Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Curso de Engenharia Software

UM MAPEAMENTO SISTEMÁTICO DE MÉTRICAS PARA METODOLOGIAS ÁGEIS SCRUM, KANBAN E XP

Autor: Breno Dantas Cruz
Orientadora: Fabiana Freitas Mendes

Brasília, DF
2013

BRENO DANTAS CRUZ

UM MAPEAMENTO SISTEMÁTICO DE MÉTRICAS PARA METODOLOGIAS ÁGEIS SCRUM, KANBAN E XP

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Orientadora:
MsC. Fabiana Freitas Mendes

Brasília, DF
2013

CIP – Catalogação Internacional da Publicação*

Cruz, Breno.

Um Mapeamento Sistemático De Métricas Para Metodologias Ágeis Scrum, Kanban e XP / Breno Dantas Cruz. Brasília: UnB, 2013. 103 p. : il. ; 29,5 cm.

Monografia (Graduação) – Universidade de Brasília

Faculdade do Gama, Brasília, 2013. Orientação: Fabiana Freitas Mendes

1. Metodologias ágeis de desenvolvimento de software 2. Métricas 3. Qualidade 4. Revisão Sistemática 5. Kanban 6. Extreme Programing 7. XP 8. Scrum

I. Mendes, Fabiana Freitas. II. Msc.

CDU Classificação

- A ficha catalográfica oficial deverá ser solicitada à Biblioteca pelo aluno após a apresentação.

UM MAPEAMENTO SISTEMÁTICO DE MÉTRICAS PARA METODOLOGIAS ÁGEIS SCRUM, KANBAN E XP

Breno Dantas Cruz

Monografia submetida como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software da Faculdade UnB Gama - FGA, da Universidade de Brasília, em ___/___/___.apresentada e aprovada pela banca examinadora abaixo assinada:

Prof[a]. Ms: Fabiana Freitas Mendes, UnB/ FGA
Orientadora

Prof. Ms: Ricardo Ajax Dias Kosloski, UnB/ FGA
Membro Convidado

Prof[a]. Ms: Elaine Venson, UnB/FGA
Membro Convidado

Brasília, DF
2013

DEDICATÓRIA

Esse trabalho é dedicado ao Senhor.

AGRADECIMENTOS

Agradeço ao meu Deus, o Eu Sou, que me livrou da morte, me levou para locais que eu nunca sonharia de ir, abençoou e me alargou as fronteiras, que estendeu a sua mão e me preservou do mal, de modo que não me sobrevenho aflição.

Ao meu pai Rogério Luiz Veríssimo Cruz e a minha mãe Eliete Lisboa Dantas Veríssimo pelo amor, conselhos valiosos e pelo apoio sempre presente na minha vida.

A minha irmã Sara Dantas Cruz que mesmo a distância me incentivou a continuar sem desistir.

Aos meus tios e tias pelo braço forte, mão amiga, socorro nas horas de desespero com e os seus concelhos e exemplos motivadores.

Aos meus avôs e avós cujo exemplos e lições de vida me inspiraram muito.

Meus primos com o apoio nas horas de desânimo.

A Prof. Ms Fabiana Freitas Mendes que me orientou na segunda etapa do meu trabalho com as suas sugestões precisas, as quais enriqueceram, sobremaneira, o conteúdo deste trabalho.

E ao Prof. Ms Ricardo Ajax Dias Kosloski que me orientou e auxiliou na primeira etapa deste trabalho.

EPÍGRAFE

Quanto melhor é adquirir a sabedoria do que o ouro! e quanto mais excelente é escolher o entendimento do que a prata - Provérbios 16:16

RESUMO

A adoção de métodos ágeis vem crescendo nos últimos anos, em especial no governo brasileiro observa-se principalmente a adoção das metodologias ágeis *Scrum*, *Kanban* e *Extreme Programming*. Com base neste contexto, foi realizado um mapeamento sistemático com o objetivo de investigar quais métricas podem ser utilizadas. As métricas encontradas foram apresentadas usando o modelo de definição de métricas da ISO/IEC 9126 e foram, também, organizadas considerando as classes que seguem a definição de Categorias de Informação do *Practical* Software *Measurement*. Além disso, um exemplo de detalhamento de métricas é apresentado mostrando como elas poderiam ser utilizadas em um contexto real. Como trabalho futuro foi identificado a possibilidade de realização de trabalhos que busquem métricas para outras metodologias, por exemplo *Feature Driven Development* ou Família *Crystal*; Além de trabalhos que busquem mais informações para as métricas encontradas.

Palavras-chave: Metodologias Ágeis; Revisão Sistemática; Mapeamento Sistemático; Métricas; Categoria de Informação; ISO/IEC 9126; *Scrum*; *Kanban*; *Extreme Programming*; XP; Método Ágil;

ABSTRACT

The adoption of agile methods has been growing in recent years. The Brazilian government mainly uses the agile methodologies Scrum, Kanban and Extremme Programing. Based on this context, it was performed a systematic mapping in order to investigate what metrics are used for those methodologies. The found metrics were presented using the metrics model from ISO / IEC 9126 and were also organized in classes, which follows the definition of information categories from the Practical Software Measurement. It is also presented an example of detailed metrics to show how they would be used in a real context. As future work it is possibility to conduct studies that seek metrics to other methodologies, such as Feature Driven Development or Crystal Family. It is also possible to perform studies, which seek more information about the found metrics.

Keywords: Agile methodologies; Systematic Review; Systematic Mapping; Metrics; Information Category; ISO / IEC 9126; Scrum; Kanban; Extreme Programming; XP; Agile Method;

LISTA DE ILUSTRAÇÕES

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Neste documento é relatada a execução de um mapeamento sistemático que criou uma lista de métricas utilizadas em processos ágeis de desenvolvimento de software. Nessa pesquisa focou-se nas metodologias *Scrum*, *Extreme Programming* (*XP*) pois são algumas das mais utilizadas no contexto de produção de software [1].

A lista de métricas obtida é importante, pois pode ser utilizada como insumo para a definição de métricas em um projeto de software auxiliando, assim, empresas a obterem maior qualidade para os seus produtos de software.

1.1 CONTEXTO E MOTIVAÇÃO DO ESTUDO

Os sistemas de software têm sido utilizados extensamente na grande maioria das atividades humanas, desde as mais simples até as mais complexas e críticas. Os produtos de software se fazem cada vez mais presentes, tornando seus usuários mais produtivos e efetivos na realização de suas atividades rotineiras ou específicas, relativas às suas atuações profissionais. Os segmentos comerciais, bancários, administrativos, governamentais e científicos têm se valido da utilização de produtos de software [2, 4 – 11p].

Sommerville et al [2, 4 – 11p], cita diversas aplicações em software no mercado, tais como: sistemas de software, aplicações em tempo real, aplicações de software embarcados, aplicações de uso pessoal, aplicações de negócios e, mais recentemente, aplicações web. A partir dessas aplicações, depreende-se que o desenvolvimento de software é um importante indutor de negócios. Tal fato pode ser comprovado pela atuação de organizações de grande porte nessa área há vários anos. Esse é o caso, por exemplo, da IBM, Microsoft, Apple, Oracle, dentre outras [3, 20 – 28p].

A demanda por metodologias para o desenvolvimento de software no mercado, impulsionaram a busca de novas abordagens de desenvolvimento de software. As primeiras técnicas de desenvolvimento de software dificilmente poderiam ser consideradas como metodologias, pois cada empresa trabalhava independentemente para auferir lucro das tecnologias baseadas na computação [4].

Em busca de uma solução para a quantidade de problemas existentes nas primeiras metodologias de desenvolvimento, em 1969, durante a realização de conferência da Organização do Tratado do Atlântico Norte (OTAN), foi proposto o termo “Engenharia de Software”. E o primeiro modelo de Engenharia de Software foi

proposto, o modelo cascata “*Waterfall*”, aproveitado das técnicas já aplicadas em outras engenharias [2, 29 – 31p].

Segundo Sommerville [2, 30 - 32p] o modelo cascata é um exemplo de modelo orientado ao planejamento. No início do processo a equipe de desenvolvimento deve planejar todas as atividades, sendo contudo importante destacar que esse planejamento inicial é causa de um dos principais problemas desse modelo, pois compromissos devem ser feitos na fase inicial, fato que causa aumento dos custos para realização de mudanças, as quais ocorrem naturalmente no decorrer dos projetos.

Devido à formalidade do planejamento exigido pelo modelo cascata, associada às necessidades de mudanças inerentes aos projetos de software, ocorreu a busca por melhoria das metodologias de desenvolvimento, culminando no método de desenvolvimento incremental. Nesse modelo as atividades de especificação, desenvolvimento e validação, caminham juntas e com troca de *feedback* entre clientes e empresa, o que constitui uma característica importante também nas abordagens ágeis [2, 32 – 34p.].

As segundo Sommerville [2, 33 – 34p] as principais características do método incremental são:

- Custo reduzido para acomodar solicitações mudanças;
- Facilidade de troca de f*eedback* com consumidor;
- Entregas rápidas de software útil;

Entretanto, o método incremental também apresenta problemas do ponto de vista do gerenciamento. Estes estão relacionados ao fatos de o processo não ser completamente visível e à tendência de degradação da estrutura do sistema conforme novos incrementos são adicionados [2, 34p]. Também por conta da invisibilidade do processo, os gerentes necessitam de entregas constantes de artefatos para medir o progresso do projeto [2, 34p].

Segundo Larman [5, 59 - 61p], com o passar dos anos os desenvolvedores descobriram que o processo de desenvolvimento tradicional de software não é eficiente na prática, devido a sua alta taxa de falha. A grande quantidade de documentos, inerente ao processo tradicional, atrapalha o desenvolvimento, pois é necessário criar diversos artefatos antes de gerar o código.

Em 2001 um grupo de desenvolvedores experientes assinou o Manifesto Ágil [6] como forma de se antepor aos problemas e a ineficiências presentes no processo tradicional de desenvolvimento de software.

O Manifesto Ágil surge no contexto em que empresas estão buscando agilidade e código de alta qualidade, devido à maior exigência em termos de níveis de qualidade por seus produtos fornecidos [7]. Assim, houve interesse pelos princípios e valores propostos no movimento ágil [8] e consequente aumento na adoção de métodos ágeis em empresas produtoras de software [1].

Os métodos ágeis estão ganhando espaço também no Brasil. Em um recente documento do Tribunal de Contas da União (TCU), é relatado esse aumento bem como é dito que as principais metodologias ágeis utilizadas pela Administração Pública Federal são *Scrum*, *Extreme Programming* (XP) e *Kanban* [9]. Entretanto, apesar da crescente adoção destas metodologias, não há conhecimento difundido quanto a utilização de métodos ágeis.

Neste contexto, as métricas são importantes, pois fornecem a informação necessária para determinar a situação dos projetos executados. Assim, este trabalho possui como questão motivadora: quais são as métricas utilizadas para metodologias ágeis, em especial, para *Scrum*, *Kanban* e XP?

1.2 OBJETIVOS GERAL E ESPECÍFICOS.

Com base na questão apresentada anteriormente, foram estabelecidos os objetivos geral e específicos abaixo descritos, com o propósito de explorar a questão motivadora desta pesquisa.

Objetivo Geral: Definir uma lista de métricas que possa ser utilizada no contexto das metodologias ágeis aplicáveis as metodologias ágeis *Scrum*, *Kanban* e XP.

Para o cumprimento do objetivo geral do trabalho de pesquisa foram estabelecidos os seguintes objetivos específicos:

a. Estudar os principais conceitos relacionados aos temas
   Resultado esperado: revisão de conceitos de qualidade, medição e desenvolvimento ágil.
b. Identificar métricas utilizadas em projetos que usam metodologias ágeis.
   Resultado esperado: lista de métricas.

1.3 METODOLOGIA DE PESQUISA

Esta pesquisa possui as seguintes fases:

a. FASE 1 – REVISÃO BIBLIOGRÁFICA: Estudo teórico associado a cada um dos objetivos específicos. Tem como propósito dar o embasamento teórico para a realização do mapeamento sistemático e compreensão dos temas em estudo;
b. FASE 2 – DEFINIÇÃO DO PROTOCOLO DE MAPEAMENTO: tem como objetivo normatizar como será conduzida a pesquisa, o seu rigor e processos de inclusão e exclusão de fontes. O protocolo de mapeamento será definido a partir das informações obtidas da revisão bibliográfica preliminar;
c. FASE 3 – REALIZAÇÃO DE MAPEAMENTO SISTEMÁTICO: Aplicação do protocolo de mapeamento sistemático;
d. FASE 4 – ELABORAÇÃO DA LISTA DE MÉTRICAS: A partir das informações obtidas do mapeamento sistemático, elaborar a lista de métricas para as metodologias ágeis *Scrum*, XP e *Kanbam*;
e. FASE 5 – ANÁLISE DOS DADOS: Analisar as informações obtidas a partir do mapeamento sistemático.
    - Classificar as métricas quanto a Categoria de Informação correspondente;
    - Analisar a quantidade de métricas obtidas;
    - Analisar quais foram as métricas que foram encontradas mais vezes.

Conforme descrito por Moresi [10], esta pesquisa pode ser classificada como:

- Pesquisa Qualitativa, onde os dados são analisados indutivamente, tendo no processo e seu significado as principais abordagens;
- Pesquisa Telematizada e Bibliográfica, pois busca informações com o uso de computador e em informações publicadas em livros, revistas e artigos científicos.

1.4 ORGANIZAÇÃO DO TRABALHO

Com base nos objetivos gerais e específicos delimitados inicialmente, as informações presentes neste documento são divididas nos seguintes capítulos:

- Capítulo 2 – Conceitos Importantes: são expostos ao leitor conceitos de qualidade, método ágil de desenvolvimento de software, *Goal Question Metric*, *Practical Software Measurement* e metodologias ágeis de desenvolvimento de software *Scrum*, *Extreme Programming* e *Kanban*, necessários para compreensão deste trabalho;

- Capítulo 3 – Protocolo de Pesquisa: é apresentado o protocolo de revisão sistemática utilizado para a execução do mapeamento sistemático;
- Capítulo 4 – Relatório de Execução: são apresentados os resultados da execução do Protocolo de Pesquisa;
- Capítulo 5 – Conclusão: são apresentadas as considerações finais relativas ao trabalho com ênfase nos resultados obtidos.

2 CONCEITOS IMPORTANTES

A seguir serão os explicados os seguintes conceitos para melhor compreensão do trabalho:

- Revisão Sistemática;
- Qualidade;
- ISO/IEC 9126;
- Métodos de Desenvolvimento de Métricas;
- Método Ágil de Desenvolvimento de Software;
- Abordagens Ágeis de Desenvolvimento de Software.

Os conceitos tratados no presente capítulo são importantes, pois visam ambientar o leitor aos temas e objetivos utilizados no decorrer deste documento.

2.1 REVISÃO SISTEMÁTICA DE LITERATURA

Revisão Sistemática de Literatura (RS) é um termo utilizado para se referir a uma metodologia de pesquisa especificamente desenvolvida para coletar e avaliar evidências pertinentes a um certo tópico [11].

Uma RS identifica, avalia e interpreta todas as pesquisas relevantes para uma determinada questão, tópico ou fenômeno de interesse para a pesquisa. Ela também pode ser utilizada para examinar a extensão que evidências empíricas apoiam ou contradizem as hipóteses da pesquisa [12].

Embora a maioria das pesquisas iniciem com algum tipo de revisão literária, caso esta não apresente abrangência e imparcialidade ela possuirá baixo valor científico. O alto valor científico é a principal vantagem de uma RS, quando comparada a revisões literárias comuns. O seu valor é assegurado pela abrangência, imparcialidade e capacidade de auditagem, obtidos seguindo um procedimento para análise dos trabalhos coletados [11, 12, 13].

Uma revisão sistemática de literatura pode ser executada por diversas razões, alguns exemplos são [11, 13, 14]:

- Identificar evidências existentes para um tratamento ou tecnologia;
- Identificar alguma lacuna em certas áreas de pesquisa existentes como forma de embasar pesquisas mais aprofundadas;
- Promover suporte para novas áreas de pesquisa.

