Sistemas & Gestão 14 (2019), pp 188-196

SISTEMAS & GESTÃO
Revista Eletrônica

GESTÃO DE RISCOS EM PROJETOS DE SOFTWARE: UMA ABORDAGEM BASEADA EM REQUISITOS NÃO FUNCIONAIS

Ana Cristina da Silva Andrade
ana_cristinas@yahoo.com.br
Instituto de Educação Tecnológica – Ietec, Belo Horizonte, Minas Gerais, Brasil.

José Luis Braga
zeluisdpiufv@gmail.com
Instituto de Educação Tecnológica – Ietec, Belo Horizonte, Minas Gerais, Brasil.

André Luiz de Castro Leal
andrecastr@gmail.com
Universidade Federal Rural do Rio de Janeiro – UFRJ, Rio de Janeiro, Rio de Janeiro, Brasil.

Fernando Hadad Zaidan
fhzaidan@gmail.com
Instituto de Educação Tecnológica – Ietec, Belo Horizonte, Minas Gerais, Brasil.

RESUMO

Em todas as etapas do processo de desenvolvimento de software existem riscos e estes podem representar uma oportunidade ou ameaça para o projeto. A prática precoce do gerenciamento de riscos em projetos de software possibilita conhecer e controlar os fatores que impactam no projeto, contribuindo assim para sua qualidade e sucesso. Esse artigo tem como objetivo propor um modelo conceitual composto pelos principais fatores de riscos em projetos de desenvolvimento de software que possibilite aos gerentes de projetos avaliar e monitorar riscos. Para atingir resultados que satisfaçam os objetivos deste trabalho foram realizadas atividades de maneira interativa de acordo com mapa mental previamente elaborado. Considerando o risco como um requisito não funcional, modelos para gerenciamento de riscos foram propostos através do NFR (non-functional requirements) Framework e Framework i*. Através de um exemplo, conclui-se que projetos que tratarem no tempo certo as operacionalizações de risco ou parte delas podem ter maior chance de sucesso.

Palavras-chave: Gestão de Projetos de Software; Gestão de Riscos; Requisitos Não Funcionais; Metas Flexíveis.

PROPPI / LATEC
DOI: 10.20985/1980-5160.2019.v14n2.1526

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

1. INTRODUÇÃO

Nas últimas décadas, falhas em projetos de desenvolvimento de software sempre foram um assunto preocupante para a engenharia de software. No Relatório CHAOS de 2015 (Hastie; Wojewoda, 2015), publicado pelo Standish Group, os seguintes números foram apresentados:

- 29% dos projetos obtiveram sucesso (concluídos no prazo, dentro do orçamento e com o escopo acordado).
- 52% dos projetos não foram executados conforme acordado (atraso na entrega, estouro de orçamento ou redução de escopo).
- 19% dos projetos falharam (foram cancelados ou não utilizados).

Esses percentuais, quando expressados em valores monetários, representam uma quantia significativa para as organizações e, numa organização de desenvolvimento de software, é um risco corporativo que pode significar sua sobrevivência.

Com isso, as organizações produtoras de software buscam novas estratégias para alcançar o sucesso dos projetos e a gestão de risco vem sendo adotada de forma a minimizar o surgimento de impedimentos que gerem declínio de produtividade e de qualidade dos softwares gerados (Silva, 2013). Um projeto de desenvolvimento de software precisa satisfazer a metas (qualidade, performance, ambiente e outras) que, normalmente, são modeladas como requisitos não funcionais (RNF).

RNF são aqueles que não estão relacionados com os serviços específicos oferecidos pelo software (o que o software faz), mas sim com as propriedades do software, como confiabilidade e tempo de resposta (como o software faz), (Sommerville, 2011).

Baseado em Chung et al. (2000), Leite (2009), Supakkul et al. (2010) e Cappelli et al. (2010), que enquadram transparência como um requisito de qualidade (não funcional), neste trabalho, o risco será considerado um RNF, ou um *softgoal*, usando a terminologia da modelagem intencional, visto que se trata de um fator subjetivo, dependente do domínio da aplicação e difícil de ser avaliado pelas partes interessadas.

Utilizando o NFR (non-functional requirements) Framework, é possível visualizar o desdobramento da subjetividade de riscos em desenvolvimento de software e, através do modelo i*, serão operacionalizadas ações e responsabilidades para mitigação de riscos.

Este trabalho considera risco como RNF e define suas operacionalizações concretas de forma a minimizar esses riscos. O objetivo é obter um modelo, composto pelas principais variáveis relacionadas com risco no desenvolvimento de software, que permita a engenheiros de software e gerentes de projeto considerar a inclusão de tratamento de riscos em projetos mais cedo no processo de desenvolvimento, atuando de maneira preventiva e aumentando as chances de sucesso do projeto.

