Uso do Algoritmo PageRank para Priorização de Requisitos de Software

Mirna Paula Silva[1], **Fabio Tirelo**[2], **Humberto Torres Marques Neto**[1]

[1]Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, MG – Brasil

[2]Google Inc.
Belo Horizonte, MG – Brasil

mirna.silva@sga.pucminas.br, ftirelo@google.com, humberto@pucminas.br

Abstract. *The presented approach aims to reduce the enforcement spent by stakeholders during the priorization process. Therefore, this paper studied the application of the PageRank algorithm on the requirement priorization process. An analysis is presented and comparisons were performed between the priorization approaches Cumulative Voting, PERT/CPM and PageRank. After comparing the experiments results, it was observed that PageRank allows stakeholders to performs a less effort priorization and adjustments in their priorities, unlike the other two compared techniques.*

Resumo. *Este trabalho apresenta uma nova abordagem para reduzir o empirismo existente na priorização de requisitos, aplicando o algoritmo PageRank em um grafo representativo das dependências entre requisitos. Uma análise é apresentada e comparações são realizadas entre as abordagens de priorização por votação acumulativa, com o uso de PERT/CPM e por meio do PageRank, ou seja, a técnica proposta neste trabalho. Após comparar os resultados, foi possível observar que o PageRank permite que os stakeholders despendam menos esforços para efetua a priorização e que realizem ajustes nas prioridades dos requisitos, diferentemente das outras duas técnicas.*

1. Introdução

Segundo Sommerville (2007), a Engenharia de Requisitos é uma subárea da Engenharia de Software responsável pela criação e gerência de requisitos que irão compor um sistema. É comum que sistemas possuam um grande número de requisitos e como forma de organizar e planejar o projeto as empresas façam o uso de um processo chamado priorização de requisitos.

Para iniciar esse processo, Karlsson, Wohlin e Regnell (1998) aconselham a definir o fator de relevância que priorizará os requisitos. Em seguida, é necessário encontrar a abordagem de priorização que mais se enquadre no sistema. Embora existam diversas abordagens, todas elas dependem da ação dos *stakeholders*, que definem de forma experimental e subjetiva a prioridade de cada requisito. A opinião de cada *stakeholder* pode ser influenciada de maneira não consciente por diversos fatores.

Dentre esses fatores tem-se que o responsável pela priorização pode não conhecer todos os requisitos existentes no sistema. Os profissionais também podem possuir diferentes experiências dados os projetos realizados anteriormente, de modo que alguns *stakeholders* podem não ter um nível adequado de conhecimento sobre a funcionalidade relacionada ao requisito (FIRESMITH, 2004). Dessa forma, as prioridades definidas por esses profissionais serão incompatíveis, pois, enquanto o *stakeholder* mais experiente classifica o requisito de acordo com sua complexidade real, o menos experiente poderá definir valores não relacionados.

Outro fator de grande influência é a definição dos valores da escala de prioridades. Wiegers (1999) destaca a importância de todos os envolvidos conhecerem e concordarem com os níveis da escala que será utilizada. Embora seja imprescindível que todos os *stakeholders* compreendam e concordem com o significado de cada nível da escala, nem sempre isso ocorre. Além desses fatores, Lehtola, Kauppinen e Kujala (2004) citam que a priorização de requisitos deve ser realizada com base em três pontos de vista: o do cliente, o da implementação e o do negócio. Porém, normalmente os responsáveis pelo projeto levam em consideração apenas o ponto de vista do cliente, o que pode resultar em conflitos durante o desenvolvimento e custos não esperados ao longo do projeto.

Dado o contexto acima, este trabalho estuda uma nova abordagem de priorização de requisitos de software. Onde a abordagem proposta visa diminuir o esforço despendido pelos *stakeholders* ao reduzir o envolvimento direto dos mesmos durante o processo. Assim, por meio do algoritmo *PageRank*, a priorização é obtida com base nas dependências reais entre requisitos e a subjetividade dos *stakeholders* possa ser um fator opcional.

Este texto está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados ao tema abordado. A Seção 3 descreve a metodologia desenvolvida para este trabalho. Na Seção 4 é apresentada uma análise, onde a abordagem de priorização desenvolvida é comparada com duas outras abordagens mencionadas na Seção 2. Finalmente, na Seção 5 está a conclusão deste trabalho e os possíveis trabalhos futuros.

2. Revisão da Literatura

A revisão da literatura está dividida da seguinte forma: a Seção 2.1 menciona conceitos sobre engenharia de requisitos; na Seção 2.2 constam abordagens de priorização de requisitos; na Seção 2.3 contém um estudo sobre o *PageRank*, algoritmo fundamental para o método proposto por este trabalho; e finalmente, na Seção 2.4 são abordados alguns trabalhos que também não estão relacionados com a priorização de páginas na *web*, mas que utilizaram o PageRank para alcancar seus objetivos.

2.1. Engenharia de Requisitos

Segundo Nuseibeh e Easterbrook (2000), o sucesso de um software depende do quão adequado ele está às necessidades do ambiente e de seus usuários. Cheng e Atlee (2007) definem que os requisitos do software ilustram essas necessidades e a engenharia de requisitos se responsabiliza pela sua criação e manutenção.

Sommerville (2007) conceitua a engenharia de requisitos como um processo que envolve as seguintes atividades: estudo da viabilidade do sistema, levantamento e análise, validação e gerência da documentação dos requisitos. Durante a atividade de gerência, os requisitos do sistema podem ser relacionados entre si e as prioridades de cada um deles

podem ser definidas. Essas duas tarefas são chamadas, respectiamente, de rastreabilidade de requisitos e priorização de requisitos.

A rastreabilidade de requisitos é descrita por Gotel e Finkelstein (1994) como uma forma de relatar e acompanhar um requisito em qualquer fase, desde suas origens até o seu desenvolvimento. Desse modo, é possível encontrar as relações de dependência entre todos os requisitos de software do sistema.

A priorização de requisitos é classificada por Karlsson (2002) como parte essencial da análise que todo engenheiro de requisitos deve executar. De acordo com a abordagem de priorização usada, os requisitos do sistema recebem níveis de prioridade que podem auxiliar os *stakeholders* ao longo do planejamento do projeto. A atividade de priorização é melhor descrita na seção abaixo.

2.2. Priorização de Requisitos

Wiegers (1999) define que a escolha dos requisitos que irão compor determinada etapa do projeto depende do quão bem sucedida foi realizada a priorização. Dessa forma, para realizar a priorização dos requisitos de um sistema, inicialmente é necessário escolher a abordagem que mais se adapte ao projeto do sistema. Existem diversas formas de priorizar e de acordo com Firesmith (2004) e Lehtola, Kauppinen e Kujala (2004), elas trabalham de maneira subjetiva e dependente dos indivíduos responsáveis pelo sistema. Segundo os autores, a subjetividade é uma consequência da divergência sobre como a priorização deve ser aplicada, quais são os fatores de relevância que garantam a prioridade, dentre outros aspectos.

