ConBRepro
X CONGRESSO BRASILEIRO DE ENGENHARIA DE PRODUÇÃO
EVENTO ON-LINE
02 a 04 de dezembro 2020

Vantagens e desvantagens da utilização do método Mudge na análise de importância de desperdício de software

Everton Michels
Engenharia de Produção – Universidade Federal de Santa Catarina
Fernando Antonio Forcellini
Engenharia Mecânica – Universidade Federal de Santa Catarina

Resumo: Diversos métodos são encontrados na literatura a fim de analisar e priorizar desperdícios de software. Entretanto, pouco se sabe sobre a diferença nesses métodos no que tange a suas vantagens e desvantagens quando estes são comparados. Desta forma, o objetivo deste estudo é identificar os principais métodos de análise e priorização de desperdícios de software, bem como, verificar as vantagens e desvantagens comparando os mesmos ao método de Mudge. Para tal, foi realizado um mapeamento sistêmico a fim de embasar os conceitos teóricos, identificar os principais métodos com base na frequência relatada, bem como, apoiar na identificação dos resultados deste trabalho. Com base nisso, os principais resultados encontrados foram a identificação das vantagens e desvantagens de se utilizar o método de Mudge comparado aos métodos *Analytic Hierarchy Process* (*AHP*) e *Planning Game* (*PG*), bem como, a análise para identificar dentre esses métodos quais tem os melhores valores econômicos. Como conclusões, pode-se verificar que a utilização de Mudge possui um valor econômico superior aos outros dois principais métodos relatados neste estudo, mesmo este método não sendo comumente utilizado na priorização de desperdícios no desenvolvimento de software.

Palavras-chave: Métodos, Priorização, Desperdício, Desenvolvimento de Software

Advantages and disadvantages of using the Mudge method to analyze the importance of software waste

Abstract: Several methods are found in the literature in order to analyze and prioritize software waste. However, little is known about the difference in these methods in terms of their advantages and disadvantages when they are compared. Thus, the objective of this study is to identify the main methods of analysis and prioritization of software waste, as well as, to verify the advantages and disadvantages comparing them to the Mudge method. To this end, a systemic mapping was carried out in order to support the theoretical concepts, to identify the main methods based on the reported frequency, as well as to support the identification of the results of this work. Based on this, the main results found were the identification of the advantages and disadvantages of using the Mudge method compared to the Analytic Hierarchy Process (AHP) and Planning Game (PG) methods, as well as, the analysis to identify among those methods which have the best economic values. As conclusions, it can be verified that the use of Mudge has an economic value superior to the other two main methods reported in this study, even though this method is not commonly used in prioritizing waste in software development.

Keywords: Methods, Prioritization, Waste, Software Development

1. Introdução

Na atualidade vivida por um mundo globalizado e constantemente mutável, as organizações estão cada vez mais preocupadas em serem mais eficientes e eficazes, não só entregando produtos e serviços de valor, como fazendo o máximo possível para evitar desperdícios.

No segmento de software não é diferente. Com a tecnologia evoluindo de forma constante, rápida e disrruptiva, as organizações procuram errar o mais cedo possível para evitar desperdícios, e priorizar o que de fato importa pro cliente (RIES, 2012).

Diversos métodos de priorização/importância são encontrados na literatura, especialmente na de desenvolvimento de software e desenvolvimento de produto/serviço, entre eles pode-se citar: AHP, *Planning Game*, *Quality Functional Deployment* (QFD), *Binary Search Tree*, *Cost-Value*, entre outros (ACHIMUGU, et al., 2014; DEVULAPALLI, 2016; SUFIAN, et al., 2018).

Estes métodos na maioria das vezes, são muito utilizados para a priorização de requisitos de software e não desperdícios. Entretanto, o que muda de fato na utilização dos mesmos é o conteúdo, e não a forma como estes são realizados, dessa forma, sejam pra priorizar requisitos, ou desperdícios, eles possuem a mesma funcionalidade.

A literatura relata em revisões sistemáticas os principais métodos de priorização de requisitos, baseado na frequência com os quais são encontrados. Em alguns estudos estes métodos são até comparados, porém não do ponto de vista de valor econômico, muito menos comparado ao método de Mudge.

Sabendo então deste gap, o objetivo deste estudo é identificar os principais métodos de priorização relatados no desenvolvimento de software, e compará-los por meio do valor econômico ao método de Mudge, e desta forma identificas suas vantagens e desvantagens.

2. Desperdício em software

Aprender a ver o desperdício é o primeiro passo para desenvolver avanços com o pensamento enxuto. Se algo não agrega diretamente o valor percebido pelo cliente, é desperdício. Se existe uma maneira de fazer sem ele, é desperdício. Em 1970, Winston Royce escreveu que as etapas fundamentais de todo o desenvolvimento de software são análise e codificação (POPPENDIECK; POPPENDIECK, 2003).

Shigeo Shingo, um dos mentores do Sistema Toyota de Produção, identificou sete tipos de resíduos de fabricação. Sua lista ajudou muitos gerentes de manufatura a encontrar resíduos onde nunca teriam pensado em procurar. Para ajudar os gerentes de desenvolvimento de software em sua busca para encontrar aquela coisa indescritível chamada desperdício, os autores traduziram os sete desperdícios de fabricação nos sete desperdícios de desenvolvimento de software, conforme visto no Quadro 1 (POPPENDIECK; POPPENDIECK, 2003).