Inicialmente este artigo apresenta conceitos de gestão de riscos, requisitos não funcionais e modelos intencionais (NFR Framework e Framework i*). Em seguida, são apresentados os principais fatores de riscos identificados no processo de desenvolvimento de software e os modelos elaborados utilizando tais fatores. E, por fim, são apresentadas as considerações finais e oportunidades de trabalhos futuros a partir deste.

2. REVISÃO DE LITERATURA

2.1 Gestão de risco

Segundo o *Project Management Body of Knowledge* (PMBOK, 2017), o risco é um evento ou condição incerta que, se ocorrer, terá um efeito positivo (oportunidade) ou negativo (ameaça) sobre pelo menos um objetivo do projeto envolvendo tempo, custo, escopo ou qualidade. Macedo e Salgado (2015), baseado em Charette (2005), definem risco como um evento ou o estado que pode causar danos, perda ou atraso num projeto de software. O gerenciamento de risco é fundamental para a gestão de projetos, sendo uma das dez áreas de conhecimento do PMBOK e também tratado por modelos de avaliação da qualidade de processos de software como ISO/IEC15504 e MPS.BR.

O gerenciamento de risco do projeto, segundo o PMBOK (2017), é composto pelos processos ilustrados abaixo, com o objetivo de aumentar a probabilidade e o impacto dos eventos positivos e diminuir a probabilidade e o impacto dos eventos adversos ao projeto.

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

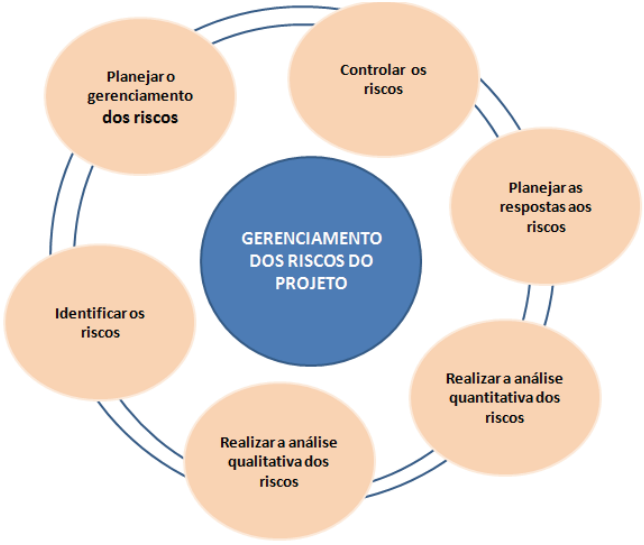

Figura 1. Visão geral dos processos de gerenciamento dos riscos do projeto
Fonte: Adaptado de Project Management Institute (2017).

2.2 Requisitos não funcionais

Na engenharia de software, requisitos são definidos como as descrições de o que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento, refletindo as necessidades dos clientes (Sommerville, 2011).

Os requisitos de software são classificados em:

- Requisitos funcionais: descrevem "o que" o sistema deve fazer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações;

- RNF: fixam restrições sobre "como" os requisitos funcionais serão implementados, ou seja, restringem "como" o sistema deve realizar o "o que", e inclui restrições de custos, performance, portabilidade, robustez e outros.

A implementação de RNF pode se propagar por todo o software. Esses requisitos definem restrições globais do software, do processo de desenvolvimento e do processo de implantação, sendo considerados globais no sentido de que eles surgem de todas as partes do sistema e de suas interações (Xavier et al., 2009), podendo afetar toda a arquitetura do sistema e não apenas componentes individuais.

Os RNF são críticos no desenvolvimento de um software. No projeto de software, se determinado requisito funcional do sistema não for implementado, o usuário pode encontrar uma maneira de contornar sua ausência. Porém, se um RNF não for atendido, pode comprometer o funcionamento de todo o sistema.

2.3 NFR framework

O NFR Framework foi proposto por Chung et al. (2000), tendo como foco a modelagem de RNFe suas operacionalizações, através da construção de um *Softgoal Interdependency Graph* (SIG), que descreve as dependências entre *softgoals* (metas flexíveis) e como eles são decompostos (Serrano, 2011). Meta flexível, sinônimo de *softgoal,* são qualidades (segurança, desempenho, confiabilidade e outras) desejadas pelos atores que não possuem critérios claros para sua satisfação, ou seja, são subjetivas e dependentes dos pontos de vista dos interessados (Oliveira et al., 2007).