Durante a priorização, existem algumas tarefas que auxiliam na execução desse processo. A comparação em pares é uma dessas atividades. Ela compõe a etapa inicial de muitas técnicas de priorização. Essa atividade consiste em formar pares com todos os requisitos existentes no sistema e em seguida comparar se um requisito tem ou não maior prioridade sobre seu par. Essa tarefa deve ser realizada com todos os pares de requisitos existentes e devido a isso, Karlsson et al. (2007) observam que essa comparação exige dos *stakeholders* muito tempo para finalizar a sua execução.

Além de atividades iniciais de comparação e avaliação, existem técnicas que permitem que outros fatores sejam avaliados e que o processo de priorização seja executado de maneira mais abrangente. O jogo do planejamento e a votação acumulada são algumas das técnicas mais conhecidas e dadas as aptações realizadas para adequa-las aos projetos, diversas outras técnicas resultaram dessas modificação (BERANDER; KHAN; LEHTOLA, 2006). Além dessas, tem-se outras técnicas que são muito utilizadas pela indústria, sendo elas a abordagem de custo-benefício, a análise de caso de negócio e o processo baseado em Valor. As técnicas mencionadas serão descritas ao longo desta seção.

Heldman (2004) descreve o jogo do planejamento como parte da metodologia de desenvolvimento ágil Programação eXtrema (XP). Ela consiste em definir o planejamento do projeto de acordo com as necessidades dos clientes e desenvolvedores. O autor descreve que a priorização dos requisitos ocorre na fase de comprometimento, que pode ser resumida em três etapas. Na primeira etapa, clientes classificam os requisitos em três pilhas com os seguintes nomes: *aqueles sem as quais o sistema não funciona*, *aquelas que são menos essenciais mas proporcionam um valor significativo para os negócios*, e *aque-*

les que seriam interessante ter. Na etapa seguinte, desenvolvedores estimam o tempo no qual a equipe irá demorar para implementar cada requisito. Definidas as etapas anteriores, na última etapa, os clientes escolhem o conjunto de requisitos que farão parte da versão atual, observando sua importância no sistema e o tempo estimado de implementação.

Já a técnica de votação acumulativa, também chamada de *100-points method*, é explicada por Leffingwell e Widrig (2003) e Ahl (2005). São distribuídos 100 pontos para cada *stakeholder* do projeto. Em seguida, cada pessoa distribui seus pontos entre os requisitos existentes, respeitando o limite máximo de pontos que um requisito pode receber de cada pessoa (caso exista esse limite). Assim que todos os pontos de todos os *stakeholders* forem distribuídos, um dos envolvidos somará a quantidade de pontos que cada requisito recebeu e aquele que possuir a maior pontuação é o de maior prioridade no sistema.

A abordagem de custo-benefício é descrito por Karlsson e Ryan (2002) como o método no qual os requisitos são priorizados de acordo com seus benefícios e custos relativos. Inicialmente é necessário definir como o benefício e o custo devem ser interpretados. Benefício é o potencial que um requisito candidato tem de contribuir para a satisfação do cliente com o sistema final. O custo está relacionado ao custo de uma implementação bem sucedida do requisito candidato. Para analisar os requisitos candidatos a priorização, eles são divididos em pares, organizados em uma estrutura hierárquica e comparados de acordo com o custo-benefício de cada um deles. É definido como prioritário aquele requisito que possui o melhor custo-benefício.

Boehm (2003) define que a análise de caso de negócio integra o processo de priorização baseado em valores. Esse método é o responsável por determinar quais requisitos resultarão no melhor retorno de investimento, ajudando os *stakeholders* a priorizar e reconciliar as suas propostas de valores. Outro ponto importante é que dado os valores investidos nos requisitos, os *stakeholders* se encontram capazes de determinar se vale a pena gastar muito tempo durante o desenvolvimento de certos requisitos.

O processo baseado em valor auxilia a engenharia de software provendo novas perspectivas, ferramentas, habilidades e critérios de sucesso para as suas atividades envolvidas (BOEHM, 2003). Esse processo foi construído sobre sete elementos-chave, sendo eles: análise das realizações benéficas, reconciliação e elicitação da proposta de valor do *stakeholder*, análise de caso de negócio, gerenciamento de oportunidades e riscos contínuos, engenharia de software e sistemas distribuídos, controle e monitoramento baseado em valores, e mudança como uma oportunidade. Esses elementos compõem o *framework* de aplicação do processo baseado em valor, onde é analisado e levado em consideração todos os valores e custos reais do sistema, ao invés de observar apenas um parâmetro do sistema.

2.2.1. Considerações Sobre as Abordagens de Priorização

Existem muitas maneiras de realizar a priorização dos requisitos de um sistema. No entanto, de acordo com Berander, Khan e Lehtola (2006), não existem muitas evidências sobre qual é o perfil de projeto ideal para ser aplicado em cada uma das abordagens de priorização. Outro ponto citado pelos autores, o resultado de uma priorização tende a

mudar conforme o modo no qual é executado, devido à subjetividade existente.

Firesmith (2004) e Lehtola, Kauppinen e Kujala (2004) listam fatores que sugerem que a subjetividade inserida na priorização é consequência da ação direta dos *stakeholders*. Dessa forma, para obter resultados mais próximos durante a execução de uma mesma abordagem, levanta-se a hipótese de reduzir a influência direta dos *stakeholders*. Porém, como será feita a priorização sem solicitar o envolvimento direto dos responsáveis pelo projeto?

A abordagem apresentada neste trabalho visa reduzir o envolvimento direto dos *stakeholders* ao modelar os requisitos do sistema em um grafo de requisitos, e aplicar o algoritmo *PageRank* para se executar a priorização. Segundo Page et al. (1999), o algoritmo *PageRank* é um método de classificação baseado no grafo estrutural da *web*, conhecido por medir a importância relativa de cada página da internet, de acordo com os sites que as referenciam. Por priorizar elementos estruturados na forma de grafo, o algoritmo *PageRank* se mostra ideal para os propósitos deste trabalho, permitindo que a subjetividade resultante da ação direta dos *stakeholders* seja um fator opcional ao longo do processo.

2.3. *PageRank*

Diferentemente de outras coleções de documentos, Page et al. (1999) definem que a *World Wide Web* contém documentos que podem estar na forma de hipertextos e prover uma quantidade considerável de informações auxiliares no topo dos textos das páginas *web*, como por exemplo, *links* estruturais e textuais.

Com o crescimento da *World Wide Web*, diversos mecanismos de busca *web* foram desenvolvidos, e um dos problemas que eles enfrentam é conseguir classificar os resultados retornados de acordo com o interesse do usuário que realizou a busca. O que torna esse problema desafiador é o fato de que apenas alguns desses resultados serão inicialmente apresentados ao usuário. Por isso, eles precisam estar priorizados antes de serem exibidos. Uma maneira de medir a importância relativa de cada página *web* é por meio do algoritmo *PageRank* desenvolvido por Page et al. (1999). Esse algoritmo é um método de classificação baseado no grafo estrutural da *web*, no qual foi inserido pelos autores na ferramenta de busca Google para auxiliar no aprimoramento das pesquisas retornadas.