As principais diferenças entre revisões sistemáticas de literatura e revisões comuns são apresentadas na Tabela 1 [11, 13]:

Tabela 1. - Quadro comparativo entre modalidades de pesquisa bibliográfica adaptado de [11, 13]

Revisão Sistemática	**Revisão de Literatura**
Define obrigatoriamente um protocolo de revisão e métodos para revisão	Não define propriamente um protocolo e métodos de revisão
Estratégia para maximizar o número de pesquisas obtidas é obrigatória	Não há uma estratégia definida para maximizar o número de pesquisas
Rigor e abrangência	Rigor e abrangência questionáveis
Repetibilidade do processo pelo leitor	É bem provável que o processo não possa ser repetido
Requerem critérios de inclusão e exclusão explícitos de estudos	Os critérios de inclusão e exclusão não são bem explicitados
Explicitam critérios de qualidade de informação para as fontes	Os critérios de qualidade podem não ser explicitados

A Revisão Sistemática de Literatura envolve uma série de atividades. Diretrizes existentes para revisões sistemáticas possuem diferenças sutis quanto ao número e ordem das atividades. Conforme Kitchenham et al. [12, 13] os estágios de uma revisão sistemática se resumem, geralmente, a três fases principais que podem ser executadas diversas vezes para refinamento. Durante a execução das fases as informações são armazenadas em uma atividade de empacotamento [11].

Na Tabela 2 são apresentadas as fases do processo de revisão sistemática.

Tabela 2. - Fases e tarefas da Revisão Sistemática de Literatura adaptado de [11]

Planejamento	**Execução**	**Resultados**	**Empacotamento**
Identificação da necessidade de revisão	Identificação da pesquisa	Especificação dos mecanismos de disseminação	Armazenar as informações
Formalização do problema	Seleção dos estudos primários		
Especificação da questão de pesquisa	Estudo de validação de qualidade		
Definição do protocolo de revisão	Estudo de validação de qualidade		
Avaliação do protocolo de revisão	Extração de dados e monitoramento	Formatação do relatório principal	
	Síntese dos dados	Avaliação do relatório	

A Figura 1 ilustra a ordem de execução de cada uma das fases de uma Revisão Sistemática.

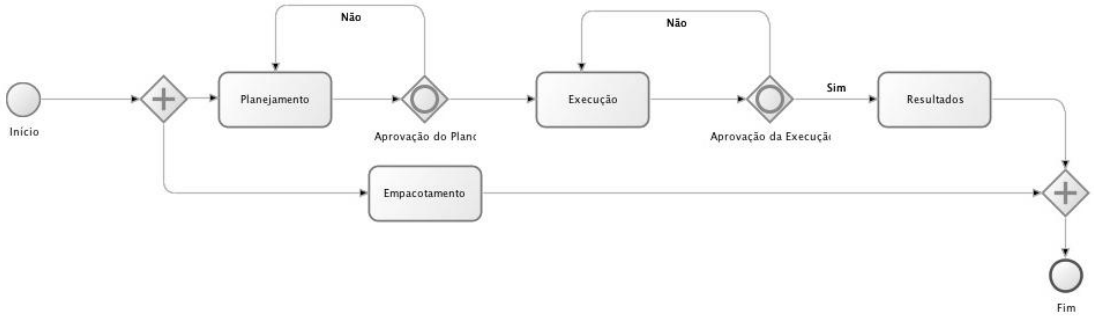

Figura 1. - Processo de Revisão Sistemática adaptado de [11]

2.1.1 MAPEAMENTO SISTEMÁTICO

Mapeamentos sistemáticos são modalidades de pesquisas que, possuem os mesmos estágios iniciais e formalidade de uma RS [14]. Entretanto, a pergunta de pesquisa de um mapeamento sistemático é mais ampla e geral do que de uma RS, devido ao seu caráter exploratório [12].

Para responder o escopo mais amplo, um mapeamento sistemático, geralmente, segue três estágios. Primeiramente são identificados estudos primários

que possam conter resultados relevantes. A partir dos documentos encontrados, são selecionados documentos para análise mais aprofundada. E, por fim, é realizada uma atividade para garantir a qualidade dos documentos selecionados [15].

Em uma RS, os estágios anteriormente descritos, seriam seguidos por fases de extração e análise dos dados. Entretanto, como há um escopo mais abrangente, as etapas de extração e classificação estão mais focadas em classificar os estudos encontrados [15].

Assim, caso nos estágios iniciais de uma RS seja identificado que a quantidade de informações relevantes para a pesquisa é diminuta, um mapeamento sistemático pode ser mais apropriado, pois este irá identificar lacunas no conhecimento, o que pode ser utilizado para alterar o cursos de pesquisas ou dar início a outras [11, 15].

As principais diferenças de uma mapeamento sistemático para uma RS , de acordo com [15], podem ser resumidos nos seguintes pontos:

- **Diferença de objetivos:** uma RS visa encontrar um estado de evidência, enquanto um mapeamento busca a classificação, análise temática e identificação de publicações ;
- **Diferença no processo:** em um mapeamento sistemático os documentos encontrados não são analisados quanto a sua qualidade, o que é feito, geralmente é uma análise temática. Essa modalidade é interessante, pois auxilia a categorizar os resultados encontrados. Entretanto, para ser realizada uma classificação de estudos em uma RS é necessário outra fase para analisar os trabalhos primários;
- **Diferença de aprofundamento:** em um mapeamento sistemático mais artigos podem ser considerados, pois grande parte não são analisados em detalhe, o que permite uma maior abrangência. Entretanto uma RS preza por qualidade e por conta disso foca em analisar os documentos encontrados em detalhes, logo possui maior aprofundamento.

Neste trabalho optou-se por um MS ao invés de RS, pois o objetivo maior desse estudo era encontrar uma lista de métricas e classificá-las sem se atentar à qualidade e detalhamento dos trabalhos retornados.

2.2 QUALIDADE EM SOFTWARE

Segundo a definição do *Institute of Electrical and Electronics Engineers* (IEEE), software é: “programa ou programas de computador, procedimentos (possivelmente documentação associada) e dados associados a operação de sistemas da computação” [17].

Desta forma, qualidade de software pode ser definida das seguintes formas [17]:

- O grau em que o sistema, componente ou processo alcança os requisitos especificados;
- O grau em que o sistema, componente ou processo alcança as necessidades dos usuários ou especificações.

Essas definições causam impacto nos seguintes conceitos de qualidade de software:

- “Qualidade significa conformidade aos requisitos” [17] ou seja, refere-se ao grau em que o software alcança especificações preparadas pelo consumidor e seu time profissional;
- “Qualidade consiste naquelas *features* do produto que alcançam as necessidades de consumidores e, por conta disso, fornecem a satisfação pelo produto. Qualidade consiste na “liberdade de deficiências” [17] ou seja, o cumprimento das necessidades reais e satisfação do usuário com o software.

Segundo Pressman [3], para a caracterização de software são empregadas medições de propriedades do programa relacionadas a: coesão, número de pontos de função, linhas de código, entre outras. A partir da análise destas características mensuráveis é possível encontrar dois tipos de qualidade, a qualidade de design e a qualidade de conformidade.

No desenvolvimento de software, a qualidade de *design* engloba requisitos, especificações e o *design* do sistema. Qualidade de conformidade está focada na implementação. Caso a implementação siga o design e o sistema resultante alcance os requisitos e os objetivos de desempenho, então a qualidade de conformidade será alta [3].

2.2.1 USO DE MEDIÇÕES EM AVALIAÇÃO DA QUALIDADE DE SOFTWARE

Conforme pesquisa de qualidade no setor de software brasileiro em 2009 realizada pelo Ministério da Ciência e Tecnologia, a norma NBR ISO 9126[1] ainda é a mais conhecida e utilizada pela indústria [18]. Desta forma, este trabalho abordará o aspecto de qualidade de software com base na ISO 9126.

As famílias de normas ISO/IEC JTC1/SC7 9126 e 14598 descrevem um modelo de qualidade, um processo de avaliação e alguns exemplos de métricas que podem ser utilizadas por organizações que almejam avaliar o produto de software [19].

A avaliação desses produtos tem sido uma das formas para obtenção de maior qualidade. Para que a avaliação seja efetiva é necessário utilizar um modelo de qualidade capaz de permitir o estabelecimento e avaliação de requisitos de qualidade, mas também um processo de avaliação bem definido e estruturado [19].

2.3 ISO 9126

A NBR ISO/IEC 9126-1 [20] é uma normatização para qualidade de software. Essa norma descreve um modelo de três partes para a qualidade do produto de software: qualidade interna, externa e qualidade em uso.

2.3.1 CONCEITOS DE QUALIDADE

A primeira parte do modelo especifica seis características para a qualidade interna e externa (ver Figura 2), as quais são subdividas em outras sub-características. Estas se manifestam externamente quando o software é utilizado como parte de um sistema, e é resultado dos atributos internos de software [20].

As seis características para qualidade interna e externa segundo a ISO/IEC 9126-1 [20] são:

- **Funcionalidade:** “capacidade do produto de software em prover funções que alcançam necessidades definidas quando o software é utilizados em condições definidas” [20].
- **Usabilidade:** “capacidade do software em ser entendido, aprendido, usado e atrativo ao usuário, quando usado sob condições especificadas” [20].

[1] Então conhecida como NBR 13596.

- **Confiabilidade:** “capacidade do software em aderir a padrões, convenções ou regulamentações” [20].
- **Eficiência:** “capacidade do software em prover o desempenho apropriado, relativo a quantidade de recursos usados, dentro de um dado contexto” [20].
- **Capacidade de Manutenção:** “capacidade do software de ser modificado. Modificações podem incluir correções, melhorias ou adaptações à mudanças no ambiente, e em requisitos e especificações funcionais” [20].
- **Portabilidade:** “capacidade do software em ser adaptado para diferentes ambientes sem aplicar ações ou meios diferentes daqueles providos para este propósito de software” [20].

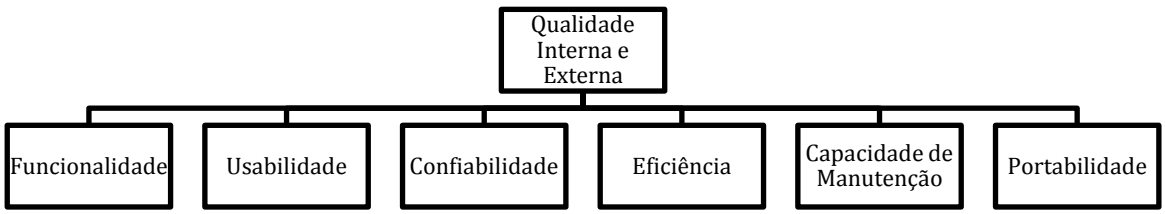

Figura 2. - Diagrama com as seis características de qualidade interna e externa adaptado de [20]

A qualidade em uso é a visão da qualidade do sistema pelo usuário, e é medida em termos do resultado da utilização do software. É a combinação entre a qualidade interna e externa para o usuário [20]. A ISO/IEC 9126-1 descreve a qualidade em uso em termos de entendimento, aprendizado, operacionalidade e atratividade e conformidade. A qualidade em uso pode ser influenciada por qualquer característica de qualidade e é bem mais abrangente do que a usabilidade.

Métricas de qualidade em uso medem a extensão que um produto alcança as necessidades de usuários específicos, (ver Figura 3) isso em termos de um determinado objetivo de efetividade, produtividade, segurança e satisfação [20].

- **Efetividade:** “capacidade do produto de software em permitir que usuários alcancem objetivos específicos com precisão dentro de um contexto especificado” [20].

- **Produtividade:** “capacidade do produto de software em permitir que o usuário gaste montantes apropriados de recursos em relação a efetividade alcançada em um contexto especificado de uso” [20].
- **Segurança:** “capacidade do produto de software em alcançar níveis de aceitação de risco que podem afetar pessoas, negócios, propriedades ou ambiente em um contexto específico de uso” [20].
- **Satisfação:** “capacidade do produto de software em satisfazer usuários em um contexto específico de uso” [20].

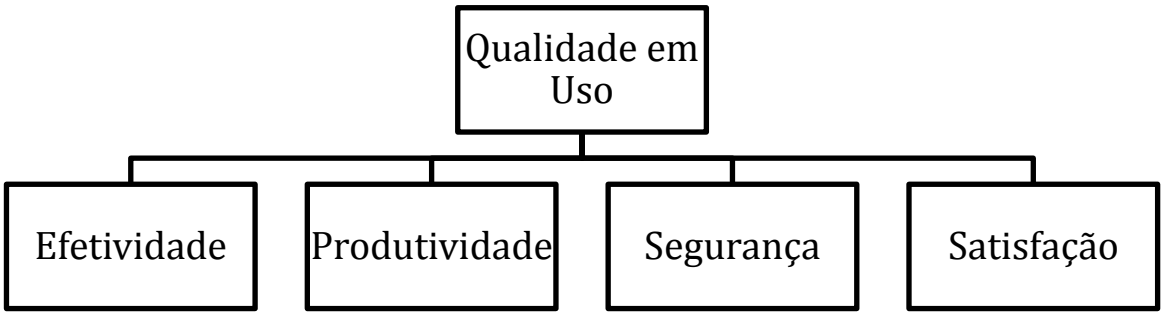

Figura 3. - Diagrama com as quatro características da qualidade em uso ISO/IEC 9126 adaptado de [20]

2.3.2 MÉTRICAS

A primeira parte da ISO/IEC 9126-1 [20] define termos para as características da qualidade de software e como essas características são decompostas em sub-características, não descrevendo, contudo, como estas devem ser medidas. Para tanto, foram criadas outras partes da norma: A ISO/IEC 9126-2 [21] define métricas para qualidade externa, a ISO/IEC 9126-3 [22] estabelece a qualidade interna e a ISO/IEC 9126-4 [23] define métricas para qualidade em uso.

As métricas de **Qualidade Interna** podem ser aplicadas para produtos de software não executáveis durante seus estágios de desenvolvimento e dão aos usuários a habilidade de medir a qualidade de entregáveis intermediários, possibilitando assim, predizer a qualidade do produto final [22].

Já as métricas de **Qualidade Externa** podem ser utilizadas para medir a qualidade do produto de software medindo o comportamento do sistema do qual ele faz parte. Somente podem ser utilizadas durante os estágios de teste do ciclo de desenvolvimento e durante estágios operacionais [21].

Finalmente as métricas de **Qualidade em Uso** medem se um produto atende ou não às necessidades e objetivos especificadas pelos usuários. Esses requisitos especificados por métricas deveriam ser utilizados como critério quando um produto é avaliado [23].

Conforme *template,* apresentado na Tabela 3, retirado das partes 2, 3 e 4 da ISO/IEC 9126 [21, 22, 23], as métricas são apresentadas e detalhadas considerando as seguintes informações:

- Nome da métrica: “Métrica correspondente na tabela de métricas internas e externas” [21];
- Propósito da métrica: “Expressado como uma pergunta que deve ser respondida pela aplicação da métrica” [21];
- Método de aplicação: “Provém um tipo de aplicação” [21];
- Medição, fórmula e dados de elementos computacionais: “Provém a fórmula medição e explica os meios para utilização dos elementos de dados” [21];
- Interpretação dos valores medidos: “Fornece o intervalo e valores preferenciais” [21];
- Tipo de métrica da escala: “Tipo da escala utilizada pela métrica. Tipos de escalas utilizados são: Nominal, Ordinal, Escala de Intervalo, Escala Absoluta” [21];
- Tipo de medida: “Os tipos utilizados são: Tamanho, Tempo e Contagem (*count*)” [21];
- Entrada para medida: “Origem dos dados utilizados para a medição” [21];
- Audiência alvo: “Identifica os usuários e os resultados da medição” [21].

Na Seção 3.11 essas informações e a Tabela 3 serão utilizados para construção das Tabelas 12 e 13 as quais apresentarão as métricas encontradas.