Neste *framework* os RNF são tratados como metas flexíveis (*softgoals*), que serão identificadas e refinadas, sendo representadas por uma estrutura gráfica inspirada nas árvores And / Or (Xavier et al., 2009). Um *softgoal* é refinado até o ponto em que as operacionalizações sejam atingidas, gerando, assim, requisitos funcionais em função da necessidade de detalhar os RNF.

Para Chung et al. (2000), as metas estão relacionadas à intencionalidade dos atores, enquanto requisitos (funcionais e não funcionais) são características implementadas por funções do software.

Através da construção do gráfico de dependências, é possível avaliar as metas e apurar se determinado requisito não funcional está sendo alcançado em um projeto específico. Porém, conforme Xavier et al. (2009), as metas representam RNF e raramente estes podem ser considerados totalmente "satisfeitos".

2.4 Framework i*

O modelo i* é intencional e tem como objetivo descrever processos que envolvem vários atores, refletindo as motivações e interesses destes atores, bem como o relacionamento entre eles. A modelagem é baseada em atores, metas, crenças, habilidades e compromissos e representam a dependência mútua nas metas, tarefas e recursos. Diferente das demais técnicas de modelagem, expressa o porquê de determinada ação ou tomada de decisão (Yu, 1995).

O Framework i* (i-estrela), proposto em 1995 por Eric Yu, é uma técnica de modelagem conceitual para descrição de processos que envolvem vários atores (Serrano, 2011). Esta técnica concentra-se no relacionamento entre atores e suas dependências, focando nas razões ou motivações que estão associadas aos comportamentos (os porquês).

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

No Framework i* os atores dependem uns dos outros para alcançarem seus objetivos, realizarem tarefas e fornecerem recursos. Através da cooperação um ator pode alcançar objetivos que sozinho seria difícil.

O i* possui representação gráfica na forma de uma rede de relacionamentos, sendo formado por dois modelos básicos: Modelo SD (dependência estratégica), que descreve relações de dependência entre os atores, e o Modelo SR (razão estratégica), que explica como os atores atingem suas metas.

3. GESTÃO DE RISCOS BASEADA EM REQUISITOS NÃO FUNCIONAIS

A literatura existente sobre gerenciamento de riscos em projetos de desenvolvimento de software indica que um dos maiores motivos de falhas neste tipo de projeto é a inadequada, ou até mesmo inexistente, avaliação dos fatores de risco.

Apoiadores de gerenciamento de riscos de software afirmam que ações para reduzir as chances de falhas de um projeto podem ser tomadas a partir da identificação e análise das ameaças ao sucesso do projeto, ao longo de todo o ciclo do processo de desenvolvimento (Schmidt, 2001).

Para avaliar os riscos de um projeto, é necessário identificar quais são, na realidade, esses riscos, e conhecer aqueles que merecem maior atenção do gerente de projeto. Porém, os gerentes de projetos encontram dificuldade para identificar os riscos mais comuns num projeto de software.

Diante deste cenário, a primeira etapa deste trabalho foi identificar, através da literatura, as principais variáveis de risco que impactam no processo de desenvolvimento de software. Dentre as opções encontradas na bibliografia disponível, Schmidt (2001) apresenta uma lista extensa de fatores de riscos em projetos de software.

Para este trabalho, a lista publicada por Schmidt (2001) foi então comparada com publicações de Lopes (2014) e Barki (1993), que também apresentam fatores de risco em projetos de software. Com base nesta comparação, os fatores referentes a Planejamento e Comunicação foram adicionados à lista de Schmidt, definindo assim o conjunto de fatores de riscos para projetos de desenvolvimento de software, apresentado na coluna "Lista fatores de risco" da tabela 1, que serviu como base para o estudo apresentado.

Identificados os principais fatores de riscos de um projeto de software, o próximo passo foi a criação de um modelo de gerenciamento de riscos através do NFR Framework. Observe que os fatores de riscos foram renomeados apropriadamente para serem tratados como *softgoals*, conforme apresentado na coluna "*Softgoals*" do Quadro 1 e, posteriormente, o *softgoal* risco foi refinado.