Quadro 1 – Os Sete Desperdícios

Os sete desperdícios da fabricação	Os sete desperdícios do desenvolvimento de software
Inventário	Trabalho Parcialmente Feito
Super-processamento	Processos extras
Superprodução	Recursos extras
Transporte	Troca de tarefas
Espera	Espera
Movimentação	Movimentação
Defeitos	Defeitos

Fonte: Adaptado de Poppendieck e Poppendieck (2003)

Desta forma, se faz necessário esclarecer de forma sintética essa associação e o que é cada um destes sete desperdícios na área de software. O primeiro deles é o desenvolvimento de software parcialmente feito, ou trabalho parcialmente feito, o qual tem

uma tendência a se tornar obsoleto e atrapalha outros desenvolvimentos que talvez precisem ser feitos. Mas o grande problema com o software parcialmente pronto é que você pode não ter ideia se ele irá ou não funcionar. Você pode até ter uma pilha de código que pode até ser testado, mas até que o software seja integrado ao restante do ambiente, você realmente não sabe quais problemas podem estar ocultos até que o software esteja realmente em produção, você realmente não sabe se ele resolverá o problema do negócio (POPPENDIECK; POPPENDIECK, 2003).

Já para o desperdício de processos extras a grande pergunta é: toda essa papelada é realmente necessária? Papelada consome recursos. A papelada diminui o tempo de resposta. A papelada oculta problemas de qualidade. A papelada se perde. A papelada degrada e se torna obsoleta. A papelada que ninguém se importa em ler não agrega valor (POPPENDIECK; POPPENDIECK, 2003).

Quanto ao desperdício de recursos extras, pode até parecer uma boa ideia colocar alguns em um sistema, caso sejam necessários. Os desenvolvedores podem adicionar um novo recurso técnico apenas para ver como ele funciona. Cada parte do código no sistema deve ser rastreada, compilada, integrada e testada toda vez que o código é tocado e, em seguida, deve ser mantido por toda a vida útil do sistema (POPPENDIECK; POPPENDIECK, 2003).

Um dos maiores desperdícios no desenvolvimento de software geralmente está na espera de que as coisas aconteçam. Atrasos no início de um projeto, atrasos na equipe, atrasos devido a documentação excessiva de requisitos, atrasos nas revisões e aprovações, atrasos nos testes e atrasos na implantação são desperdícios (POPPENDIECK; POPPENDIECK, 2003).

O desenvolvimento é uma atividade que requer grande concentração; portanto, caminhar pelo corredor leva muito mais tempo do que se imagina. Provavelmente, o desenvolvedor levará várias vezes o tempo para restabelecer o foco necessário para obter a resposta de uma pergunta. É por esse motivo que as práticas de desenvolvimento ágil de software geralmente recomendam que uma equipe trabalhe em uma única sala de trabalho onde todos tenham acesso a desenvolvedores, testadores e clientes ou representantes de clientes (POPPENDIECK; POPPENDIECK, 2003).

Por fim, um dos desperdícios mais comuns no software é a geração de defeitos, e ele pode ser medido pelo produto do impacto do defeito e o tempo que passa sem ser detectado. Um defeito crítico detectado em três minutos não é uma grande fonte de desperdício. Um pequeno defeito que não é descoberto há semanas é um desperdício muito maior (POPPENDIECK; POPPENDIECK, 2003).

Outros desperdícios de software são encontrados na literatura (Relatórios inúteis; desenvolver requisitos errados; falta de confiança em pedir para monitorar tudo; sem/ou difícil acesso às informações do cliente; pré-estudos e prova de conceito que não levam ao desenvolvimento; terceiros, fornecedores externos; trabalho parcialmente realizado; relatórios de problemas; retrabalho, etc.) e relatados por alguns autores (PESSÔA, et al. 2009; AL-BAIK; MILLER, 2014; POWER; CONBOY, 2014; BEHROOZI; KAMANDI, 2016; ALAHYARIA; GORSCHEK; SVENSSON, 2019), no entanto, todos acabam por categorizar os mesmos nos sete desperdícios de software explicitados por Poppendieck e Poppendieck (2003).

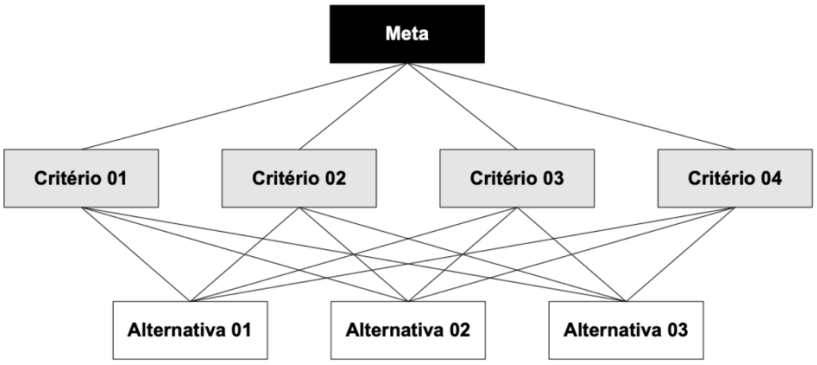
3. Métodos de priorização

Este tópico tem o intuito de explicitar de forma sintética os conceitos relacionados aos dois métodos de priorização mais frequentemente utilizados na literatura para priorização de desperdício de software (AHP e PG), conforme encontrado no Mapeamento Sistemático (MS), bem como, para relatar o conceito do método de Mudge (1971).

