UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
CURSO SUPERIOR DE TECNOLOGIA DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Ana Carolina de Oliveira Ribeiro

DESENVOLVIMENTO DE PRODUTO DE BACKLOG COM APOIO DE CROWDSOURCING

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2016

Ana Carolina de Oliveira Ribeiro

DESENVOLVIMENTO DE PRODUTO DE BACKLOG COM APOIO DE CROWDSOURCING

Trabalho de Conclusão de Curso apresentada ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Alexandre L'Erário

CORNÉLIO PROCÓPIO
2016

AGRADECIMENTOS

Gostaria, primeiramente, de agradecer à Deus que sempre esteve presente em minha vida. Agradeço aos meus pais, pois sem eles eu não teria chegado até aqui, a eles minha imensa gratidão.

Agradeço a meu orientador, Prof. Dr. Alexandre L'Erário, pelo apoio, incentivo, e por todo conhecimento transmitido durante o desenvolvimento desse trabalho.

Gostaria de deixar registrado também, o meu agradecimento ao meu namorado que me ajudou muito durante todos esses anos de faculdade e aos meus amigos que de alguma maneira colaboraram para que eu nunca desistisse de meus sonhos, pois acredito que sem o apoio deles seria muito difícil vencer essa etapa. Enfim, agradeço a todos que de alguma forma contribuíram para que eu chegasse até aqui.

RESUMO

Ribeiro, Ana. **Desenvolvimento de produto de Backlog com apoio de Crowdsourcing** 2016. 49 f. Trabalho de Conclusão de Curso (Graduação) – Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

O *Crowdsourcing* é um modelo de negócio para concepção e produção de produtos e serviços utilizando um conhecimento oriundo de origens múltiplas e heterogêneas, denominada como multidão. Entretanto, a configuração do ambiente em *Crowd*, não garante implicitamente que padrões de qualidade e processos empregados na produção mais formal dos produtos sejam aplicados. Os métodos ágeis atualmente são muito utilizados para conduzir o desenvolvimento dos projetos de software. Essa pesquisa propõe analisar a aderência de *Scrum* em ambientes de *Crowdsourcing*, no que tange a composição dos requisitos. Inicialmente foi feita a descrição dos conceitos, ferramentas e métodos utilizados. No fim, uma análise e apresentação dos resultados gerados.

Palavras-chave: Scrum. Crowdsourcing. Graduação.

ABSTRACT

RIBEIRO, Ana. **Product Development Backlog with Support of Crowdsourcing** 2016. 49 f. Trabalho de Conclusão de Curso (Graduação) – Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2016.

Crowdsourcing is a business model for design and production of goods and services using knowledge from multiple, heterogeneous sources, known as “crowd”. However, the environment setup for crowd does not implicity ensure that quality standards and processes employed in a more formal production of the products are applied. Currently, agile methods are widely used to guide the development of software projects. This research intends to analyze the adoption of Scrum on Crowdsourcing environments, regarding the composition of requirements. Initially, the description of the concepts, tools and methods used was made. Finally, an analysis and presentation of the generated results.

Keywords: Scrum. Crowdsourcing. Graduation.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE SIGLAS

BPMN	Business Process Model and Notation
ER	Engenharia de Requisitos

Sumário

1 INTRODUÇÃO

O interesse pelo desenvolvimento distribuído de software, que é o desenvolvimento de um produto de software de forma remota por um grupo de pessoas que colaboram para construir um produto em comum, cresceu nos últimos anos. Tal grupo pode ser composto por pessoas de diferentes localidades que juntas trabalham por um mesmo objetivo. O interesse por esse meio de desenvolvimento aumenta de forma gradativa. (L'Erario, 2009).

Segundo (L'Erario 2009) as justificativas para as organizações aderirem ao desenvolvimento distribuído de software são: o custo de produção, a proximidade com clientes, a disponibilidade de mão de obra, entre outros motivos.

Um dos processos que pode ser empregado para o desenvolvimento de distribuído de software é o *Scrum*. Seria então, um método ágil para colaborar na melhoria dos processos, conduzindo o desenvolvimento do projeto e focando sempre na qualidade (BOOTLA *et al.*, 2015).

Corroborando a ideia de desenvolvimento distribuído, houve um crescimento no interesse por *Crowdsourcing*. Tal afirmativa pode ser mapeada a partir da quantidade de publicações do tema encontradas no site, segue o link http://ieeexplore.ieee.org/Xplore/home.jsp. Atualmente existem 933 artigos publicados relacionados a este tema e existe uma grande quantidade publicados entre 2013 e 2015.

O *Crowdsourcing* é utilizado para encontrar soluções para problemas, novas ideias, por meio do uso da tecnologia comumente, a internet. Conta com a opinião de várias pessoas, utiliza-se da sabedoria das multidões em vez de usar somente da opinião de pessoas da equipe ou dos chefes de empresas. (CHUENE; MTSWENI, 2015),(RUOYU, LU, 2011).

Segundo (ANEGEKUH; SUN; IFEACHOR, 2014) uma maneira de otimizar e ajudar na solução de problemas, visando ser mais rápida e barata. O

nome *Crowdsourcing* vem de “multidão” e “terceirização”, onde usa-se da opinião de pessoas anônimas e não somente dos chefes de empresas. Pode ser aplicado em qualquer meio, inclusive no desenvolvimento distribuído de software.