2.3.1. Entendendo o *PageRank*

A *web* é composta por páginas e *links* que fazem referências a outras páginas. Uma página pode possuir diversos *links*, nos quais são classificados em *links* de avanço e *links* de retorno. Um *link* de avanço é uma referência feita a outra página, e um *link* de retorno é uma referência recebida, originada em outra página (PAGE et al., 1999).

As páginas *web* variam quanto ao número de *links* de retorno que elas possuem. O *PageRank* utiliza-se dessa topologia como um indicador de pontuação a ser dado a qualquer página. Ou seja, uma página recebe a sua pontuação de acordo com a quantidade de *links* de retorno que ela possui, independentemente do seu contexto. Um exemplo semelhante é dado por Bianchini, Gori e Scarselli (2005), onde os índices de citações usados em publicações científicas, onde quanto maior o número de citações, maior a importância da obra.

O *PageRank* define que uma página terá uma alta classificação se a soma das classificações de seus *links* de retorno for alta. Com isso, uma página bem classificada ou possui muitos *links* de retorno de classificação baixa/média, ou possui poucos e bem classificados. A Figura 1 representa de maneira simplificada a propagação da classificação realizada pelo *PageRank*.

Logo que a página obtém a sua classificação, esse valor é dividido entre os seus *links* de avanço, que apontarão para outras páginas. Esse algoritmo é recursivo e de acordo com Page et al. (1999), pode ser iniciado a partir de qualquer conjunto de páginas.

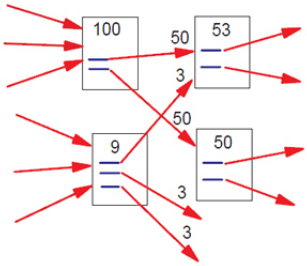

Figura 1. Cálculo simplificado do *PageRank* (PAGE et al., 1999)

2.3.2. Definição do Algoritmo PageRank

Conforme descrito no trabalho de Page et al. (1999), considere o algoritmo abaixo:

$$PR(P_1) = (1-d) + d \times \sum_{i=1}^{n} \left(\frac{PR(P_i)}{N(P_i)} + ... + \frac{PR(P_n)}{N(P_n)} \right)$$

Onde $PR(P_1)$ é o *PageRank* da página P_1. $PR(P_i)$ é o *PageRank* da página P_i que aponta para a página P_1. $N(P_i)$ é o número de *links* que saem da página P_i. Seja d o *damping factor* no qual pode assumir valores entre 0 e 1. O *damping factor* d representa a probabilidade do usuário que está clicando em *links* aleatoriamente, interromper a sua ação e ir para outra página *web*. Estudos realizados por Page et al. (1999) testaram diferentes valores para o *damping factor* e é definido como padrão o valor 0,85. Esse algoritmo trabalha recursivamente e em cada iteração todos os *ranks* são computados. O *PageRank* se encerra quando a diferença entre o *rank* atual e o *rank* anterior, ou seja, o fator de convergência, for um valor pouco significativo.

2.4. Aplicações do *PageRank* Fora do Contexto Inserido na Internet

Além de sua aplicação nas ferramentas de busca para priorizar páginas *web*, os autores Jin et al. (2009), Li e Yi (2009) encontraram maneiras de utilizá-lo fora do contexto inserido na *web*. E embora estejam inseridas em contextos distintos, essas aplicações mantêm o foco principal do *PageRank* que é a priorização a partir dos relacionamentos entre os elementos a serem priorizados.

O primeiro trabalho a ser citado é o dos autores Jin et al. (2009). A gerência de requisitos é um fator determinante para se estipular o planejamento de um projeto e os

custos do mesmo antes de se iniciarem as implementações. Assim, esse trabalho foca na análise de preocupações e impactos causados pelos requisitos, de modo que é apresentado uma abordagem na qual prioriza essas análises por meio do *PageRank*. Para que isso fosse possível, inicialmente foram definidas as preocupações na medida em que os requisitos ganhavam suas especificações. Em seguida, a partir das preocupações já listadas, foi necessário adequá-las a estrutura de um grafo, para só então aplicar o *PageRank* e obter a priorização dessas preocupações.

Outro trabalho que foge do escopo da *web*, é o dos autores Li e Yi (2009). Nele foi medido a complexidade dos relacionamentos entre elementos de um sistema por meio do algoritmo *PageRank*. Onde por meio do nível de complexidade existente entre certos elementos, é possível analisar e encontrar vulnerabilidades do sistema em etapas precoces do desenvolvimento. Nesse trabalho os autores optaram por utilizar matrizes. Essas matrizes correspondem aos relacionamentos de dependência entre as classes, que foram extraídos a partir do diagrama de classes. Tendo construído as matrizes, o *PageRank* foi aplicado e os relacionamentos foram priorizados como o esperado.

A partir da análise desses trabalhos, é possível afirmar a solidez desse processo de priorização. Assim, diante dessas abordagens, surgiu a hipótese de priorizar os requisitos de um sistema por meio do algoritmo *PageRank*.

3. Metodologia

O objetivo deste trabalho é propor uma abordagem que visa reduzir os esforços realizados pelos *stakeholders* ao diminuir o seu envolvimento direto durante a execução do processo de priorização. Com isso, a priorização é feita levando em consideração as dependências entre requisitos e a subjetividade dos *stakeholders* possa ser um fator opcional do processo. Para atingir esse objetivo, foi proposto um grafo de requisitos que permita a aplicação direta da abordagem proposta.

3.1. Definição do Grafo de Requisitos

Esta seção descreve o processo de construção do grafo de requisitos, com base nas dependências existentes entre eles. A análise realizada para construir esse grafo é a mesma utilizada na construção de matrizes de rastreabilidade de requisitos. A representação dos relacionamentos em uma matriz de rastreabilidade é realizada da seguinte forma: requisitos que se relacionam devem receber o valor 1 nas suas respectivas linhas e colunas. Se não há nenhum tipo de relacionamento entre eles, então o valor 0 deve ser atribuído.

Matriz de Rastreabilidade			
	A	B	C
A	0	1	1
B	1	0	1
C	0	1	0

Figura 2. Exemplo de matriz de rastreabilidade

A Figura 2 é um exemplo de matriz de rastreabilidade. Esse exemplo mostra as relações de dependência entre os requisitos A, B e C de um sistema. O requisito A é dependente de B e C, e para representar esse relacionamento na matriz, foi atribuído o valor 1 na linha A e colunas B e C. A mesma análise é feita com B e o valor 1 é atribuído

na linha B, colunas A e C. Finalmente, o requisito C depende apenas de B. Assim, é atribuído o valor 1 na linha C, coluna B. Nas células onde não há nenhum tipo de relacionamento entre requisitos, o valor 0 é atribuído. É possível ver essa atribuição entre C e A, na linha C e coluna A.

Na estrutura de grafo definida neste trabalho, cada requisito é representado por um vértice e suas relações são representadas por arestas. Para reproduzir corretamente os relacionamentos existentes entre os requisitos de um sistema, se um requisito depende de outro, então uma aresta irá originar nele e apontará para sua dependência. Neste trabalho, essa representação de aresta será chamada de *link* de avanço. Na Figura 3, é possível visualizar o grafo resultante da matriz de rastreabilidade da Figura 2.