Quadro 1. Fatores de riscos de projeto de desenvolvimento de software x *softgoals*

Lista de fatores de risco	Softgoals
1. Ambiente corporativo	**Ambiente corporativo**
Mudança no ambiente empresarial e organizacional	Volatilidade ambiente corporativo
Incompatibilidade entre cultura empresarial e novos processos	Incompatibilidade entre cultura empresarial e novos processos
Falta de valor de negócio e apoio	Ausência de valor de negócio e apoio
Ambiente corporativo instável (pressões competitivas alteram radicalmente os requisitos do usuário)	Instabilidade ambiente corporativo
Mudanças propriedade e/ou alta gerência	Modificabilidade de propriedade e/ou alta gerência
Inexistência Alinhamento Estratégico	Inexistência Alinhamento Estratégico
2. Patrocínio / propriedade	**Propriedade**
Falta de comprometimento alta direção	Ausência de comprometimento alta direção
Falta de aceitação do projeto	Ausência de aceitação do projeto
Falta comprometimento do usuário	Ausência comprometimento do usuário
Conflito entre departamentos	Incompatibilidade entre departamentos
Falta de aprovação de todas as partes	Ausência de aprovação de todas as partes
3. Gestão de relacionamento	**Relacionabilidade**
Falta de gerenciar as expectativas do usuário	Ausência de gerenciamento das expectativas dos usuários
Envolvimento inadequado dos usuários	Envolvimento inadequado dos usuários
Falta de cooperação dos usuários	Ausência de cooperação dos usuários
Falha na identificação/envolvimento de todos stakeholders	Falha na identificação / envolvimento de todos stakeholders
Aumento nas expectativas dos usuários	Aumento de expectativas dos usuários
Gerenciando múltiplas relações com as partes interessadas	Gerenciabilidade de múltiplas relações com as partes interessadas
A falta de experiência adequada dos usuários-chaves	**Inexperiência dos usuários-chaves**
4. Gerenciamento de projetos	Gerenciabilidade
Não gerir ou gerir inadequadamente mudanças	Ausência de gestão / gestão inadequadamente mudanças
Falta de habilidade/poder para gerenciar o projeto	Ausência de habilidade/poder para gerenciar o projeto

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

Metodologia inexistente /inadequada	Metodologia inexistente/inadequada
Definição inadequada dos papéis e responsabilidades	Ineficiente definição de papéis e responsabilidades
Controle pobre ou inexistente	Ineficiente / inexistente controle
Gerenciamento de riscos inexistente/inadequado	Gerenciabilidade de riscos inexistente/inadequada
Escolha da estratégia de desenvolvimento errada	Ausência de assertividade na escolha da estratégia de desenvolvimento
5. Escopo	**Escopo**
Objetivos / Escopo mal entendidos e/ou mal definidos	Ineficiente definição/entendimento de escopo e objetivos
Mudanças de escopo / objetivos	Modificabilidade de escopo / objetivos
Má definição ou definição incompleta	Ineficiente / Incompleta definição
Foco somente tecnológico / Ignorar requisitos de negócios	Foco exclusivamente tecnológico
Muitas linhas de comunicação	Variabilidade de linhas de comunicação
6. Requisitos	**Requisitos**
Falta de Requisitos Congelados (Mudanças)	Instabilidade
Má definição / entendimento	Ineficiente definição / entendimento
Falta de domínio/conhecimento do assunto	Ausência de domínio/conhecimento do assunto
7. Financiamento	**Custo**
Custo de desenvolvimento mal estimado	Custo de desenvolvimento mal estimado
Ausência de orçamento para custo de manutenção	Ausência de orçamento para custo de manutenção
8. Cronograma (agendamento)	Tempo de desenvolvimento
Prazo mal estimado	Tempo de desenvolvimento mal estimado
Prioridade inferior a outros projetos	Prioridade inferior a outros projetos
9. Processo de desenvolvimento	**Metodologia**
Metodologia inexistente/inadequada	Metodologia inexistente/inadequada
Nova metodologia / tecnologia	Imaturidade da metodologia / tecnologia
10. Pessoal (Personnel)	**Pessoas**
Falta de conhecimento / competência	Ausência de conhecimento / competência
Falta de competência / habilidade para gerenciamento	Ausência de competência / habilidade para gerenciamento
Relacionamento ruim da equipe	Baixa afinidade da equipe
11. Pessoal (Sttafing)	**Pessoal (Sttafing)**
Pessoal envolvido insuficiente / inapropriado	Pessoal envolvido insuficiente / inapropriado
Rotatividade de pessoas	Rotatividade de pessoas
Uso excessivo de terceiros	Alto número de terceiros
Falta de conhecimento / competência e disponibilidade dos envolvidos	Ausência de conhecimento / competência dos envolvidos
12. Tecnologia	**Tecnologia**
Novas tecnologias	Novas tecnologias
Instabilidade da arquitetura técnica	Instabilidade da arquitetura técnica
13. Dependências externas	**Dependências externas**
Dependências externas não atendidas	Dependências externas não atendidas
Múltiplos Fornecedores	Múltiplos Fornecedores
Falta de controle sobre terceiros / fornecedores	Ausência de controle sobre terceiros / fornecedores
14. Planejamento	**Planejamento**
Planejamento inexistente / inadequado	Inexistente / inadequado
15. Comunicação	**Comunicação**
Comunicação inexistente / inadequada	Inexistente / inadequado

Fonte: Elaborado pelos autores.