3.1 *Analytic Hierarchy Process*

O método AHP se dá pela decomposição de um problema em uma hierarquia de critérios mais facilmente analisáveis e comparáveis de modo independente, conforme pode ser visto na Figura 1. A partir do momento em que essa hierarquia lógica está construída, os tomadores de decisão avaliam sistematicamente as alternativas por meio da comparação, de duas a duas, dentro de cada um dos critérios (SAATY, 2008a; SAATY, 2008b).

Figura 1 – Exemplo de hierarquia de critérios/objetivos



Fonte: Vargas (2010)

Desta forma, a principal característica do método AHP é o uso de comparações entre pares, que são usadas para comparar as alternativas com relação aos vários critérios e para estimar os pesos destes critérios (VARGAS, 2010).

Para fazer comparações, é necessária uma escala de números que indique quantas vezes mais importante ou dominante um elemento está do que outro elemento em relação ao critério ou propriedade em relação à qual eles são comparados, conforme mostra a Figura 2 (SAATY, 2008a).

Figura 2 - Escala Fundamental de Números Absolutos

1	Igual Importância	As duas atividades contribuem igualmente para o objetivo
3	Importância pequena de uma para a outra	A experiência e o julgamento favorecem levemente uma atividade em relação à outra.
5	Importância grande ou essencial	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra.
7	Importância muito grande ou demonstrada	Uma atividade é muito fortemente favorecida em relação à outra.
9	Importância absoluta	A evidência favorece uma atividade em relação à outra com o mais alto grau de certeza.
2,4,6,8	Valores intermediários	Quando se procura uma condição de compromisso entre as duas definições.

Fonte: Adaptado de Saaty (2008a)

As prioridades são obtidas na forma exata, aumentando a matriz para grandes potências e somando cada linha e dividindo cada uma pela soma total de todas as linhas, ou aproximadamente adicionando cada linha da matriz e dividindo por seu total.

3.2 Planning Game

O Planning Game (PG) é comumente usado no planejamento e na decisão do que desenvolver em um projeto *Extreme Programming* (XP). No PG, os requisitos (escritos nos chamados cartões de histórias) são extraídos do cliente. Quando os requisitos são elicitados, eles são priorizados pelo cliente em três pilhas diferentes: (1) aquelas sem as quais o sistema não funcionará; (2) aquelas que são menos essenciais, mas fornecem valor comercial significativo; e (3) aquelas que seriam bom ter (BECK, 2000).

Ao mesmo tempo, os desenvolvedores estimam o tempo necessário para implementar cada requisito e, além disso, classificam os requisitos em três pilhas: (1) aqueles que eles podem estimar com precisão, (2) aqueles que eles podem estimar razoavelmente bem e (3) aqueles que eles não podem estimar (BECK, 2000).

Com base nas estimativas de tempo, ou escolhendo as cartas e calculando a data de lançamento, os clientes priorizam os requisitos dentro das pilhas e depois decidem quais requisitos devem ser planejados para o próximo lançamento (NEWKIRK; MARTIN, 2001). Assim, a técnica usa um algoritmo de classificação, semelhante à atribuição de números, para particionar os requisitos em uma das três pilhas. Em seguida, os requisitos em cada pilha são comparados entre si para obter uma lista classificada (KARLSSON, 1996).

O resultado do método é uma lista ordenada de requisitos. Isso significa que os requisitos são representados como uma classificação em uma escala ordinal, sem nenhuma informação sobre quanto mais importante um requisito é do que outro (KARLSSON, et al., 2007).

3.3 Mudge

O método de Mudge é comumente utilizado para avaliar funções. O método busca priorizar as funcionalidades por ordem de importância, as quais representam as necessidades definidas pelo cliente (PEREIRA FILHO, 1994).

O método compara as funcionalidades por pares. O mesmo tem início relacionando a funcionalidade “X” com uma funcionalidade “Y”, e assim, determina-se qual é a mais importante. A letra da funcionalidade escolhida como mais importante é colocada no cruzamento das respectivas funcionalidades da tabela de comparação, sempre vinculando a linha, com a coluna, conforme mostra a Figura 3.