Esse grafo mostra todos os requisitos representados como vértices e suas relações. Considerando que A depende de C, a relação entre eles é obtida por meio de um *link* que se origina em A e aponta para C. Em caso de requisitos que possuem dependência mútua, como por exemplo, A e B, ou B e C, esse relacionamento será representado por um *link* em A que aponta para B, e outro em B que aponta para A. O mesmo será feito na relação mútua entre B e C.

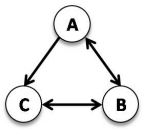

Figura 3. Grafo de requisitos construído com base no exemplo da Figura 2.

3.2. Aplicação do *PageRank* no Grafo de Requisitos

Conforme mencionado na Seção 2.3, o *PageRank* é o algoritmo que calcula o *rank* das páginas *web*, onde uma de suas métricas é a quantidade de *links* que apontam para cada uma delas. A computação desse algoritmo só se torna possível ao considerar a *web* uma estrutura similar a um grafo. Dessa forma, as páginas são os vértices e os links são as arestas.

Para obter todos os requisitos priorizados, é necessário executar o algoritmo *PageRank*. De modo análogo à estrutura da *web*, na qual considera vértices e arestas, torna-se possível aplicar o *PageRank* no grafo de requisitos descrito na seção anterior. Assim, o algoritmo recebe como dado de entrada um grafo e calcula todos os *ranks* de acordo com a estrutura desse grafo. O *rank* de prioridades é definido com base nos relacionamentos que cada requisito possui.

Após algumas iterações, o algoritmo se encerra e exibe como resultado o *rank* final de todos os requisitos do sistema. A execução só pode ser encerrada quando a diferença entre o *rank* atual e o anterior, ou seja, o fator de convergência, for um valor pouco significativo. Esse valor pouco significativo é o limite de convergência do algoritmo e foi definido com o valor épsilon. Desse modo, se o fator de convergência for inferior ao limite de convergência, então o *PageRank* finalmente atingiu a sua iteração final e não haverá mais *ranks* a serem calculados.

3.3. Ajustando *Ranks* com Vértices Artificiais

Para que seja possível inserir no grafo de requisitos informações externas que influenciem na priorização, será introduzido neste contexto o conceito de vértices artificiais. Os vértices artificiais são os responsáveis por influenciar na prioridade de determinado requisito com base nos pesos e fatores de priorização estipulados para ele. Por exemplo, se foi estipulado que um requisito deve receber uma prioridade extra devido ao seu custo, então essa prioridade extra será representada no grafo de requisitos por um vértice artificial.

No entanto, a inserção de vértices artificiais é uma atividade opcional. Ela proporciona para os *stakeholders* a possibilidade de também considerem a prioridade técnica e a prioridade de negócio dos requisitos durante o processo de priorização.

Para realizar essa atividade, é necessário definir quais são os requisitos do sistema que receberão esse ajuste. Ajustar uma prioridade consiste em inserir um vértice artificial no grafo, visando aumentar o *rank* do requisito selecionado, sem que seus relacionamentos de dependência sejam desestruturados. Dessa forma, os *ranks* de suas dependências também deverão herdar a prioridade extra que lhe foi atribuída por meio do vértice artificial.

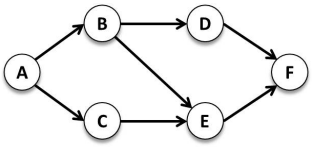

Figura 4. Exemplo de um grafo em que requisitos recebem prioridades iguais.

Considerando o grafo da Figura 4, é possível observar que os *ranks* dos vértices B e C serão iguais, uma vez que ambos recebem a mesma quantidade de *links*. Assim, para evitar que empates desse tipo prejudiquem o desenvolvimento do sistema, é definido de acordo com algum fator externo, que o requisito B deverá receber prioridade extra. A Figura 5.A ilustra a inserção de um vértice artificial VA no grafo para somar uma prioridade extra ao *rank* do requisito B.

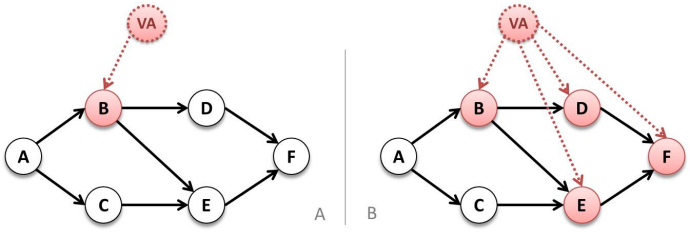

Figura 5. A: Ajustando o *rank* de B inserindo um vértice artificial. B: Ajustando o *rank* do fecho transitivo de B.

Para manter a coerência dos relacionamentos dos vértices do grafo, é necessário que as dependências de B também tenham suas prioridades ajustadas. Desse modo, para encontrar as dependências de B, é necessário definir o seu fecho transitivo. Considerando o grafo $G = (V, E)$, seu fecho transitivo é o grafo $G^* = (V, E^*)$, no qual E^* contém *links* entre (u, v), se e somente se, G possui um caminho (ou pelo menos um *link*) de u para v. Portanto, para aumentar a prioridade desses requisitos é necessário que sejam criados

links artificiais, que conectem o vértice artificial aos vértices do fecho transitivo. A Figura 5.B ilustra o ajuste dos *ranks* dos vértices D, E e F que pertencem ao fecho transitivo de B, por meio da inserção de *links* artificiais que os conectem com o vértice artificial VA.

Dessa forma, considerando o mesmo grafo $G = (V, E)$, é possível definir que o seu conjunto de vértices é $V(G) = R \cup A$, onde R representa todos os requisitos e A representa os vértices artificiais inseridos no grafo. O conjunto de *links* existentes no grafo pode ser definido como $E(G) = X \cup Y \cup Z$. A relação de dependência entre vértices de requisitos é descrita por $X = \{(r_1, r_2) \in R^2 | r_1 \; depende \; de \; r_2\}$, onde r_1 depende de r_2 e possui *link* de avanço representando esse relacionamento. $Y = R \times A$ é a relação entre todos os requisitos a serem ajustados e seus respectivos vértices artificiais. Finalmente, a relação entre requisitos do fecho transitivo e vértices artificiais é definida por $Z = \{(a, r) \in A \times R | \exists r' \in R : \; r' \in \Gamma^+(r)\}$, no qual, se existe *link* artificial do vértice artificial a para o requisito r, então possivelmente existe um requisito r' em R que pertence ao fecho transitivo de r. Dessa forma, também há *link* artificial de a para r'.

É importante mencionar que todos os vértices e *links* artificiais inseridos fazem parte da estrutura do grafo, portanto eles também participam do cálculo do *PageRank*. Porém, por serem estruturas artificiais, eles não compõe o *rank* final. Uma vez que todos os vértices artificiais foram inseridos e todos os *links* definidos no grafo, o cálculo do *PageRank* deve ser executado para obter o *rank* final de prioridades.

4. Estudo sobre um Sistema de *Petshop* Online

Esta seção apresenta um estudo, na qual foram realizados três análises de priorização de requisitos: o PERT/CPM, a votação acumulativa e o *PageRank*. Essas análises foram executados em um sistema de *petshop* online, e seus resultados foram apresentados e comparados, visando analisar a abordagem de priorização proposta neste trabalho.

