Identificando a qualidade de Requisitos de Sistemas utilizando Redes Bayesianas

Guilherme Vilela, Thiago Schumacher Barcelos, Alexandra A. de Souza.

Pós Graduação Lato Sensu em Gestão de Sistemas de Informação – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Guarulhos.

gvilela1108@hotmail.com, tsbarcelos@ifsp.edu.br, alexandra.souza@ifsp.edu.br

Abstract. *During the life cycle of a software project it is common that problems with the system requirements are identified, causing problems such as: increase in project cost, delay in deliver, rework in codes, among others. In this scenario companies are looking for alternatives to improve this process, such as the adoption of new software development paradigms, but they can not completely eliminate the uncertainty regarding the quality of the requirement. This work seeks to demonstrate how the use of bayesian networks, whose pilot prototype had an accuracy of 72.13%, can facilitate the identification of the quality of the requirements in the initial phase of the project and leads to minimize the impacts on the project budget.*

Resumo. *No ciclo de vida de um projeto de software é comum que sejam identificados problemas com os requisitos do sistema, ocasionado problemas como: aumento de custo do projeto, atraso no prazo de entrega, retrabalho em códigos, dentre outros. Diante desse cenário as empresas buscam alternativas para melhorar este processo, como a adoção de novos paradigmas de desenvolvimento de software, contudo não conseguem eliminar totalmente a incerteza em relação à qualidade do requisito. Este trabalho busca demonstrar como a utilização de redes bayesianas, cujo protótipo piloto teve 72,13% de acurácia, pode facilitar a identificação da qualidade dos requisitos na fase inicial do projeto e leva a minimizar os impactos no orçamento dos projetos.*

1. Introdução

Os requisitos de software podem se constituir como um dos principais fatores de insucesso em projetos de desenvolvimento. Conforme [The Standish Group 2015] muitos dos problemas encontrados nos projetos de desenvolvimento de software que falham, ou seja, que não produzem um sistema de software funcional, se dá por fatores como tamanho do projeto, excesso de requisitos e metas irreais. Esses fatores conjuntamente indicam uma alta complexidade e correspondem por 54% de projetos que falharam de acordo com a pesquisa Chaos Report, conforme apresentado na Tabela 1. Deve-se observar que esse fator de insucesso deve-se ao fato que inúmeras mudanças nos requisitos provocam um impacto no orçamento de um projeto devido ao retrabalho.

Tabela 7: CHAOS REPORT – Entrega por Complexidade de Projeto. Adaptado [The Standish Group 2015].

	Sucesso	Desafiador	Falha
Muito Complexo	15%	57%	28%
Complexo	18%	56%	26%
Médio	28%	54%	18%
Fácil	35%	49%	16%
Muito Fácil	38%	47%	15%

[Blaschek 2002] também afirma que os requisitos nos projetos de software são a principal fonte de problemas, quando não atendem corretamente o solicitado pelo cliente, ou provocando insatisfação devido a atrasos no cronograma, custos que superam as estimativas e retrabalho da equipe de desenvolvimento.

Segundo [Zardetto 2016] embora não exista nenhuma relação entre o sucesso de um projeto de software e a criação de uma documentação abrangente, a elaboração de um documento detalhado de requisitos baseado em informações que não estejam claramente definidas e estabilizadas podem ocasionar nos problemas apontados acima, e por este motivo as metodologias ágeis buscam modelar os requisitos de forma clara e objetiva para que seja fácil compreendê-los para minimizar estes problemas.

A partir da constatação que uma má qualidade na especificação de requisitos pode ter um impacto significativo no sucesso de um projeto, surge a questão de como mitigar este problema da qualidade de sistemas. A aplicação de técnicas de inteligência artificial apresenta um potencial para auxiliar nessa solução. Segundo [Luger 2013], a Inteligência Artificial pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente, ou seja, a capacidade de dispositivos de raciocinar, decidir e solucionar problemas, então, utilizar Inteligência Artificial em conjunto com uma modelagem do conhecimento extraído a partir de integrantes da equipe pode proporcionar uma gestão melhor da qualidade dos requisitos, o que consequentemente poderá melhorar a qualidade geral do software.

Embora existam estudos que ilustram formas de utilizar IA para determinados processos, como procedimentos de diagnósticos médicos [Saheki 2005], sistemas para educação [Ferreira 2018], aplicações de redes neurais [Rahman, et al 2019], entre outros, não foram muito abordados formas de ligar o uso de IA para melhorar a qualidade dos requisitos de um software em um projeto.