Figura 3 – Exemplo da Aplicação de Mudge

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	PESO	%
A	B 5	A 3	A 1	A 3	A 1	A 1	A 1	A 3	A 1	A 3	A 3	A 3	A 1	A 5	29	11%
	B	B 5	B 5	B 5	B 5	B 5	B 5	B 5	B 5	B 5	B 5	B 5	B 5	B 5	70	27%
		C	D 1	E 1	F 1	G 1	H 1	C 3	C 1	C 3	C 3	C 3	C 1	C 3	17	7%
			D	E 1	F 1	G 1	H 1	D 3	D 1	D 3	D 3	D 3	D 1	D 3	18	7%
				E	E 1	E 1	E 1	E 3	E 1	E 3	E 3	E 3	E 1	E 3	22	9%
					F	F 1	F 1	F 3	F 1	F 3	F 3	F 3	F 1	F 3	21	8%
						G	G 1	G 3	G 1	G 3	G 3	G 3	G 1	G 3	20	8%
							H	H 3	H 1	H 3	H 3	H 3	H 1	H 3	19	7%
								I	J 1	I 3	I 1	I 1	N 1	I 3	8	3%
									J	J 3	J 3	J 3	J 1	J 3	14	5%
										K	K 1	M 1	N 1	K 3	4	2%
											L	L 1	N 1	O 1	1	0%
												M	N 1	M 3	4	2%
													N	N 3	7	3%
														O	1	0%
														TOTAL	255	100%

Fonte: Domingues, Sellitto e Lacerda (2013, p. 382)

Além disso, seguindo da letra, coloca-se a diferença na importância das funções, as quais são expressas pelos fatores 1, 3 ou 5 conforme os pesos (PEREIRA FILHO, 1994):

- Peso 1 = funcionalidade ligeiramente mais importante que a comparada;
- Peso 3 = funcionalidade mais importante que a comparada;
- Peso 5 = funcionalidade muito mais importante que a comparada.

Esse procedimento de comparação, é repetido por todas as linhas e colunas da tabela, de forma pareada entre linha/coluna, até que todas as funcionalidades linha/coluna tenham disso comparadas. A análise é finalizada ao somar os fatores-peso de cada uma das funcionalidades, e assim, informando o valor total na coluna de peso. A funcionalidade mais importante será justamente a que tiver o maior peso na coluna de total de peso (BASSO, 1991).

Para se encontrar o percentual de importância de cada funcionalidade, basta dividir o fator peso de cada funcionalidade pelo somatório de todos os fatores peso. Isso possibilita a um relato gráfico das funcionalidades por ordem de importância (DOMINGUES; SELLITTO; LACERDA, 2013).

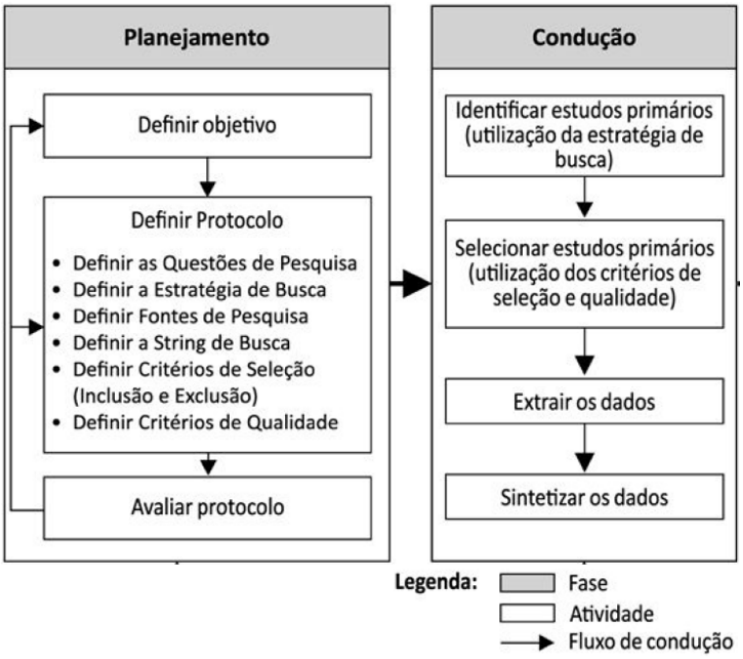
4. Metodologia

O Mapeamento Sistemático (MS), tem por objetivo identificar e classificar estudos que se relacionam com um tópico específico de pesquisa (KITCHENHAM; CHARTERS, 2007).

Desta forma, o MS é capaz de gerar resultados que possibilitem a identificação de lacunas na área ou tópico pesquisado, e com isso tornar claro o caminho para possíveis trabalhos futuros (KITCHENHAM; CHARTERS, 2007; PETERSEN et al., 2008; KITCHENHAM et al., 2010).

Para tal, o MS segue uma estrutura conforme é mostrada na Figura 4. O objetivo levantado para o MS foi o seguinte: Identificar estudos que utilizem métodos de priorização de desperdícios no desenvolvimento de software.

Figura 4 – Estrutura do Mapeamento Sistemático



Fonte: Adaptado de Nakagawa et al. (2017, p. 19)

Como um MS se diferencia em partes de uma Revisão Sistemática, principalmente pela revisão sistemática ser mais detalhada e específica, o tópico a ser abordado na pesquisa para o MS foi: priorização de desperdício e desenvolvimento de software.

A estratégia de busca definida para o MS contempla as seguintes atividades: definição das fontes de pesquisa; definição dos métodos de busca e descrição da aplicação dos métodos de busca.