Tabela 3. - *Template* de Métricas Obtidas adaptado de [21, 22, 23]

Métricas								
Nome da Métrica	**Propósito da Métrica**	**Método de Aplicação**	**Medição, fórmula e dados de elementos computacionais**	**Interpretação dos Valores Medidos**	**Tipo de Métrica da Escala**	**Tipo de Medida**	**Entrada para Medida**	**Audiência Alvo**
Métrica correspondente na tabela de métricas internas e externas	Expressado como uma pergunta que deve ser respondida pela aplicação da métrica	Provém um tipo de aplicação	Provém a fórmula medição e explica os meios para utilização dos elementos de dados	Fornece o intervalo e valores preferenciais	Tipo da escala utilizada pela métrica. Tipos de escalas utilizados são: Nominal, Ordinal, Escala de Intervalo, Escala Absoluta	Os tipos utilizados são: Tamanho, Tempo e Contagem (*count*)	Origem dos dados utilizados para a medição	Identifica os usuários e os resultados da medição

2.4 MÉTODOS PARA DESENVOLVIMENTO DE MÉTRICAS

Projetos de software atualmente ainda excedem prazos e orçamentos ou requerem maiores níveis de recursos do que os inicialmente planejados. Assim, as equipes trabalham de forma desestruturada, o que resulta em níveis de qualidade baixos ou desconhecidos [24].

Como forma de entender e controlar os processos de desenvolvimento, a medição de software define, coleta e analisa dados a respeito do processo de desenvolvimento de software e de seus produtos resultantes. As medições são relevantes, pois espera-se que elas produzam um entendimento sobre o processo que permita controlá-lo, além de suprir informações relevantes para o seu aperfeiçoamento [24].

Abordagens que envolvem medições são fundamentais para avaliações de qualidade de software, por exemplo, o *Practical Software Measurement* (PSM) é utilizado para a criação de métricas. O PSM visa identificar e estabelecer as medições necessárias e importantes para atividades de desenvolvimento de software [25].

Neste trabalho focou-se no PSM devido ao conceito de Categoria de Informação. Por conta disso, os conceitos referentes a essa metodologia serão apresentados mais a seguir.

2.4.1 PRACTICAL SOFTWARE MEASUREMENT (PSM)

O processo de gerenciamento de software é o gerenciamento bem sucedido dos processos que trabalham associados ao desenvolvimento, manutenção e suporte de produtos de software e sistemas de software. “Bem sucedido” significa que tanto produtos quanto serviços produzidos pelos processos alcançam totalmente os requisitos internos e externos do cliente, além de alcançar os requisitos da organização [25].

Para auxiliar o gerenciamento de projetos de software foi criado o *Practical Software Measurement* (PSM) em 1994 sob o patrocínio do Departamento de Defesa Norte Americano. O PSM define um conjunto de princípios e práticas focadas nas responsabilidades chave do projeto [25].

O PSM possui os seguintes conceitos para medição e melhora de processos, projetos e produtos de software [25]:

- **Necessidade de Informação:** uma necessidade específica, a qual é necessária para apoiar o processo de tomada de decisão do projeto [26];
- **Conceito Mensurável:** uma ideia sobre as entidades que poderiam ser medidas para satisfazer uma necessidade de informação [26];
- **Construtor de Medição:** eventualmente, um conceito mensurável será formalizado como um construtor de medição, o qual especifica o que será medido e como os dados serão combinados, a fim de produzir os resultados que satisfaçam as necessidades de informação [26];
- **Processo de Medição:** orientado às necessidades de informação da organização. Para cada necessidade de informação, o processo gera um produto de informação para sanar a necessidade [26];
- **Procedimento de Medição**: define a mecânica de coleta e organização dos dados coletados, que são necessários para um construtor de medição. A execução deste produz os produtos de informação que respondem as necessidades de informação [26];
- **Plano de Medição**: para tanto, é necessária a identificação de dados que auxiliem a satisfazer a necessidade de informação. Esses dados podem ser obtidos medindo não só elementos processo de software, mas também características do produto, estes elementos são chamados de entidades de software. Os itens: necessidades de informação, construtor de medição e procedimento de medição, são combinados em um plano de medição [26].

A figura 5 ilustra os conceitos anteriormente mencionados. Observe a sequencia necessária para chegar ao plano de medição.

Figura 4. - Esta figura resume o processo de criação de métricas do PSM adaptado de McGarry [26]

O PSM é uma abordagem para medição de software orientada às necessidades de informações organizacionais aderente à ISO/IEC 15939 e, como ela, possui dois componentes: um Modelo de Informação de Medição e um Processo

de Medição. Logo, objetiva medir e melhorar processos, projetos e produtos de software [26].

Em nível prático o PSM aborda dois problemas [27]:

1. Como especificar de uma maneira formal as medidas que deverão ser utilizadas no projeto, ou seja, especificar formalmente os indicadores a serem usados;
2. Como deverá ser conduzido o processo de medição;

Para atingir tais objetivos, o PSM faz uso do Modelo de Informação e o Processo de Medição [25]:

1. **Modelo de Informação:** uma estrutura para a definição das medidas que deverão ser utilizadas no projeto, conforme é apresentado na Figura 6 onde é mapeado a estrutura necessária para se alcançar os atributos começando com a necessidade de informação. Para isso o PSM trabalha com o conceito de necessidades de informações mensuráveis que, por sua vez, resultam de esforços dos gerentes para influenciar as saídas de projetos, processos e iniciativas oriundas de objetivos de medição. As necessidades de informações são usualmente derivadas de duas fontes: objetivos que os gerentes procuram atingir em seus projetos e obstáculos que impedem de atingir tais objetivos [26];

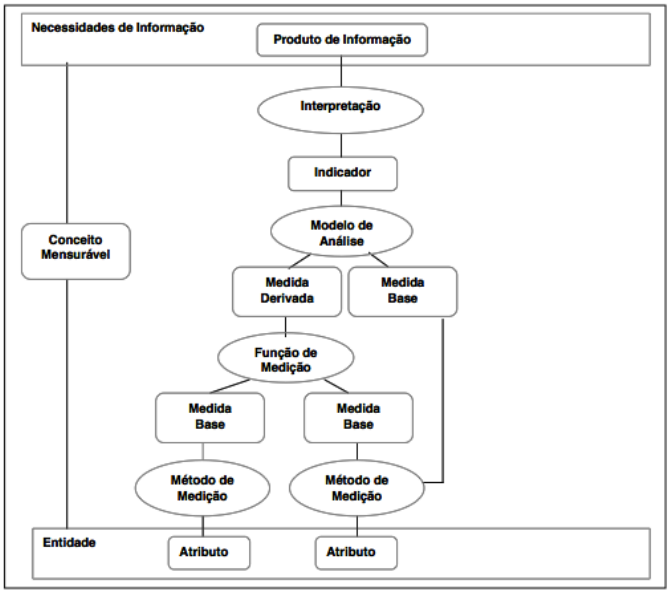

Figura 5. - Modelo de Informação de Medição [28]

2. **Modelo de Processo:** serve de guia a implementação do PSM.É o processo para o estabelecimento, planejamento, execução e avaliação da medição do projeto, organização ou empreendimento como um todo. O PSM inclui um conjunto de medidas já utilizadas com sucesso pela indústria. Essas correspondem a categorias previamente definidas que, por sua vez, são alinhadas às necessidades de informações mensuráveis. Assim as métricas selecionas são agrupadas nas seguintes categorias de medição [26];

   As cinco perspectivas que são centrais para o processo de medição [25]:

   - Desempenho – valores característicos que são observados quando medidos os atributos dos produtos e serviços, advindos do processo. Pode ser descrito de duas formas, ao se medir os atributos que os processos produzem e ao se medir os atributos do processo em si [29];
   - Estabilidade – refere-se à habilidade de uma organização em produzir produtos de acordo com um plano e em melhorar processos, visando à produção de produtos melhores e mais competitivos. Um processo previsível é sinônimo de sob controle. Processos podem variar em formas conhecidas e não aleatórias e mesmo assim satisfazer a definição de processo controlado de Shewhart[2] [25];
   - Conformidade – refere-se ao processo estar claramente definido, efetivamente suportado, fielmente executado, reforçado [25];
   - Capacidade – refere-se aos valores e características que irão variar conforme o passar do tempo. Quando um processo é estável, este possuirá meios previsíveis dentro de intervalos previsíveis. Então é possível examinar se o processo está ou não atingindo os objetivos [25];
   - Melhora e Investimento – refere-se a objetivos de negócio e estratégias, os quais juntamente com dados factuais sobre os atributos de qualidade do produto e performance do processo, são as chaves que levam a ações que aperfeiçoam o processo de software [25].

A questão de tornar a medição de software efetiva, parte primeiramente da escolha de medidas certas, ou seja, medidas que auxiliem a organização a operar

[2]Um fenômeno é dito controlado quando através de experiências passadas, é possível prever, pelo menos dentro de limites, quanto é esperado que o fenômeno varie no futuro.

com sucesso no mercado [27]. Nessa linha de pensamento foram identificadas cinco medidas consideradas essenciais: tamanho, produtividade, tempo, esforço e confiabilidade.

A experiência mostra que a maior parte das necessidades de informação podem ser agrupadas em áreas gerais, chamadas categorias de informação, que são básicas para a maioria dos projetos. Assim que uma necessidade de informação é identificada o próximo passo é mapeá-la a uma determinada categoria de informação [26].

Segundo McGarry [26], as categorias de informação do PSM ajudam a selecionar as medidas apropriadas, com a alocação de cada necessidade de informação identificada para uma ou mais categorias de informação, conforme tabela.

Dessa forma o PSM define duas tabelas, aqui apresentadas nas Tabelas 4 e 5. A Tabela 4 apresenta os campos: categoria de informações, conceitos mensuráveis e questões associadas [26].

Tabela 4. - Categoria de Informação, Conceito Mensurável, Métricas Associadas adaptado de [26].

Categoria Informação	**Conceito Mensurável**	**Questões Associadas**
Planejado e Progresso	Completude de marcos, Desempenho no caminho crítico Progresso de unidade de trabalho Capacidade Incremental	O projeto está seguindo alcançando os marcos? As datas das tarefas críticas ou entregas não estão sendo cumpridas? Como estão as atividades específicas e progresso dos produtos? A capacidade está sendo entregada como planejado, em builds e releases incrementais?
Recursos e Custo	Esforço pessoal Desempenho financeiro Recursos de ambiente e suporte	O esforço dispendido está de acordo com o plano? Há equipe suficiente com as habilidades necessárias? O projeto está alcançando o orçamento e materiais disponíveis?
Tamanho do produto e estabilidade	Tamanho físico e estabilidade Tamanho funcional e estabilidade	Quanto está mudando o tamanho do produto, conteúdo, características físicas, ou interface Quanto está mudando os requisitos e funcionalidades associadas

Categoria Informação	Conceito Mensurável	Questões Associadas
Qualidade do Produto	Quão corretas são as funcionalidades Capacidade de manutenção Eficiência Portabilidade Usabilidade Confiança	O produto é bom o suficiente para ser entregue ao usuário? Os problemas identificados estão sendo resolvidos? Quanta manutenção o sistema precisa? O sistema alvo faz uso eficiente dos recursos do sistema? Em que extensão as funcionalidades podem ser recebidas em outras plataformas? A interface do usuário está adequada e apropriada para os operadores? Os erros dos operadores estão dentro de margens aceitáveis? Quão frequente o serviço do usuário é interrompido? As taxas de falhas estão dentro das margens aceitáveis?
Efetividade Tecnológica	Adequação tecnológica Volatilidade tecnológica	A tecnologia atual alcança todos os requisitos alocados, ou serão necessárias tecnologias adicionais? A nova tecnologia se faz um risco, pois pode necessitar de muitas mudanças?
Satisfação do Consumidor	Feedback do consumidor Suporte do consumidor	Qual é a percepção do desempenho neste projeto? Quão rápido os pedidos de mudança do consumidor estão sendo endereçados?

A Tabela 5, continua o raciocínio iniciado na Tabela 4, e apresenta os campos: categorias de informações, conceitos mensuráveis e medições prospectadas na indústria de software [26].

Tabela 5. - Categoria da Informação, Conceito Mensurável, Métricas Prospectadas adaptada de [26]

Categoria Informação	**Conceito Mensurável**	**Métricas Prospectadas**
Planejado e Progresso	Completude de marcos, Desempenho no caminho crítico Progresso de unidade de trabalho Capacidade Incremental	Datas dos marcos Tempo ocioso Requisitos traçados Requisitos testados Abertura de problemas relatados Fechamento de problemas relatados Revisões completas Requisição de mudança abertas Requisição de mudanças fechadas Unidades desenhadas Unidades codificadas Unidades integradas Tentativas de casos de teste Casos de teste que passaram Itens de ação em aberto Itens de ação completos Componentes integrados Funcionalidades integrados
Tamanho do produto e estabilidade	Tamanho físico e estabilidade Tamanho funcional e estabilidade	Tamanho do banco de dados Componentes Interfaces Linhas de código Requisitos Mudanças funcionais Pontos de função
Qualidade do Produto	Quão corretas são as funcionalidades Capacidade de manutenção Eficiência Portabilidade Usabilidade Confiança	Detectar Idade dos defeitos Nível de desempenho técnico Tempo para restaurar Complexidade Ciclomática Utilização *Throughput* Tempo de resposta Adequação a padrões Erros de operadores Tempo médio para falha

Categoria Informação	**Conceito Mensurável**	**Questões Associadas**
Desempenho do processo	Adequação do processo Eficiência do processo Eficácia do processo	Referência da taxa de maturidade Achados das auditorias do processo Produtividade Tempo do ciclo Defeitos contidos Defeitos que Escaparam Esforço de retrabalho Componentes de retrabalho
Efetividade Tecnológica	Adequação tecnológica Volatilidade tecnológica	Cobertura de requisitos Mudanças na *baseline*
Satisfação do Consumidor	*Feedback* do consumidor Suporte do consumidor	Taxa de satisfação Taxa de premiações Requisições de suporte Tempo de suporte

2.5 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE ESTUDADAS

As metodologias ágeis seguem os valores e princípios do Manifesto Ágil [6]. Ele foi criado em 2001 por um grupo de desenvolvedores experientes como forma de se antepor às metodologias tradicionais dando início ao movimento ágil.

O Manifesto Ágil apresenta quatro valores os quais são apresentados na Tabela 6.

Tabela 6. - Valores do Movimento Ágil adaptado de [4]

Priorizar	**Ao invés de**
Indivíduos e iterações	Processos e Ferramentas
Software funcionando	Documentação complexa
Colaboração do consumidor	Negociação contratual
Responder as mudanças	Seguir um plano

Dos valores apresentados na Tabela 6 foram derivados doze princípios os quais são apresentados na Tabela 7.

Tabela 7. - Princípios do Movimento Ágil adaptado de [4]

Princípios fundamentais do manifesto ágil
Satisfazer o cliente, por meio de entregas constantes e incrementais
Receber os pedidos de mudanças
Entregar software funcionando com frequência
Desenvolvedores e responsáveis pelo negócio devem trabalhar perto no dia-a-dia do desenvolvimento
Construir projetos ao redor de indivíduos motivados
Estabelecer rotinas de interação pessoal entre os responsáveis pelo desenvolvimento.
Estabelecer software funcionando, pois é a principal medida de progresso
Promover o desenvolvimento sustentável e ágil
Atentar continuamente à excelência técnica e ao bom design, pois, desta forma, otimiza-se a agilidade
Simplicidade
Equipes auto-organizacionais
Regularmente a equipe reflete sobre o seu desempenho, em busca de se tornar mas efetiva

Todas as metodologias buscam estar aderentes aos valores do Manifesto Ágil e atendem a alguns, mas não necessariamente todos os princípios.

Nesta seção serão apresentados os conceitos das metodologias ágeis *Scrum, Extreme Programming* (XP) e *Kanban.* Essas foram estudadas por conta do relatório realizado pelo Tribunal de Contas da União (TCU) que apontou como principais metodologias ágeis utilizadas pela Administração Pública Federal [9], o qual é um grande consumidor de produtos de software.

