de garantia da qualidade ao longo do ciclo de vida do desenvolvimento do produto de software acarreta em economia financeira. Portanto, introduz a percepção de riscos financeiros associados a garantia da qualidade.

Posteriormente, Dromey (1995) propôs um novo modelo, voltado a aspectos da qualidade na atividade de implementação. Ele argumentou que os modelos anteriores não forneciam um mecanismo lógico para relacionar os fatores e as estruturas do código-fonte. Influenciado pelo paradigma da orientação a objetos e a componentização, Dromey propôs um modelo genérico para viabilizar a operacionalização entre os atributos de qualidade e as propriedades inerentes do código-fonte. A ideia geral do modelo consiste em um conjunto de componentes, um conjunto de propriedades da qualidade dos componentes e um conjunto com a descrição em alto nível dos atributos de qualidade. Esse modelo não é hierárquico e o autor descreve seis relações binárias entre esses conjuntos e propõe um mapeamento desde dos atributos de qualidade descritos em alto nível até, por exemplo, uma estrutura sintática de uma linguagem de programação, como uma expressão, variável ou método. Contudo, não há validação desse modelo, seja em uso ou experimental, o que representa um ponto fraco do estudo. Cabe ressaltar que, esse modelo já utiliza do arcabouço teórico da ISO/IEC 9126 (1991) como referência conceitual.

Esses trabalhos pioneiros de McCall e Bohem serviram base para criação da primeira norma internacional de padronização da qualidade do produto de software, a ISO/IEC 9126, cuja primeira versão foi publicada em 1991 e atualizada em 2001 (ISO/IEC 9126, 2001), já contemplando aspe A ISO/IEC 9126 organizou os fatores em um modelo hierárquico composto por características e subcaracterísticas, divido em:

1. atributos de qualidade interna e externa – atributos de qualidade interna são propriedades do sistema observadas de forma estática, ou seja, sem a necessidade de ser executado. Já os atributos de qualidade externa são observados durante a execução do sistema. As características mapeadas nessas dimensões de qualidade são Funcionalidade (atendimento aos requisitos), Confiabilidade, Usabilidade, Eficiência, Facilidade de Manutenção e Portabilidade;
2. atributos de qualidade em utilização – são propriedades observáveis por meio da utilização do sistema por usuários finais. As características mapeadas nessa dimensão de qualidade são: Eficácia, Produtividade, Segurança e Satisfação.

A ISO 9126 foi substituída pela ISO/IEC 25010 (2010), a versão vigente. Essas normas surgiram no intuito de padronizar a terminologia conceitual da qualidade do produto de software além de, servirem como guia para definição de modelos avaliação da qualidade do produto de software. Essas duas normas, em especial a ISO 9126, são os modelos mais de qualidade de produto mais estudados na literatura (Ouhbi et al., 2014). A ISO 25010 define e reorganiza novas características e subcaracterísticas, além de, consolidar a visão da qualidade de sistemas de software na perspectiva da qualidade de produto e qualidade em uso. Nas figuras 2 e 3 são apresentadas as Características e Subcaracterísticas da Qualidade do Produto

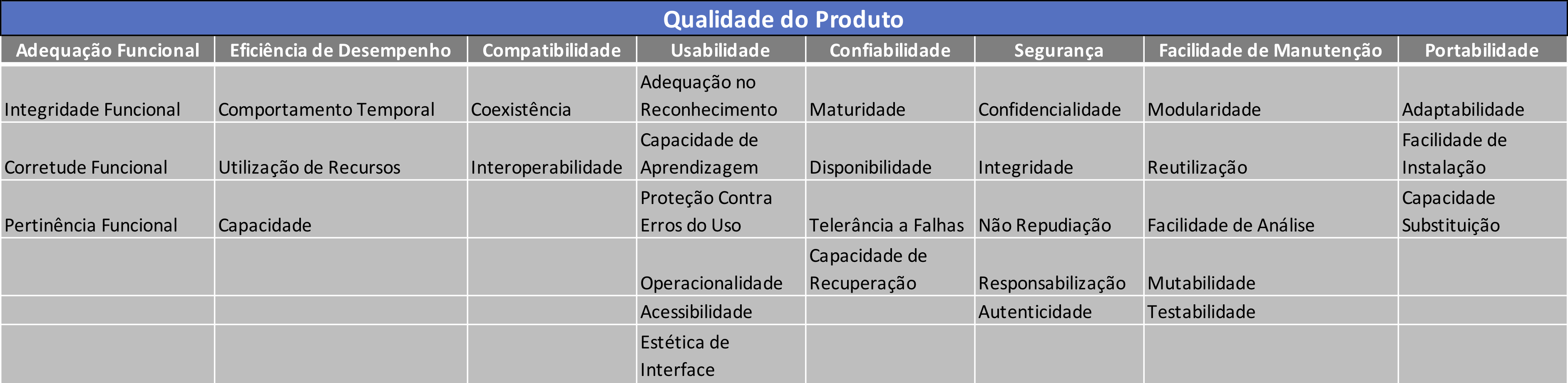


Figura 2: Características e Subcaracterísticas da Qualidade do Produto - ISO 25010