Desta forma, as fontes as quais serão feitas as buscas foram: *Scopus* e *Web of Science* (WOS). Os métodos de busca por sua vez foram a busca automática, e o *snowballing*. Por fim, a *String* de busca definida para o MS conforme o objetivo do mesmo e o tópico de pesquisa foi a seguinte: (*waste AND software development AND method*) *and* (*prioritization or importance*), as quais se concentraram no título, resumo e palavras-chave.

Com base no objetivo do MS, e visando encontrar estudos que contemplem tanto a parte teórica a respeito de métodos de priorização e desperdício de software, quanto a parte empírica de sua utilização, os critérios para a inclusão de estudos a princípio estão voltados para revisões tradicionais, pesquisa-ação e estudos de caso.

Estes estudos são tanto de periódicos, conferências ou livros conceituados e referenciados nos temas. E devem contemplar os seguintes questionamentos (NAKAGAWA et al., 2017):

a) O estudo relata sobre desperdício de software?

b) O estudo relata sobre métodos de priorização de importância?

c) O estudo aborda a integração destes dois temas?

Da mesma forma, os critérios para exclusão dos estudos contemplam as seguintes características (NAKAGAWA et al., 2017):

a) O estudo não relata sobre os temas deste trabalho.

b) O estudo não relata sobre a integração entre os temas deste trabalho.

c) O estudo é uma versão anterior de um estudo mais completo sobre a mesma pesquisa.

d) O estudo é a descrição de um curso, editorial, resumo de palestra, workshop ou tutorial;

A fim de não correr o risco de introduzir um viés ao MS, optou-se nesse ponto por não incluir nenhum critério de qualidade ao mesmo.

Nesta fase do MS optou-se por trazer todas as suas atividades de forma integrada no mesmo tópico. Para a identificação e seleção dos estudos primários foi seguido o processo conforme mostrado na Figura 5.

Desta forma, o primeiro passo foi realizar a busca de estudos nas bases bibliográficas citadas anteriormente, bem como, utilizando *snowballing*. A busca com a *String*: (*waste AND software development AND method*) *and* (*prioritization* or *importance*) resultou em uma quantidade de 8 estudos na *Scopus* e 3 na WOS, totalizando assim 11 estudos.

O próximo passo foi passar pela Etapa 0 (zero), a qual realiza a eliminação dos artigos duplicados, resultando assim em 10 estudos.

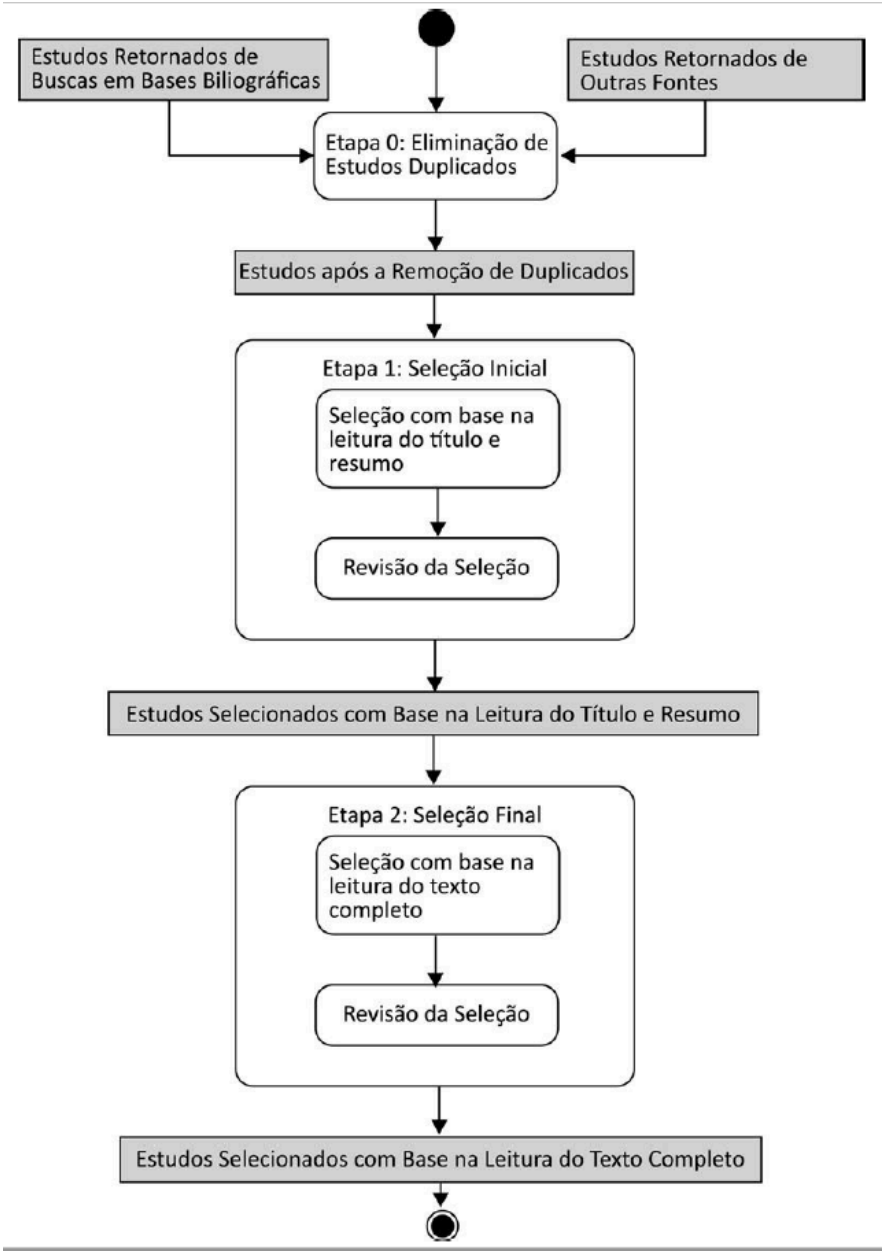
O passo seguinte na condução foi a realização da Etapa 1 (um), a qual visa a seleção para a leitura do título e resumo e posteriormente a revisão dessa seleção, a fim de confirmar se os estudos selecionados após a leitura estão de acordo com os critérios definidos. Desta forma, restou então 1 estudo aderente aos critérios de exclusão estabelecidos.