2.5.1 METODOLOGIA SCRUM

A metodologia de *Scrum* teve origem em uma comunidade voltada para a prototipação. Isto porque esta comunidade queria uma metodologia que apoiasse um ambiente em que os requisitos não só estivessem incompletos no início do projeto, mas também pudessem sofrer mudanças rápidas no decorrer do desenvolvimento [4]. Esta metodologia tem foco no processo de gerenciamento de projetos [28].

A metodologia *Scrum* é classificada como um framework estrutural que vem sendo utilizado para o gerenciamento de projetos complexos desde o início da década de 1990. Este framework permite empregar vários processos ou técnicas numa abordagem iterativa e incremental [9].

A metodologia *Scrum* é fundamentada nas teorias empíricas de controle de processo, ou empirismo. Ele afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido. O *Scrum* emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos. [31].

A implementação do controle de processo empírico está apoiado em três pilares:

- Transparência: “Aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados” [31];
- Inspeção: “Os usuários *Scrum* devem, frequentemente, inspecionar os artefatos *Scrum* e o progresso em direção a detectar variações. Esta inspeção não deve, no entanto, ser tão frequente que atrapalhe a própria execução das tarefas” [31];
- Adaptação: “se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o produto resultado será inaceitável, o processo ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve o possível para minimizar mais desvios” [31].

Quatro eventos formais são descritos pelo *Scrum* para inspeção e adaptação, estes são: Reunião de Planejamento da *Sprint*, Reunião Diária, Reunião de Revisão da *Sprint*, Retrospectiva da *Sprint* [31].

O principal evento da metodologia *Scrum* é a *Sprint*, um período de tempo certo de um mês ou menos, no qual uma versão incremental potencialmente utilizável do produto, é criado. *Sprints* tem durações coerentes em todo o período de desenvolvimento. Sendo uma nova Sprint iniciada imediatamente após a conclusão da Sprint anterior [31].

As *Sprints* são compostas pelas seguintes eventos formais:

- Reunião de Planejamento da Sprint: “possui um tempo estipulado de no máximo oito horas para um Sprint de um mês de duração. Para *Sprints* menores, este evento é normalmente menor. O *Scrum Master* garante que o evento ocorra e que os participantes entendam seu propósito” [31];
- Reunião Diária do *Scrum:* “um evento com de duração definida de 15 minutos, para que a Equipe de Desenvolvimento possa sincronizar as suas atividades

e criar um plano para as próximas 24 horas. Esta reunião é feita para inspecionar as atividades realizadas desde a última reunião diária" [31];

- Retrospectiva da *Sprint*: "é executada ao final de cada *Sprint* para inspecionar o que foi desenvolvido e adaptar o *Backlog* do Produto se necessário. Durante essa reunião o time *Scrum* e as partes interessadas colaboram sobre o que foi feito na *Sprint*. Ela é uma reunião informal, e não uma reunião de *status*, e a apresentação do incremento destina-se a motivar obter comentários e promover a colaboração" [31].
-

Figura 6. - Eventos formais do Scrum adaptado de [32]

Durante uma *Sprint* existem práticas que devem ser seguidas, estas são [31]:

- "Não são feitas mudanças que possam por em perigo o objetivo da *Sprint*";
- "As metas de qualidade não diminuem"; e
- "O escopo pode ser clarificado e renegociado entre o *Product Owner* e o time de desenvolvimento".

Uma Sprint pode ser cancelada antes do tempo planejado somente por autorização do *Product Owner*. A Sprint também poderá ser cancelada se o objetivo da Sprint se tornar obsoleto. Ressaltando que com o cancelamento de uma *Sprint*, qualquer item de *backlog* completo e considerado como terminado é revisado [31].

O Time *Scrum* é composto pelos seguintes atores:

- *Product Owner* (PO – dono do produto) este ator trabalha intimamente com a equipe de desenvolvedores para auxiliar na identificação e priorização das funcionalidades do sistema. Ele é responsável por maximizar o valor do produto e do trabalho da equipe de desenvolvimento, além de ser o único

responsável por gerenciar o *Backlog* do Produto (*Product Backlog*) [9, 31]. Esse documento contém as funcionalidades do sistema identificadas e priorizadas. Além destas, também compõem o *Product Backlog* as correções de defeitos encontrados, os requisitos não funcionais e tudo o que for preciso para alcançar o produto final [9].

- Equipes de Desenvolvimento, as quais são estruturadas, auto-organizáveis e formada por profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto ao final de cada *Sprint* [31].
- *Scrum Master*, este é responsável por garantir que o *Scrum* seja entendido e aplicado, assegurando que o time *Scrum* adere à teoria, práticas e regras da metodologia [9, 31]. Ele também trabalha em parceria com o PO e com a Equipe de Desenvolvimento, auxiliando o *Product Owner* a encontrar técnicas para gerenciar efetivamente o *Backlog* do Produto. Além disso, ele auxilia a comunicação clara acerca da visão, objetivo, e itens do *Backlog* do Produto para a equipe de desenvolvimento. O *Scrum Master* também auxilia o Equipe de Desenvolvimento com treinamentos para autogerenciamento, interdisciplinaridade, além de ensinar e liderar a Equipe de Desenvolvimento a produzir produtos de alto valor [31];

O *Backlog* do Produto é uma lista ordenada de tudo que deve ser necessário no produto, e é uma origem única dos requisitos para qualquer mudança a ser feita no produto. O *Product Owner* é responsável pelo *Backlog* do produto, incluindo seu conteúdo, disponibilidade e ordenação [31]. Ele nunca está completo. Os primeiros desenvolvimentos apenas estabelecem os requisitos inicialmente conhecidos e melhor entendidos. O *Backlog* do produto evolui tanto quanto o produto e o ambiente no qual ele será utilizado evoluem. O *Backlog* do Produto é dinâmico; mudando constantemente para identificar o que o produto necessita para ser mais apropriado, competitivo e útil [31].

O *Backlog* da Sprint é um conjunto de itens do *Backlog* do produto selecionados para a Sprint, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da Sprint. O *Backlog* da Sprint é a previsão do time de Desenvolvimento sobre qual funcionalidade estará no próximo incremento e sobre o trabalho necessário para entregar essa funcionalidade em um incremento “Pronto” [31]. O *Backlog* da Sprint torna visível todo o trabalho que o time de desenvolvimento identifica como necessário para atingir o objetivo da Sprint [31].

O *Scrum* possui três fases. A primeira é o pré-jogo, que incluem planejamento e desenvolvimento da arquitetura; na segunda fase ocorre o desenvolvimento; e na terceira fase, pós-jogo, é realizado o fechamento da release [33].

2.5.2 MÉTODO KANBAN

A metodologia *Kanban* para desenvolvimento de software foi proposta por David J. Anderson e define um *framework* para melhoria incremental de processos e sistemas em organizações [9].

Segundo Anderson [34] o *Kanban* foi primeiramente utilizado para manutenção de software, por conta da sua filosofia em aperfeiçoamento contínuo. Alcançado por meio de mudanças incrementais e evolucionárias, que respeitam o processo atual, os seus papéis, suas responsabilidades e seus cargos [9, 34].

Kanban pode ser utilizado para introduzir uma mudança para o ciclo de desenvolvimento de software existente ou para a metodologia de gerenciamento de [35].

O *Kanban* é uma metodologia para impulsionar mudança, mas é pouco prescritiva, o que a diferencia de metodologias como XP e *Scrum*. Ele não uma cópia ou modelo de implementação de processo e também não é um substituto para adoção de *Scrum* ou XP. A adoção do *Scrum* pode ser utilizada como ponto de partida para a utilização de *Kanban*, e este pode ser um catalisador da melhoria do processo adotado [9, 35].

Essa abordagem não define um conjunto específico de passos ou de funções para serem seguidos no processo de desenvolvimento e nem prega uma revolução nos processos existentes [9, 35]. Mas encoraja uma mudança gradual entendida e aceita em consenso pelas equipes, tendo em vista pequenos incrementos no processos a fim de alcançar grandes melhorias no sistema [9, 35].

A metodologia *Kanban* baseia-se no conceito que é necessário conhecer o seu fluxo de trabalho (*workflow*) tendo cada item em produção separado. Além de limitar o número de itens em produção (*Work in Progress* (WIP) em cada estágio do projeto. Considerando viável o início de um trabalho somente quando o predecessor é entregue ou removido da produção [35].

Conforme referência [36] algumas vantagens da utilização do *Kanban* são:

- Aumento da satisfação do consumidor;
- Aumento da produtividade;
- Redução dos custos;
- Aumento da capacidade de resposta a mudanças.

2.5.3 METODOLOGIA EXTREME PROGRAMMING (XP)

A metodologia *Extreme Programming* (XP) foi originalmente desenvolvida tendo em vista a sua utilização por equipes pequenas. Sendo mais tarde utilizados para projetos maiores e distribuídos [32] .

Segundo *Sommerville*, a metodologia *Extremme Programming* (XP) é uma das abordagens ágeis mais conhecidas e utilizadas no mundo. XP Recebeu esse nome devido a utilização de boas práticas em níveis extremos. As práticas utilizadas são [2, 32]:

- **Desenvolvimento Incremental:** pequenas *releases* frequentes do sistema onde os requisitos e funcionalidades são baseadas em *user stories* ou cenários do consumidor [2, 32];
- **Envolvimento do Consumidor:** representantes do consumidor trabalham perto da equipe de desenvolvimento sendo responsável por definir testes de aceitação do sistema [2, 32];
- **Valorização de Pessoas:** programação em pares, propriedade coletiva do código e jornadas de trabalho diárias de oito horas focam mais em pessoas do que no processo [2, 32];
- **Aceitação de Mudanças:** releases constantes aos consumidores, desenvolvimento baseado em testes (TDD), refatoração evitam a degeneração do código e integração contínua do código [2, 32];
- **Manter Simplicidade:** é suportada pela refatoração constante e melhora da qualidade do código e pela utilização de *design* e não necessariamente antecipam mudanças ao software [2, 32].

A metodologia XP baseia-se em quatro princípios básicos: simplicidade, comunicação, feedback e coragem [2, 32] e em doze práticas de suporte: planejamento, fases pequenas, metáfora, design simples, testes, refatoração, programação em pares, propriedade coletiva, integração contínua, semana de 40 horas, cliente próximo aos desenvolvedores e padrões de código [32, 37].

2.6 CONSIDERAÇÕES FINAIS

Qualidade é um foco importante em desenvolvimento de software e tem sido gerida a partir do uso de medições. Assim sendo, o estudo de métodos para identificação e estabelecimento das medições corretas, de acordo com os focos de qualidade relevantes para a organização de desenvolvimento, têm sido cada vez mais utilizados e, portanto, serão estudadas no próximo Capítulo deste trabalho.

Desta forma, o princípio básico da metodologia ágil é a entrega do software de alta qualidade testado e rodando. Outros focos de qualidade podem ser relevantes para a satisfação do usuário, independente do desenvolvimento ser feito ou não por meio de paradigmas ágeis, tais como: usabilidade e portabilidade. Mas dependerão dos seus requisitos não funcionais a serem estabelecidos em cada caso de desenvolvimento.

3 PROTOCOLO DO MAPEAMENTO SISTEMÁTICO

O objetivo deste mapeamento sistemático foi identificar métricas de qualidade de software utilizadas pelas metodologias ágeis de desenvolvimento de software *Scrum*, *Extreme Programming* e *Kanban*. As próximas seções tratam de detalhar o tema da pesquisa abordado.

É importante mencionar que as definições apresentadas neste capítulo fazem parte da fase de planejamento do Mapeamento Sistemático denominada “Refinar Objetivo de Pesquisa” é apresentada na Figura 8.

3.1 CICLO DA PESQUISA

Um protocolo para execução Mapeamento Sistemático foi desenvolvido e executado para coletar evidências com o intuito de responder a seguinte pergunta de pesquisa proposta: quais são as métricas utilizadas para metodologias ágeis, em especial, para *Scrum*, *Kanban* e XP?

A partir dos estudos encontrados criou-se um ciclo de pesquisa para buscar as principais métricas para as metodologias ágeis *Scrum*, XP e *Kanban*. A ordem de execução das fases do ciclo são apresentas na Figura 8. A descrição de cada uma das fases é apresentada a seguir:

1. **Refinar Objetivo de Pesquisa**: aqui o processo de seleção das fontes foi definido, assim como, a *string* de busca e os critérios de seleção;
2. **Pesquisar em Engenhos de Busca**: nesta a *string* foi executada;
3. **Selecionar os Artigos Encontrados**: os resultados encontrados foram analisados e foram selecionados conforme os critérios definidos na fase 1;
4. **Analisar os Resultados**: resultados obtidos foram analisados e, caso seja identificado alguma inconformidade com os resultados encontrados, o processo é repetido.

Figura 7. - Ciclo de Execução do Mapeamento Sistemático

3.2 PROBLEMA

Segundo o Relatório Técnico do TCU [9], as principais metodologias ágeis de desenvolvimento de software utilizadas pelo Governo Federal são: *Scrum*, XP e *Kanban*. Apesar de serem utilizadas pelo governo, não existe, segundo este mesmo relatório, uma expertise quanto à sua utilização e, portanto, o conhecimento das métricas existentes relacionadas às metodologias não está, igualmente, difundido.

3.3 QUESTÃO

Buscou-se responder a seguinte pergunta de pesquisa: quais são as métricas utilizadas para metodologias ágeis, em especial, para *Scrum*, *Kanban* e XP?

3.4 PALAVRAS-CHAVE

As palavras-chave foram derivadas a partir da questão de pesquisa. Para cada metodologia foram identificados os sinônimos e termos alternativos, os quais também entraram como palavras chaves. A Tabela 8 apresenta todas as palavras-chave consideradas nessa pesquisa apresentando somente as em inglês.

Tabela 8. - Palavras-Chave

Inglês	
Agile Software Development Process	Metrics
Scrum	
Extreme Programming (XP)	
Kanban	

Observe que a Tabela 6 traz as palavras-chaves em duas colunas distintas, pois as strings de busca foram criadas a partir da combinação das palavras-chaves da primeira com a segunda coluna.

3.5 CRITÉRIOS PARA SELEÇÃO FONTES DE PESQUISA

A seleção das fontes seguiu os seguintes critérios:

- As fontes de pesquisa devem ser de língua inglesa ou portuguesa;
- Os trabalhos deverão estar disponíveis gratuitamente por convênios com a Universidade de Brasília ou simplesmente gratuitas;
- As fontes de pesquisa devem estar disponíveis em bibliotecas digitais;

3.6 MÉTODOS DE PESQUISA

O processo utilizado para a procura por estudos primários incluiu buscas eletrônicas através de engenhos de busca das bibliotecas digitais, onde foi utilizada *string* de busca formulada, presente na Seção 3.10.

3.7 LISTA FONTES DE PESQUISA

Para este mapeamento sistemático foram pesquisadas nas base de dados *IEEEXplore*, por ter boa cobertura de importantes periódicos e conferências, e *Scopus*, por ser uma base de indexação geral de trabalhos [13].

3.8 PROCESSO DE SELEÇÃO

O procedimento adotado para seleção dos estudos primários desta pesquisa está dividido em duas etapas. Na primeira um conjunto de estudos obtidos através das pesquisas eletrônicas foram analisados. Na segunda parte foi obtido um conjunto de artigos considerados relevantes, resultantes da primeira etapa de seleção, os quais são igualmente analisados e classificados.

Os critérios para seleção dos artigos na pesquisa foram apresentados na Tabela 9, esses critérios são executados em passos consecutivos.

Tabela 9. - Critérios de Seleção

Catalogação Preliminar	**Critérios de Inclusão**	**Critérios de Exclusão**
1) Leitura do Título; 2) Leitura do Abstract/Resumo; 3) Leitura da Introdução; 4) Leitura da Estrutura (seções) do Trabalho; 5) Leitura da Conclusão do Trabalho; 6) Leitura de Todo o Trabalho;	1) As fontes de pesquisa deverão ser da área da computação ou administração; 2) Devem citar ou descrever o uso de métricas para processo ágil de desenvolvimento de software; 3) As referências devem estar disponíveis gratuitamente por convênios com a UnB ou simplesmente gratuitas; 4) Devem citar ou descrever as métricas utilizadas em processo ágil de desenvolvimento de software; 5) Métricas relevantes para governos;	1) Artigos que não contenham ou façam menção a cerca de métrica, medição, qualidade, métodos ágeis, *Scrum*, *Kanban*, *XP* ou Extreme *Programming*; 2) Publicações dos engenhos de busca que forneçam somente o resumo e/ou *abstract* não entraram na pesquisa.