Observe que os refinamentos realizados de acordo com o Quadro 1 irão compor o catálogo de riscos de software utilizado neste trabalho.

Com o SIG de riscos elaborado, é possível compreender que, ao gerenciar riscos em ambiente corporativo, propriedade, relacionabilidade, gerenciabilidade, escopo, custo, tempo de desenvolvimento, metodologia, pessoas, recursos do projeto, tecnologia, dependências externas, planejamento e comunicação, os riscos do projeto estarão sendo gerenciados. Neste caso, temos uma contribuição positiva entre as dependências e, se todas as dependências forem atendidas, então a raiz também será.

O SIG de riscos permite visualizar os *softgoals*, ou metas flexíveis, referentes ao domínio que se está querendo gerir, sendo o primeiro passo para gestão de riscos de software.

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

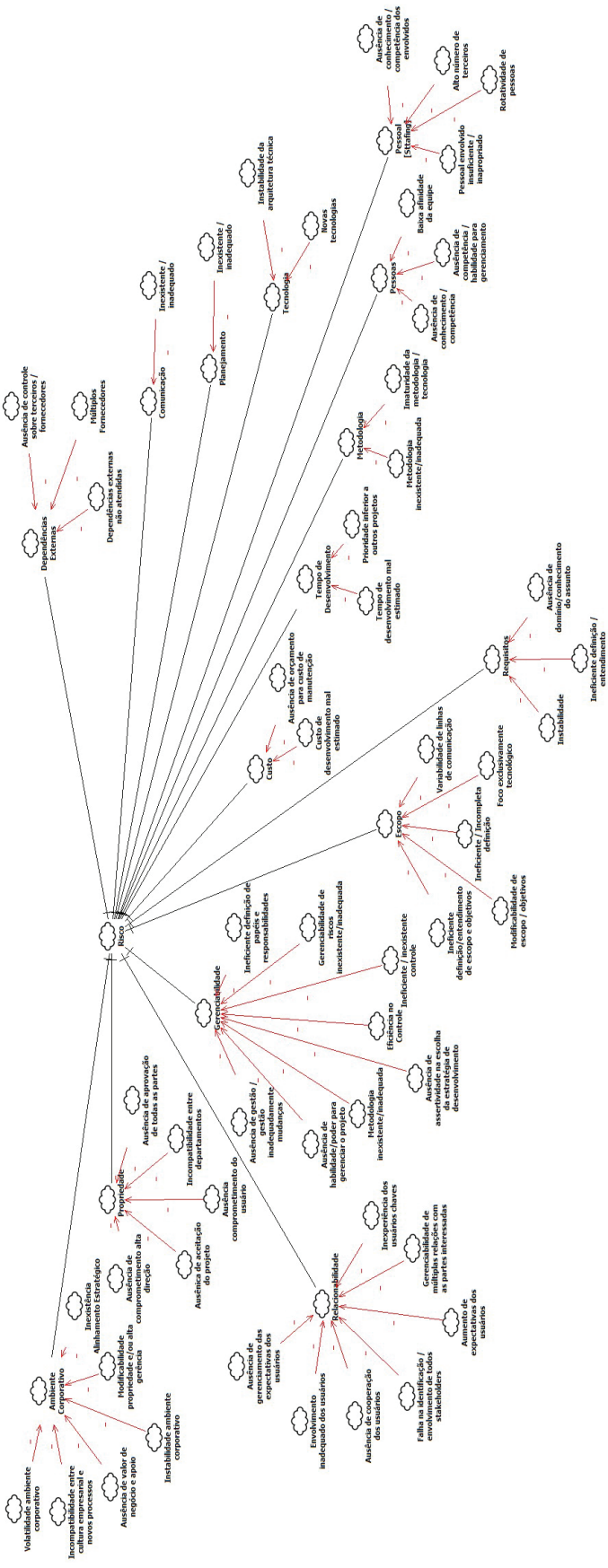

Figura 2. SIG de gerenciamento de riscos em projetos de software.
Fonte: Elaborado pelos autores (2018).

