# Introdução

## Introdução

A qualidade de produto de software é objeto de estudo da engenharia de software há aproximadamente quatro décadas. A partir dos primeiros e seminais modelos de avaliação de qualidade de produto de software, definidos por Boehm et al. (1976) e McCall et al. (1977), os subsequentes foram inspirados na proposta de hierarquização de características/fatores e subcaracterísticas (atributos de qualidade), propostas por estes, onde essa organização favorece a observação do fenômeno da qualidade de produto de software e estabelece conexões lógicas com as descrições das características e subcaracterísticas (“teorias”).

Esse conhecimento acumulado do longo do tempo é evidenciado por: diferentes modelos de avaliação de qualidade de produto propostos; um vasto conjunto de métricas e medidas foram definidas (Kitchenham, 2010) (Elberzhager et al., 2012) (Tahir e MacDonell, 2012) (Mendonça et al., 2013); estudos sobre os relacionamentos entre diferentes atributos de qualidade (características e subcaracterísticas) evidenciaram a importância e necessidade de se entender os *tradeoffs* associados, uma vez que, implica em riscos financeiros, com impacto direto no custo do desenvolvimento (Al-Daajeh et. al, 2012-b); a utilização de abordagens orientadas a análise de dados, de forma a apoiar a tomada decisão técnico-gerencial, acerca da qualidade de produto sistemas de software (Zhang et al., 2011) (Abdellatif et al. 2015); 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, as diferentes necessidades de informação de engenheiros de software, gerentes de produto, usuários finais e demais interessados nas informações sobre o produto (Buse e Zimmermann, 2012) (Sjoberg et al. 2012) e seu alinhamento com a estratégia de negócio das organizações.

Ainda que os avanços sejam notáveis, os estudos sempre observaram a qualidade de produto em uma perspectiva unidimensional, focando preferencialmente em características relacionadas à qualidade interna. Argumentamos que as relações entre as características da qualidade devem ser explicitamente expostas e analisadas simultaneamente, por meio de uma abordagem orientada a dados, a fim de, capturar efetivamente todo o espectro de qualidade. Essa é a principal diferença do modelo proposto nesta pesquisa, em relação aos estudos prévios que tivemos conhecimento até o presente momento.

Considerando o exposto, apresentamos o MeasureSoftGram. As contribuições essenciais dessa proposta de tese de doutrado 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, poderação e agregação), em espaços multidimensionais, construído a partir da aplicação de conceitos da álgebra multilinear. Importante ressaltar que esse conjunto de regras é válido para qualquer quantidade de métricas ou medidas. Ou seja, o modelo opera o mesmo conjunto de regras transformações caso possua apenas uma única medida ou, caso possua centenas;
3. um mecanismo, que chamamos de equalizador da qualidade, para expor os relacionamentos entre as 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. Por conseguinte, visa apoiar os tomadores de decisão.

Outras características do modelo:

* O sistema trabalha com uma escala razão, com valores que variam entre [0,...,1], sendo 0-percepção de qualidade mínima e 1-percepção de qualidade máxima;
* O modelo de análise contempla um método de medição que propicia coletar, tratar e transformar, medidas básicas e derivadas, expressas nos atributos da qualidade, descritas nos modelos de referência;
* Todas medidas são normalizadas, com base em critérios de decisão que podem ser definidos: a partir de opinião de especialistas; mineração e análise estatística de dados históricos do próprio produto ou em repositórios de software que fornecem grandes volumes de dados, como o Github[[1]](#footnote-2), por exemplo, que hospeda mais de 100 milhões de projetos de software. Dessa forma, o modelo de análise viabiliza tipos de medição subjetivos e objetivos e que, por fim, proporciona que coleta seja realizada de forma automática ou manual;
* As mesmas regras de agregação são aplicadas em qualquer nível de herarquia dos atributos de qualidade. Como o modelo faz uso de uma abordagem *botton-up*, então as medidas são agregadas até o último nível hierárquico dos atributos de qualidade. Elas se dão por meio de aplicações matemáticas em espaços tensoriais, como a norma tensorial[[2]](#footnote-3) e o produto de tensores (Kronecker)[[3]](#footnote-4). Na Figura 1 apresentamos uma abstração simplificada do MeasureSoftGram.