3.9 EXTRAÇÃO E CLASSIFICAÇÃO DOS DADOS

Os artigos selecionados foram classificados em três grupos, grupo “sim” onde o primeiro pesquisador não teve dúvida quanto a inclusão no estudo; grupo “não” de artigos que não foram selecionados por conta dos critérios de seleção e que não houve dúvidas quanto a sua exclusão; e grupo “classificar” que agrupou os artigos que o primeiro pesquisador ficou em dúvida quanto a classificação.

Os artigos classificados no grupo “classificar” foram analisados por outro pesquisador com o intuito de reduzir o viés individual da pesquisa, conforme sugerido por Kitchenham [13].

3.10 CONSTRUÇÃO DA *STRING* DE BUSCA

A *string* utilizada nos engenhos de busca foi construída para encontrar métricas relevantes para as metodologias ágeis *Scrum*, XP e *Kanban*. Desta forma, a estratégia para construção da *string* de busca baseou-se nos seguintes passos [8]:

1. A partir das questões de pesquisa, derivou-se as principais palavras-chaves apresentadas na Tabela 8;
2. Identificou-se sinônimos e termos alternativos as palavras chaves;
3. Usou-se o conector booleano *OR* para incorporar palavras alternativas e sinônimas;
4. Usou-se o conector booleano *AND* para ligar palavras chaves.

A Tabela 10, a seguir, apresenta *string* de busca resultante do processor descrito anteriormente.

Tabela 10. - *Strings* de Busca em Inglês

Inglês
SCRUM AND METRICS
((XP OR EXTREME PROGRAMMING) AND METRICS)
AGILE SOFTWARE DEVELOPMENT PROCESS AND METRICS
KANBAN AND METRICS

Cada uma das *strings* acima foram conectadas pelo operador lógico "*OR*" formando, assim, a *string* de busca:

- (SCRUM AND METRICS) OR ((XP OR EXTREME PROGRAMING) AND METRICS) OR (AGILE SOFTWARE DEVELOPMENT PROCESS AND METRICS) OR (KANBAN AND METRICS)

3.11 SUMARIZAÇÃO DOS RESULTADOS

Os resultados da pesquisa foram analisados da seguinte forma:

- Análise quantitativa: nesta análise foi apresentada a quantidade de itens obtidos que passaram pelos critérios de inclusão definidos, classificando-os segundo as categorias definidas na Seção 3.8;
- Análise qualitativa: nesta análise foram relacionadas as métricas apresentadas com o formato de descrição de métricas da ISO/IEC 9126 parte 2, 3 e 4 [21, 22, 23] apresentado na Tabela 3. As métricas também foram classificadas com base na metodologia, apresentado na Seção 4.3, em que pode ser empregada e quanto a Categoria de informação, apresentado na Seção 4.4 [26].

3.12 FERRAMENTAS

Os artigos selecionados, com base nos critérios de inclusão, foram armazenados na ferramenta "Zotero" [38]. As informações relevantes de cada um dos trabalhos selecionados foram também armazenadas na ferramenta. Além disso, também foi adicionado o resumo do trabalho na seção "abstract" e as seguintes informações na aba geral:

- Palavras-chave do trabalho;
- Arquivo e local onde foi obtido;
- Métricas identificadas;

4 RESULTADOS ENCONTRADOS

O objetivo deste capítulo é relatar os resultados encontrados a partir da execução do protocolo de revisão sistemática apresentado no capítulo anterior. Para tanto, é apresentado nas seções seguintes um relatório da execução do Mapeamento Sistemático, os artigos selecionados, as métricas encontradas e a interpretação destas métricas.

4.1 RELATÓRIO DE EXECUÇÃO

A execução do protocolo de pesquisa selecionou 21 documentos obtidos após execução do processo de seleção descrito na Seção 3.8. Os artefatos foram classificados quanto a base de dados de origem, *IEEEXplore*, *Scopus* e qual metodologia ágil abordada.

4.1.1 CONSIDERAÇÕES SOBRE A STRING DE BUSCA

A execução da *string* de busca para a base de dados da *IEEEXplore* sofreu modificações, pois mesmo utilizando o operador lógico OR o engenho de busca não estava retornando o número de artigos esperado, ou seja, um conjunto com a soma dos artigos de cada metodologia. Por conta disso, a *string* de busca apresentada na Seção 3.10 foi dividida nas partes a seguir:

- SCRUM AND METRICS
- (XP OR EXTREME PROGRAMMING) AND METRICS
- AGILE SOFTWARE DEVELOPMENT PROCESS AND METRICS
- KANBAN AND METRICS

O ciclo da pesquisa descrito na Seção 3.1 foi executado duas vezes. Na primeira vez a *string* de busca foi somente executada na base de dados da IEEE, após análise dos resultados atestou-se que a *string* para a metodologia XP foi projetada incorretamente. Ele não apresentava parênteses para separar os sinônimos “XP” e “EXTREME PROGRAMMING” da operação lógica “AND” com a palavra “METRICS” o que ocasionou no retorno de 667 artigos, logo seria inviável a classificação e extração dos resultados em tempo hábil. Para resolver esse problema a *string* foi corrigida para a seguinte: “(XP OR EXTREME PROGRAMMING) AND METRICS”. Esta *string* foi novamente executada e retornou 42 artigos.

4.2 ARTIGOS SELECIONADOS

A partir da execução do protocolo de pesquisa foram obtidos os seguintes resultados:

- 74 artigos, foram retornados na base de dados *IEEEXplore,* entretanto destes 20 foram selecionados;
- Já na base de dados da *Scopus,* foram retornados 58 artigos, destes apenas 1 foi selecionado; Isso se deve à quantidade de artigos que não possuíam as informações necessárias para a extração de dados. Por conta disso, conforme definido nos Critérios de Exclusão presentes na Tabela 9, os artigos não foram selecionados;

É apresentado, a seguir, a Tabela 11 que contém as informações extraídas de cada documento selecionado. Essas informação são importantes, pois elas foram utilizadas como insumo para construção das lista de métricas encontradas, as quais serão apresentadas, mais a frente.

A Tabela 11 apresenta as seguintes informações dos documentos selecionados: Artigo Analisado, Métricas Encontradas, Comentários e Fonte. Os comentários se referem a considerações realizadas e justificativas para o artigo ter sido selecionado.

Tabela 11. - Artigos Selecionados

Artigos Analisado	Métricas Encontradas	Comentários	Fonte
Making Agile Development Work in a Government Contracting Environment	*Earned Value, Budgeted Cost for Work Schedule (BCWS), Budgeted Cost for Work Performed (BCWP), Actual Cost for Work Performed (ACWP), Cost Variance, Schedule Variance, Cost and Schedule Performance Indices, Estimate at Completion and estimate to Completion*	Apresenta métricas para o gerenciamento de metodologias ágeis de desenvolvimento de software. Cabe ressaltar que não explicita a metodologia.	IEEE
An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes	*Weighted methods per class (WMC), Depth of inheritance tree (DIT), Lack of cohesion of methods (LCOM), Number of Local Methods (NLM), Coupling Through Abstract Data Type (CTA), Coupling Through Message Passing (CTM)*	Apresenta métricas para qualidade em código, estas podem ser aplicadas a metodologia ágil XP	IEEE
Analyses of an agile methodology implementation	*Defect Rate, Relative Schedule Deviation, Relative Cost Deviation, Project Cost Change, Customer and Developer Satisfaction*	Apresenta métricas para o gerenciamento de metodologias ágeis de desenvolvimento de software. Cabe ressaltar que não explicita a metodologia.	IEEE
Introducing an Agile Process in a Software Maintenance and Evolution Organization	*Velocity*	O artigo cita a métrica *velocity* para XP. entretanto não traz definiçãos das métricas	IEEE
The Role of a Projec-Based Capstone Course	*Role Time Measure (RTM), Role Communication Measure (RCM), Role Management Measure (RMM)*	O artigo apresenta métricas para projetos ágeis de desenvolvimento de software	IEEE
Agile Metrics at the Israeli Air Force	*Product size, Pulse, Burn-down, Faults*	O artigo apresenta métricas para metodologia XP	IEEE
Stretching Agile to fit CMMI Level 3	*Velocity, work-in-progress (WIP)*	O artigo apresenta métricas para o processo de desenvolvimento buscando uma semelhança como CMMI.	IEEE
Agile Software Testing in a Large-Scale Project	*running and tested features,*	O artigo faz menção mas não entra em detalhes de nenhuma forma, entretanto mostra que a métrica pode ser utilizada para se determinar o *product size*	IEEE

Artigos Analisado	Métricas Encontradas	Comentários	Fonte
Appropriate Agile Measurement: Using Metrics and Diagnostics to deliver Business Value	*Business Value Delivered, Velocity*	O artigo traz a definição das métricas apresentadas de uma forma bem detalhada.	IEEE
Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes	*Weighted Methods Per Class, Depth of Inheritance Tree, Number of Children, Coupling Between Objects, Response for a Class, Lack of Cohesion of Methods, Attribute Hiding Factor, Method Hiding Factor, Attribute Inheritance Factor, Method Inheritance Factor, Avg. Num. Ancestors, Cohesion Among Methods, Class Interface Size, Data Access Metric, Direct Access Metric, Direct Class Coupling, Measure of Aggregation, Measure of Functional Abstraction, Number of Methods*	O trabalho cita diversas métrica e as suas descrições. As métricas apresentadas são para qualidade do produto.	IEEE
Establishing the Agile PMO: Managing variability across Projects and Portfolios	*Time-to-Market, Costumer Satisfaction, Value Delivered per Release*	Apresenta o uso das métricas, mas não a sua descrição	IEEE
An Empirical Study of the Relationship of Stability Metrics and the QMOOD Quality Models Over Software Developed Using Highly Iterative or Agile Software Processes	*System Design Instability (SDI metric)*	Apresenta métricas utilizadas para gerenciamento de projetos.	IEEE
Using a Validation Model to Measure the Agility of Software Development in a Large Software Development Organization	*Productivity, agility*	Apresenta métricas para qualidade em código, estas podem ser aplicadas para o gerenciamento de metodologias ágeis em geral	IEEE

Artigos Analisado	Métricas Encontradas	Comentários	Fonte
Theory of Relative Dependency: Higher Coupling Concentration in Smaller Modules and its Implications for Software Refactoring and Quality	*Lines of code, coupling between object classes, depth of inheritance tree,*	O artigo apresenta uma série de métricas para tamanho do código, acoplamento e complexidade, entretanto não entra em muitos detalhes nas suas descrições.	IEEE
Enterprise Scrum: Scaling Scrum to the Executive Level	*Enterprise Story Points, Net Present Value, Gross Domestic Product*	Apresenta métricas para o *Scrum*	IEEE
The Evolution of ANT Build Systems	*Source Size (KSLOC), Build System Size (KSBLOC), Timespan, Number of Releases, Shortest Rel. Cycle, Longest Rel Cycle, Release Style, Static Build Lines of Code (SBLOC), Target Count, Task Count, File Count, Halstead Complexity, Build Graph Length, Build Grapth Depth, Target Coverage, Dynamic Build Lines of Code (DBLOC)*	O artigo faz menção de métricas para sistema. Estas podem ser utilizadas para aferir a qualidade do produto.	IEEE
The Impact of Service Cohesion on the Analyzability of Service-Oriented Software	*Lack of Cohesion in Methods (LCOM), Cohesion Among Methods in a Class (CAMC), Normalized Hamming Distance (NHD), Service Interface Data Cohesion (SIDC), Service Interface Usage Cohesion (SIUC) Service Interface Sequential Cohesion (SISC), Total Interface Cohesion of a Service (TICS), Analyzability Metric (Failure Analysis Efficiency), Subjective Cohesion ranks (CR), Sevice Interface Implementation Cohesion (SIIC), Service Interface Usage Cohesion (SIUC),*	O artigo traz diversas métricas que podem ser aplicadas para os contextos estudados.	IEEE
Agile & Kanban In Coordination	*Velocity, Pseudo-Velocity, Cycle-Time, Pseudo-Cycle-Time*	O artigo faz menção de métricas para sistema. Estas podem ser utilizadas para aferir a qualidade do produto.	IEEE
Metrics to evaluate & monitor Agile based software development projects	*Velocity, Obstacles Removed per Iteration, Time To Obstacle Removal*	Apresenta métricas clássicas e métricas utilizando a lógica *fuzzy*. Tem que se observar se esse estudo realmente apresenta métricas válidas.	IEEE

Artigos Analisado	**Métricas Encontradas**	**Comentários**	**Fonte**
Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft	*Velocity, Work Capacity, Focus Factor, Percentage of Found Work, Accuracy of Estimation, Accuracy of Forecast, Target Value Increase, Success at Scale, Win/Loss Record*	Apresenta métricas e suas definições	IEEE
Identification and application of Extract Class refactorings in object-oriented systems	*Entity Placement Value, Association between entities, mutual association, method interations, coupling between objects, Coupling Factor, Lack of Cohesion Methods*	Apresenta o uso de métricas para aferir a qualidade do código.	SCOPUS

4.3 MÉTRICAS ENCONTRADAS

De posse dos documentos, foram extraídas as métricas neles presentes. Semelhantemente aos artefatos, cada uma das métricas foi classificada nas classes de Planejamento e Progresso, Recursos e Custo, Tamanho do Produto e Estabilidade, Desempenho do Processo, Satisfação do Cliente e Desenvolvimento, e Qualidade do Produto, conforme especificado na Seção 3.11 referente a sumarização dos resultados.

4.3.1 MÉTRICAS ENCONTRADAS

Foram extraídas 87 métricas dos 21 documentos selecionados, para cada métrica encontrada nos documentos, buscaram-se respostas necessárias para a descrição de uma métrica pela ISO/IEC 9126 [21, 22, 23], conforme apresentado na Tabela 3. Cabe ressaltar que nem todos os artigos selecionados possuem todas as informações necessárias para a descrição de uma métrica conforme a ISO/IEC 9126, logo algumas informações ficaram faltando, como por exemplo a entrada “Tipo de Métrica da Escala”.

As métricas encontradas foram descritas utilizando o modelo da ISO/IEC 9126, apresentado na Tabela 3. Deste modelo foram consideradas as informações nome da métrica, o propósito da métrica, medição, fórmula e dados de elementos computacionais, tipo de medição, entrada para medição, audiência alvo e identificador foram armazenados no principal produto deste trabalho onde são representados os resultados do mapeamento sistemático e da análise dos documentos encontrados.

Algumas métricas não tiveram todas as suas informações encontradas. Essa ausência indica que os documentos selecionados somente citam ou referenciam as métricas, não apresentando todas as informações necessárias para uma descrição detalhada de uma métrica.

A Tabela 12, contém todas as métricas encontradas que possuíam informações suficientes para preencher um ou mais campos do *template* de métrica da ISO/IEC 9126. Essa tabela é importante, pois ela resume junto com a Tabela 13 todos os resultados selecionados deste mapeamento sistemático.

Para a leitura das siglas se atente ao nome da métrica e as siglas anteriormente apresentadas.