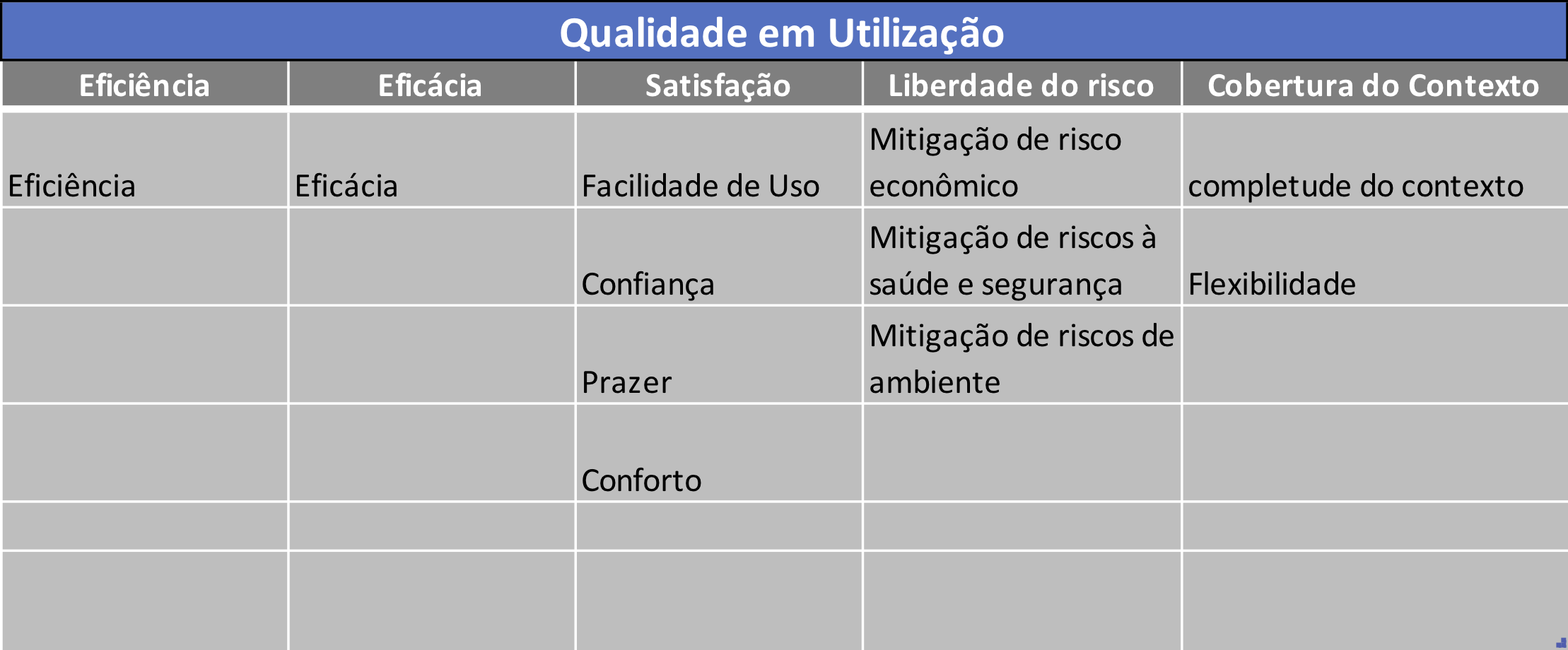


Figura 3: Características e Subcaracterísticas da Qualidade em Utilização - ISO 25010

A ISO 25010 se consolidou como o modelo com maior cobertura de características relacionadas a qualidade do produto (Miguel J. et al., 2014). Ela não cobre características definidas em modelos específicos a um determinado contexto.

juntamente com o de Dromey influenciaram

Na última década, diferentes modelos e métodos de medição foram propostos com intuito de endereçar a operacionalização entre as definições conceituais das características e suas subcaracterísticas, descritas nas ISO supracitadas (tidas como abstratas), com suas respectivas medidas e métricas. Frequentemente, estudos apontavam essa condição como um limitador que dificultava a aplicação de métodos objetivos de medição (calculados a partir de regras numéricas) e por conseguinte, restringia a adoção de tais modelos por parte da indústria. Nesse sentido, destacam-se principalmente o modelo SQALE, inicialmente proposto por Mordal-Manet et al. (2009) e metamodelo Quamoco, proposto por Wagner et al. (2012). Ambos são hierárquicos, permitem modelar as características e subcaracterísticas definidas nos modelos de referência e se baseiam nessa estrutura de informação como um meio de avaliar a qualidade de um componente de software. Esses modelos endereçaram tratamentos para as questões discutidas por Kitchenham (2010), Juristo e Moreno (2010), Lavazza e Morasca (2016) e (Dybå et al., 2012). Tanto o SQALE quanto o Quamoco serviram de inspiração para a definição dos modelos mais recentemente propostos, o QATCH (Siavvas et al., 2017) e o Q-Rapids (López et al., 2018), que, do ponto de vista do modelo de avaliação e método de medição, são os trabalhos mais relacionados com esta pesquisa

Em relativa concomitância temporal aos estudos dos modelos de qualidade de produto, há um extenso arcabouço de conhecimento acumulado em estudos sobre medidas e métricas de software. As primeiras métricas e medidas criadas datam da década de 70. Considerando a atividade de implementação/codificação como naturalmente propensa a erros, Akiyama (1971) apud Bird, Menzies e Zimmermam (2015) define que o número de defeitos em um programa de computador era uma função do número de Linhas de Código-Fonte (LOC). Já McCabe (1976) defendeu que a quantidade de LOC era menos importante que a complexidade do código-fonte. Ele argumentou que o código-fonte era mais propenso a erros na medida em que sua complexidade ciclomática (CC) era maior do que 10. Desde então, diferentes métricas e medidas continuaram sendo propostas, podendo ser classificadas entre dois principais grupos: (I) estáticas, ou seja, capturadas diretamente de artefatos de software, sem a necessidade de implantação e uso do software; e (II) dinâmicas, capturadas a partir do comportamento percebido por meio do uso do software. Diferentes estudos secundários discutiram o uso de técnicas de analise estática (Mendonça et al., 2013) e dinâmica (Tahir e MacDonell, 2012). Contudo, para que se obtenha uma percepção mais holística e fidedigna da qualidade do produto de software é necessário que haja a combinação dessas diferentes técnicas de análise, como argumentado por Elberzhager et. al. (2012).