Com base no panorama apresentado, este trabalho tem como objetivo propor uma classificação supervisionada da qualidade de requisitos de sistemas ágeis por meio de uma rede bayesiana. É apresentado um experimento com a utilização de 247 requisitos extraídos de cinco projetos de uma empresa do ramo de seguros, onde a acurácia da classificação da qualidade dos requisitos é avaliada. Dessa forma, o trabalho é norteado a responder a seguinte pergunta de pesquisa: “Redes bayesianas podem mitigar os problemas de requisitos em projetos de sistemas?”.

2. Material e Métodos

A estratégia utilizada para este trabalho chegar ao seu objetivo foi dividida em dois momentos.

O primeiro momento foi através de uma revisão bibliográfica, onde foram analisados os principais problemas encontrados em requisitos de sistemas, e formas de utilizar uma rede bayesiana para realizar a classificação da qualidade dos requisitos.

Já o segundo momento foi um experimento onde, por meio de um protótipo de rede bayesiana supervisionado para classificar a qualidade de requisitos, avaliamos alguns requisitos de projetos reais, oriundos de uma equipe do departamento de TI de uma empresa do ramo de seguros situada na cidade de São Paulo, e extraímos os resultados para responder a pergunta de pesquisa.

3. Requisitos de Sistemas

Segundo [Ávila e Spínola 2007] existem diversas definições sobre o que é um requisito:

- Um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar para atingir os seus objetivos;
- As descrições das funções e restrições são os requisitos do sistema;
- Um requisito é uma propriedade que o software deve exibir para resolver algum problema no mundo real;
- Uma condição ou uma capacidade que deve ser alcançada ou estar presente em um sistema para satisfazer um contrato, especificação ou outro documento.

Com base nestas definições, podemos entender que os requisitos são o conjunto de necessidades explicitadas pelo cliente que deverão ser atendidas para solucionar um determinado problema de negócio no qual o cliente faz parte.

Porém, de acordo com [Prikladnicki 2004], na maioria das empresas não existe um processo formal definido para realizar a coleta profunda de requisitos, o que pode levar com que o software seja desenvolvido em um tempo muito maior do que o planejado, e, além disto, não existem procedimentos bem definidos sobre a solução que um requisito deve possuir. Isto pode impedir a coleta de métricas e prejudicar a produtividade, ou seja, na prática, os problemas ocorrem, pois faltam requisitos ou foram levantados de forma insuficiente a contemplar todas as funcionalidades.

Com base nesses conceitos, constatamos que o cenário atual de desenvolvimento de software ainda está longe do ideal. Isto traz diversas consequências, como elevados custos no desenvolvimento do projeto, e isso é a consequência direta do retrabalho provocado durante um projeto, em nível de requisitos, projeto, codificação e teste, causados por uma definição mal elaborada do domínio do problema nas fases iniciais do desenvolvimento, [Ávila e Spínola 2007].

Ao observar a Figura 1, podemos verificar que à medida que um projeto vai passando de fase o custo para corrigir um defeito se eleva de uma forma considerável, e tais fatores que ajudam a explicar a forte influência da alta complexidade como fator de insucesso de projetos, tal como indicado na Tabela 1, pois quanto mais tardio o defeito é encontrado mais caro fica o projeto, levando a custos insustentáveis em alguns casos.

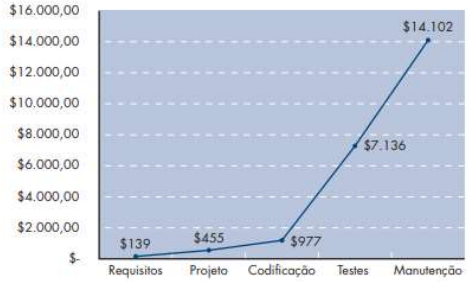

Figura 1: Custo relativo para correção de erros e defeitos - Adaptado [Boehm 2001].

[Sommerville 2018] afirma que em praticamente todos os sistemas os requisitos mudam, principalmente após o sistema ser instalado, onde estas mudanças são consequências de erros e omissões nos requisitos originais. Estas mudanças além de já aumentarem o custo do projeto, podem possuir impactos em outros requisitos já entregues fator que aumentaria ainda mais estes custos.

Com esta análise, percebemos o quão importante é a necessidade de melhorar a identificação da qualidade dos requisitos na fase inicial do projeto, pois é nela em que o custo é o menor de todo o ciclo de vida de um software. Com isso, podemos começar a discutir formas de identificar a qualidade de um requisito, para assim buscar alternativas para reduzir a chance de insucesso causada pela especificação incorreta de requisitos de software.