Tabela 12. - Métricas Completas Encontradas

Nome da Métrica	**Propósito da Métrica**	**Medição, Fórmula e Dados de Elementos Computacionais**	**Tipo de Medição**	**Entrada para Medição**	**Audiência Alvo**	**Metodologia**	**Id**
Earned Value (Budgeted Cost for Work Performed – (BCWP)	Prover informação para planejamento e controle de projetos complexos. Medindo quanto valor foi produzido dentro de um dado intervalo de tempo.	Não Encontrado	Não Encontrado	Tamanho (*story points*), Esforço (homem/mês)	Gerentes de projeto e a companhia	Ágeis	1
Budgeted Cost for Work Schedule (BCWS)	Este é o plano que representa o orçamento total	Não Encontrado	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	2
Actual Cost for Work Performed (ACWP)	Este é o custo de todo o trabalho bem sucedido realizado	Não Encontrado	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	3
Cost Variance (CV)	É a diferença entre custo planejado e o custo real	CV = ACWP - BCWS	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	4
Schedule Variance (SV)	É a diferença entre custo do investimento e o valor retornado	SV = BCWP - ACWP	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	5

Nome da Métrica	**Propósito da Métrica**	**Medição, Fórmula e Dados de Elementos Computacionais**	**Tipo de Medição**	**Entrada para Medição**	**Audiência Alvo**	**Metodologia**	**Id**
Cost and Schedule Performance Indices (SPI)	São os índices de desempenho normalizados	SPI = BCWP/BCWS	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	6
Estimate at Completion (EAC)	Valor que provem a estimativa do valor do custo total e custo para se completar	EAC = Custo Atual + Estimativa do Custo do Trabalho restante	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	7
Defect Rate	O número de defeitos realizados pela equipe e por cada programador durante uma iteração e/ou release e durante todo o projeto	Não Encontrado	Não Encontrado	número de defeitos, esforço gasto com *bug fixing*, esforço por pessoa	Equipes e para cada programador	Ágeis	8
Relative Schedule Deviation	Esta métrica mostra como o tempo real gasto em desenvolvimento corresponde ao tempo planejado.	((tempo real - tempo planejado)/ tempo planejado) * 100	Não Encontrado	Tempo real, tempo planejado	Interessados no planejamento	Ágeis	9
Relative Cost Deviation	Semelhantemente a métrica anterior, mas na perspectiva do custo	((custo real - custo planejado)/custos planejados) * 100	Não Encontrado	Custo real, custo planejado	Interessados no planejamento	Ágeis	10

Nome da Métrica	**Propósito da Métrica**	**Medição, Fórmula e Dados de Elementos Computacionais**	**Tipo de Medição**	**Entrada para Medição**	**Audiência Alvo**	**Metodologia**	**Id**
Project Cost Change	Mostra como o custo do projeto piloto corresponde ao custo de baseline do projeto	(1-custopp/custobp)*100	a companhia decide quais parâmetros incluir nos cálculos do custo do projeto	*Custopp*, *custobp*	Interessados no planejamento	Ágeis	11
Customer and Developer Satisfaction	Este conjunto de métricas mostra como a proximação do problema esta sendo aceita por clientes e desenvolvedorese como ela afeta a qualidade do rpoduto do ponto de vista do consumidor	Satisfação cliente/desenvolvedor classificada entre 0-100%	Questionário	%	equipe de desenvolvimento	Ágeis	12
Velocity	Esta métrica é a medida de progresso do time. É calculada somando o número de Story Points de cada User Story completadas durante a iteração. Permitir a equipes de desenvolvimento conhecer a sua velocidade de produção	*Velocity* é definida como: V = Σ das estimativas originais de todo o trabalho aceitável	Não Encontrado	*Story points*	Não Encontrado	XP/Ágeis/SCRUM/KANBAN	13

Nome da Métrica	Propósito da Métrica	Medição, Fórmula e Dados de Elementos Computacionais	Tipo de Medição	Entrada para Medição	Audiência Alvo	Metodologia	Id
Product Size, Running and tested features	Esta métrica representa o montante de trabalhos completos.	Não Encontrado	Não Encontrado	*Test points*	Não Encontrado	XP/Ágeis	14
Pulse	Esta métrica tem como objetivo medir quão contínua é a integração	Não Encontrado	Não Encontrado	*Check-ins operations/day*	Não Encontrado	XP	15
Burn-Down	Esta métrica apresenta o montante de trabalho restante versus a quantidade de recursos humanos restante	Não Encontrado	Não Encontrado	Não Encontrado	Não Encontrado	XP	16
Work-in-progress	Esta métrica ajuda a prever o tento de liderança (*lead time*), logo efeitos futuros no cronograma do projeto.	Não Encontrado	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	17
Business Value Delivered	Os desenvolvedores não conseguem determinar o valor do software em isolamento. Assim esta métrica apresenta o valor das *features* ou grupos de *features*	Cálculo do Net Present Value (NPV), Cálculo do Internal Rate of Return (IRR), Cálculo do Return of Investment (ROI)	Não Encontrado	Não Encontrado	Gerência e empresa	Ágeis	18

Nome da Métrica	Propósito da Métrica	Medição, Fórmula e Dados de Elementos Computacionais	Tipo de Medição	Entrada para Medição	Audiência Alvo	Metodologia	Id
Attribute hiding factor	Não Encontrado	[1-numero total de atributos visíveis de uma classe/número total de atributos num set]	Não Encontrado	Não Encontrado	Não Encontrado	XP	19
System Design instability, SDI metric	Esta métrica apresenta a medida da evolução do software orientado a objeto. Tem como objetivo analisar as mudanças no design a nível do sistema de uma iteração para outra	Σ(percentual de classes adicionadas + percentual de classes que foram deletadas + percentual de classes que tiveram o nome mudado de uma iteração para outra)	Não Encontrado	Não Encontrado	Não Encontrado	XP	20
Productivity	Esta métrica é a divisão do montante produzido pelo custo	Amount of production/cost	Não Encontrado	Não Encontrado	Empresa, gerência, equipe de desenvolvimento	SCRUM	21
Enterprise Story points	Não Encontrado	Não Encontrado	Não Encontrado	*Story points*	Não Encontrado	Ágeis	22
Net present value(NPV)	Não Encontrado	Não Encontrado	Não Encontrado	Não Encontrado	Não Encontrado	SCRUM	23
Gross Domestic product (GDP)	Esta métrica é uma forma de se medir a produtividade da equipe	GDP = NPV/esforço	Não Encontrado	Dólares/pessoa	Não Encontrado	SCRUM	24

Nome da Métrica	Propósito da Métrica	Medição, Fórmula e Dados de Elementos Computacionais	Tipo de Medição	Entrada para Medição	Audiência Alvo	Metodologia	Id
service interface data cohesion (SIDC)	Esta métrica foi projetada para quantificar a Comunicação, assim como indiretamente refletir coesão.	SIDC = (Common(Param(Sop(si))))+Common(returnType(Sop(si))))/(total(Sop(si))*) onde: Sop(si) é o conjunto de todas as operações expostas na interface si do serviço s. Common (Param(SOP(si))) é a função que calcula o número de pares de serviços de operações que tem o mesmo tipo de retorno. Total(SOp(si)) é a função que retorna o número de todas as possíveis combinaçãoes de pares operação para o serviço de interface si.	Não Encontrado	Não Encontrado	Não Encontrado	SCRUM	25
Service interface sequential cohesion (SISC)	Está métrica quantifica as propriedades sequenciais de padrões de operações de serviço	SISC(s) = SeqConnected(Sop(si))/ Total(Sop(si))	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	26

Nome da Métrica	Propósito da Métrica	Medição, Fórmula e Dados de Elementos Computacionais	Tipo de Medição	Entrada para Medição	Audiência Alvo	Metodologia	Id
Total interface cohesion of a servisse (TICS)	Está métrica cobre todas os possíveis aspectos de coesão de serviços para interface.	TICS(s) = (SIDC(s) + SIUC(s) + SIIC(s) + SISC (s))/4	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	27
Analyzability Metric (Failure Analysis Efficiency – FAE)	Não Encontrado	FAE = Sum(T)/N	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	28
Subjective Cohesion ranks	Determinação do nível de coesão de 1 a 5. Sendo 1 coesão baixa e 5 coesão alta.	É requerido que os participantes indiquem o nível de 1 a 5	Não Encontrado	Não Encontrado	Não Encontrado		29
Service Interface Implementation Cohesion (SIIC)	Esta métrica cobre a implementação de funcionalidades (*features*) operação serviço.	SIIC(s) = |IC(s)|/(|(C U I U P U H U BPS)|*|SO(si)|)	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	30
Service Interface usage Cohesion (SIUC)	Esta métrica quantifica o uso de padrões de operações de serviço, assim está diretamente relacionado com coesão.	SIUC = Invoked(clients, Sop(si)/(|clients|*|Sop(si)|)	Não Encontrado	Não Encontrado	Não Encontrado	Ágeis	31

Nome da Métrica	Propósito da Métrica	Medição, Fórmula e Dados de Elementos Computacionais	Tipo de Medição	Entrada para Medição	Audiência Alvo	Metodologia	Id
Cycle-time	Esta métrica pode ser utilizada como padrão para se determinar a saída de resultados de uma equipe. São utilizados *story points* para cada item para calcular o *cycle time* para cada *story size*	Não Encontrado	A partir desta métrica é possível dar aos *product owners* um exato lead time para cada *story* .	Não Encontrado	Não Encontrado	Ágeis	32
Pseudo-cycle-time	Esta métrica pode ser utilizada como padrão para se determinar a saida de resultados de uma equipe. São utilizados story points para cada item para calcular o cycle time para cada story size	Não Encontrado	A partir desta métrica é possível dar aos *product owners* um exato lead time para cada *story* .	Não Encontrado	Não Encontrado	Ágeis/SCRUM/KANBAN	33

Nome da Métrica	Propósito da Métrica	Medição, Fórmula e Dados de Elementos Computacionais	Tipo de Medição	Entrada para Medição	Audiência Alvo	Metodologia	Id
Entity Placement Value (EPC)	Classificação dos candidatos a refatoração quanto ao seu impacto na qualidade de design	EPC = (Σei ε Distance(ei, C)/|entities ε C|)/(Σei ε Distance(ei, C)/|entities ˜ε C|)	A *Entity Placement Value* para uma classe C (EPc) é a taxa da sua distância média entre uma entidade que pertence a uma classe C para a distancia média das entidades não pertencentes a esta classe	Não Encontrado	Não Encontrado	Ágeis/SCRUM/KANBAN	34

A Tabela 12 pode servir de insumo para definição e seleção de métricas de um projeto. São ditas “algumas”, pois nem todas possuem suas informações completas, logo seria necessário a realização de estudos mais aprofundados para completar as suas definições de tal forma que sua utilização se tornaria viável.

A Tabela 13 apresenta as métricas encontradas consideradas incompletas. Essas métricas receberam essa classificação por terem sido extraídos poucos dados sobre elas. Isso reforça a conclusão anterior, que os documentos, geralmente, somente citam ou referenciam as métricas sem apresentar o seu detalhamento ou razão da escolha.

Também é importante ressaltar que, embora incompletos, as métricas apresentadas na Tabela 13 podem ser utilizadas como insumo para pesquisas que busquem completar os seus detalhes.

A informação encontrada pode ser qualquer um dos campos do *template*, podendo ser um propósito ou qualquer outra informação. Elas foram colocadas aqui tal qual foram encontradas nos artigos.

Tabela 13. - Métricas Incompletas

Nome da Métrica	**Informação Encontrada**	**Metodologia**	**Id**
Cohesion among methods	A medida de coesão que é baseada na similaridade entre assinatura de métodos na classe	Ágeis	36
Class interface size	O número de métodos públicos na classe	XP	37
Data access metric	A taxa de atributos privados ou protegidos para o número total de atributos declarados na classe	XP	38
Direct Class coupling	O número de classes que aceitam instância de uma dada classe como parâmetro mais as classes incluindo atributos de uma dado	XP	39
Measure of Aggregation	O percentual de dados declarados no sistema que os tipos são classes definidos pelo usuário.	XP	40
Measure of functional abstraction	A taxa de atributos privados ou protegidos para o número total de atributos declarados na classe	XP	41
Number of Methods	O número de métodos em uma classe	XP	42
Time-to-market, Cycle Time	Esta métrica apresenta o tempo necessário para a entrega do produto	Ágeis	43
Constumer satisfaction	Taxa de satisfação do cliente em relação ao produto	Ágeis	44
Value Delivered per Release	Quantidade de valor entregue por release	Ágeis	45

Nome da Métrica	**Informação Encontrada**	**Metodologia**	**Id**
Estimation Quality Factor	Mede a precisão das estimativas do projeto, a estimativa de erro		46
Role Time Measure	Mede da taxa de desempenho para desenvolvimento/papel	SCRUM/ KANBAN	47
Role Communication Measure	Mede o nível de comunicação entre a equipe a cada etapa de desenvolvimento	SCRUM/ KANBAN	48
Role Management Measure	mede o nível de gerenciamento do projeto	SCRUM/ KANBAN	49
Static Build Lines of Code	O número de linhas de código em um arquivo de especificação de build	XP/KANBAN/Ágeis	50
Target Count	O número de alvos para *build* (*build target*) no arquivo de especificação	XP/KANBAN/Ágeis	51
Task Count	O número de tarefas no arquivo de especificação da build	XP/KANBAN/Ágeis	52
File Count	O número de arquivos de especificação no sistema de build (build system)	XP/KANBAN/Ágeis	53
Halstead Complexity	A quantidade de informação contida em uma build de sistema (volume), a dificuldade mental associada com o entendimento do arquivo de especificação da build do sistema	XP/KANBAN/Ágeis	54
Build Grapth Length	O tamanho de um grafo de uma *build*. Pode ser em termos do número total de tarefas executadas ou o número total de alvos executados	XP/KANBAN/Ágeis	55
Build Grapth Depth	A profundidade da *build* em termos do nível máximo de profundidade em termos de referencias feitas	XP/KANBAN/Ágeis	56
Target coverage	O percentual de alvos em um sistema de *build* que são exercidos por padrão ou alvos para limpeza (*default or clean target*)	XP/KANBAN/Ágeis	57
Dynamic Build Lines of Code	O percentual de código no build system que é exercido por padrão ou por alvo de limpeza (*default or clean target*)	XP/KANBAN/Ágeis	58
Cohesion among methods in a class	Esta métrica mede o grau que corresponde entre tipos de parâmetros de cada método dentro de uma dada classe	Ágeis	59
Normalized Hamming Distance	Pode ser considerada uma extensão de CAMC. Ela e *Cohesion among methods in a class* correspondem fortemente com *Lack of Cohesion methods*. Assim provem uma alternativa para medição da coesão dentro de um certo estágio de design.	Ágeis	60

Nome da Métrica	**Informação Encontrada**	**Metodologia**	**Id**
Time to obstacle removal	É a soma do tempo de todos os obstáculos não resolvidos dividido pelo número de obstáculos não resolvidos	Ágeis	61
Work Capacity	A soma de todo o trabalho durante a *sprint c*ompleto ou não.	Ágeis	62
Focus Factor	Velocity ÷ Work Capacity	Ágeis	63
Percentage of Found Work	Σ(Estimativas Orginais do Trabalho Adotado) + (Previsão Para a *Sprint*)	Ágeis	64
Accuracy of Estimation	1 - (Σ(Deltas Estimados) + (Previsão Total))	Ágeis	65
Accuracy of Forecast	(Σ(Estimativas Originais) Σ (Σestimativas Originais + Σtrabalho Adotado + Σtrabalho Encontrado)	Ágeis	66
Target value increase	*Velocity* da *sprint* atual + *Velocity* original	Ágeis	67
Success at scale	Para cada ponto na escala de Fibonacci (Fp) = (Σ No. de tentativas aceitas da escala Fp) + (No. de todas as tentativas da escala Fp)	Ágeis	68
Win/Loss Record	Uma *sprint* é considerada um ganho somente se: a) foi aceito um mínimo de 80% do estimativa original; e b) Trabalho encontrado + adotado durante a *sprint* permanece a 20% ou menos da previsão inicial.	Ágeis	69
Association Between entities	Tem como objetivo medir as similaridades de componentes	XP	70
mutual association	Tem como objetivo medir as similaridades de componentes	XP	71
Estimate to Completion	Valor que provem a estimativa do valor do custo total e custo para se completar	Ágeis	72
Weighted methods per class	É a somas das complexidades de todos os métodos locais	XP	73
Depth of Inheritance tree	Esta métrica mede quantas classes ancestrais na hierarquia podem afetar a classe em estudo	XP	74
Lack of cohesion of methods	A medição da contagem de pares de variáveis instanciadas. Também pode ser definida como o número de pares de métodos em uma classe que não possuem referências a atributos em comum	XP	75
Number of Local Methods	O número de métodos locais, que são acessíveis for a da classe (métodos públicos)	XP	76
Coupling throught Abstract Data Type	O número total de classes que são utilizadas como tipos de dados abstratos nas declarações das classes	XP	77
Coupling Through Message Passing	Esta medida afere o número de mensagens diferentes enviadas a partir de classes para outras, exclui mensagens enviadas para objetos criados como objetos locais em métodos locais	XP	78

Nome da Métrica	Informação Encontrada	Metodologia	Id
Pseudo-velocity	Esta métrica é essencial para o planejamento das *releases*. Ela é obtida a partir de fatias de tempo do quadro de *Kanban* da equipe	SCRUM/ KANBAN /Ágeis/XP	79
Obstacles removed per iteration	É o número de obstáculos removidos numa única iteração	Ágeis	80
Number of children	Uma contagem do número de filhos de uma dada classe	XP	81
Coupling between objects	Contagem as outras classes que os atributos ou métodos são utilizados por uma dada classe	XP	82
Agility	Esta métrica combina o modelo de Validação (*Validation model*) e modelos existentes de eficiência. Ela descreve a habilidade de rapidamente, e num estágio inicial, construir funcionalidade e qualidade num software	Ágeis/SCRUM/KANBAN	83
Lines of code	Esta métrica é uma forma de se medir o tamanho do software. Ela mede o tamanho físico em linhas de código excluindo linhas em branco e comentários	XP	84
Coupling between object classes	Esta métrica apresenta o número de outras classes que utiliza os métodos e atributos da presente classe	XP	85
Faults	Número de erros durante uma iteração	XP	86
Response for a class	O número de todos métodos locais de uma classe mais todos os métodos diretamente chamados na classe.	XP	87

A seguir a Figura 8 apresenta a classificação das métricas encontradas quanto a metodologia identificada. Essa figura é importante, pois apresenta a quantidade de métricas relacionada a metodologia identificada.