Nas pesquisas efetuadas, conforme capítulo 2, não houve contribuições que adotam um *framework* como *Scrum* em um ambiente de *Crowdsourcing*. Um dos desafios identificados neste projeto é a aderência do *Scrum* no que se diz a requisitos para ambientes de desenvolvimento distribuído de software que utilizam o *Crowdsourcing*.

1.1 Problematização

Atualmente, muitas empresas utilizam métodos ágeis para conduzir o desenvolvimento de um produto. As empresas contam com esses métodos, visando buscar produtividade e qualidade, podendo sempre contar com um *feedback* de tudo que tem acontecido entre a equipe durante o desenvolvimento do produto. Isso é possível através de métodos ágeis, como por exemplo, *Scrum*.

Em contramedida percebe-se possível utilidade na inserção de ferramentas que possam auxiliar no desenvolvimento de software nas empresas, buscando uma melhoria em seus produtos e também colaborando no desempenho da equipe que está participando do projeto. Isso será possível utilizando o *Scrum* e *Crowdsourcing* integrados, adaptando seus métodos pelas equipes de desenvolvimento. Diante disso: Qual o valor dos métodos de *Scrum* e *Crowdsourcing* para auxiliar alunos em graduação a aprimorar suas técnicas para trabalhar em grupo e utilizar isso no mercado de trabalho, visando alcançar o sucesso desejado nos projetos seja em atividades acadêmica ou então no mercado de trabalho.

1.2. Justificativa

O *Scrum* é um *framework* muito utilizado desde de 1990, desenvolve e mantém produtos complexos, favorece em alguns fatores o desenvolvimento do

produto, sendo assim, traz diversas melhorias para a execução do projeto, seja por melhorar a qualidade ou por proporcionar uma maior agilidade da equipe, essa agilidade ocasiona um andamento maior no projeto. (MUNDRA; MISRA; DHAWALE, 2013). Existe uma quantidade significativa no crescimento do *Scrum* e muitos projetos que comprovam a eficácia de tal método. Realizada uma pesquisa no site http://ieeexplore.ieee.org/Xplore/home.jsp, nota-se a existência de 448 artigos relacionados a este tema.

A globalização faz com que o mundo atual sofra constantes mudanças tecnológicas que fazem com que as empresas sofram uma necessidade exacerbada em buscar novas formas de desenvolverem seus produtos, visando estarem sempre atualizadas. Assim então surgiu o termo *Crowdsourcing* (BITENCOURT; MELO; BERNARDES, 2014). O *Crowdsourcing* tem sido um modelo de sucesso utilizado em inúmeras áreas como sistemas de informação, comercialização e operacionalização. O crescente interesse acadêmico e industrial por este modelo já foi comprovado. Inúmeros projetos utilizam-se deste modelo, como por exemplo o caso da Linux, Fiat Mio que são exemplos da eficácia do *Crowd*.

O *Crowdsourcing* é um modelo que pode usar da sabedoria de inúmeras pessoas para colaboração no desenvolvimento de seus projetos, isso pode reduzir os custos, reduzir o tempo de desenvolvimento, sendo então útil para a solução de problemas num modelo de produção. Esta, é uma estratégia para a melhoria dos métodos atuais, pois é possível obter uma captura das necessidades do cliente.

As plataformas de *Crowdsourcing* são excelentes para um projeto, pois integram pessoas com as relações de mercado, o que é muito importante pois pode ocasionar o desenvolvimento melhor do produto. O fato das pessoas contribuírem para as ideias e os casos de sucessos existentes em relação a este modelo, fazem com que o crescimento do Crowdsourcing seja comprovado, e sua eficácia também.

Os problemas encontrados diante o desenvolvimento de projetos de software é evidente, estes problemas podem comprometer o andamento, a qualidade e a eficiência do projeto. Visando isto, e visando também o impacto positivo causado pelo *Scrum*, a necessidade de mudanças e a popularização do *Crowdsourcing*, este trabalho será executado para que aconteça a integração de ambos modelos para que se torne possível uma melhoria nas atividades acadêmicas ou então no mercado de trabalho.

1.3 Objetivos

O objetivo deste trabalho é desenvolver um processo cujo propósito seja gerar um *Product Backlog* a partir de requisitos coletados por meio de técnicas de Crowdsourcing.

1.3.2 Objetivos específicos

- Identificar como o *Scrum* trata requisitos.
- Identificar como o *Crowdsourcing* trata requisitos.
- Verificar se o *Scrum* dá suporte em um ambiente *Crowdsourcing*.
- Mapear como executar um projeto *Crowdsourcing*, o processo e a instrumentação do mesmo.
- Executar o experimento em um ambiente de graduação.
- Identificar qual foi o resultado final dos alunos participantes do experimento.

2. Fundamentação Teórica

Este capítulo visa trazer os conceitos relacionados ao tema proposto no trabalho, para alcançar um melhor entendimento sobre ele. A seguir os temas *Scrum* e o que engloba este método, o *Crowdsourcing* e em seguida a *Engenharia de Requisitos.*

2.1 Scrum

Scrum é um *framework* pertencente à área de gestão e adequado para desenvolvimento de projetos de software. Este, favorece em alguns fatores o desenvolvimento do produto, sendo assim, traz diversas melhorias para a execução do projeto, seja por melhorar a qualidade ou por proporcionar uma maior agilidade da equipe (MUNDRA; MISRA; DHAWALE, 2013).