Por fim, foi realizada a Etapa 2, a qual foi responsável pela seleção dos estudos para leitura completa dos mesmos e posterior revisão. O resultado final de estudos selecionados para o MS totalizou então 1 trabalho (ALAHYARI; GORSCHEK; SVENSSON, 2019).

O estudo foi registrado em uma planilha eletrônica, bem como em uma ferramenta para análise (*Mendeley*), os quais foram separados conforme a etapa adequada que se

encontram, e com as respectivas informações registradas conforme sugerido por Nakagawa et al. (2017), caracterizando assim a etapa de extração de dados.

Figura 5 – Etapas de Seleção dos Estudos



Fonte: Nakagawa et al. (2017, p. 65)

Como não houve um resultado substancial de estudos encontrados que integrem o desperdício de software, com métodos de priorização/importância, optou-se por fazer um mapeamento mais amplo ainda, no qual os estudos fossem direcionados não a integração entre métodos de priorização/importância e desperdício, mas apenas a métodos de priorização na área de software.

Esse MS focou em revisões da literatura, a fim de verificar estudos sintéticos que buscassem relatar os métodos resumidamente, bem como, verificar a frequência de suas ocorrências. Isso foi feito pois independente da utilização dada ao método, como por exemplo, priorização de requisitos, o mesmo poderia ser utilizado para priorização de desperdícios, pois o que muda em si é o conteúdo e não o método.

Desta forma, seguindo a metodologia, optou-se nesse momento por sintetizar os resultados destes MS's no Quadro 2.

Quadro 2 – Síntese dos MS's

Autor	Ano	Método	Métodos Mais Citados
Alahyari, Gorschek e Svensson	2019	Revisão Sistemática da Literatura	7 Desperdícios de Software
Sufian, et al.	2018	Revisão Sistemática da Literatura	*AHP, Numeral Assiggnment Technique, Cumulative Voating, Planning Game, Binary Search Tree*
Hujainah, et al.	2018	Revisão Sistemática da Literatura	*AHP, Numeral Assiggnment Technique, Cumulative Voating, Planning Game, Cost-Value*
Devulapalli	2016	Revisão Sistemática da Literatura	*AHP, Cumulative Voating, Cost-Value, Planning Game, Weigers*
Achimugu, et al.	2014	Revisão Sistemática da Literatura	*AHP, QFD, Cumulative Voating, Planning Game, Binary Search Tree*
Aasem, Ramzan e Jaffar	2010	Revisão Sistemática da Literatura	*AHP, Cumulative Voating, Cost-Value, Planning Game, Numeral Assiggnment Technique*

Fonte: Elaborado pelo autor

5. Discussão e resultados

A maioria da literatura citada no MS, abordou quatro métodos principais para priorização de requisitos (AHP, *Planning Game*, *Cost-Value* e *Cumulative Voating*), bem como, combinações deles.

A estrutura de tomada de decisão aplicada no AHP, o qual é aplicável a muitos domínios com parâmetros específicos do domínio do problema definido, muitas vezes constituiu base para o método de *Cost-Value*, e o conceito de priorização em camadas.

Observou-se nos estudos encontrados também que, quando os métodos foram combinados, os mesmos não possuíam uma linha de base comum nos projetos, em que as combinações podiam ser válidas. A natureza dos projetos e o nível de abstração dos requisitos variaram amplamente para deduzir o uso prático destas combinações.

Os métodos recentes propostos explicitam o quão complexo pode ser o processo de aplicação dos mesmos. Observou-se que os mesmos não oferecem a flexibilidade de re-planejar, re-priorizar de maneira simples.

Para identificar dentre os métodos vistos, qual seria o principal deles, utilizou-se o conceito de valor econômico. Desta forma, ficou estabelecida a equação abaixo para balizar a decisão da escolha do método mais eficiente/eficaz para este estudo, onde B são os benefícios e C são os custos/esforços (MILES, 2015).

$$VE = \sum B \; / \sum C$$

Para explicitar como se chegou até o respectivo valor econômico, optou-se por relatar o desenvolvimento do mesmo na Tabela 1. Nesta tabela constam os métodos aqui explicitados, pois estes foram os principais relatados na literatura de software, com base na sua frequência de ocorrência (ACHIMUGU, et al., 2014; DEVULAPALLI, 2016; SUFIAN, et al., 2018).