======

.

=======

Acompanhando a evolução dos estudos sobre qualidade de produto de software, métricas e medidas, diferentes ferramentas também foram desenvolvidas, de forma a automatizar o método de medição, como por exemplo, algumas mais recentemente desenvolvidas: SonarQube[[1]](#footnote-2); TeamScale[[2]](#footnote-3); CodeClimate[[3]](#footnote-4); IBM Rational Team Concert[[4]](#footnote-5) e Microsoft Code Analysis[[5]](#footnote-6). Vale ressaltar que essas ferramentas foram criadas oferecendo suporte a ciclos curtos de desenvolvimento, com entregas e liberação de versões de produto de software em fluxos de trabalho com atividades organizadas em uma cadência contínua. Como essas ferramentas apoiam a observação da qualidade baseada nos modelos existentes, elas também acabam operando de forma unidimensional.

### Estudos sobre as relações entre as características da qualidade do produto de software

Diferentes estudos se preocuparam em investigar a relação entre as características que descrevem a qualidade de produto, a partir de evidências experimentais. Essas características, a depender do contexto, podem ter diferentes significados e importância para diferentes interessados na informação e as organizações de software precisam lidar com isso no desenvolvimento de seus produtos. Henningsson e Wohlin (2002) conduziram um estudo onde concluíram, a partir de *surveys* realizados com participantes da academia e indústria, que existem dependências entre as características da qualidade. Cabe ressaltar que McCall et al., (1977) também chegaram a conclusões semelhantes. Uma das principais conclusões desse estudo realizado por Henningsson e Wohlin (2002) foi que existe uma lacuna na percepção observada na literatura, onde as relações entre os atributos de qualidade são descritas de forma genérica, e o tácito da indústria. Já Svahnberg e Henningsson (2009) investigaram as relações entre as características de qualidade a partir da triangulação de três fontes: artigos; *survey* com a indústria e um estudo sobre avaliação da qualidade em padrões arquiteturais. As características de qualidade foram extraídas da ISO/IEC 9126 (2001). Neste estudo eles confirmaram a relação negativa (-) entre as características de usabilidade X eficiência de desempenho e funcionalidade X eficiência de desempenho. Por outro lado, encontram conflitos por exemplo, entre: adequação funcional X portabilidade; facilidade de manutenção X confiabilidade. Uma das principais conclusões desse estudo revela que é possível encontrar relações entre as características de qualidade a partir de diferentes fontes de informação e que a consistência semântica entre as descrições das características deve ser cuidadosamente investigada, de forma a reforçar a confiança acerca das classificações. Como sugerido nesse estudo, não se pode utilizar as classificações existentes indiscriminadamente (às cegas), ainda que haja um aparente mapeamento com os normativos propostos pela ISO. Os autores argumentam que as especificidades dos diferentes contextos devem ser consideradas. Já nos estudos realizados por Al-daajeh et al. (2012-a) foi proposto um arcabouço conceitual para, baseado na abstração e descrição de táticas arquiteturais, descritas por meio de cenários, que visam a priorização e calibração dos atributos de qualidade (características). Para definição e avaliação dos cenários, foi aplicado um questionário que tiveram 88 respondentes entre alunos de mestrado em engenharia de software, pesquisadores e diferentes perfis de profissionais da prática. Para isso, foram escolhidas as caraterísticas de facilidade de manutenção X segurança e os respondentes deveriam fazer um julgamento sobre a o grau de impacto de uma determinada subcaracterística representada em um cenário. Em seguida, de acordo com o método de medição proposto, é produzida uma matriz com a avaliação de impacto de uma determinada característica, detalhada em suas subcaracterísticas e representada nos cenários, frente às demais características. Um achado interessante nesse estudo foi a indicação de que a intensidade da relação entre as características(+ positiva; - negativa; o neutra) podem ser intercambiáveis, ou seja, quando o foco da priorização foi a característica de facilidade de manutenção, então foi observado que sua relação com as características de confiabilidade e segurança era fortemente positiva (+). Já quando a característica de segurança foi priorizada, a relação se manteve positiva, porém com menor intensidade. Al-daajeh et al. (2012-a) conclui que a compreensão dos relacionamentos entre as características auxilia a tomada de decisões mais assertivas, além de que a técnica de táticas e cenários arquiteturais auxiliam a atividade de priorização e definição do objetivo de medição. Embora esse estudo proveja um nível mais detalhado (“mais baixo nível”) de operacionalização no que se refere a priorização dos atributos de qualidade, ainda careceu de dados reais de projetos. Já Haoues et al. (2017) foi o primeiro a investigar as características e subcaracterísticas descritas na ISO/IEC 25010 (2010). Para tanto foram selecionados e analisados, um conjunto de 113 estudos (artigos) extraídos das bases digitais IEEE Xplore, ACM Digital Libray, Science Direct e CiteSeer.

Após a etapa de seleção, foi aplicada uma técnica de recuperação de informação conhecida como análise formal de conceitos (*FCA*), baseada na teoria matemática de conjuntos ordenados e seus entrelaçamentos[[6]](#footnote-7). A FCA se baseia duas concepções: tabelas de dados e entrelaçamentos. Uma tabela de dados representa uma relação binária entre objeto (ex: artigo) e chave (palavras-chave/conceitos). Já os entrelaçamentos são utilizados para prover uma visão estruturada da relação e classificação dos conceitos investigados. Primeiramente os autores identificaram os padrões arquiteturais mais utilizados, reportados na literatura. Em seguida, propuseram uma classificação de acordo com a ISO/IEC 25010 (2010). Por fim, analisaram os relacionamentos entre diferentes características e subcaracterísticas descritas na ISSO 25010. Dessa maneira, Haoues et al. (2017) propuseram uma matriz que descreve esses relacionamentos. Como havia um contexto específico do estudo de Haoues et al. (2017), que eram os padrões arquiteturais, a partir da interpretação desta matriz realizamos algumas adaptações considerando os estudos Henningsson e Wohlin (2002), Svahnberg e Henningsson (2009) e AAl-Daajeh et al. (2012-b) e chegamos a seguinte matriz de relacionamentos ilustrada na Figura 2.