A análise da abordagem proposta consiste em três etapas. A primeira está relacionada com a análise de posições inválidas existentes no resultados das priorizações. O termo posição inválida se refere a impossibilidade de se implementar um determinado requisito naquele momento, uma vez que suas dependências ainda não foram implementadas. Muitas vezes essas situações não são impeditivas para a continuidade do processo, porém tendem a causar transtornos, já que alternativas devem ser desenvolvidas para solucionar o problema. A segunda etapa, consiste em verificar se as técnicas resolvem interdependência entre requisitos, ou seja, requisitos mutuamente dependentes. Finalmente, a terceira etapa consiste em verificar se as técnicas permitem que as prioridades dos requisitos sejam ajustadas e analisar o nível de dificuldade de se executar essa tarefa.

4.1. O Sistema de *Petshop* Online

O *petshop* online é um sistema desenvolvido e disponibilizado pela IBM Corporation, podendo ser encontrado em IBM (2003). A documentação disponível para o *petshop* consiste nos seguintes artefatos: modelo de negócio, especificação de requisitos, especificação de interface gráfica, plano de testes, código-fonte e arquivos da aplicação compilados.

Para os propósitos deste trabalho, apenas o documento de especificação de requisitos foi considerado e dele foram extraídos 28 requisitos funcionais não priorizados.

Esses requisitos identificados descrevem ações dos seguintes casos de uso: pesquisar produtos, comprar animais e manter conta de usuário. A Figura 6 mostra a lista dos requisitos extraídos.

Petshop Online - Requisitos Funcionais	
ID do Requisito	**Descrição do Requisito**
R01	Exibir controle de paginação em resultados de busca.
R02	Formatação padronizada das páginas
R03	Exibir sinalização (*) de campos obrigatórios
R04	Navegar no menu principal (tela inicial): Categorias, Pesquisar, Carrinho de Compras, Login, Ajuda
R05	Efetuar Login
R06	Editar dados da conta (Usuário logado)
R07	Exibir produtos (raça)
R08	Adicionar produtos ao carrinho de compras
R09	Exibir itens (animal)
R10	Adicionar itens ao carrinho de compras
R11	Visualizar carrinho de compras
R12	Remover itens e/ou produtos do carrinho
R13	Fechar o carrinho e iniciar o pagamento
R14	Exibir campos sobre "dados do cliente"
R15	Exibir campos sobre "dados do cliente" preenchidos se usuário estiver logado
R16	Exibir campos sobre "endereço de entrega"
R17	Solicitar confirmação dos "dados do cliente" e "endereço de entrega"
R18	Exibir resumo da compra e dos dados (cliente e endereço de entrega). Efetuar Compra.
R19	Realizar cadastro de novos usuários
R20	Exibir sistema em Português, Inglês e Espanhol
R21	Funcionalidade "Minha Lista"
R22	Exibir "Minha Lista" na tela carrinho de compras
R23	Exibir banner de dicas
R24	Exibir tela de ajuda
R25	Realizar pesquisa
R26	Exibir resultado da pesquisa
R27	Gerenciar categorias e produtos
R28	Gerenciar idiomas, cartões de crédito e banner de dicas.

Figura 6. Lista dos requisitos extraídos da documentação do *petshop*.

De acordo com os casos de uso mencionados, ao utilizar a aplicação *petshop* online, um usuário pode comprar animais. Para realizar essa ação, é necessário estar autenticado no sistema e efetuar a busca pelo animal desejado. Também é permitido que novos usuários se cadastrem no sistema e usuários já cadastrados atualizem seus dados. A Figura 7 mostra os requisitos e suas respectivas dependências.

4.2. Priorização de Requisitos com PERT/CPM

A primeira análise realizada consiste em priorizar os 28 requisitos identificados, utilizando a abordagem PERT/CPM. Segundo a descrição de Kerzner (2009), a rede PERT/CPM é a representação gráfica de uma sequência lógica do planejamento, na qual são exibidas as dependências das atividades do projeto e o tempo estimado para que cada uma delas seja concluída. Essa representação gráfica é realizada por meio de um grafo que consiste em nós ou blocos, e arestas ou linhas orientadas. Nesse grafo, os nós representam as atividades e as arestas representam as relações de dependência entre essas atividades. O tempo estimado de cada atividade é representado com um número localizado em cada uma de suas arestas. De acordo com essa estrutura de grafo, se uma atividade A aponta

Lista de Dependências dos Requisitos	
Requisitos	**Dependências**
R1	R7, R9, R26
R2	R4, R7, R9, R11, R18, R20, R22, R24
R3	R5, R6, R14, R15, R16, R19
R4	R5, R7, R9, R11, R19, R24, R25, R27
R5	R19
R6	R5, R19
R7	R25, R27
R8	R7, R26, R27
R9	R25, R27
R10	R9, R26, R27
R11	R8, R10, R12, R27
R12	R8, R10, R27
R13	R11, R27
R14	R13, R19, R28
R15	R5, R13, R19, R28
R16	R14, R15
R17	R14, R15, R16
R18	R11, R14, R15, R16, R17, R27, R28
R19	------
R20	R28
R21	R5, R18, R27
R22	R5, R11, R18, R19, R21, R27
R23	R5, R19, R28
R24	R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R18, R19, R20, R22, R23, R25, R26, R27, R28
R25	R27
R26	R7, R9, R25, R27
R27	------
R28	------

Figura 7. Tabela com a matriz de rastreabilidade dos requisitos do sistema.

para uma atividade *B*, significa que *B* é a próxima atividade a ser executada após a conclusão de *A* (Figura 8).

Para adaptar a representação do PERT/CPM ao escopo deste trabalho, será considerado que na composição do grafo, os nós representarão os requisitos do sistema e as arestas representarão as relações de dependência entre eles. O PERT/CPM foi escolhido para integrar esse teste por dois motivos. O primeiro, é pelo fato dessa ser uma técnica antiga, e ainda assim, utilizada pelas empresas. O segundo motivo, é pela semelhança estrutural dessa técnica com a abordagem proposta na metodologia deste trabalho (Seção 3), na qual também considera um grafo, onde os vértices são conectados de acordo com suas relações de dependência.

Portanto, para realizar esse estudo foi construído o grafo de dependências entre requisitos e estimada a mesma quantidade de tempo para todas as arestas existentes. Após

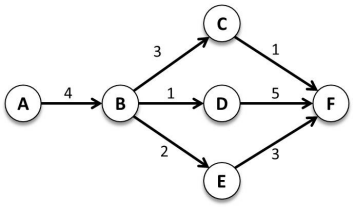

Figura 8. Rede do PERT/CPM e as estimativas de tempo de cada atividade.

estruturar o grafo, é possível observar que ele é um grafo acíclico, permitindo aplicá-lo uma ordenação topológica. A ordenação topológica é muito utilizada para obter uma sequência linear válida dos vértices, onde, nessa análise, significa obter a ordem em que os requisitos devem ser implementados. Essa ordenação é o resultado da priorização. A Figura 9 mostra a priorização do PERT/CPM em uma sequência linear válida, resultante da ordenação topológica.