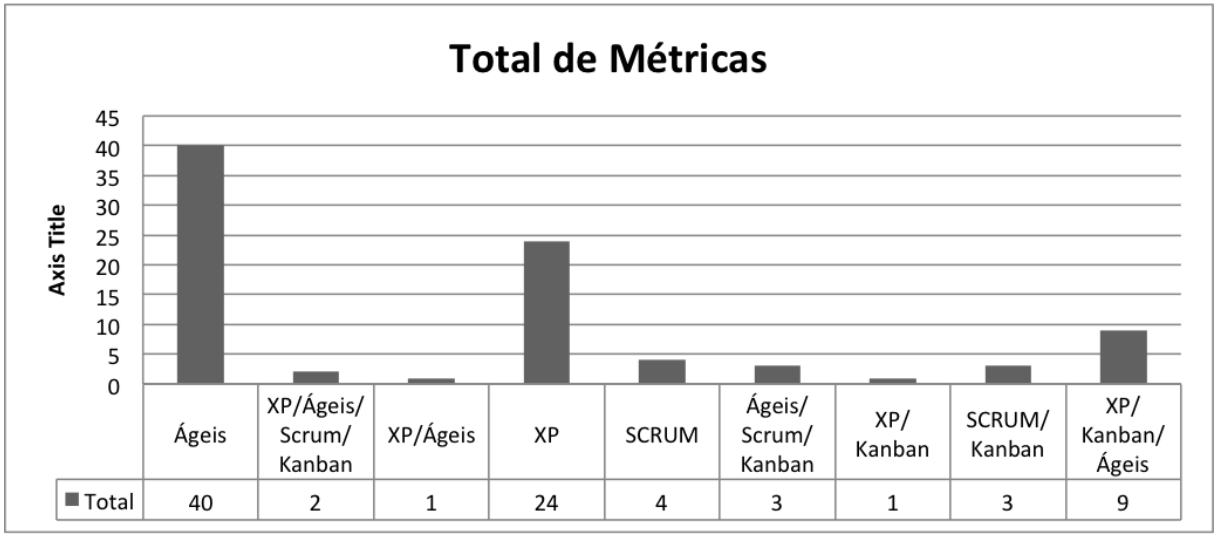

Figura 8 - Classificação das Métricas Quanto a Metodologia

As métricas que foram mais referenciadas estão apresentadas na Figura 9 a seguir. Ela apresenta o número de vezes que a métrica foi encontrada e o nome da métrica.

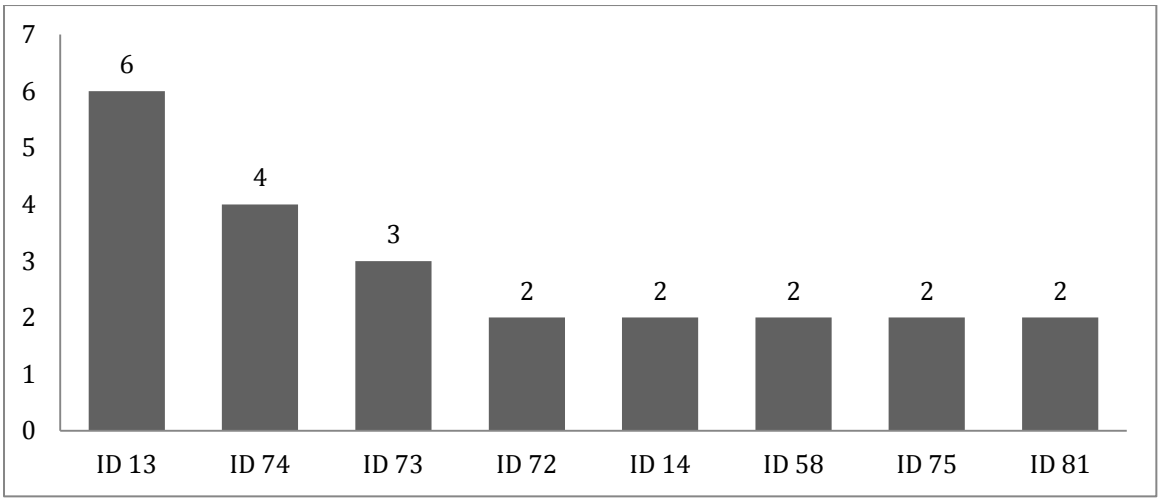

Figura 9 - Métricas Mais Encontradas.

Legenda: ID 13:Velocity; ID 74: Lack of Cohesion Among Methods; ID 73: Depth Inheritance Tree; ID 72: Weighted Methods per Class; ID 14: Product Size; ID 58: Cohesion Among Methods In a Class; ID 75: Number of Local Methods; ID 81: Coupling Between Objects.

4.4 INTERPRETAÇÃO DOS RESULTADOS

Para interpretação dos resultados foram criadas classes para organização das métricas. A definição das classes seguiu o conceito de “Categoria de Informação”, áreas gerais de agrupamento [26], pois segundo McGarry [26], as categorias de informação auxiliam a seleção de medidas apropriadas aos projetos.

A Tabela 14 apresenta o nome de cada categoria de informação juntamente com as suas descrições.

Tabela 14 - Classes de Classificação

Classe	Descrição
Planejamento e Progresso	métricas que buscam responder questões relativas ao planejamento do projeto
Recursos e Custo	métricas que buscam responder questões relativas ao custo e recursos disponíveis do projeto
Tamanho do Produto e Estabilidade	métricas que respondem questões sobre o tamanho e estabilidade dos produtos em desenvolvimento
Desempenho do Processo	métricas que respondem questões relativas ao desempenho do processo de desenvolvimento

Classe	Descrição
Satisfação do Cliente e Desenvolvedores	métricas que respondem questões relacionadas a satisfação dos clientes com o produto e satisfação dos desenvolvedores no projeto
Qualidade do Produto	métricas que respondem questões a cerca da qualidade do produto do produto, tais como: Capacidade de manutenção, eficiência, portabilidade e usabilidade
Efetividade Tecnológica	métricas que medem a adequação tecnológica

Para cada métrica foram analisadas as suas informações para, assim, ser possível armazená-la em uma das classes. Para a classificação observou-se se as métricas encontradas apresentavam semelhanças em relação àquelas apresentadas na Tabela 5. Caso semelhanças fossem observadas quanto ao seu propósito, ela foi adicionada a classe correspondente. A Figura 10 apresenta um gráfico com o número de métricas armazenadas em cada classe.

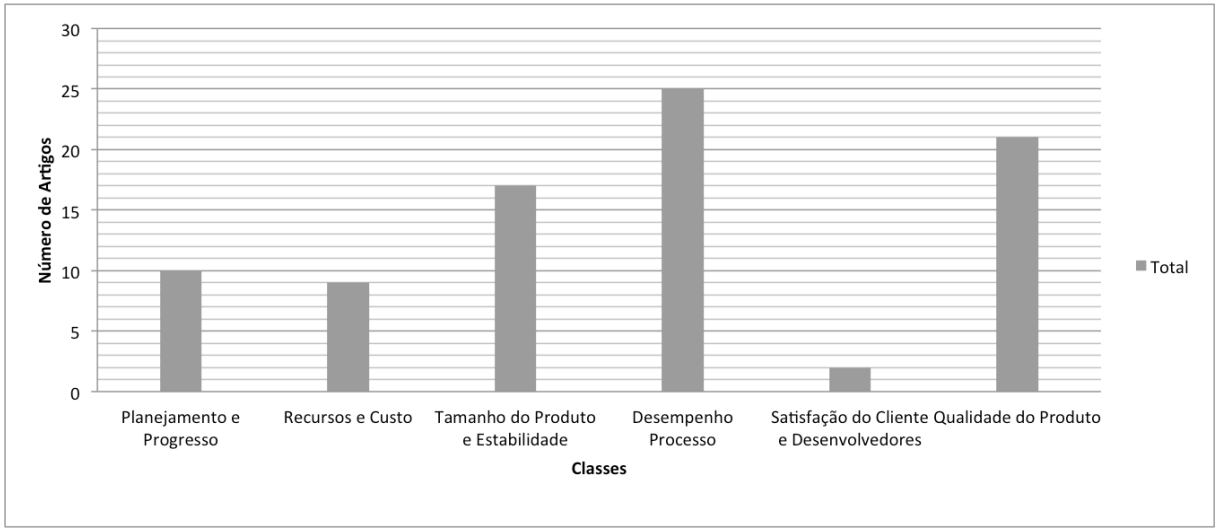

Figura 10. - Número de Métricas em Cada Classe

A partir das classificações apresentadas na Figura 10 pode-se observar que foi encontrada uma grande quantidade de métricas para a classe Processo, um baixo número para a classe Satisfação do Cliente e Desenvolvedores e nenhuma métrica para a classe efetividade tecnológica. A Tabela 15 mostra a classificação das métricas encontradas, os números são os identificadores das métricas provenientes da classificação apresentada nas Tabelas 12 e 13.

A partir das informações apresentadas na Tabela 15 e Figura 10 é possível concluir que há maior preocupação para geração de métricas para aferir a qualidade do processo de desenvolvimento e menor preocupação com a satisfação do cliente. Uma explicação é a maior diversidade e quantidade de métricas para o desempenho de processos, o reflete em maior preocupação com o controle e entendimento dessa classe. Essa afirmativa não é observada na classe satisfação do cliente e desenvolvedor, logo não foi observado o mesmo grau de preocupação de geração de métricas a satisfação do cliente.

Tabela 15. - Classificação das Métricas Encontradas

Planejamento e Progresso	**Recursos e Custo**	**Tamanho do Produto e Estabilidade**	**Desempenho do Processo**	**Satisfação do Cliente e Desenvolvedores**	**Qualidade do Produto**
1	2	22	6	12	19
5	3	36	8	43	20
9	4	41	14		25
13	7	49	15		26
16	10	50	18		27
17	11	51	21		28
42	23	52	24		29
71	83	53	32		30
78	84	54	33		31
80		55	44		34
		57	45		35
		73	46		37
		74	47		38
		75	48		39
		76	60		40
		77	61		56
		86	62		58
			63		59
			64		69
			65		70
			66		72
			67		
			68		
			79		

Planejamento e Progresso	**Recursos e Custo**	**Tamanho do Produto e Estabilidade**	**Desempenho do Processo**	**Satisfação do Cliente e Desenvolvedores**	**Qualidade do Produto**
			85		

Essa conclusão vai de encontro com os princípios ágeis de satisfação do cliente e código de alta qualidade, conforme apresentado na Seção 2.5.1, pois, o número de métricas encontrado por esse trabalho não evidencia essa premissa.

Uma forma de suprir essa a falta de métricas para as Categorias de Informação Satisfação do Cliente e Eficiência Tecnológica seria a utilização das métricas apresentadas na Tabela 4 e Tabela 16 por McGarry [26], onde são apresentadas métricas para essas classe.

Tabela 16 - Categoria de Informação com Métricas Adaptado de [26]

Categoria Informação	**Métricas Prospectadas**
Efetividade Tecnológica	Cobertura de requisitos Mudanças na *baseline*
Satisfação do Consumidor	Taxa de satisfação Taxa de premiações Requisições de suporte Tempo de suporte

Embora essas métricas sejam adequadas para essas categorias de Informação, não necessariamente elas serão interessantes para as metodologias *Scrum*, *Kanban* e/ou XP. Pois cada uma delas apresenta um foco, por exemplo o *Scrum* que foca em gerenciamento. Assim conforme é apresentado mais a frente no Capítulo 5 essa sugestão pode ser encarada como uma possibilidade de trabalho futuro. As métricas sugeridas poderiam ser testadas em casos reais para verificar a sua utilidade para as metodologias.

5 CONCLUSÃO

Conforme apresentado incialmente, esse trabalho teve por foco a buscar uma forma de auxiliar organizações a realizar o acompanhamento de seus projetos que façam uso das metodologias *Scrum*, *Kanban* e *Extreme Programming* (XP). Para nortear o desenvolvimento dos estudos necessários foi apresentada a seguinte questão motivadora: quais são as métricas utilizadas para metodologias ágeis, em especial, para *Scrum*, *Kanban* e XP?

Para responder a essa pergunta foram estudadas, no Capítulo 2, as metodologias ágeis *Scrum*, *Kanban* e XP na Seção 2.5, estudado o PSM apresentado na Seção 2.4; os conceitos de qualidade apresentados pela ISO/IEC 9126 na Seção 2.3.

No Capítulo 3 foi apresentado o protocolo de mapeamento sistemático o qual foi elaborado com o objetivo de identificar as métricas utilizadas pelo *Scrum*, *Kanban* e pelo XP.

Finalmente no Capítulo traz os principais resultados encontrados nesse trabalho, os quais foram resumidamente:

- Lista de métricas utilizadas para as metodologias ágeis *Kanban*, *Scrum* e XP presentes nas Tabelas 12 e 13;
- Classificação das métricas em dois grupos, sendo o primeiro grupo o de métricas completas, apresentado na Tabela 12 onde várias informações foram obtidas através do mapeamento; e o segundo, métricas incompletas, apresentada na Tabela 13 onde foram agrupadas métricas onde não foi identificada quantidade relevante de informação. A partir desse agrupamento, foi possível concluir que nem todas as informações para a classificação das métricas podem ser identificadas, que as mesmas métricas podem ser utilizadas para metodologias diferentes. E que, aparentemente, há uma maior preocupação com desempenho do processo.

Apesar dos resultados alcançados, este trabalho possui algumas restrições. Uma delas diz respeito da utilização da norma ISO/IEC 9126 para as definições das métricas. Atualmente já existe outra norma que a substitui [18]. Além desta é importante notar que existem outras metodologias ágeis que não foram contempladas pela pesquisa e que devem ser consideradas para trabalhos futuros.

As três principais conclusões deste trabalho foram: primeiramente ao fato que as métricas encontradas podem ser utilizadas por equipes que façam uso de *Kanban*, *Scrum* e/ou XP para auxiliar com o gerenciamento de seus processos, contribuindo para o incremento da qualidade. Que conforme apresentado na Tabela 15, foi encontrado um número baixo de métricas para satisfação do cliente e um número maior para qualidade do processo e do produto, logo os resultados encontrados não condizem com os princípios ágeis, apresentados na Tabela 6, onde busca-se satisfazer os clientes. E, finalmente, que foi constatado que várias métricas podem ser utilizadas por diversas metodologias. As métricas não são restritas a uma só abordagem.