4. Conceitos da Rede Bayesiana

De acordo com [Bastos e Oliveira 2017] redes bayesianas são modelos gráficos probabilísticos que possibilitam o estudo de problemas em que há dependência entre as variáveis caracterizadas como causa e efeito.

Segundo [Gonçalves 2008] uma rede bayesiana pode ser modelada como um classificador, calculando a probabilidade de P(C|V), onde C representa a classe analisada e V o conjunto de variáveis que descrevem os padrões. O classificador mais importante dentre os classificadores Bayesianos é o *Naive Bayes*, descrito em [DUDA; HART 1973].

Conforme [Friedman, et al. 1997], a classificação é feita aplicando o teorema de Bayes para calcular a probabilidade de C dado uma particular instância de A1,..., An e então o predizendo a classe com a maior probabilidade a posterior. O processo de aprendizagem do *Naive Bayes* é feito de maneira indutiva, apresentando um conjunto de dados de treinamento e calculando a probabilidade condicional de cada atributo Ai, dado a classe C.

Segundo [Ferneda 2006] existem duas formas básicas de aprendizado de redes: aprendizado supervisionado e aprendizado não supervisionado. No aprendizado supervisionado, um agente externo (professor) apresenta à rede neural alguns conjuntos de padrões de entrada e seus correspondentes padrões de saída. Para cada entrada, o professor indica explicitamente se a resposta calculada é boa ou ruim. A resposta fornecida é comparada à resposta esperada. O erro verificado é informado à rede para que sejam feitos ajustes a fim de melhorar suas futuras respostas. Já na aprendizagem não supervisionada, não existe um agente externo para acompanhar o processo de

aprendizado. Neste tipo de aprendizagem, somente os padrões de entrada estão disponíveis para a rede. A rede processa as entradas e, detectando suas regularidades, tenta progressivamente estabelecer representações internas para codificar características e classificá-las automaticamente.

Com base nestes conceitos, podemos entender que as redes bayesianas podem ajudar no processo de classificação dos requisitos, porém como dito anteriormente precisamos definir a estratégia para criação da rede para que a mesma possa realizar as classificações de forma correta. Para este trabalho utilizaremos o conceito da rede supervisionada.

5. Aplicação da Rede Bayesiana

Para este trabalho a estratégia que adotamos para responder a pergunta de pesquisa foi utilizar uma rede bayesiana, onde tivemos como apoio 247 requisitos oriundos de cinco projetos reais, desenvolvidos em sequência, de uma equipe do departamento de TI de uma empresa do ramo de seguros situada na cidade de São Paulo. Na Tabela 2, vemos a disposição de requisitos e uma breve explicação de cada projeto.

Tabela 2: Projetos Utilizados

Projeto	Descrição do Projeto	Total de Requisitos
1	Criação de um gerenciador de sinistros para a área usuária	144
2	Melhorias funcionais para o sistema projetado no projeto 1	40
3	Inclusão de uma regulamentação obrigatória pelo órgão controlador do ramo de seguros no sistema de sinistros	17
4	Inclusão de um novo produto e necessidade de adaptar o sistema de sinistros	25
5	Melhorias e adição de novas regras do órgão regulamentador no sistema de sinistros	21

Conforme descrito anteriormente, no aprendizado supervisionado precisamos indicar a entrada e saída de dados, ou seja, todos os requisitos foram classificados em questão de qualidade em “Bom” e “Ruim”, que são as saídas esperadas. Esta classificação foi realizada em conjunto de um dos autores do artigo com a equipe de TI da empresa. A acurácia da rede foi medida da seguinte forma: em cada rodada de treinamento da rede separávamos de forma aleatória 75% de requisitos para testes e 25% para treinamentos. Cada rodada correspondeu ao conjunto de requisitos de um dos projetos analisados e a construção da rede foi acumulativa, ou seja, a cada iteração foram somados os requisitos da iteração anterior e novamente separados na mesma proporção para testes e treinamentos. Dessa forma, procuramos simular uma linha temporal dos projetos e como a rede se comportaria em um cenário mais próximo da realidade, com a construção de uma base de conhecimento acumulada a partir dos projetos.

Para realizar o processamento prévio do texto dos requisitos e, consequentemente, a preparação da entrada para a construção da rede foi utilizado o conceito do processamento natural de linguagem (PNL) que, segundo [Indurkhya e Damerau 2010] é uma área da computação que tem como objetivo extrair representações e significados mais completos de textos livres escritos em linguagem natural. Para aplicar a PNL utilizaremos a linguagem Python com a biblioteca NLTK, que segundo [Bird et al. 2009] tem como objetivo fornecer uma estrutura intuitiva e consistente junto com blocos de construção substanciais oferecendo aos usuários um conhecimento prático da PNL.