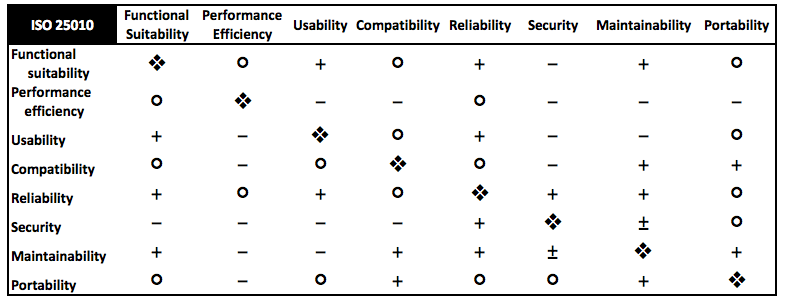


Figura 4: Relacionamento entre as características de qualidade de produto da ISO 25010

É importante registrar que esta é a evidência de referência para o modelo de proposto nesta pesquisa para lidar com os relacionamentos entre características e sub-características. Utilizaremos essas relações quando não houver dados disponíveis, com significância estatística, a serem extraídos dos ambientes de desenvolvimento e experimentação contínua. A partir do momento que o projeto comece a gerar dados, a intenção é que utilizemos o conhecimento das relações obtidas por meio destes. Em relação aos estudos sobre os relacionamentos entre características e subcaracterísticas da qualidade de produto, os principais trabalhos relacionados a esta pesquisa são: Al-daajeh et al. (2012-a); AAl-Daajeh et al. (2012-b) e Haoues et al. (2017).

Finalmente, o uso de tensores tem aplicação clássica em campos da física e matemática como, por exemplo, estudos sobre tensão mecânica, gravitacional. Na área de computação, especialmente nesta última década, o uso de tensores, segundo Papalexis et al. (2016), se mostrou eficiente, denotando a maturidade da sua aplicação em áreas como, por exemplo: processamento de sinais e visão computacional (Dimitri et al., 2010); processamento de linguagem natural (KaiWei et al., 2013); cidades inteligentes (Wang et al., 2014); análise de dados em redes sociais (Hansen et al., 2015); mineração de dados na web (Agrawal et al., 2015); e sistemas de recomendação (Pantraki e Kotropoulos, 2015). Apoiado por essas ferramentas computacionais, há também aplicação na área de medicina, como por exemplo, em estudos de sinais cerebrais em análises de exames de eletroencefalograma (Acar et al., 2007). Essa gama de aplicação e uso de tensores se deve sua habilidade em representar e modelar dados multimodais ou multiaspectos. Como resultado, a decomposição de tensores, tipicamente utilizada para tratar sua ordem, e por consequência lidar com sua alta dimensionalidade, se mostrou satisfatoriamente apropriada nesses contextos de aplicação, uma vez que auxiliou revelar relações latentes entre os dados, de forma a apoiar "*insights*" não possíveis de serem alcançados por estruturas diferentes das tensoriais. Além disso, existem eficientes operações para lidar com a esparcidade e otimização de dados como os apresentados no *survey* conduzido por Papalexis et al. (2016). Não é do nosso conhecimento, até o presente momento, que tensores tenham sido utilizados para modelar a qualidade de software.

## Diferentes necessidades de informação e semelhantes necessidades de tomar decisões sobre o produto de software

Toda medição deve ser orientada a um objetivo, que por sua vez, deve atender a uma necessidade de informação.

A área de engenharia de software se caracteriza como um rico contexto para se obter dados. Há de se observar, por exemplo, diferentes fontes de informação relacionadas ao produto de software como por exemplo: repositórios de código-fonte, resultados de execução de testes, bases de registro de defeitos, registros de log de operação e de transação, avaliações de experiência do usuário. No que se refere à extração e tratamento de dados oriundos dessas fontes de informação destaca-se o alto potencial de automação. Uma vez instrumentalizadas essas etapas (extração e tratamento), é possível obter informações tempestivas, que auxiliam engenheiros de software e gerentes a tomarem decisões sobre o produto de software, baseada em análise de dados do próprio produto.

Então, além de medidas que orientam decisões no âmbito do projeto e do pro- cesso, com predominante interesse daqueles que gerenciam o desenvolvimento do produto, há também, àquelas relacionadas ao produto, sendo esta, de predominante interesse da equipe de desenvolvimento. O entendimento de ambas é complementar e permite que a equipe técnica e de gestão tomem decisões mais conscientes acerca do comportamento do software em produção. Portanto, projetos de software podem ser monitorados durante todo o ciclo de vida do produto e, com isso, podem fornecer um vasto conjunto de métricas e medidas, de forma a fornecer insumos que apoiem a tomada de decisão técnico-gerencial (Buse e Zimmermann, 2012). Em vista disso, existem diferentes perfis interessados em informações sobre o produto de software e, que a partir do seu espaço de decisão por exemplo: i) patrocinadores decidindo sobre a dotação de orçamento para investimentos no desenvolvimento do produto software; ii) engenheiros de Software, envolvidos no desenvolvimento e manutenção, tomam decisões que tem impacto direto na qualidade interna e externa do produto de software; iii) gerentes decidem qual desenvolvedor deve ficar responsável por implementar determinada funcionalidade;

Nos últimos anos houve a ascendência da área chamada de *Software Analytics* (SA), cujas pesquisas focam na análise de dados extraídos de diferentes artefatos elaborados ao longo do ciclo de vida do software, portanto, uma abordagem orientada a dados, de forma a obter informações que direcionem as ações na tomada de decisões (Zhang et al., 2011). Abdellatif et al. (2015) enfatizam que SA se distingue da análise direta de um determinado artefato, porque trata da análise de muitos e diferentes artefatos, de forma a obter informações que apoiem a tomada de decisões sobre o produto de software. Nesse estudo secundário é possível observar ainda que o código-fonte é o artefato que desperta maior interesse por parte dos pesquisadores. Os autores também relatam que existem poucos trabalhos que focam em aspectos gerenciais.