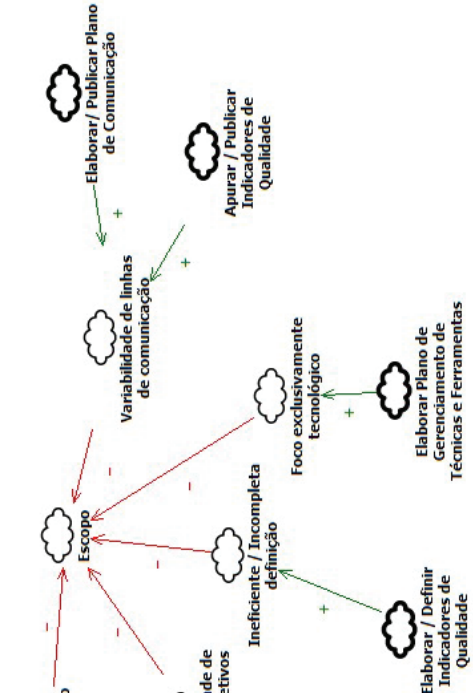

Figura 3. SIG de riscos: operacionalizações das metas de gerenciamento de escopo
Fonte: Elaborado pelos autores (2018).

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

Além de mostrar os desdobramentos dos riscos, apresenta também o inter-relacionamento entre diversos *softgoals*, bem como entre operacionalizações, e os impactos negativos e positivos entre eles.

A árvore apresentada acima poderá ser utilizada pelos gerentes de projetos como *framework* no momento da identificação de riscos de um projeto de desenvolvimento de software. Através de sua aplicabilidade, é possível verificar se os fatores de risco mais comuns em projetos de softwares estão sendo gerenciados e, ainda, gerar uma estrutura analítica dos riscos (EAR) completa e detalhada, uma vez que a árvore engloba fatores técnicos, organizacionais, gerenciais e externos.

A EAR é uma ferramenta para o gerenciamento de riscos a ser elaborada de acordo com cada projeto. Segundo Hillson et al. (2006), a EAR pode ser definida como um agrupamento que organiza e define os riscos do projeto, e possibilita a compreensão dos riscos assumidos pelo projeto.

Como o SIG de riscos apresentado na Figura 2 não detalha as operacionalizações de todas as metas, a Figura 3 apresenta o recorte do SIG de riscos com a inserção das operacionalizações que serão tratadas neste estudo para o caso de gerenciamento do escopo. Por exemplo, seguindo o grafo, a operacionalização para foco exclusivamente tecnológico (que influencia escopo) é elaborar Plano de Gerenciamento de Técnicas e Ferramentas.

A seguir é apresentado um exemplo de interação de agentes aplicado sobre o domínio de gerenciamento de escopo, no qual os atores do processo foram identificados e o modelo SD apresentado na Figura 4 foi desenvolvido.

Ao analisar o modelo SD, observe que as metas e metas flexíveis estão interligadas através das dependências, correlações e contribuições. É possível visualizar os atores do processo, seus objetivos (metas) e as metas flexíveis.

A Figura 5 apresenta um exemplo de aplicabilidade dos modelos elaborados. Para tal, foi desenvolvido o modelo de processo baseado na notação *Business Process Modeling Notation* (BPMN), referente ao gerenciamento do escopo de um projeto. Neste, utilizando as operacionalizações propostas no SIG de riscos e as dependências do SD de gerenciamentos de escopo, os RNF foram inseridos de forma explícita no processo de negócio.