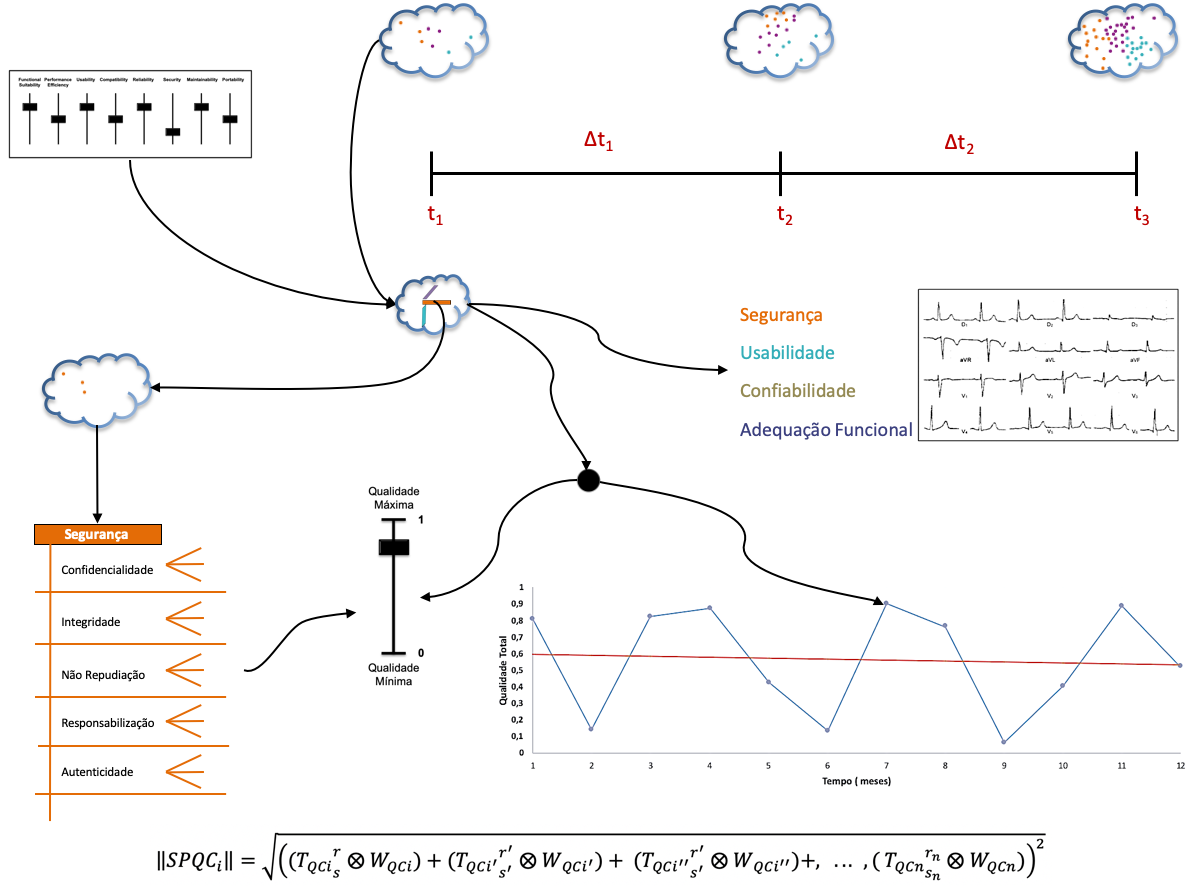


Figura 1: Ilustração simplificada da proposta

## Motivação

Analisando estudos dos modelos de qualidade de produto de software percebemos que: i) há pouca evidência experimental que suporte a análise das relações entre as características de qualidade (Henningsson e Wohlin, 2002); ii) existe uma dificuldade em agregar estudos sobre os relacionamentos entre as características de qualidade de produto (Svahnberg e Henningsson, 2009) ; iii) as evidências acerca das relações entre as características de qualidade, em sua maioria, são descritas em alto nível e foram obtidas por meio de *surveys*. Em alguns desses estudos a população foi de profissionais da indústria, em outros, de artigos (Al-Daajeh et al., 2012-a) (Aldaajeh, 2012) (Haoues 2017); iv) não foram encontradas evidências sobre o uso de dados reais de produtos de software na análise dos relacionamentos entre as características e, principalmente, a qualidade de produto de software vem sendo observada de maneira unidimensional, ou seja, observando uma ou duas características de qualidade por tempo, sem considerar os efeitos colaterais ou influências nas demais características. Essas lacunas nos levam a refletir sobre quão fidedigna, de fato, é a representação da realidade desse fenômeno, no que diz respeito a tomada de decisões assertivas(boas) sobre a aceitação de releases.