Menzies et al. (2015) apresentaram uma compilação dos estudos, tanto no âmbito acadêmico, quanto da indústria, que visam endereçar as diferentes necessidades de informação de gestores e equipe de desenvolvimento. Nesse compêndio, há estudos voltados a construção de modelos que expressem o comportamento do software a partir de dados extraídos do produto. Já Bird et al. (2015) apresentam estudos que possuem o foco na criação de métodos e ferramentas para a mineração e extração de dados de artefatos de software. A partir dessas referências, é possível observar que há investigação da aplicação de SA em praticamente todas as etapas da produção do software.

Assim, um dos segredos do sucesso para adoção de SA em um ambiente de desenvolvimento é identificar as reais necessidades de informação da equipe, e traduzi-las em objetivos de medição e ações analíticas, de forma a produzir análises e interpretações úteis para os diferentes perfis (Zhang et al., 2011).

(em elaboração).... Reler e concluir subseção:

- (Dybå et al., 2012) - Contexto

- (Al-Daajeh et. al, 2012-b). – Trade-offs

- (Choma e Guerra, 2017) – Analytics Patterns

## Ciclo de Vida de Desenvolvimento de Sistemas Contemporâneos

O software enquanto produto, nos dias atuais, é executado em diferentes tipos de dispositivos, por exemplo: servidores, dispositivos móveis, console de jogos eletrônicos, automóveis. A partir de seu uso, o software produz dados em tempo real, o que, juntamente com dados obtidos ainda nos ciclos de desenvolvimento, possibilitam a realização de análises de dados sobre o produto. O software não é mais desenvolvido de forma centralizada e monolítica. A forma de se desenvolver software mudou. Tem sido cada vez mais percebida de maneira holística. De acordo com essa visão, as atividades e tarefas do processo de desenvolvimento são organizadas e executadas para estabelecer uma cadência contínua do fluxo do trabalho. Normalmente, tais ambientes oferecem um conjunto entrelaçado de atividades de desenvolvimento e implantação e tarefas organizadas e executadas continuamente e não mais estruturadas como um conjunto de fases, atividades e tarefas, organizadas e executadas discretamente (Fitzgerald e Stol, 2017). Ressalta-se, entretanto, que não há consenso na academia, tampouco na indústria, sobre a nova mentalidade(mindset) de como se desenvolver software. De um lado, há uma mentalidade orientada a processos, influenciada pela corrente de pensamento positivista. Do outro lado, por assim dizer, há aqueles que possuem uma mentalidade de fluxo de trabalho e entrega de releases numa cadência contínua, favorecendo a liberação da entrega de versões do produto de software em intervalos curtos e frequentes no tempo. Esse tipo de mentalidade é mais orientado às correntes de pensamento interpretativista e teoria crítica (Easterbrook et al., 2007). Tal visão se alinha a "escola de pensamento lean", originada na indústria de manufatura (Morgan e Liker, 2006)

Ao longo dos últimos 20 anos, na esteira dessa nova mentalidade e impulsionadas pelas práticas de engenharia de software e valores adotados pelas comunidades de desenvolvimento ágil (Beck K. et al., 2001) e de desenvolvimento lean (Poppendieck M. e Poppendieck T, 2006), além das comunidades de software livre, um novo termo passa ser estabelecido: a engenharia de software contínua. Passou-se a perceber que práticas dessas diferentes comunidades convergiam para o entendimento de que as atividades do processo de desenvolvimento devem ser continuamente executadas, considerando o alinhamento estratégico do negócio ao desenvolvimento do produto, com auto nível de automação e colaboração, alcançados por meio da utilização de práticas de DevOps (Fitzgerald e Stol, 2017) (Luz et al., 2018). Vale ressaltar que, Boehm (1978) já apontava que a limitação das tecnologias de automação daquela época impunha restrições ao avanço das pesquisas em qualidade de produto.

Aproximadamente nos últimos 10 anos, foram propostas estratégias de experimentação em engenharia de software, em larga escala, por diferentes organizações e grupos de pesquisa, como Microsoft (Kohavi et al., 2009), Google (Tang et al., 2010), , além do movimento Lean Startups (Bosh, 2012). A ideia básica é desenvolver uma nova versão de um produto (tratamento) e compará-la com a versão em uso (controle), por meio da execução de experimentos controlados, conhecido como testes do tipo A/B. A partir de uma amostra aleatorizada da população de usuários, as hipóteses definidas são verificadas e a decisão de qual versão será disponibilizada a população de usuários é tomada a partir dos resultados do teste estatístico de hipótese.

Posteriormente, essas estratégias de observação e tomada de decisão passaram a ser utilizadas em contextos de desenvolvimento contínuo. Plataformas tecnológicas livres e abertas utilizadas em grandes organizações foram desenvolvidas e disponibilizadas e uso da experimentação ganhou patamar de escala (Kohavi et al., 2013) (Bakshy et at., 2014). Segundo Kevic et al. (2017) ao longo de 2,5 anos foram executados 21.220 experimentos controlados somente no produto Bing.