Em 2011 foi realizada uma pesquisa em que conseguiram descobrir que 54% de 120 empresas de 35 países utilizaram *Scrum* no desenvolvimento de seus projetos, sendo assim, percebe-se que existe uma grande aceitação e utilização desse framework (BOOTLA *et al.*, 2015). O *Scrum* é dividido em equipes, segundo (MUNDRA; MISRA; DHAWALE, 2013) as equipes são compostas de 3 a 9 integrantes. Porém este número pode variar, dependendo do ponto de vista em questão, mas sempre as equipes devem ser constituídas por poucos integrantes.

Assim como um processo de produção, por exemplo, indicado por (SLACK, 2001), o *Scrum* é configurado por um conjunto de atividades, papéis e artefatos.

Os papéis que compõem o *Scrum* são: *Product Owner*, *Scrum Master*, *Scrum Team*. Também existem os artefatos *Product Backlog*, *Release Burndown Chart*, *Sprint Backlog*, *Sprint Burndown Chart*, *Task Board, Potentially Shippable Product Increment*. As atividades que compõem o *Scrum*: *Sprint Planning Meeting*, *Sprint*

Retrospective, *Sprint Review Meeting*, *The Daily Scrum*, *Estimating the Product Backlog, Prioritizing the Backlog*, *Release Planning* (EAGLESHAM, 2015).

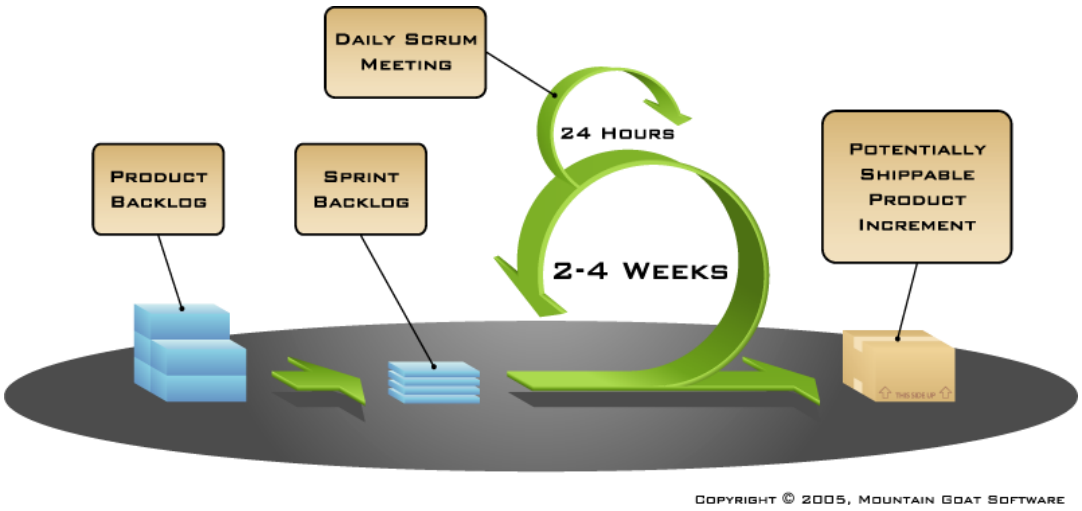

Figura 1 – Processo do *Scrum*

Fonte: Adaptada no site disponível em http://epf.eclipse.org/wikis/scrum, acesso em agosto, 2015.

O *Product Backlog* é uma lista com todas as funcionalidades desejadas a serem desenvolvidas no produto, incorpora os pedidos realizados pelo cliente. Os itens desejados também podem ser conhecidos por “estórias”. (KNIBERG, 2007). No quadro abaixo segue o exemplo de um *Product Backlog*:

Id	Nome	Importância	Est	Como demonstrar	Notas
1	Depósito	30	5	Logar-se, abrir a página de depósito, depositar R$ 10,00, ir para a página do meu saldo e verificar que este aumentou em R$ 10,00.	Precisa de um diagrama UML de sequência. Não é necessário se preocupar com criptografia por enquanto.

2	Verificar seu próprio histórico de transações	10	8	Logar-se, clicar em “transações”. Fazer um depósito. Voltar para transações, verificar se o novo depósito é listado.	Usar paginação para evitar consultas muito grandes ao banco de dados. Projetar de forma similar à página de visualização de usuários.

Quadro 1 – Product Backlog
Fonte: Adaptado de (KNIBERG, 2007)

• **Id:** é uma identificação, serve para ter o controle sobre as estórias (itens) caso os nomes sofram alguma alteração.

• **Nome:** Nome do item, onde normalmente é escrito também de uma maneira descritiva, para que o *Product Owner* e os desenvolvedores compreendam do que se trata.

• **Importância:** Pontuação de importância do item para o *Product Owner*. Quanto mais pontos, mais importante.

• **Est:** Estimativa Inicial de quanto tempo será necessário inicialmente para a implementação do item.

• **Como demonstrar:** Um teste, uma descrição de como a *Sprint* será executada.

• **Notas:** Outras informações gerais. A nota normalmente é bem breve.