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

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

Criados a partir dos modelos inicialmente propostos por por Boehm et al. (1976) e McCall et al. (1977), além daquele proposto por Dromey (1995), os modelos normatizados, ISO/IEC 9126 (2001) e sua atualização ISO/IEC 25010 (2010), tornaram-se referência conceitual e normativa. Entretanto, existem vários outros propostos a partir da adaptação desses, como por exemplo, aqueles que propuseram novas características, específicas do contexto de software livre, como exemplos o OpenBRR (Wasserman, 2006) e SQO-OSS (Samoladas, 2008), além de outros apresentados na revisão realizada por Miguel J. et al. (2014).

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[[4]](#footnote-5); TeamScale[[5]](#footnote-6); CodeClimate[[6]](#footnote-7); IBM Rational Team Concert[[7]](#footnote-8) e Microsoft Code Analysis[[8]](#footnote-9). 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.

Ademais, 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[[9]](#footnote-10). 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.

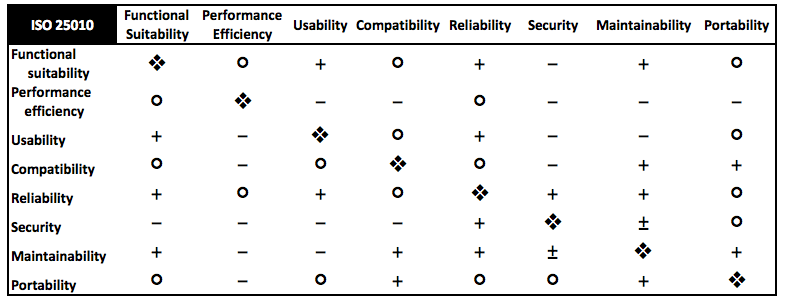


Figura 2: Relacionamento entre as características de qualidade de produto da ISO 25010

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.

(em elaboração).... Aqui serão introduzidas as necessidades de informação endereçam os objetivos de medição e como o analytics tem contribuído para isso

## 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*.

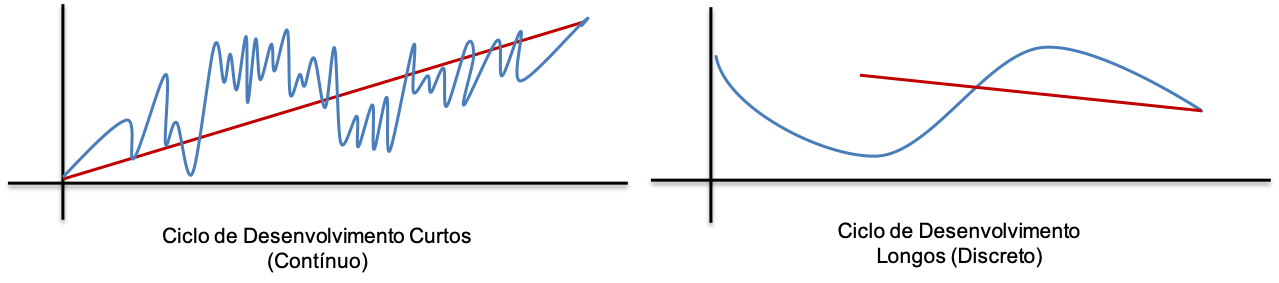


Figura 3: Pontos de observação da qualidade de versões de produtos de software

## Delineamento do problema

Em nossa sociedade contemporânea, as características pervasivas e ubíquas do software levam o software a "todo lugar". O software roda em servidores, dispositivos móveis, sistemas embarcados e caminha para uma nova realidade onde não só mais o software se comunicará com software, mas coisas se comunicarão com coisas. Além disso, informações geradas por usuários a respeito do uso do software estão disponíveis em fóruns de discussão específicos, redes sociais, lojas virtuais que disponibilizam produtos de software. Diante disso, analisar tão somente aspectos de qualidade interna, externa e em uso, isoladamente, não é suficiente para que um gestor de produto ou engenheiros de software decidam sobre a aceitação ou não de um release.