Além disso, são identificados os benefícios (priorização e robustez) para este estudo, bem como, os esforços (tempo, facilidade e entendimento) relacionados a utilização de cada método para a priorização de desperdícios de software.

Os benefícios foram caracterizados como: priorização (o ato de ordenar uma lista qualquer, nesse caso os desperdícios, por meio do seu grau de importância), e robustez (o grau com

que se consegue comparar os itens dessa mesma lista por meio de escalas a fim de se chegar a uma priorização da importância).

O esforço de tempo refere-se ao tempo gasto para construir o método com base na mesma quantidade de desperdícios identificados. Já o de facilidade, é a facilidade com a qual o método é construído para o seu devido uso. Por fim, o entendimento, é o quão fácil é de se compreender o método sem um conhecimento prévio.

Com isso, foi estabelecida uma escala Likert com valores de 1 (mais fraco) a 5 (mais forte) a fim comparar os itens com os respectivos métodos, usando o critério de mais forte para os benefícios, e mais fraco para os esforços, e desta forma chegar ao valor final de todos.

Tabela 1 – Análise do Valor Econômico

		Método		
		AHP	**Planning Game**	**Mudge**
	Escala	Razão	Ordinal	Razão
Benefícios	Priorização	5	5	5
	Robustez	5	1	4
	Tempo	4	2	3
Esforço	Facilidade	3	1	2
	Entendimento	3	2	2
	Cálculo Benefício	(5*0,5)+(5*0,5) = 5	(5*0,5)+(1*0,5) = 3	(5*0,5)+(4*0,5) = 4,5
	Cálculo Esforço	(4*0,33)+(3*0,33) +(3*0,33) = 3,3	(2*0,33)+(1*0,33) +(2*0,33) = 1,65	(3*0,33)+(2*0,33)+(2*0,33) = 2,31
	Valor Econômico	**5 / 3,3 = 1,51**	**3 / 1,98 = 1,8**	**4,5 / 2,31 = 1,9**

Fonte: Elaborada pelo autor

Com base na análise realizada na Tabela 1, e levando-se em conta a fórmula utilizada para mensurar o valor econômico dos métodos, pode-se verificar que o método que mais oferece a relação custo/benefício é o de Mudge, seguido do Planning Game e do AHP consequentemente. Para identificar as vantagens e desvantagens de cada método, optou-se por sintetizá-las no Quadro 3.

Quadro 3 – Síntese das Vantagens e Desvantagens

Método	Vantagem	Desvantagem
AHP	Possui uma granularidade fina, é sofisticado, algoritmo de execução, produz resultados mais confiáveis, baseados em uma escala de proporção, é tolerante a falhas e inclui uma verificação de consistência, comparação por pares	Complexo, difícil entendimento, não escalável, necessidade de programar, consome muito tempo
Planning Game	Facilidade de uso, rápido e fácil entendimento do método	Não mede bem com alto valor de itens, não escalável, não compara escalas de razão, não faz comparação por pares
Mudge	Facilidade de uso, médio tempo para execução e fácil entendimento do método, comparação por pares	Não escalável, baseado somente no julgamento humano

Fonte: Elaborado pelo autor

6. Conclusão

Este estudo teve por objetivo identificar os principais métodos de priorização de requisitos, bem como, comparar os mesmos ao método Mudge com base no valor econômico. Para tal, o estudo foi dividido em linhas gerais em 2 grandes etapas, sendo elas: Revisão Bibliográfica e Comparativo.

A Revisão Bibliográfica foi feita a partir de um Mapeamento Sistemático (MS) da literatura, utilizado principalmente para sustentar de forma geral o referencial deste estudo. O MS serviu não só para embasar a parte teórica do estudo, como também auxiliar na identificação dos principais métodos de priorização de requisitos no desenvolvimento de software. Por sua vez, o comparativo serviu para analisar os dois principais métodos de priorização de requisitos de software encontrados na literatura, com base na sua frequência, em relação ao método Mudge.

Desta forma, após a análise destes três métodos pode-se constatar que baseados no valor econômico definido para a comparação, o método de Mudge foi o mais bem avaliado, seguido do Planning Game, e por fim o método AHP. O método de Mudge teve a melhor relação custo/benefício. Já o Planning Game teve um baixo esforço realizado, em contrapartida também não teve muito benefício. Por fim, o método AHP se mostrou o mais benéfico, porém, também o mais custoso, e isso o levou a terceira posição na avaliação.

Com base na literatura e no comparativo, descobriu-se que o AHP mesmo sendo o mais mal colocado, é a abordagem mais promissora. Ele produz os resultados mais confiáveis, baseados em uma escala de proporção, é tolerante a falhas e inclui uma verificação de consistência. A verificação de consistência é muito importante, pois o julgamento humano está longe de ser perfeito. Por outro lado, o AHP pode ser problemático se ampliado para projetos maiores.

As principais dificuldades encontradas neste estudo foram, a inexistente ocorrência na literatura a respeito da integração dos temas deste estudo, que mesmo não sendo primordial, pode haver uma diferença pros estudos encontrados. Outro ponto a se relatar é a falta de comprovação empírica do comparativo. A análise e resultado encontrados foram baseados na revisão bibliográfica.