PERT/CPM: Tabela de Resultados

	Rank	ID do Requisito
Prioridade Alta	1º	R27
	2º	R19
	3º	R25
	4º	R28
	5º	R05
	6º	R09
	7º	R07
	8º	R26
	9º	R10
	10º	R08
	11º	R12
	12º	R11
	13º	R13
	14º	R14
	15º	R15
	16º	R16
	17º	R17
	18º	R18
	19º	R21
	20º	R22
	21º	R06
	22º	R20
	23º	R04
	24º	R23
	25º	R24
	26º	R03
	27º	R01
Prioridade Baixa	28º	R02

Figura 9. Tabela com a priorização resultante do PERT/CPM.

A ordenação topológica apresenta mais de uma sequência linear válida. Dessa forma, é necessário definir qual o critério de seleção para o resultado final. Caso um requisito tenha recebido uma prioridade mais elevada do que suas dependências, ele se encontra em posição inválida. O termo posição inválida se refere a impossibilidade de se implementar um determinado requisito naquele momento, uma vez que suas dependências ainda não foram implementadas. A existência dessas posições inválidas na priorização

final reduz a consistência do resultado. Portanto, a ocorrência de posições inválidas é o critério de seleção escolhido para definir qual será a sequência linear válida da ordenação topológica. Assim, a sequência que possuir a menor quantidade dessas ocorrências é a que melhor se adequa a priorização final.

Ao observar o resultado final desse estudo na Figura 9, tem-se que o requisito $R27$ é o mais prioritário e está situado na primeira posição do *rank*. O $R27$ é o requisito responsável pelo gerenciamento de animais e produtos na aplicação *petshop* online, e por isso, vários outros requisitos dependem dele. O requisito com a segunda maior prioridade é o $R19$, no qual é o responsável pelo cadastramento de novos usuários no sistema e também possui muitos outros requisitos que dependem dele.

O requisito com a menor prioridade é o $R02$. Esse requisito define que todas as páginas devem possuir formatação padronizada de cores e fontes. O $R02$ influencia apenas na interface do sistema e nenhum outro requisito depende dele. Devido a isso, ele se encontra situado na última posição do *rank*. A priorização final obtida por meio do PERT/CPM não apresentou nenhuma posição inválida. Dessa forma, o resultado da priorização do PERT/CPM será considerada base de comparação durante as análises junto aos outros dois experimentos.

4.3. Priorização de Requisitos com Votação Acumulativa

A análise contida nesta seção simula como a priorização é executada nas empresas. Foi solicitado que 23 alunos do curso de graduação em Ciência da Computação se comportassem como *stakeholders* e priorizassem os 28 requisitos encontrados. Apenas alunos matriculados na disciplina de Engenharia de Software I foram selecionados.

Antes de se iniciar a priorização dos requisitos, todos os alunos foram informados que apenas a técnica votação acumulativa deveria ser utilizada. Também foi informado que todas as restrições definidas deveriam ser obrigatoriamente respeitadas ao longo do processo.

Nessa análise, foram distribuídos 100 pontos para cada *stakeholder* e era sua responsabilidade distribuir todos os pontos entre os 28 requisitos. Os pontos deveriam ser distribuídos de acordo com a prioridade do requisito. Assim, o requisito que demanda alta prioridade deveria receber 10 pontos. Do mesmo modo, o que possui prioridade baixa deveria receber 1 ponto. Durante essa distribuição, as seguintes restrições deveriam ser respeitadas:

- Todos os requisitos devem ser pontuados;
- Todos os 100 pontos devem ser distribuídos;
- A pontuação dada a cada requisito deve ser um número inteiro;
- Cada requisito pode receber no mínimo: 1 ponto (baixa prioridade);
- Cada requisito pode receber no máximo: 10 pontos (alta prioridade);
- No máximo 25% dos requisitos (7 requisitos) podem receber 10 pontos;
- No máximo 15% dos requisitos (4 requisitos) podem receber 1 ponto;
- No máximo 25% dos requisitos (7 requisitos) podem ter pontos iguais.

Essas restrições foram definidas com o objetivo de evitar situações em que o processo de priorização pudesse ser invalidado, seguindo a recomendação de Ahl (2005). Por exemplo, todos os requisitos poderiam receber a mesma pontuação, ou apenas alguns

serem pontuados. Outra situação evitada é a existência de requisitos excessivamente pontuados, por exemplo, os 2 primeiros requisitos recebem 50 pontos e o restante 0. Dessa forma, o processo fica invalidado, pois os requisitos que receberam 0 pontos não chegaram a ser avaliados, ele foram simplesmente ignorados.

Finalmente, após todos os *stakeholders* terem concluído esse processo, é necessário agrupar todos os resultados de priorização. Para isso, é essencial somar todos os pontos que cada requisito recebeu. A priorização final é obtida por meio da quantidade de pontos que cada requisito possui (AHL, 2005). Aqueles que possuírem as maiores pontuações, são os mais prioritários. De modo análogo, os requisitos com as pontuações mais baixas são os menos prioritários. A Figura 10 mostra a priorização resultante dessa análise.

Votação Acumulativa: Tabela de Resultados

	Rank	ID do Requisito	Pontos Recebidos
Prioridade Alta	1º	R04	119
	2º	R19	114
	3º	R07	108
	4º	R12	103
	5º	R14	102
	6º	R16	99
	7º	R08	97
	8º	R09	96
	9º	R11	95
	10º	R25	94
	11º	R05	93
	12º	R05	93
	13º	R10	91
	14º	R27	90
	15º	R17	87
	16º	R28	87
	17º	R13	85
	18º	R26	80
Prioridade Baixa	19º	R03	76
	20º	R15	75
	21º	R18	71
	22º	R21	68
	23º	R22	58
	24º	R20	55
	25º	R24	54
	26º	R01	38
	27º	R23	36
	28º	R02	36

Figura 10. Tabela com a priorização resultante da votação acumulativa.

A partir dos resultados obtidos, $R04$ é o requisito prioritário, diferentemente do PERT/CPM que obteve $R27$ nessa posição. Os *stakeholders* desse estudo decidiram atribuir maior prioridade ao requisito responsável pelo desenvolvimento da página principal da aplicação *petshop* online. Essa decisão provavelmente foi tomada considerando o que seria mais importante a partir ponto de vista do cliente. Contudo, em ambos os estudos o requisito $R19$ foi definido como o segundo mais relevante. $R19$ é o responsável por cadastrar novos usuários no sistema e também pode ter sido considerado importante para o cliente. Ao realizar essa análise considerando apenas o ponto de vista do cliente, 14 posições inválidas foram geradas ao longo da priorização. O requisito $R07$ é

o responsável por exibir os produtos cadastrados no sistema e está situado em posição inválida, devido a sua dependência pelo requisito $R27$. O $R27$ realiza o gerenciamento de produtos e deveria ser implementado antes de $R07$ para que essa dependência fosse respeitada.