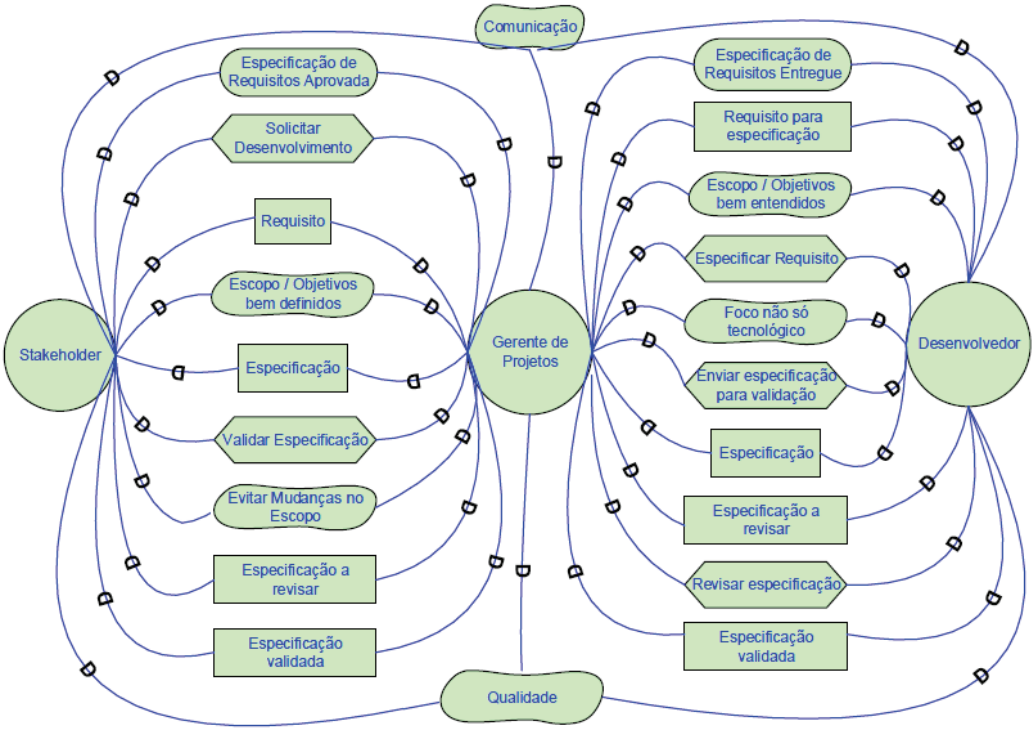

Figura 4. Modelo SD de Gerenciamento de Escopo
Fonte: Elaborado pelos autores (2018).

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

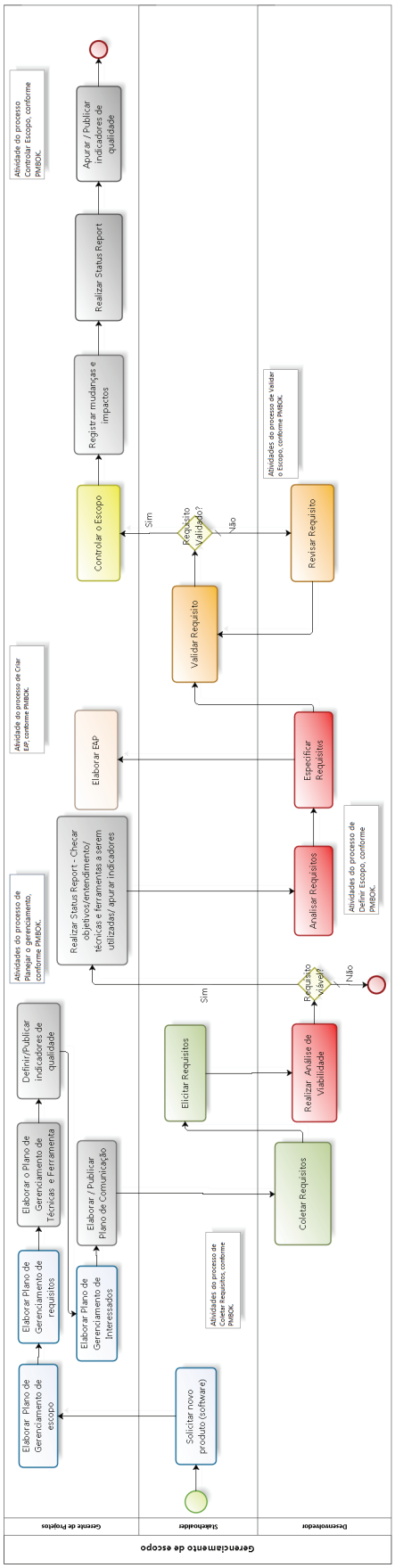

Figura 5. Modelo de processo de negócio de gerenciamento de escopo com inserção dos elementos de risco de Escopo inseridas de forma explícita no processo de negócio.
Fonte: Elaborado pelos autores (2018).

Observe que foi possível chegar a um modelo do processo com maior nível de detalhamento, visto que atividades do gerenciamento de riscos, que até então faziam parte do conhecimento tácito dos envolvidos no processo, foram inseridas explicitamente no processo, sem afetar a eficiência do processo original.

4. CONSIDERAÇÕES FINAIS

A principal contribuição deste trabalho foi o modelo conceitual (qualitativo) de riscos em projeto, agrupado em uma única estrutura de grafo, para facilitar seu entendimento e aplicação prática em projetos. Foi possível reunir as principais contribuições de todos os trabalhos citados em uma única estrutura que, com o tempo, poderá ser considerada um *framework* conceitual para ajudar projetistas e gerentes de projeto a enxergar melhor as situações de riscos em cada projeto, e a tratá-las adequadamente.

As variáveis identificadas foram agrupadas no catálogo de risco, utilizado para criar o SIG de riscos, através da NFR Framework. Este catálogo é dinâmico e representa o primeiro passo para elaboração de um catálogo mais completo. O SIG de riscos permitiu mostrar uma forma de validar os requisitos de riscos através da análise de rede de metas flexíveis, com a utilização do catálogo de risco.