A falta de entendimento acerca dos dados gerados ao longo do ciclo de vida do software é muitas vezes a causa de más decisões. Aliado a isso, a falta de qualificação dos dados previamente às análises contribui sobremaneira a análises e interpretações errôneas. Um desafio comum para os profissionais da área de software é a identificação e correção de más decisões antes que estas possam produzir efeitos indesejáveis. Isso provavelmente ocorre porque é mais fácil julgar decisões em engenharia de software após o acontecimento do fato, do que monitorar e avaliar, antes que seu impacto seja conhecido (Hoover et al., 2010).

Portanto, a falta de compreensão ou clareza sobre as informações que dizem respeito às características da qualidade de produto do software, principalmente a falta de conhecimento sobre suas relações e influências mútuas, contribui para a tomada de decisões não assertivas (ruins ou erradas), o que compromete:

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

Finalmente, a observação da qualidade de produto de software de forma unidimensional (linear), impede a percepção das relações entre suas características e subcaracterísticas de forma conjunta, simultânea. Portanto, os efeitos das relações e também os colaterais, são desconhecidos e isso afeta a qualidade. Consequentemente, compromete a tomada de decisão.

## Objetivo e Questão da Pesquisa

Acreditamos que seja necessário envidar esforços em trazer a observação da qualidade de produto de software para essa nova realidade de mundo, de forma a apoiar a tomada de decisão de aceitação de releases, tanto pelas equipes de desenvolvimento quanto pelos gestores de desenvolvimento de produto, a partir de evidências. Para tanto, definimos o seguinte objetivo:

|  |  |
| --- | --- |
| **Analisar** | características de qualidade de sistemas de software (produto e uso) de forma multidimensional |
| **com o propósito** | de caracterizar fatores, medidas e suas relações, a partir de dados obtidos de versões candidatas de produtos de software (desenvolvimento e uso) de modo a apoiar a tomada de decisão, técnico-gerencial, sobre a aceitação de versões finais de produto |
| **com respeito à** | eficiência |
| **do ponto de vista** | do pesquisador |
| **no contexto** | de repositórios de projetos de software livre |

Tendo em vista que: i) o problema levantado afeta a avaliação da qualidade de projetos e produtos de software e ii) organizações que lidam com o desenvolvimento de software necessitam compreender melhor os fatores que afetam a qualidade do produto de software, além de suas relações e influências mútuas, a questão de pesquisa norteadora e ser respondida nesta proposta de tese é:

*Como observar a qualidade de produto de software, de forma multidimensional, de modo a apoiar a tomada de decisão técnico-gerencial, com relação à aceitação de versões de produtos de software?*

## Metodologia

De acordo com Wohlin e Aurum (2015) existem diferentes pontos de decisão na definição de uma metodologia de pesquisa, que podem ser agrupadas em três grandes fases: estratégica, tátical e operacional.

(em elaboração) ....

# 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. Anais...2012Disponível 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, 2010Disponível em: <http://doi.acm.org/10.1145/1882362.1882379>

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 Software Engineering - Software Measurement Process. International Organization for Standardization and International Electrotechnical Commission. [S.l.]. 2002

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>

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.

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

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, 2011Disponível 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://github.com> [↑](#footnote-ref-2)
2. <http://mathworld.wolfram.com/Norm.html> [↑](#footnote-ref-3)
3. <http://mathworld.wolfram.com/KroneckerProduct.html> [↑](#footnote-ref-4)
4. <https://www.sonarqube.org/> [↑](#footnote-ref-5)
5. <https://www.cqse.eu/en/products/teamscale/landing/> [↑](#footnote-ref-6)
6. <https://codeclimate.com/> [↑](#footnote-ref-7)
7. <https://jazz.net/products/rational-team-concert/> [↑](#footnote-ref-8)
8. <https://marketplace.visualstudio.com/items?itemName=VisualStudioPlatformTeam.MicrosoftCodeAnalysis2017> [↑](#footnote-ref-9)
9. <http://mathworld.wolfram.com/Lattice.html> [↑](#footnote-ref-10)