A experimentação contínua, ainda possui o foco em observar características de qualidade em uso e com isso, prioriza a experiência do usuário que é medida a partir do uso do software. Portanto, a experimentação contínua pressupõe a mentalidade de que toda mudança no produto requer a definição de hipóteses que possibilitem avaliar os impactos da nova versão para o usuário final.

A questão importante a ser destacada nesta pesquisa, em relação aos ambientes de engenharia de software contínua, é que, a característica de lançamentos frequentes e em ciclos curtos de desenvolvimento, melhora a capacidade de observação e compreensão do fenômeno, uma vez que aumenta a quantidade de pontos de observação, ao longo do tempo. Com isso, a capacidade de intervenção do time de desenvolvimento é antecipada, frente a novas versões corretivas ou evolutivas. Assim, não se posterga ou retarda as intervenções, o que também contribui para minimizar os impactos dos riscos financeiros no custo do desenvolvimento ou manutenção. Portanto, embora a utilização do MeasureSoftGram não seja restrita a ambientes de engenharia de software contínua, são nesses ambientes que se obtém maior acurácia com seu uso. A Figura 3 ilustra esse *rationale*.

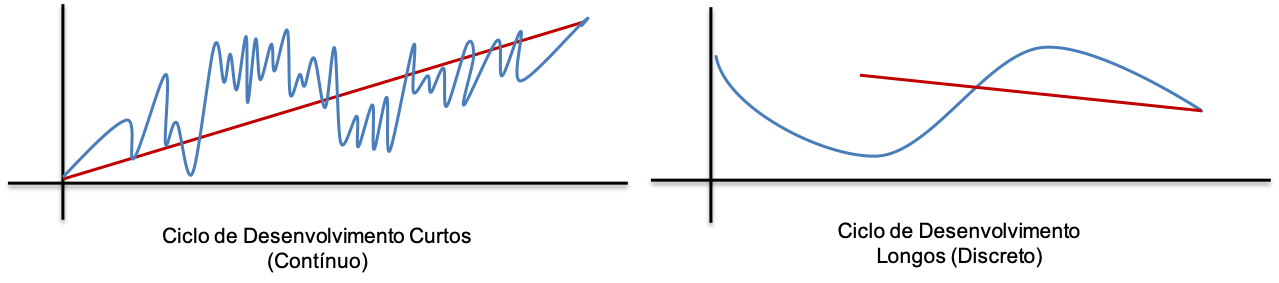


Figura 5: Pontos de observação da qualidade de versões de produtos de software

# Referências Bibliográficas

ABDELLATIF, T. M.; CAPRETZ, L. F.; HO, D. Software Analytics to Software Practice: A Systematic Literature Review. Proceedings of the First International Workshop on BIG Data Software Engineering. Anais...: BIGDSE ’15.Piscataway, NJ, USA: IEEE Press, 2015 Disponível em: <http://dl.acm.org/citation.cfm?id=2819289.2819300>

ACAR, E.; Canan, A-B; Haluk, B.; Rasmus, B., Bülent, Y. 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. Disponível 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-WWW ’15 Companion.New York, NY, USA: ACM, 2015 Disponível em: <http://doi.acm.org/10.1145/2740908.2743060>

ALDAAJEH, S. H.; AL-QUTAISH, R. E.; AL-QIREM, F. A tactic-based framework to evaluate the relationships between the software product quality attributes. International Journal of Software Engineering, v. 5, n. 1, p. 5–26, 2012.

ALDAAJEH, S. et al. Communing Different Views on Quality Attributes Relationships’ Nature. European Journal of Scientific Research, v. 68, n. 1, p. 101–109, 2012.

AKIYAMA, F. An Example of Software System Debugging. Inform Process Congress. p. 353-359, 1971

BAKSHY, E.; ECKLES, D.; BERNSTEIN, M. S. Designing and Deploying Online Field Experiments. Proceedings of the 23rd ACM conference on the World Wide Web. Anais...2014

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

BASILI, V. R. A Personal Perspective on the Evolution of Empirical Software Engineering. In: Perspectives on the Future of Software Engineering. [s.l.] Springer Berlin Heidelberg, 2013. p. 255–273.

BECK, K. et al. Manifesto for Agile Software DevelopmentManifesto for Agile Software Development, 2001. Disponível em: <http://www.agilemanifesto.org/>

BIRD, C.; MENZIES, T.; ZIMMERMANN, T. The Art and Science of Analyzing Software Data. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

BOEHM, B. W. Characteristics of Software Quality. [s.l.] North-Holland, 1978.

BOEHM, B. W.; BROWN, J. R.; LIPOW, M. Quantitative Evaluation of Software Quality. Proceedings of the 2Nd International Conference on Software Engineering. Anais...: ICSE ’76.Los Alamitos, CA, USA: IEEE Computer Society Press, 1976Disponível em: <http://dl.acm.org/citation.cfm?id=800253.807736>

BOSCH, J. Building Products as Innovation Experiment Systems. (M. A. Cusumano, B. Iyer, N. Venkatraman, Eds.)Software Business. Anais...Berlin, Heidelberg: Springer Berlin Heidelberg, 2012

BOSCH, J. et al. The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups. (B. Fitzgerald et al., Eds.)Lean Enterprise Software and Systems. Anais...Berlin, Heidelberg: Springer Berlin Heidelberg, 2013

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

BUSE, R. P. L.; ZIMMERMANN, T. Analytics for Software Development. Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. Anais...: FoSER ’10.New York, NY, USA: ACM, 2010 Disponível em: <http://doi.acm.org/10.1145/1882362.1882379>

CHOMA, J.; GUERRA, E. M.; DA SILVA, T. S. Patterns for Implementing Software Analytics in Development Teams. Proceedings of the 24th Conference on Pattern Languages of Programs-PLoP ’17.USA: The Hillside Group, 2017 Disponível em: <http://dl.acm.org/citation.cfm?id=3290281.3290308>

CZERWONKA, J. et al. CODEMINE: Building a Software Development Data Analytics Platform at Microsoft. [S.l.]: IEEE Computer Society. July 2013

DE FRANÇA, B. B. N.; JERONIMO JUNIOR, H.; TRAVASSOS, G. H. Characterizing DevOps by Hearing Multiple Voices. Proceedings of the 30th Brazilian Symposium on Software Engineering. Anais...: SBES ’16.New York, NY, USA: ACM, 2016Disponível em: <http://doi.acm.org/10.1145/2973839.2973845>

DE MENDONCA, Vinicius Rafael Lobo et al. Static analysis techniques and tools: A systematic mapping study. 8th International Conference on Software Engineering Advances-ICSEA, 2013.