Conclui-se que os modelos e exemplos apresentados podem contribuir para os gestores de projetos identificarem e gerenciarem os riscos do projeto em fase inicial, o que irá gerar um alerta prévio dos possíveis problemas, possibilitando uma ação preventiva e contribuindo positivamente para a qualidade e sucesso do produto de software.

Ainda há muito estudo a ser feito referente à abordagem de gestão de riscos utilizando requisitos não funcionais. Como trabalho futuro, sugere-se a aplicação do modelo às demais áreas de gerenciamento de projeto do PMBOK, visto que o exemplo apresentado é somente referente ao gerenciamento de escopo. E, um pouco mais desafiador, seria a aplicação das variáveis e modelos até aqui apresentados para criação de um sistema de agentes intencionais.

REFERÊNCIAS

Barki, H. et al. (1993). Toward an assessment of software development risk. Journal of management information systems, Vol. 10, No. 2, pp. 203-225.

Chung, L. et al. (2000). Non-functional requirements in software engineering. Springer Science & Business Media, New York.

Hastie, S.; Wojewoda, S. (2015). Standish Group 2015 Chaos Report Q&A with Jennifer Lynch. Disponível em: https://www.

Revista Eletrônica Sistemas & Gestão
Volume 14, Número 2, 2019, pp. 188-196
DOI: 10.20985/1980-5160.2019.v14n2.1526

infoq.com/articles/standish-chaos-2015. Acesso em 22 abr. 2019.

Hillson, D. et al. (2006). Managing project risks using a cross risk breakdown matrix. Risk Management, Vol. 8, pp. 61-76.

Leite, J. C. S. D. P. (2009). Software Transparency. In Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Lopes, J. D. S. (2014). Um modelo para apoio à decisão em avaliação de riscos em projetos de software utilizando simulação com dinâmica de sistemas. Dissertação de Mestrado, Universidade Federal Viçosa, Viçosa, MG, Brasil.

Macedo, M. H. B.; Salgado, E. G. (2015). Gerenciamento de risco aplicado ao desenvolvimento de software. Sistemas & Gestão, Vol. 10, No. 1, pp. 158-170.

Oliveira, A. et al. (2007). Engenharia de requisitos intencional: tornando o software mais transparente. XXI Simpósio Brasileiro de Engenharia de Software, 15-19 out. 2007, João Pessoa, PB.

PMBOK, G. (2017). Um guia do conjunto de conhecimentos em gerenciamento de projetos. In Project Management Institute.

Schmidt, R. et al. (2001). Identifying software project risks: an international delphi study. Journal of Management Information Systems, Vol. 17, No. 4, pp. 5-36.

Serrano, M. (2011). Desenvolvimento Intencional de Software Transparente Baseado em Argumentação. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ.

Silva, F. L. A. D. (2017). Análise do Impacto do Gerenciamento de Riscos no Sucesso de Projetos: Um Estudo de Caso em uma Organização de Desenvolvimento de Software. Dissertação, Universidade Federal de Pernambuco, Recife, PE.

Sommerville, I. (2011). Engenharia de Software. 9 ed. Pearson Prentice Hall, São Paulo.

Supakkul, Sam et al. An NFR pattern approach to dealing with NFRs. In: 2010 18th IEEE International Requirements Engineering Conference. IEEE, 2010. pp. 179-188.

Xavier, L. et al. (2009). Integração de Requisitos não Funcionais a Processos de Negócios: Integrando BPMN e NFR. In: Anais do WER10 - Workshop em Engenharia de Requisitos, Cuenca, Ecuador, April 12-13, 2010.

Yu, E. (1995). Modelling strategic relationships for process reengineering. Tese, Universidade de Toronto, Toronto, Canadá.

Cappelli, C. et al. (2010, March). Transparency versus security: early analysis of antagonistic requirements. In Proceedings of the 2010 ACM symposium on applied computing, pp. 298-305.

Recebido: 14 abr. 2019

Aprovado: 09 maio 2019

DOI: 10.20985/1980-5160.2019.v14n2.1526

Como citar: Andrade, A. C. S.; Braga, J. L.; Leal, A. L. C. et al. (2019), "Gestão de riscos em projetos de software: uma abordagem baseada em requisitos não funcionais", Sistemas & Gestão, Vol. 14, No. 2, pp. 188-196, disponível em: http://www.revistasg.uff.br/index.php/sg/article/view/1526 (acesso dia mês abreviado. ano).