4.4. Priorização de Requisitos com *PageRank*

O terceiro estudo realizado é a priorização de requisitos utilizando o *PageRank*, de acordo com a abordagem definida na metodologia deste trabalho. Todos os 28 requisitos extraídos da documentação do *petshop* online foram inseridos em um grafo e suas rastreabilidades definidas. Nessa análise será utilizado o resultado original da priorização e nenhum requisito terá a sua prioridade ajustada por meio dos vértices artificiais, uma vez que essa é uma atividade opcional do processo.

Uma vez que o grafo de requisitos foi construído, é possível executar o *PageRank* usando-o como dado de entrada. Foram necessárias apenas 12 iterações antes do limite de convergência ser atingido e o algoritmo parar sua execução. A Figura 11 mostra a pontuação final dos *ranks* e o resultado da priorização.

PageRank: Tabela de Resultados

	Rank	ID do Requisito	Pontuação PageRank
Prioridade Alta ↑	1º	R27	1.675
	2º	R19	0.924
	3º	R25	0.595
	4º	R28	0.562
	5º	R05	0.469
	6º	R09	0.400
	7º	R07	0.400
	8º	R11	0.390
	9º	R26	0.374
	10º	R14	0.370
	11º	R15	0.370
	12º	R13	0.340
	13º	R10	0.308
	14º	R08	0.308
	15º	R16	0.259
	16º	R18	0.247
	17º	R12	0.240
	18º	R24	0.183
	19º	R17	0.180
	20º	R06	0.178
	21º	R21	0.174
	22º	R20	0.173
	23º	R22	0.173
	24º	R04	0.165
	25º	R23	0.157
	26º	R03	0.150
	27º	R01	0.150
↓ Prioridade Baixa	28º	R02	0.150

Figura 11. Tabela com a priorização resultante do *PageRank*.

De acordo com a computação realizada pelo *PageRank*, cada posição do *rank* tem uma pontuação compatível. Devido a grande quantidade de cálculos realizados durantes as iterações, a pontuação final dos *ranks* tende a ser um valor baixo. O resultado da priorização desse estudo é similar aos resultados do PERT/CPM, de modo que ambos

definiram os requisitos $R27$ e $R02$, como o mais e o menos prioritário, respectivamente. Além desses, outros 9 requisitos foram classificados semelhantemente nas duas abordagens.

Após analisar a priorização resultante, foram identificados 4 requisitos em posições inválidas. No entanto, se um requisito que não possui muitos dependentes se encontra em uma posição inválida, é possível corrigir a sua posição aplicando o ajuste de *ranks*. Nessa situação, a correção é feita ao inserir um vértice artificial no grafo, e conectá-lo ao vértice no qual o inválido depende. Desse modo, a dependência que antes tinha *rank* inferior, passará a ter um valor maior que o inválido, corrigindo essa invalidez.

4.5. Comparação Entre as Análises de Priorização

Para demonstrar a eficácia da priorização por meio do *PageRank*, foi realizada uma comparação entre as análises executadas nesse estudo. Em cada análise foram priorizados os mesmos 28 requisitos extraídos da documentação do *petshop* online e os resultados obtidos não foram os mesmos. No entanto, como o *PageRank* e o PERT/CPM trabalham com as relações de dependência entre requisitos, é mais provável que eles tenham resultados semelhantes.

Quando uma abordagem priorização chega ao fim de sua execução, é importante verificar a consistência de seu resultado. Considera-se consistente o resultado que permite que todos os requisitos sejam implementados na sequência apresentada pela priorização, sem haver conflitos de dependência. Quando esses conflitos são encontrados, eles são chamados de posições inválidas. O termo posição inválida se refere a impossibilidade de se implementar um determinado requisito naquele momento, uma vez que suas dependências ainda não foram implementadas. Muitas vezes essas situações não são impeditivas para a continuidade do processo, porém tendem a causar transtornos, já que alternativas devem ser desenvolvidas para solucionar o problema.

Todas as análises foram verificadas e apenas o PERT/CPM não apresentou posições inválidas. A Figura 12 apresenta uma tabela com informações relevantes sobre todos os três estudos realizados.

Comparação entre os experimentos realizados			
	PERT/CPM	**Votação Acumulativa**	**PageRank**
Posições inválidas encontradas:	0	14	4
Resolve interdependência entre requisitos?	Não	Não	Sim
Permite ajuste no *rank* dos requisitos?	Sim. Manual	Não	Sim. Automático

Figura 12. Tabela resumida com informações relevantes sobre a priorização do PERT/CPM, do *PageRank* e da Votação Acumulativa.

Como mostra a Figura 12, na técnica votação acumulativa foram encontradas 14 posições inválidas, no PERT/CPM tais posições não foram encontradas e no *PageRank* verificou-se a existência de 4 delas. A votação acumulativa é subjetivamente baseada apenas ações e decisões dos *stakeholders*. Desse modo, espera-se que essa técnica resulte em

um alto número de posições inválidas. Ambas as abordagens, PERT/CPM e *PageRank*, utilizam as relações de dependência existentes entre os requisitos para encontrar suas prioridades. Dessa forma, no *PageRank* essa utilização apenas reduziu o número de posições inválidas, uma vez que a priorização dos requisitos também depende dos cálculos realizados pelo algoritmo. Dentre esses cálculos, temos a variável *damping factor* que representa o comportamento de um usuário navegando na Internet. No entanto, como neste trabalho não temos usuários internautas, é provável que o *damping factor* seja o responsável pela existência de posições inválidas nessa abordagem. Embora o *PageRank* não tenha apresentado o melhor resultado inicialmente, as vantagens dessa abordagem podem ser vistas durante a priorização dos requisitos. Diferentemente das duas outras abordagens, o *PageRank* soluciona durante suas iterações a duplicidade envolvida entre requisitos interdependentes. O mesmo não é resolvido pela votação acumulativa, pois ela depende dos *stakeholders* para tomar a decisão de qual requisito assumirá maior prioridade. Considerando a subjetividade dessa técnica, mesmo que os *stakeholders* tomem uma decisão, essa não pode ser considerada totalmente confiável. No PERT/CPM, interdependências também não são resolvidas, pois como na votação acumulativa, ele também depende da decisão do *stakeholder*.

Uma outra vantagem apresentada pela abordagem do *PageRank* é a possibilidade de se analisar o sistema levando em consideração fatores técnicos e de negócio. Desse modo, os *stakeholders* tem a liberdade de escolher quais requisitos precisam ser ajustados para realizar priorização, sem se preocupar em ter que adaptar a técnica de priorização utilizada, ou estudar uma nova técnica. Essa flexibilização auxilia os *stakeholders* durante a priorização, inserindo os fatores resultantes de análises subjetivas que muitas vezes são necessários para o projeto.

Trabalhar com sistemas que possuem muitos requisitos pode se tornar um transtorno quando se utiliza técnicas como a votação acumulativa e o PERT/CPM, pois ambos são executados manualmente e dependem dos *stakeholders* para tomarem decisões. Considerando um sistema complexo que possui mais de 150 requisitos, priorizá-los por meio de uma abordagem manual não é viável. Desse modo, devido a escalabilidade do *PageRank*, aplicá-lo nesses requisitos não seriam um problema. Uma vez que o algoritmo *PageRank* lida com grafos que representam a *web*, ou seja, possuem muitos vértices, ele consegue lidar com sistemas que possuem muitos requisitos sem ser necessário que os *stakeholders* despendem um esforço extra.