Em seguida, podemos falar que o conteúdo da lista do *Product Backlog* é definido pelo *Product Owner* (P.O), este por sua vez, é responsável por determinar e priorizar o conteúdo existente na lista. Foca-se então somente em determinadas partes para que assim ocorra o desenvolvimento do produto em questão. Encontra-se aí então uma das atividades do método *Scrum*, o *Prioritizing the Backlog*, onde os itens são priorizados pelo PO.

O *Sprint* é considerado uma iteração, ele então pode ser denominado por um período de tempo onde determinadas tarefas são executadas. Os *Sprints* podem ter durações que variam entre 2 a 4 semanas As equipes durante o processo todo definem várias *Sprints* até que percebam que o produto em questão está pronto (MUNDRA; MISRA; DHAWALE, 2013).

Já o *Sprint Backlog* é uma lista de coisas a serem feitas com uma duração de tempo do presente até o fim da *Sprint*. Deixando a equipe atualizada. O que ainda falta ser feito no *Sprint* é colocada em um gráfico, este por sua vez conhecido como *Sprint Burndown Chart* (TECNOLOGIA; UNISANTA; GUSTAVO, 2009).

O *Sprint Planning meeting* é onde o *Scrum Team* e o *Product Owner* decidem o que será executado na próxima *Sprint*.

O *Impediment Backlog* é uma lista com os problemas que impedem a equipe de progredir, esses problemas são passados ao *Scrum Master*. E então essa lista é dividida em duas partes, uma em *Team Impediment* que são os problemas que podem ser resolvidos e o *Organization Impediment*, problemas que não poderão ser resolvidos. Temos também a *Estimating the Product Backlog*, estimativas do *Product Backlog*, que são medidas em pontos (EAGLESHAM, 2015).

O *Sprint Review Meeting*, é realizado ao final de cada *Sprint*. Nessa reunião de revisão do *Sprint*, mostra-se o que a equipe executou. Nesta reunião a equipe de desenvolvimento, o *Scrum Team,* mostram o que executaram (EAGLESHAM, 2015).

O *Sprint Restropective*, é realizado ao final de um *Sprint* para ver o que foi útil e bem executado e o que pode ser descartado. A partir disso toma-se as decisões que possam ocasionar melhorias ao desenvolvimento do projeto (EAGLESHAM, 2015).

A cada dia do *Sprint*, temos a *Daily Scrum* que é conhecida como reunião diária. Normalmente, essas reuniões são realizadas no mesmo ambiente, na mesma hora e todos os integrantes devem participar dela. A duração dela é em média 15 minutos. Nela é discutido tudo que foi realizado até o momento. O que será feito amanhã, e o que foi feito e também nele podemos pensar se existirá algum empecilho pelo caminho, estes que também são jogados em pauta para discussão. Os empecilhos em questão são discutidos pelo *Scrum Master*, um outro integrante importante, responsável por ajudar na remoção de problemas enfrentados pelos integrantes. O *Scrum Master*, também pode tratar dos *stakeholders* que são todos aqueles que estão interessados no software em questão. Falando sobre isso podemos citar também a *Task Board*, um conselho de tarefas, onde todo o trabalho realizado pela equipe durante a Sprint é amostrado, atualizado continuamente. As alterações são feitas na *Daily Scrum* (BERCZUK, 2007)(KNIBERG, 2007). Na Daily

Scrum três perguntas se fazem presentes, segundo (TECNOLOGIA; UNISANTA; GUSTAVO, 2009) que são elas:

- O que fez para o projeto desde a última reunião?
- O que fará para o projeto até a próxima reunião?
- Há algum obstáculo para conseguir seu objetivo? Precisa de ajuda?

Esse obstáculo que estamos falando depois é passado as mãos do Scrum Master.

Temos também o *Scrum Team*, formado por 5 a 9 integrantes, esta equipe é responsável pela parte de desenvolvimento do produto. Dentro desta equipe temos integrantes com diferentes funções, segundo (TECNOLOGIA; UNISANTA; GUSTAVO, 2009) são elas: programador, testador, designer gráfico, dentre outros. *Release Planning*, normalmente é executado pelo *Scrum Team* onde defini-se no início de um projeto as expectativas para ele. A equipe inicialmente desenvolve os planejamentos (EAGLESHAM, 2015).

Temos também o *Release Burndown Chart*, onde em um projeto *Scrum*, a equipe acompanha todo o progresso onde ocorrem atualizações em um gráfico burndown, onde no final de cada sprint executam essas atualizações. Exemplo no gráfico:

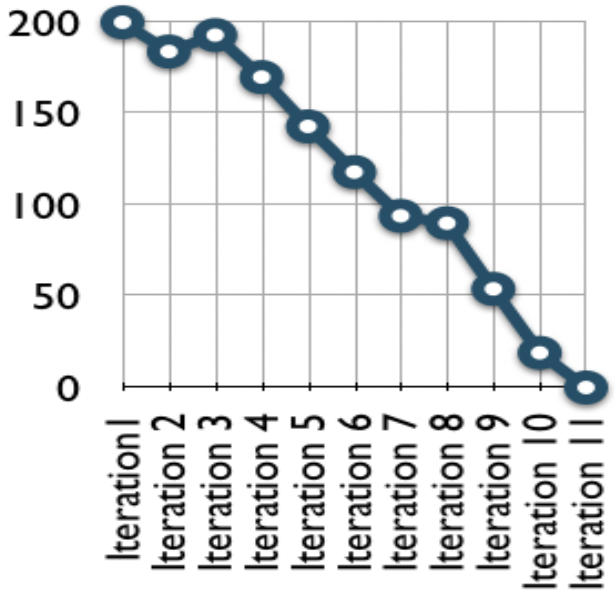

Figura 2 - Gráfico *Burndown*

Fonte: Adaptada no site disponível em http://epf.eclipse.org/wikis/scrum, acesso em agosto, 2016.