DEBOIS, P. Devops: A software revolution in the making. The Journal of Information Technology Management, v. 24, p. 3–5, 2011.

DIMITRI N., Kleanthis N. Mokios, Nicholas D. Sidiropoulos, and Alexandros Potamianos. 2010. Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures. IEEE Transactions on Audio, Speech, and Language Processing, 18, 6 (2010), 1193–1207. DOI: http://dx.doi.org/10.1109/tasl.2009.2031694

DINGSØYR, T. D. T. Agile Project Management: From Self-Managing Teams to Large-Scale Development. 37th IEEE/ACM International Conference on Software Engineering, {ICSE} 2015, Florence, Italy, May 16-24, 2015, Volume 2. Anais...2015Disponível em: <https://doi.org/10.1109/ICSE.2015.299>

DROMEY, R. G. (1995). A model for software product quality. IEEE Transactions on Software Engineering, 21(2), 146–162. doi:10.1109/32.345830.

EASTERBROOK, S. et al. Selecting Empirical Methods for Software Engineering Research. [s.l.] Springer, 2007.

ELBERZHAGER, Frank; MÜNCH, Jürgen; NHA, Vi Tran Ngoc. 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.

EVANGELIA Pantraki and Constantine Kotropoulos. 2015. Automatic image tagging and recommendation via PARAFAC2. In 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP’15). IEEE, 1–6. DOI:http://dx.doi.org/10.1109/mlsp.2015.7324363

ACAR E., Canan Aykut-Bingol, Haluk Bingol, Rasmus Bro, and Bu ̈lent Yener. 2007. Multiway analysis of epilepsy tensors. Bioinformatics 23, 13 (2007), i10–i18. DOI:http://dx.doi.org/10.1093/ bioinformatics/btm210

FAGERHOLM, F.; GUINEA, A.S.; Mäenpää, H.; Munch J. The RIGHT Model for Continuous Experimentation. Journal of Systems and Software, v. 123, p. 292–305, jan. 2017.

FABIJAN, A.; Pavel D.; Olsson, H. H.; Bosch, J., The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-driven Organization at Scale. Proceedings of the 39th International Conference on Software Engineering. Anais...: ICSE ’17.Piscataway, NJ, USA: IEEE Press, 2017 Disponível em: <https://doi.org/10.1109/ICSE.2017.76>

FELDERER, M. et al. Industrial Evaluation of the Impact of Quality-driven Release Planning. Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM, New York, USA, 2014, Disponível em: <http://doi.acm.org/10.1145/2652524.2652579>

FENTON, N. & B. 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, n. Complete, p. 176–189, 2017.

HANSEN, S.; PLANTENGA, T. D.; KOLDA, T. G. Newton-based optimization for Kullback-Leibler nonnegative tensor factorizations. Optimization Methods and Software, v. 30, n. 5, p. 1002–1029, 2015.

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, 12th International Conference on Software Quality

HOOVER, C. L.; ROSSO-LLOPART, M.; TARAN, G. Evaluating Project Decisions: Case Studies in Software Engineering. [s.l.] Addison-Wesley, 2010.

INSTITUTE, P. M. Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK). Pennsylvania, 2014.

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

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

ISO/IEC:15939 Systems and Software Engineering - Measurement Process. International Organization for Standardization and International Electrotechnical Commission. [S.l.]. 2017

JURISTO, N.; MORENO, A. M. Basics of Software Engineering Experimentation. 1st. ed. [s.l.] Springer Publishing Company, Incorporated, 2010.

KAI-WEI Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP’13). 1602–1612

KEVIC, K.; Kevic, K.; Murphy, B.; Williams, L.; Beckmann, J.; Characterizing Experimentation in Continuous Deployment: A Case Study on Bing. Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track-ICSE-SEIP ’17.Piscataway, NJ, USA: IEEE Press, 2017 Disponível 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. Controlled Experiments on the Web: Survey and Practical Guide. Data Min. Knowl. Discov., v. 18, n. 1, p. 140–181, fev. 2009.

KOHAVI, R.; LONGBOTHAM, R.; WALKER, T. Online Experiments: Practical Lessons. IEEE Computer, v. 43, p. 82–85, 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-KDD, New York, NY, USA: ACM, 2013 Disponível em: <http://doi.acm.org/10.1145/2487575.2488217>

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, 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

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

LUZ, W. P.; PINTO, G.; BONIFÁCIO, R. Building a Collaborative Culture: A Grounded Theory of Well Succeeded Devops Adoption in Practice. Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement-ESEM, New York, NY, USA: ACM, 2018 Disponível em: <http://doi.acm.org/10.1145/3239235.3240299>

MARINHEIRO, A.; BERNARDINO, J. OpenBRR Evaluation of an Open Source BI Suite. Proceedings of the International C\* Conference on Computer Science and Software Engineering. Anais...: C3S2E ’13.New York, NY, USA: ACM, 2013Disponível em: <http://doi.acm.org/10.1145/2494444.2494463>

MCCABE, T. J. A Complexity Measure. IEEE Trans. Softw. Eng., v. 2, n. 4, p. 308–320, jul. 1976.