Após comparar todas as análises e apontar as vantagens resultantes da utilização do *PageRank*, é possível observar que entre o PERT/CPM e a votação acumulativa, o *PageRank* se mostrou a abordagem mais favorável para ser realizar a priorização de requisitos. Isso ocorre pois, o *PageRank* reduz o envolvimento direto dos *stakeholders* ao longo do processo, é escalável, oferece um ajuste na prioridade dos requisitos com base na análise de fatores externos, e apresenta resultados priorizados sem depender da subjetividade inserida nas decisões tomadas pelos *stakeholders*.

5. Conclusão e Trabalhos Futuros

Executar a priorização de requisitos de software tem sido considerada uma tarefa muito desafiadora, principalmente, por exigir que os *stakeholders* exerçam ações exaustivas e às vezes subjetivas. Embora existam diversas técnicas de priorização, raramente elas aderem

à todas as necessidades do projeto e de seus respectivos *stakeholders*. Isto pode ocorrer devido a limitações da própria técnica ou até mesmo por questões de preferências pessoais ou organizacionais. A fim de evitar a utilização de técnicas que não satisfazem todas as exigências do projeto e/ou dos seus *stakeholders*, foi desenvolvido neste trabalho uma nova abordagem baseada no uso do algoritmo *PageRank*. Essa abordagem visa reduzir o esforço despendido pelos *stakeholders*, diminuindo o seu envolvimento direto, e ainda permitir que a subjetividade existente na análise dos *stakeholders* ser torne um fator opcional durante a atividade de priorização de requisitos de software.

Após comparar os resultados obtidos nas análises de utilização das técnicas, *PageRank*, PERT/CPM e votação acumulativa, é possível observar que entre o PERT/CPM e a votação acumulativa, o *PageRank* se mostrou a abordagem mais favorável para ser realizar a priorização de requisitos. Isso é ocasionado pela redução de envolvimento direto dos *stakeholders* durante a utilização do *PageRank*. Além disso, a abordagem proposta é escalável, oferece um ajuste na prioridade dos requisitos com base na análise de fatores externos e apresenta resultados priorizados sem depender da subjetividade inserida nas decisões tomadas pelos *stakeholders*.

A principal contribuição deste trabalho é mostrar a viabilidade de se aplicar o algoritmo *PageRank* no contexto de priorização de requisitos de software. Além disto, o principal objetivo alcançado, foi o desenvolvimento de uma abordagem de priorização que não exige uma atuação exaustiva dos *stakeholders* podendo ser adaptada a diferentes tipos de projeto de software.

Como trabalhos futuros temos o aprimoramento da técnica apresentada na metodologia e o estudo de novo valores para o *damping factor*, de modo a obter resultados que não apresentem nenhuma posição inválida. Também integra os trabalhos futuros a aplicação da técnica em um grande sistema, para avaliação dos impactos que um alto número de requisitos poderão causar.

Referências

AHL, V. An experimental comparison of five prioritization methods. *Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden*, 2005. 2005.

BERANDER, P.; KHAN, K. A.; LEHTOLA, L. Towards a research framework on requirements prioritization. *SERPS*, 2006. v. 6, p. 18–19, 2006.

BIANCHINI, M.; GORI, M.; SCARSELLI, F. Inside pagerank. *ACM Transactions on Internet Technology (TOIT)*, 2005. ACM, v. 5, n. 1, p. 92–128, 2005.

BOEHM, B. Value-based software engineering: reinventing. *ACM SIGSOFT Software Engineering Notes*, 2003. ACM, v. 28, n. 2, p. 3, 2003. ISSN 0163-5948.

CHENG, B.; ATLEE, J. Research directions in requirements engineering. In: IEEE. *Future of Software Engineering, 2007. FOSE'07*. Washington, DC: IEEE Computer Society, 2007. p. 285–303.

FIRESMITH, D. Prioritizing requirements. *Journal of Object Technology*, 2004. v. 3, n. 8, p. 35–48, 2004.

GOTEL, O.; FINKELSTEIN, C. An analysis of the requirements traceability problem. In: IEEE. *Requirements Engineering, 1994., Proceedings of the First International Conference on*. Colorado Springs, CO: IEEE, 1994. p. 94–101.

HELDMAN, K. *PMP: project management professional: study guide*. Alameda, CA: Sybex Inc, 2004. ISBN 0782143237.

IBM. Developing pet store using rup and xde. 2003. December 2003. Disponível em: <http://www.ibm.com/developerworks/rational/library/1072.html>.

JIN, Y. et al. Applying pagerank algorithm in requirement concern impact analysis. In: IEEE. *2009 33rd Annual IEEE International Computer Software and Applications Conference*. [S.l.], 2009. p. 361–366.

KARLSSON, J. Software requirements prioritizing. In: IEEE. *Requirements Engineering, 1996., Proceedings of the Second International Conference on*. Colorado Springs, CO, 2002. p. 110–116.

KARLSSON, J.; RYAN, K. A cost-value approach for prioritizing requirements. *Software, IEEE*, 2002. IEEE, v. 14, n. 5, p. 67–74, 2002.

KARLSSON, J.; WOHLIN, C.; REGNELL, B. An evaluation of methods for prioritizing software requirements. *Information and Software Technology*, 1998. Elsevier, v. 39, n. 14-15, p. 939–947, 1998. ISSN 0950-5849.

KARLSSON, L. et al. Pair-wise comparisons versus planning game partitioning experiments on requirements prioritisation techniques. *Empirical Software Engineering*, 2007. Springer, v. 12, n. 1, p. 3–33, 2007. ISSN 1382-3256.

KERZNER, H. *Project management: a systems approach to planning, scheduling, and controlling*. New Jersey, NY: Wiley, 2009.

LEFFINGWELL, D.; WIDRIG, D. *Managing Software Requirements: a use case approach*. New Jersey, NY: Pearson Education, 2003. ISBN 032112247X.

LEHTOLA, L.; KAUPPINEN, M.; KUJALA, S. Requirements prioritization challenges in practice. *Product focused software process improvement*, 2004. Springer, p. 497–508, 2004.

LI, F.; YI, T. Apply pagerank algorithm to measuring relationship's complexity. In: IEEE. *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*. [S.l.], 2009. v. 1, p. 914–917.

NUSEIBEH, B.; EASTERBROOK, S. Requirements engineering: a roadmap. In: ACM. *Proceedings of the Conference on the Future of Software Engineering*. New York, NY, 2000. p. 35–46.

PAGE, L. et al. The pagerank citation ranking: bringing order to the web. 1999. Stanford InfoLab, 1999.

SOMMERVILLE, I. Engenharia de software, 8a edição. *Prentice Hall, São Paulo, Brasil*, 2007. v. 4, p. 47–48, 2007.

WIEGERS, K. First things first: prioritizing requirements. *Software Development*, 1999. v. 7, n. 9, p. 48–53, 1999.