Podemos ver que o eixo horizontal indica os *Sprints*, e o eixo vertical a quantidade de trabalho. Este gráfico foi planejado em 11 *Sprints* durante o período

de duas semanas. Começaram com 200 pontos, o primeiro *Sprint* ocorreu da melhor maneira, o segundo então nem tanto, talvez por excesso de trabalho, algo ter sido adicionado ao projeto ao longo do desenvolvimento. Deste ponto o projeto continuou bem, porém durante a *Sprint* 7 o projeto suavizou, mas logo foi retomado. Podemos então ver que este gráfico pode ser muito útil pelas equipes.

O *Scrum* pode colaborar na elicitação de requisitos, existem estudos relacionando o *Scrum* com a engenharia de requisitos, como por exemplo em (ALAA, 2014) onde nota-se a importância do *Scrum Master* e proprietários do projeto para a rastreabilidade dos requisitos. A engenharia de requisitos é um processo da engenharia de software que engloba a elicitação, especificações, elaboração, validação e gerenciamento de requisitos para um sistema de software.

Gerenciamento de requisitos, inclui a parte de configuração de software, documentação, tabelas que mostram a origem dos requisitos. E ainda esse processo da engenharia ainda necessita no fim de um projeto adquirir e estabelecer os requisitos finais para um sistema (GAUR; SONI; BEDI, 2010).

2.2 Crowdsourcing

O *Crowdsourcing* está relacionado à produção coletiva, o poder da multidão, informação em massa, para ajudar no desenvolvimento de processos e na resolução de seus possíveis problemas. O *Crowdsourcing* surge para dar um nome a estas iniciativas que visam a busca do conhecimento em massa, através de plataformas Web a um inúmero grupo de indivíduos. O termo surgiu em 2006 por Jeff Howe, onde a palavra em si vem de multidão (Crowd) e terceirização (outsourcing), segundo ele o termo pode representar o ato de uma empresa, onde as tarefas realizadas pelos funcionários passam a ser terceirizadas por outras pessoas, em meio de usar da opinião da população para desencadear os problemas. HEFFAN; LYDON, 2012) Um exemplo de utilização de Crowdsourcing, pode ser o da plataforma Linux (PENG; BABAR; EBERT, 2014).

No http://ieeexplore.ieee.org/Xplore/home.jsp existe 933 artigos relacionados a este tema. É um conceito que nos últimos anos tem sido popularizado pela engenharia de software e tende a crescer. A ideia hoje do Crowdsourcing então, é utilizar da opinião de várias pessoas para ajudar no desenvolvimento de software e

não utilizar apenas de funcionários ou chefes da empresa para adquirir essas informações colaborativas. Segundo (PENG; BABAR; EBERT, 2014) as pessoas também se tornam “prestadoras de tarefa”, assim facilitando o desenvolvimento dos projetos.

Em desenvolvimento de software, o *Crowdsourcing* pode agir solicitando serviços on-line de uma rede indefinida. Usa-se da internet, da tecnologia, para facilitar este processo. Toda atividade *Crowdsourcing* é constituída por 4 pilares, que serão citados abaixo, dois se referem aos aspectos humanos, estes são a multidão e o *Crowdsourcer* (HOSSEINI *et al.*, 2015). Grandes empresas de tecnologia de informação tem utilizado o *Crowdsourcing* (PENG; BABAR; EBERT, 2014).

A tecnologia é uma condição para as organizações sociais atuais evoluírem, a difusão de redes, a comunicação digital, tudo vem progredindo com o passar do tempo. Pode-se comparar isto a evolução da eletricidade ou do motor elétrico por exemplo, que com as novas tecnologias tudo foi se aperfeiçoando e trazendo melhores resultados (CASTTELS, 2006).

Com essa evolução da tecnologia a internet possibilita a utilização de plataformas online de *Crowdsourcing* para captura de ideias da multidão. Empresas globais adotaram o *Crowd* visando obter um feedback dos usuários para ajudar na solução de seus problemas ou no próprio desenvolvimento de software. Pessoas que trabalham de maneira colaborativa, precisam estar cientes de tudo que ocorre no projeto e na equipe, isso tem o nome de consciência de grupo (HOSSAIN, 2012).

A comunicação na plataforma *Crowdsourcing* acontece através da troca de informações entre solicitantes e provedores, essa troca acontece para que os profissionais possam negociar sobre os requisitos que compõe o projeto. As pessoas que fazem parte do projeto, a multidão, podem estar localizadas geograficamente longe uma das outras, então utilizam da plataforma para proceder com a troca de informações. A plataforma também pode ter várias colaborações e os desenvolvedores podem colaborar de muitas maneiras, esses por sua vez, podem vir a possuir tarefas diferentes por exemplo, mas geralmente integram as informações que são parecidas. Esta plataforma apoia a gestão e a coordenação das pessoas fornecendo instalações para as tarefas *crowdsourced* e ainda possui a capacidade de supervisionar os compromissos das pessoas ligadas a ela (HOSSAIN, 2012).