MCCALL, J.; K. RICHARDS, P.; F. WALTERS, G. Factors in Software Quality. Volume I, II and III. Concepts and Definitions of Software Quality. US Rome Air Development Center Reports, US Department of Commerce, USA, p. 168, 1977.

MENEELY, A.; SMITH, B.; WILLIAMS, L. Validating Software Metrics: A Spectrum of Philosophies. ACM Transactions on Software Engineering and Methodology (TOSEM)., v. 21, n. 4, p. 24:1--24:28, fev. 2013.

MENZIES, T. et al. Sharing Data and Models in Software Engineering. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

MENZIES, T.; ZIMMERMANN, T. Software Analytics: So What? {IEEE} Software, v. 30, n. 4, p. 31–37, 2013.

MIGUEL, J. P.; MAURICIO, D.; RODRIGUEZ, G. A Review of Software Quality Models for the Evaluation of Software Products. CoRR, v. abs/1412.2977, 2014.

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

MORGAN, J.; LIKER, J. K. The Toyota Product Development System: Integrating People, Process, and Technology. [s.l.] Taylor & Francis, 2006.

NEELY, S.; STOLT, S. Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy). 2013 Agile Conference, {AGILE} 2013, Nashville, TN, USA, August 5-9, 2013. Anais...2013Disponível em: <https://doi.org/10.1109/AGILE.2013.17>

OLSSON, H. H.; ALLAHYARI, H.; BOSCH, J. Climbing the “Stairway to Heaven” - A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. (V. Cortellessa, H. Muccini, O. Demirörs, Eds.)EUROMICRO-SEAA. Anais...IEEE Computer Society, 2012Disponível em: <http://dblp.uni-trier.de/db/conf/euromicro/euromicro2012.html#OlssonAB12>POPPENDIECK

OUHBI, S. et al. Evaluating Software Product Quality: A Systematic Mapping Study. 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement. Anais...out. 2014

PAGE, L.; Brin, S.; Motwani, R.; Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web**.** [s.l.] Stanford InfoLab, nov. 1999. Disponível em: <http://ilpubs.stanford.edu:8090/422/>.

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. Disponível em: <https://doi.org/10.1109/MLSP.2015.7324363>

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.

POPPENDIECK, M.; POPPENDIECK, T. Implementing Lean Software Development: From Concept to Cash (The Addison-Wesley Signature Series). [s.l.] Addison-Wesley Professional, 2006.

Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. 2015. A study of distinctiveness in web results of two search engines. In 24th International Conference on World Wide Web, Web Science Track. ACM. DOI:http://dx.doi.org/10.1145/2740908.2743060

REPÚBLICA, P. Lei na 8.666/93, de 21 de Junho de 1993, 1993. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/L8666cons.htm>

Samantha Hansen, Todd Plantenga, and Tamara G. Kolda. 2015. Newton-based optimization for Kullback- Leibler nonnegative tensor factorizations. Optimization Methods and Software 30, 5 (April 2015), 1002– 1029. DOI:http://dx.doi.org/10.1080/10556788.2015.1009977

SAMOLADAS, I. et al. The SQO-OSS quality model: measurement based open source software evaluation. Open source development, communities and quality. Anais...2008

SCHERMANN, G. et al. We’re doing it live: A multi-method empirical study on continuous experimentation. Information and Software Technology, v. 99, p. 41–57, jul. 2018.

SJØBERG, T. D. 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>

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.

SLTI/MP. Instrução Normativa No 04, Secretaria de Logística de Tecnologia da Informação, Ministério do Planejamento 2014.

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

TANG, D. et al. Overlapping Experiment Infrastructure: More, Better, Faster Experimentation. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Anais...: KDD ’10. New York, NY, USA: ACM, 2010, Disponível em: <http://doi.acm.org/10.1145/1835804.1835810>

TCU2471 TRIBUNAL DE CONTAS DA UNIÃO. Acórdão 2.471/2008- Plenário, 2008. Disponível em:<http://portal3.tcu.gov.br/portal/page/portal/TCU/comunidades/tecnologia_informacao/boas_praticas/TCUacordao2471.pdf>

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

WOHLIN, C.; AURUM, A. Towards a Decision-making Structure for Selecting a Research Design in Empirical Software Engineering. **Empirical Softw. Engg.**, v. 20, n. 6, p. 1427–1455, dez. 2015.

Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’14). ACM, New York, NY, 25–34. DOI:http://dx.doi.org/10.1145/2623330.2623656

ZHANG, D. et al. Software Analytics As a Learning Case in Practice: Approaches and Experiences. Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering. Anais...: MALETS ’11.New York, NY, USA: ACM, 2011 Disponível em: <http://doi.acm.org/10.1145/2070821.2070829>

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-KDD ’14.New York, NY, USA: ACM, 2014 Disponível em: <http://doi.acm.org/10.1145/2623330.2623656>

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>

WASSERMAN, A.; PAL, M.; CHAN, C. Business Readiness Rating for Open Source. Proceedings of the EFOSS Workshop. Anais...Como, Italy: [s.d.]

1. <https://www.sonarqube.org/> [↑](#footnote-ref-2)
2. <https://www.cqse.eu/en/products/teamscale/landing/> [↑](#footnote-ref-3)
3. <https://codeclimate.com/> [↑](#footnote-ref-4)
4. <https://jazz.net/products/rational-team-concert/> [↑](#footnote-ref-5)
5. <https://marketplace.visualstudio.com/items?itemName=VisualStudioPlatformTeam.MicrosoftCodeAnalysis2017> [↑](#footnote-ref-6)
6. <http://mathworld.wolfram.com/Lattice.html> [↑](#footnote-ref-7)