Com base no exposto, como sugestão para trabalhos futuros pode ser interessante a aplicação empírica dos métodos aqui relatados em casos reais de desperdícios de software, para assim poder comprovar ou refutar os resultados aqui expostos. Outra sugestão é a combinação dos métodos, para verificar se elas seriam mais eficientes e eficazes do que a sua aplicação de forma isolada.

Referências

AASEM, Muhammad; RAMZAN, Muhammad; JAFFAR, Arfan. Analysis and optimization of software requirements prioritization techniques. In: **2010 International Conference on Information and Emerging Technologies**. IEEE, 2010. p. 1-6.

ACHIMUGU, Philip et al. A systematic literature review of software requirements prioritization research. **Information and software technology**, v. 56, n. 6, p. 568-585, 2014.

ALAHYARI, Hiva; GORSCHEK, Tony; SVENSSON, Richard Berntsson. An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations. **Information and Software Technology**, v. 105, p. 78-94, 2019.

AL-BAIK, Osama; MILLER, James. Waste identification and elimination in information technology organizations. **Empirical Software Engineering**, v. 19, n. 6, p. 2019-2061, 2014.

BASSO, J. **Engenharia e análise do valor**. São Paulo, Instituto IMAM, 1991.

BECK, Kent. **Extreme programming explained: embrace change**. addison-wesley professional, 2000.

BEHROOZI, Nazli; KAMANDI, Ali. Waste elimination of agile methodologies in web engineering. In: **2016 Second International Conference on Web Research (ICWR)**. IEEE, 2016. p. 102-107.

DEVULAPALLI, Sita. A Study of Factors and Proposal of new Framework for Requirements Prioritization for Successive Releases of Application Software Products. 2016.
DOMINGUES, Jeferson; SELLITTO, Miguel Afonso; LACERDA, Daniel Pacheco. Análise de valor e engenharia de valor: estudo de caso em serviços. **Revista Base (Administração e Contabilidade) da UNISINOS**, v. 10, n. 4, p. 373-385, 2013.
HUJAINAH, Fadhl et al. Software Requirements Prioritisation: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges. **IEEE Access**, v. 6, p. 71497-71523, 2018.
KARLSSON, Joachim. Software requirements prioritizing. In: **Proceedings of the Second International Conference on Requirements Engineering**. IEEE, 1996. p. 110-116.
KARLSSON, Lena et al. Pair-wise comparisons versus planning game partitioning—experiments on requirements prioritisation techniques. **Empirical Software Engineering**, v. 12, n. 1, p. 3-33, 2007.
KITCHENHAM, Barbara; CHARTERS, Stuart. Guidelines for performing systematic literature reviews in software engineering version 2.3. **Engineering**, v. 45, n. 4ve, p. 1051, 2007.
KITCHENHAM, Barbara et al. Systematic literature reviews in software engineering–a tertiary study. **Information and software technology**, v. 52, n. 8, p. 792-805, 2010.
MILES, Lawrence D. **Techniques of value analysis and engineering**. Miles Value Foundation, 2015.
MUDGE, Arthur E. **Value engineering: a systematic approach**. McGraw-Hill, 1971.
NAKAGAWA, Elisa Yumi et al. **Revisão sistemática da literatura em engenharia de software: Teoria e Prática**. Elsevier Brasil, 2017.
NEWKIRK, James; MARTIN, Robert C. **Extreme programming in practice**. Addison Wesley Longman, 2001.
PESSÔA, Marcus VP et al. Understanding the waste net: a method for waste elimination prioritization in product development. In: **Global Perspective for Competitive Enterprise, Economy and Ecology**. Springer, London, 2009. p. 233-242.
PEREIRA FILHO, Rodolfo Rodrigues. Análise do valor: processo de melhoria contínua. **São Paulo: Nobel**, 1994.
PETERSEN, Kai et al. Systematic mapping studies in software engineering. In: **Ease**. 2008. p. 68-77.
POPPENDIECK, Mary; POPPENDIECK, Tom. **Lean Software Development: An Agile Toolkit: An Agile Toolkit**. Addison-Wesley, 2003.
POWER, Ken; CONBOY, Kieran. Impediments to flow: Rethinking the lean concept of 'waste'in modern software development. In: **International Conference on Agile Software Development**. Springer, Cham, 2014. p. 203-217.
RIES, Eric. **A startup enxuta**. Leya, 2012.
SAATY, Thomas L. Decision making with the analytic hierarchy process. **International journal of services sciences**, v. 1, n. 1, p. 83-98, 2008.
SAATY, Thomas L. Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process. **RACSAM-Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales. Serie A. Matematicas**, v. 102, n. 2, p. 251-318, 2008.
SUFIAN, Muhammad et al. A Systematic Literature Review: Software Requirements Prioritization Techniques. In: **2018 International Conference on Frontiers of Information Technology (FIT)**. IEEE, 2018. p. 35-40.
VARGAS, Ricardo. utilizando a programação multicritério (ahp) para selecionar e priorizar projetos na gestão de portfólio. In: **PMI Global Congress**. 2010.