O *Crowdsourcing* exige um número diversificado de opiniões e solucionadores de problemas. A taxonomia é uma crítica porque traz informações e objetos para estudo, ela é capaz de aumentar a compreensão e conceitos de ideias complexas.

Existe então, uma taxonomia para *Crowdsourcing*, o que leva a uma desordem tanto relacionada ao conceito de *Crowd* quanto a maneira de se aplicar para utilização. Estudando isso, percebe-se que o *Crowdsourcing* é constituído em quatro pilares.

O primeiro pilar, segundo (HOSSEINI *et al.*, 2014) é conhecido pela multidão (*Crowd*), que são as pessoas que constituem as atividades de *Crowdsourcing*. O segundo pilar é o *Crowdsourcer*, uma pessoa ou entidade sem fins lucrativos que vai atrás de poder e sabedoria das pessoas para uma determinada tarefa. O terceiro pilar é a tarefa *Crowdsourcing*, que é a atividade da qual as pessoas, a multidão, irão participar. E o quarto e último pilar é a plataforma *Crowdsourcing* que é o sistema em si, que pode ou não ser um software, onde o *Crowd* vai ser executado. A partir destes quatro pilares, é possível analisar questões específicas de cada pilar.

No primeiro pilar, o *Crowd* (multidão), nota-se que dentro das pessoas que participam das atividades de *Crowdsourcing*, estas, possuem 5 características distintas. Diversidade, ou seja, significa que pessoas distintas são utilizadas para realizar as tarefas, essa diversidade é dividida em quatro categorias, espacial que significa selecionar pessoas com origens e localidades distintas. Gênero, que significa selecionar as pessoas de acordo com um gênero específico. Diversidade de idade que seria selecionar pessoas desconsiderando a faixa etária e perícia diversidade que seria selecionar pessoas de experiências diferentes.

A segunda característica é *unknown-ness*, ou seja, o fato da pessoa ser anônima. Dentro do Crowdsourcing o anonimato da pessoa tem dois significados. Primeira é quando o *Crowd* não sabe quem está realizando o *Crowdsourcer*. A segunda é quando o *Crowd* participa da tarefa e não conhece os outros membros. A terceira característica é *largeness*, grandeza, ou seja, abrangente, em determinadas situações *largeness* não deve ser abundante para evitar confusão na multidão. A quarta característica é a *suitability*, ou seja, conveniência, o *crowd suability* é quando acontece um ajuste para acertar as habilidades a serem executadas para realizar uma tarefa (HOSSEINI *et al.*, 2014). Tabela referente ao primeiro pilar:

1. Diversity
1.1. Spatial Diversity
1.2. Gender Diversity
1.3. Age Diversity
1.4. Expertise Diversity
2. Unknown-ness
2.1. Not Known to Crowdsourcer
2.2. Not Known to Each Other
3. Largeness
3.1. Number Fulfils the Task
3.2. Number Not Abundant
4. Undefined-ness
5. Suitability
5.1. Competence
5.2. Collaboration
5.3. Volunteering
5.4. Motivation
5.4.1. Mental Satisfaction
5.4.2. Self-Esteem
5.4.3. Personal Skill Development
5.4.4. Knowledge Sharing
5.4.5. Love of Community

Quadro 2 – Primeiro pilar crowd

Fonte: Adaptado de (HOSSEINI *et al.*, 2014)

No segundo pilar o *Crowdsourcer*, pessoa, instituição sem fins lucrativos. Onde destacam-se quatro características. A primeira característica é *incentives provision*, ou seja, um incentivo, algo para dar mais ânimo ao trabalho, o *crowdsourcer* pode fornecer incentivos. Temos, três tipos de incentivos, o incentivo financeiro, o que é mais proeminente, o segundo, incentivo social, o terceiro incentivo entretenimento. A segunda característica é *open call*, ou seja, uma audição

em aberta para qualquer pessoa, isto significa que o *Crowdsourcing* atividade é aberto ao público em geral. A terceira característica é a *ethicality provision*, um ato moral que representa as normas de conduta de um determinado grupo. Existem três atos, citados na tabela 3. A quarta e última característica é *privacy provision* que como o próprio nome diz é a privacidade, significa a vida privada ou assunto pessoal, em *Crowdsourcing* a privacidade deve existir entre os participantes. Tabela referente ao segundo pilar:

1. Incentives Provision
1.1. Financial Incentives
1.2. Social Incentives
1.3. Entertainment Incentives
2. Open Call
3. Ethicality Provision
3.1. Opt-out Procedure
3.2. Feedback to Crowd
3.3. No Harm to Crowd
2. Privacy Provision

Quadro 3 – Segundo pilar crowdsourcer

Fonte: Adaptado de (HOSSEINI *et al.*, 2014)

No terceiro pilar, a atividade terceirizada fornecida pelo *crowdsourcer*, encontra-se oito características. Tabela referente ao terceiro pilar:

1. Traditional operation
1.1. In-house
1.2. Outsourced
2. Outsourcing Task
3. Modularity
3.1. Atomic Tasks
3.2. Divisible to Micro Tasks
4. Complexity
4.1. Simple Tasks

4.2. Complex Tasks
5. Solvability
5.1. Simple for Humans
5.2. Complex for Computers
6. Automation Characteristics
6.1. Difficult to Automate
6.2. Expensive to Automate
7. User-driven
7.1. Problem Solving
7.2. Innovation
7.3.Co-creation
8. Contribution Type
8.1. Individual Contribution
8.2. Collaborative Contribution

Quadro 4 – Terceiro pilar the crowdsourced task

Fonte: Adaptado de (HOSSEINI *et al.*, 2014)

No quarto e último pilar, na plataforma *Crowdsourcing*, encontra-se quatro características distintas. *Crowd-related-interactions*, *Crowdsourcer-related interactions*, *Task-related facilities*, *Platform-related facilities*. Neste último pilar, pode-se notar ainda uma escassez de abordagens relacionadas as plataformas *Crowdsourcing* (HOSSEINI *et al.*, 2014).

Foi comprovado a partir de estudos que o *Crowdsourcing* pode ajudar no desenvolvimento de software, especialmente na parte de engenharia de requisitos, pois a multidão pode ser algo que favorece o que é projetado, e dessa maneira atender as necessidades do cliente. Existem vários estudos que tentaram utilizar do poder da multidão para resolver os problemas da engenharia de requisitos, esses, incluem a *Requirements-driven social adaptation*, *feedback-based requirements engineering*, *stakeholders discovery*, *requirements identification*, *empirical validation* (HOSSEINI *et al.*, 2015).

2.3 Engenharia de requisitos

Os requisitos são um conjunto de ideias que devem descrever de maneira clara e concisa todos os aspectos relevantes para o desenvolvimento de um sistema. Os requisitos devem fornecer informações suficientes para permitir que os desenvolvedores implementem um software que atenda às necessidades de seus clientes. Segundo (SOMMERVILLE, 2010), um requisito é do tipo funcional quando descreve um serviço ou funcionalidade que um sistema deve realizar e, não-funcional, quando existem restrições em relação ao sistema e o desenvolvimento do mesmo.

A análise de requisitos é a primeira atividade a ser executada no desenvolvimento de um software, é também responsável por definir as funções que um sistema deve realizar. Os requisitos servem para definir o que o sistema deve fazer ao invés de como será feito. É muito importante entender o que o cliente precisa ou então o que ele acha que precisa. O levantamento dos requisitos se torna de suma importância, pois é a partir daí que se resulta o desenvolvimento de um sistema (GIOVANELLA, 2013).

No desenvolvimento de software, a elicitação dos requisitos é feita por todas as partes envolvidas no desenvolvimento do sistema, clientes, analistas e desenvolvedores. Estudos têm comprovado que a qualidade do sistema está relacionada ao desenvolvimento do mesmo, ou seja, equivalente ao quanto seu processo de desenvolvimento é hábil. (TOBERGTE; CURTIS, 2013).

O levantamento de requisitos tem papel fundamental no desenvolvimento de um software, pois é capaz de prevenir falhas no processo de desenvolvimento de um sistema. Em (ARRUDA) , é possível notar que a etapa de requisitos é de suma importância pois 40% a 60% dos erros encontrados em sistemas é causado nessa etapa, pela falta de definição adequada dos requisitos.

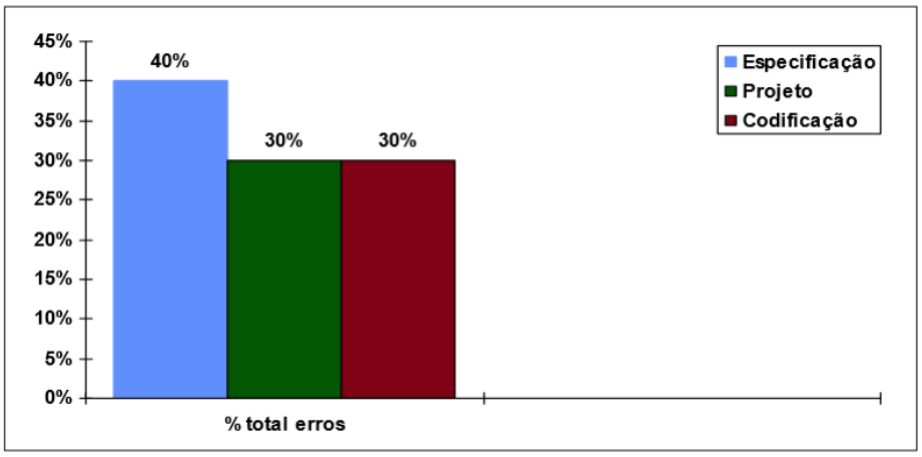

Figura 3 – Percentual de erros na ER

Fonte: Adaptado de (CAMPOS, 2013)

A engenharia de requisitos (ER) leva em consideração as exigências e necessidades de seus clientes para o desenvolvimento de software, para a ER é de extrema importância que todos os projetos de software tenham seus requisitos devidamente definidos, e que haja uma boa interação entre o software e o cliente (MELOROSE; PERROY; CAREAS, 2015). Ter requisitos bem definidos auxilia no entendimento do que precisa ser executado, sendo assim é possível elicitar, entender e validar as necessidades e expectativas dos clientes e posteriormente, documenta-las.