O primeiro passo após a classificação dos requisitos é a realização do tratamento dos dados, onde precisamos remover as inconsistências dos requisitos, entendido como inconsistência os requisitos que não estão no formato correto, previamente classificado e

os registros que não possuem requisitos. Para esta rede não foi necessário remover quaisquer elementos dos textos dos requisitos, uma vez que precisaríamos de toda a forma escrita para fazer a rede entender a escrita de diversos autores dos requisitos.

Após a limpeza das inconsistências, partimos para a etapa de preparação de treinamento da rede, neste trabalho optamos pela abordagem *bag-of-words*, onde segundo [Matsubara et al. 2003] cada documento é representado como um vetor de palavras que ocorrem no documento e com isto transformá-los em uma lista de valores numéricos representando a frequência da cada palavra, para que assim a partir destas frequências podemos calcular a pontuação para realizar a classificação. Para realizar esta tarefa de gerar a representação de frequência de termos utilizamos o pacote CountVectorizer da biblioteca scikit-learn, que segundo [Pedregosa et al. 2011], é uma biblioteca que possui ferramentas eficientes para mineração e análise de dados.

Após realizar a geração da representação de frequências utilizamos o algoritmo de *Naive Bayes*, BernoulliNB, que também está disponível na biblioteca scikit-learn, para gerar nosso classificador. Segundo [Pedregosa et al. 2011] o BernoulliNB, foi projetado para atender a dados discretos ele é projetado para recursos binários/booleanos e implementa os algoritmos ingênuos de treinamento e classificação de Bayes para dados que são distribuídos de acordo com as distribuições multivariadas de Bernoulli; isto é, pode haver vários recursos, mas cada um é assumido como sendo uma variável de valor binário. O BernoulliNB pode ter melhor desempenho em alguns conjuntos de dados, especialmente aqueles com documentos mais curtos.

Com isto realizamos a construção de nosso classificador, porém agora precisamos realizar a medição de resultados, a fim de validar a rede e verificar a eficiência da mesma.

6. Resultados

Para realizar a avaliação da nossa rede bayesiana, foi necessário percorrer todos os requisitos separados para testes e comparar o resultado real com o resultado obtido a partir da rede, onde em cada um destes resultados contabilizamos os acertos. Para obter os resultados dividimos os projetos e a cada nova interações da rede foram adicionados os requisitos do novo projeto, na Tabela 3, conseguimos observar os resultados obtidos a partir desta rede.

Tabela 3: Resultados da Análise da Rede Bayesiana

	Análise Rede Bayesiana						
Projeto	**Requisitos Projeto**	**Total de Requisitos**	**Requisitos Bons Reais**	**Requisitos Bons Rede**	**Requisitos Ruins Reais**	**Requisitos Ruins Rede**	**Acurácia da Rede**
Projeto 1	144	144	107	78	37	66	54,28%
Projeto 2	40	184	137	126	47	58	68,88%
Projeto 3	17	201	145	127	56	74	63,26%
Projeto 4	25	226	160	136	66	90	60,00%
Projeto 5	21	247	174	178	73	69	72,13%
Totais	247	247	174	178	73	69	72,13%

Percebemos que a cada nova interação a acurâcia da rede varia, iniciamos com uma acurácia de 54,28%. Apesar do aumento na primeira interação, inclusão dos requisitos do segundo projeto, entre a segunda e a terceira o valor decai. Isto se deve à adição de requisitos de novas regras e produtos diferentes do apresentado no primeiro projeto, para realizarmos os testes, portanto a rede não possuía ainda conhecimento prévio sobre os mesmos. Por este motivo na nossa última interação, finalizamos com uma acurácia de 72,13%, pois a rede já possuía um conhecimento prévio dos requisitos de uma

maior variedade de projetos e, dessa forma, estava preparada para avaliar corretamente os mesmos de acordo com o padrão da linguagem exposto em cada requisito destes sistemas.

7. Conclusões

Após entendermos um dos principais problemas nos projetos de software atualmente, que são os requisitos e entendermos o conceito de redes bayesianas, conseguimos aplicar estes conhecimentos para fim de validar se é possível verificar a qualidade de um requisito de software e assim mitigar os problemas nas empresas.

Com a análise desta rede, conseguimos verificar que, embora com poucos requisitos, foi possível identificar a qualidade deles com um nível de acerto considerável. Mesmo que ainda ocorram resultados classificados incorretamente, verifica-se um grau de confiança aceitável. Ainda percebemos que à medida que incluímos novos requisitos de sistemas tende a diminuir a acurácia, visto que a rede realiza o ajuste dinâmico das probabilidades para gerar um novo resultado, porém a mesma se adapta rapidamente e numa iteração nova, percebemos que a confiabilidade e eficiência aumentam.