Os resultados encontrados abrem caminho para trabalhos futuros, por exemplo, para outras pesquisas que busquem métricas para outras metodologias ágeis como, *Feature Driven Development*, ou *Crystal Family*. Outra possibilidade é a realização de pesquisas em casos reais para as métricas encontradas.

Outro possível trabalho futuro é a realização de uma pesquisa direcionada para completar as informações das métricas obtidas, completas e incompletas. Assim como exemplo de resultado para esse trabalho sugerido, foi montada a Tabela 17. Esta tabela foi construída a partir das informações da ISO/IEC 9126 parte 2 [21]. Ela é um exemplo de métrica completa com todas as informações necessárias para a sua descrição e compreensão.

Tabela 17. - Exemplo de Métrica Completa adaptado de [21]

Métricas								
Nome da Métrica	**Propósito da Métrica**	**Método de Aplicação**	**Medição, fórmula e dados de elementos computacionais**	**Interpretação dos Valores Medidos**	**Tipo de Métrica da Escala**	**Tipo de Medida**	**Entrada para Medição**	**Audiência Alvo**
Failure Analysis Efficiency (ID 28)	O usuário consegue determinar de forma eficiente a causa da falha? O técnico responsável pela manutenção consegue encontrar de forma eficiente a causa da falha? Qual a facilidade para se analisar a causa da falha?	Observar o comportamento do usuário ou responsável pela manutenção que está tentando resolver as falhas.	X = Sum(T)/N T = Tout – Tin Tout = Tempo para se encontrar a causa da falha. Tin = Tempo para se receber o indicativo de falha N = Número de falhas registradas	0<=X Quanto menor melhor	Taxa = (Ratio)	T = Tempo Tin, Tout= Tempo N = Count X = tempo/count	Relatório de resolução de problema Relatório de operação	Desenvolvedor Responsável pela Manutenção Operador

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **HAMED, A. ABUSHAMA, H. 2013.** POPULAR AGILE APPROACHES IN SOFTWARE DEVELOPMENT: Review and Analysis. International Conference on Computing, Electrical and eletronic Engineering. 2013

[2] **SOMMERVILLE, I. 2009.** SOFTWARE ENGINEERING. 9ed. Boston: PEARSON EDUCATION INC. 2009. ISBN-13: 978-0-13-703515-1, ISBN-10: 0-13-703515-2.

[3] **PRESSMAN, R. 2001.** SOFTWARE ENGINEERING: A Practitioner`s Approach. 5ed. New York: THE MCGRAW-HILL. 2001. ISBN 0073655783.

[4] **SERANA. 2007.** AN INTRODUCTION TO AGILE SOFTWARE DEVELOPMENT - San Mateo. SERENA SOFTWARE, INC. 2007.

[5] **LARMAN, C. 2002.** APPLYING UML AND PATTERNS TRAINING COURSE: A Desktop Seminar From Craig Larman. 3ed. Upper River Saddle: Prentice Hall. 2002. ISBN 0-13-148906-2.

[6] **BECK, K. GRENNING, J. MARTIN, R. BEEDLE, M. HIGHSMITH J. MELLOR, S. BENNEKUM, A. HUNT, A. SCHAWABER, K. COCKBURN, A. JEFFRIES, R. SUTHERLAND, J. CUNNINGHAM, W. KERN, J. THOMAS, D. FOWLER, M. MARIACK, B. 2001.** MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT. Disponível em: <http://agilemanifesto.org> acesso em: <27/11/2013>

[7] **SIMÕES, C. 2013.** SISTEMÁTICA DE MÉTRICAS, QUALIDADE E PRODUTIVIDADE. Developer`s Magazine. 1999. Disponível no site http://www.bfpug.com.br/Artigos/sistematica_metricas_simoes.htm

[8] **DYBÅ, T. DINGSØYR, T. 2008.** EMPIRICAL STUDIES OF AGILE DEVELOPMENT: A systematic review. Trondheim: Information and Software Technology: 2008.

[9] **BRASIL**. **TRIBUNAL DE CONTAS DA UNIÃO. 2013**: Relatório Técnico TC 010.663/2013-4.

[10] **MORESI, E. 2003.** METODOLOGIA DA PESQUISA. Brasília: Universidade Católica de Brasília: Pró-Reitoria de Pós-Graduação - PRPG. 2003

[11] **BIOLCHINI, J. MIAN, P. NATALI, A. TRAVASSOS, G. 2005.** SYSTEMATIC REVIEW IN SOFTWARE ENGINEERING. Rio de Janeiro: COPPE/UFRJ. 2005

[12] **KITCHENHAM, B. CHARTERS, S. 2007.** GUIDELINES FOR PERFORMING SYSTEMATIC LITERATURE REVIEWS IN SOFTWARE ENGINEERING. Keele: KEELE UNIVERSITY. 2007.

[13] **KITCHENHAM, B. BRERETON, P. 2013**. A SYSTEMATIC REVIEW OF SYSTEMATIC REVIEW PROCESS RESEARCH IN SOFTWARE ENGINEERING. Staffordshire: Elsevier. 2013.

[14] **MENDES, F. 2010.** MELHORIA DE PROCESSOS DE TECNOLOGIA DA INFORMAÇÃO MULTI-MODELO. Goiânia: Instituto de Informática/UFG. 2010.

[15] **BUDGEN, D. TURNER, M. BRERETON, P. KITCHENHAM, B. 2008** USING MAPPING STUDIES IN SOFTWARE ENGINEERING. Proceedings of PPIG Psycology of Programming Interest Group 2008.

[16] **PETERSEN, K. FELDT, R. MUJTABA, S. MATTSSON, M. 2008.** SYSTEMATIC MAPPING STUDIES IN SOFTWARE ENGINEERING. 12th International Conference on Evaluation and Assessment in Software. 2008.

[17] **GALIN, D. 2004.** SOFTWARE QUALITY ASSURANCE: From theory to implementation. 1ed. London: PEARSON EDUCATION LIMITED. 2004. ISBN: 0-201-70945-7.

[18] **SOUSA, E. MARINHO, D. 2010.** PESQUISA DE QUALIDADE DE SOFTWARE BRASILEIRO 2009. Brasília: SECRETARIA DE POLÍTICA DE INFORMÁTICA, MINISTÉRIO DA CIÊNCIA E TECNOLOGIA. 2010.

[19] **KOSCIANSKI, A. VILLAS-BOAS, A. RÊGO, C. ASANOME, C. SCALET, D. ROMERO, D. CIESLAK, J. PALUDO, M. FROSSARD, R. VOSTOUPAL. 1999.** GUIA PARA UTILIZAÇÃO DAS NORMAS SOBRE AVALIÇÃO DE QUALIDADE DE PRODUTO DE SOFTWARE - ISO/IEC 9126 E ISO/IEC 14598. Curitiba: ABNT - Associação Brasileira de Normas Técnicas. 1999.

[20] **ISO/IEC. 2001.** SOFTWARE ENGINEERING - PRODUCT QUALITY - Part 1: Quality model. Geneva: ISO/IEC. 2001.

[21] **ISO/IEC. 2002.** SOFTWARE ENGINEERING - PRODUCT QUALITY - Part 2: External Metrics. Geneva: ISO/IEC. 2002.

[22] **ISO/IEC. 2001.** SOFTWARE ENGINEERING - PRODUCT QUALITY - Part 3: Internal Metrics. Geneva: ISO/IEC. 2002.

[23] **ISO/IEC. 2001.** SOFTWARE ENGINEERING - SOFTWARE PRODUCT QUALITY - Part 4: Quality in Use Metrics. Geneva: ISO/IEC. 2001.

[24] **SOLINGEN, R. BERGHOUT, E. 1991.** THE GOAL/QUESTION/METRIC METHOD: a practical guide for quality improvement of software development. London: McGraw-Hill Publishing Company. 1991.

[25] **FLORAC, W. PARK, R. CARLETON, A. 1997.** PRACTICAL SOFTWARE MEASUREMENT: Measuring for Process Management and Improvement. Pittsburgh. Carnegie Mellon University. 1997.

[26] **MCGARRY, J. 2002.** PRACTICAL SOFTWARE MEASUREMENT: objective information for decision makers. 1ed. Boston: ADDISON-WESLEY. 2002. ISBN 0201715163.

[27] **ROCHA, A. SOUZA, G. BARCELLOS, M. 2012.** MEDIÇÃO DE SOFTWARE E CONTROLE ESTATÍSTICO DE PROCESSOS. Brasília: MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO. 2012.

[28] **GAMBA, M. BARBOSA, C.** APLICAÇÃO DE MÉTRICAS DE SOFTWARE COM SCRUM. Criciúma: UNIVERSIDADE DO EXTREMO SUL CATARINENSE.

[29] **VERSIONONE. 2013,** AGILE METHODOLOGIES FOR SOFTWARE DEVELOPMENT. Disponível no site <http://www.versionone.com/Agile101/Agile-Development-Methodologies-Scrum-Kanban-Lean-XP/> acesso: 30 abr. 2013.

[30] **AMBLER, S. 2005.** QUALITY IN AN AGILE WORLD. Software Quality Professional. 2005.

[31] **SCHWABER, K. SHUTHERLAND, J. 2013.** UM GUIA DEFINITIVO PARA O SCRUM: As Regras do Jogo. Scrum.org: Improving the Professions of Software Development

[32] **HIGHSMITH, J. 2002.** AGILE SOFTWARE DEVELOPMENT ECOSYSTEMS. Indianapolis: ADDISON-WESLEY. 2002. ISBN 0-201-76043-6

[33] **SIRSHAR, M., ARIF, F. 2012.** EVALUATION OF QUALITY ASSURANCE FACTORS IN AGILE METHODOLOGIES. Rawalpindi: INTERNATIONAL JOURNAL PUBLISHERS GROUP. 2012.

[34] **ANDERSON, D. 2008.** THE KANBAN PRIMER: A Cultural Evolution in Software. 2008.

[35] **KNIBERG, H. SKARIN, M. 2010.** KANBAN AND SCRUM MAKING THE MOST OF BOTH. C4Media. 2010. ISBN 978-0-557-13832-6

[36] **MAND, C. 2013**. USING THE LEAN MODEL FOR PERFORMANCE IMPROVEMENT, Milwaukee. Disponível em: < http://videos.med.wisc.edu/files/Mand.pdf> acesso 30 abr. 2013.

[37] **RAMOS, R. 2013.** "DESENVOLVIMENTO ÁGIL," UNIVASF. Disponível em: < http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ESI2009_2/Aula13_14_DesenvolvimentoAgil.pdf> acesso: 30 abr. 2013.

[38] **TAKATS, S. STILLMAN, D. KORNBLITH, S. 2013.** CHESLACK-POSTAVA, F. ZOTERO. 2013

ANEXO I: LISTA DOS ARTIGOS ENCONTRADOS NO MAPEAMENTO SISTEMÁTICO

ANDERSON, D. 2005. STRETCHING AGILE TO FIT CMMI LEVEL 3. IEEE. Agile Development Conference. 2005.

DOWNEY, S. SUTHERLAND, J. 2013. SCRUM METRICS FOR HYPERPRODUCTIVE TEAMS: HOW THEY FLY LIKE FIGHTER AIRCRAFT. IEEE: 46th Hawaii International Conference on System Sciences. 2013.

DUBINSKY, Y. HAZZANN, O. 2005. THE ROLE OF A PROJEC-BASED CAPSTONE COURSE. St. Louis. 2005.

DUBINSKY, Y. TALBY, D. HAZZAN, O. KEREN, A. 2005. AGILE METRICS AT THE ISRAELI AIR FORCE. IEEE: Proceedings of the Agile Development Conference. 2005

FOKAEFS, M. TSANTALIS, N. STROULIA, E. CHATZIGEORGIOU, A. 2012. IDENTIFICATION AND APPLICATION OF EXTRACT CLASS REFACTORINGS IN OBJECT-ORIENTED SYSTEMS. Elsevier: The Journal of System and Software 85. 2012.

GREENING, D. 2010. ENTERPRISE SCRUM: SCALING SCRUM TO THE EXECUTIVE LEVEL. Proceeding of the 43rd Hawaii International Conference on System Sciences. 2010.

HARTMANN, D. DYMOND R. 2006. APPROPRIATE AGILE MEASUREMENT: USING METRICS AND DIAGNOSTICS TO DELIVER BUSINESS VALUE. IEEE Agile Conference. 2006.

HENDERSON, M. ALLEMAN, G. 2003. MAKING AGILE DEVELOPMENT WORK IN A GOVERNMENT CONTRACTING ENVIRONMENT. Salt Lake City: Agile Development. 2003

IKOMA, M. OOSHIMA, M. TANIDA, T. OBA, M. SAKAI, S. 2009. USING A VALIDATION MODEL TO MEASURE THE AGILITY OF SOFTWARE DEVELOPMENT IN A LARGE SOFTWARE DEVELOPMENT ORGANIZATION. Vancouver. 2009.

ILIEVA, S. IVANOV, P. STEFANOVA, E. 2004. AN EMPIRICAL VALIDATION OF OBJECT-ORIENTED METRICS IN TWO DIFFERENT ITERATIVE SOFTWARE PROCESSES. IEEE: EUROMICRO Conference. 2004.

ILIEVA, S. IVANOV, P. STEFANOVA, E. 2004. ANALYSES OF AN AGILE METHODOLOGY IMPLEMENTATION. IEEE: 30th EUROMICRO Conference. 2004.

KORU, A. EMAM, K. 2009. THEORY OF RELATIVE DEPENDENCY: HIGHER COUPLING CONCENTRATION IN SMALLER MODULES AND ITS IMPLICATIONS FOR SOFTWARE REFACTORING AND QUALITY. IEEE Software. 2009.

MCINTOSH, S. ADAMS, B. HASSAM, A. 2010. THE EVOLUTION OF ANT BUILD SYSTEMS. IEEE Computer Society. 2010

OLAGUE, H. ETZKORN, L. GHOISTON, S. QUATTLEBAUM, S. 2007. EMPIRICAL VALIDATION OF THREE SOFTWARE METRICS SUITES TO PREDICT FAULT-PRONENESS OF OBJECT-ORIENTED CLASSES DEVELOPED USING HIGHLY ITERATIVE OR AGILE SOFTWARE DEVELOPMENT PROCESSES. IEEE. Computer Society. Vol.33, no6. 2007

OLK, R. 2011. AGILE & KANBAN IN COORDINATION. IEEE: Agile Conference. 2011.

PEREPLETCHIKOV, M. RYAN, C. TARI, Z. 2010. THE IMPACT OF SERVICE COHESION ON THE ANALYZABILITY OF SERVICE-ORIENTED SOFTWARE. IEEE Computer Society. 2010

RODEN, P. VIRANI, S. ETZKORN, L. MESSIMER, S. 2007. AN EMPIRICAL STUDY OF THE RELATIONSHIP OF STABILITY METRICS AND THE QMOOD QUALITY MODELS OVER SOFTWARE DEVELOPED USING HIGHLY ITERATIVE OR AGILE SOFTWARE PROCESSES. IEEE: International Working Conference on Source Code Analysis and manipulation. 2007.

SEDEHI, H. MARTANO, G. 2012. METRICS TO EVALUATE & MONITOR AGILE BASED SOFTWARE DEVELOPMENT PROJECTS. IEEE: Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement. 2012.

SVENSSON, H. HÖST, M. 2005. INTRODUCING AN AGILE PROCESS IN A SOFTWARE MAINTENANCE AND EVOLUTION ORGANIZATION. IEEE: Proceedings fo the Ninth European Conference on Software Maintenance and Reegineering. 2005.

TALBY, D. KEREN, A. HAZZAN, O. BUBINSKY, Y. 2006. AGILE SOFTWARE TESTING IN A LARGE-SCALE PROJECT. IEEE: Computer Society. 2006.

TENGSHE, A. NOBLE, S. 2007. ESTABLISHING THE AGILE PMO: MANAGING VARIABILITY ACROSS PROJECTS AND PORTFOLIOS. IEEE: Agile. 2007.