Portanto, com este trabalho vimos que é sim possível mitigar os problemas de requisitos de sistemas, desde que apoiado com uma forma de validação dos mesmos, e para isto foi visto neste trabalho que uma rede bayesiana pode suprir esta validação, embora tenhamos utilizado requisitos oriundos de projetos de metodologias ágeis, acreditamos que independente da metodologia uma rede bayesiana pode auxiliar na validação dos requisitos, desde que treinada corretamente.

Para trabalhos futuros, deseja-se melhorar o algoritmo do protótipo para minimizar a fase supervisionada. Além disto, prever impactos nos sistemas a partir da validação destes requisitos para mitigar ainda mais os problemas oriundos de falhas de requisitos nos projetos de software por meio da sua classificação automática, a realização de um estudo comparativo deste algoritmo com outras técnicas de IA a fim de validar qual seria a melhor abordagem também seria uma ótima sugestão.

Referências

ÁVILA, Ana Luiza; SPÍNOLA, Rodrigo Oliveira (2007). Artigo Engenharia de Software - Introdução à Engenharia de Requisitos. Engenharia de Software Magazine Edição 01, Brasil, p. 46-52.

BASTOS, Paulo R. F. de M.; OLIVEIRA, Denise F. S. de (2017), Redes Bayesianas Aplicações em Confiabilidade e no diagnostico de perdas não técnicas, Appris, 1º ed.

Bird, S., Klein, E., and Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media, Inc., 1st edition.

Blaschek, José Roberto (2002), Gerência de Requisitos: O principal problema dos projetos de software - ISLIG-Rio. Acessado em: 07/03/2019, Disponível em: http://www.bfpug.com.br/islig-rio/

Boehm, B., and V. Basili (2001), "Software Defect Reduction Top 10 List," IEEE Computer, vol. 34, no. 1, January, pp. 135–137.

DUDA, R. O.; HART, P. E. (1973) Pattern Classication and Scene Analysis. [S.l.]: John Wiley Sons Inc. Hardcover.

FERNEDA, Edberto (2006) Redes neurais e sua aplicação em sistemas de recuperação de informação. Ci. Inf. – Brasília - Brasil, v. 35, n. 1, p. 25-30, jan./abr.

FERREIRA, Hiran Nonato Macedo (2018). Uma Abordagem Híbrida Baseada em Redes Bayesianas e Ontologias para Modelagem do Estudante em Sistemas Adaptativos e Inteligentes para Educação. 2018. Tese (Doutorado em Ciência da Computação) - Universidade Federal de Uberlândia, Uberlândia.

FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. (1997) Bayesian network classifiers. Machine Learning, v. 29, n. 2-3, p. 131–163.

GONÇALVES, André Ricardo (2008). Redes Bayesianas. Dissertação – UNICAMP, São Paulo.

INDURKHYA, N. and DAMERAU, F. J. (2010). Handbook of Natural Language Processing. Chapman & Hall/CRC, 2nd edition.

MATSUBARA, Edson T. et al (2003). PreTexT: uma ferramenta para pré-processamento de textos utilizando a abordagem bag-of-words, ICMC, São Carlos.

LUGER, George F. (2013). Inteligência Artificial. São Paulo: Pearson Education do Brasil, 6º edição.

Pedregosa et al (2011), Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830.

PRIKLADNICKI, Rafael (2004), Problemas, Desafios e Abordagens do Processo de Desenvolvimento de Software. 69f. Dissertação (Mestrado) – PUC-RS, Porto Alegre.

RAHMAN, Abdur, et al (2019), Classifying Non-functional Requirements using RNN Variants for Quality Software Development, MalTeSQuE, p. 25-30

SAHEKI, André Hideaki (2005). Construção de uma rede bayesiana aplicada ao diagnóstico de doenças cardíacas. Dissertação (Mestre em Engenharia) - Escola Politécnica de São Paulo, São Paulo.

SOMMERVILLE, Ian. (2018) Engenharia de Software, São Paulo, Pearson Education do Brasil, 10º edição.

The Standish Group (2015), CHAOS Report – The Standish Group. Acessado em: 02/08/2019, Disponível em: https://www.standishgroup.com/

Zardetto, Patricia Paula (2016), Definição de requisitos em metodologias ágeis. IBM Academy of Technology Affiliated. Acessado em: 03/07/2019, Disponível em: https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/mp258?lang=en