Na ER é possível notar que o *Crowdsourcing* é capaz de auxiliar na etapa de elicitação de requisitos do desenvolvimento de um projeto. O *Crowdsourcing* facilita o envolvimento de um amplo número usuários que podem moldar os requisitos de software e colaborar com alternativas de como cumprir tais requisitos, isso faz com que aumente o volume e a diversidade dos mesmos. Com isso é possível obter uma ideia devidamente definida sobre os usuários em relação as suas expectativas sobre o sistema a ser desenvolvido (HOSSEINI *et al.*, 2015).

Em suma para que haja um sistema funcional e eficaz é necessário um levantamento de requisitos e o mapeamento desse processo. A elicitação dos requisitos pode não garantir que o software atenda todas as necessidades do cliente, mas previne o surgimento de falhas e inconsistências durante o desenvolvimento de seu projeto, melhorando assim o processo de desenvolvimento de produtos de software.

2.4 Trabalhos relacionados

Em (HOSSEINI *et al.*, 2015) defende o uso de *Crowdsourcing* na elicitação de requisitos, e tenta descobrir maneiras de ajustar o *Crowdsourcing* para aperfeiçoar os requisitos. Usam dois grupos focais constituídos por 14 integrantes, sendo eles, usuários e desenvolvedores. Realizam então uma pesquisa on-line envolvendo 34 pessoas da comunidade de engenharia de requisitos. Para execução disso, separaram os aspectos humanos do *Crowdsourcing* e então os norteou para descrever os requisitos. Os dois grupos, passaram por duas sessões distintas na Universidade de Bournemouth no Reino Unido, um mesmo questionário foi feito a eles, entretanto, cada pessoa tem seu nível e distinções de conhecimento.

Os 14 integrantes foram divididos em dois grupos de 7 e então entregaram o questionário com um determinado tempo para responder as perguntas da entrevista e do debate. Tudo foi gravado e depois transcrito e então analisaram todas as respostas por via de um método de análise de conteúdo. Esta análise foi feita por 2 investigadores, e se fez presente um terceiro para que não houvesse problemas. Com os resultados, identificou-se um conjunto de relações entre recursos *Crowdsourcing* e os atributos de requisitos de qualidade induzido, 34 relações foram notadas, classificaram em 10 categorias, onde cada categoria expressava uma característica do *Crowdsourcing*.

No final, considerando a falta de conhecimento sobre o *Crowdsourcing* para aumentar a qualidade dos requisitos induzidos, foi confirmada a relação entre o *Crowd* e requisitos. Conclui-se então que existe um potencial significativo do *Crowd* para descrever requisitos. Nota-se também que adotar o *Crowd*, trouxe diversas dificuldades, o que traz uma maneira correta de configura-lo.

Em (VLAANDEREN *et al.*, 2010), um estudo de caso da vida real é feito para mostrar como os métodos ágeis podem ser aplicados a gestão de produto. As experiências aprendidas da empresa são descritas através de lições obtidas para ajudar os outros a aplicarem os princípios ágeis, isso através da utilização do *Scrum*. Nesse estudo, destaca-se a esperança de que outras empresas possam ser ajudadas a partir deste estudo de caso e aplicar e medir os efeitos da refinaria de requisitos.

A funcionalidade do produto então em questão é em partes, durante o processo acontece a definição dos detalhes e especificações. A primeira parte então é a *vision* que é a partida para o ciclo de vida de cada produto. Nesta etapa levantam-se ideias pela empresa, cliente ou algum interessado, quando a ideia atinge o produto, em seguida converte-se isso a um tema. Outra parte é o *theme*: onde ocorre a execução formal da visão, descrevendo-a em mais detalhes. O gerente do produto define a funcionalidade.

O tema então deve descrever o problema de negócio e as questões que então surgem. Essa descrição não pode exceder uma página. Uma visão pode ser complexa e expressa por meio de vários temas. O tema é revisado pelos integrantes da equipe de desenvolvimento. A outra e última parte é a definição de requisitos, onde ocorre a descrição detalhada dos requisitos, essa parte então é dividida em três etapas. A primeira é realizada pela equipe SPM (gestão de produtos de software), onde, esta equipe realiza a tradução dos conceitos em uma lista definições de requisitos. Nesta parte deve-se garantir o critério de ajuste, para que haja compatibilidade entre os requisitos.

Depois das definições, ocorre-se a elaboração em requisitos pela equipe de desenvolvimento. Para isso ser possível, cada definição de requisitos é executada pela primeira vez num determinado *Sprint* de desenvolvimento. O time *Scrum* então, garante a clareza para compreensão de cada requisito por todos os membros da equipe, podendo existir nessa parte diagramas e informações necessárias para ajudar na aplicabilidade dos requisitos. Concluiu-se então que para garantir a SPM ágil, o tamanho da tarefa, estrutura *Backlog*, refinaria dos requisitos, tudo isso tem que estar presente para a execução bem sucedida.

Foi realizada uma pesquisa no http://ieeexplore.ieee.org/Xplore/home.jsp e foi verificado que existem trabalhos que relacionam requisitos e Crowdsoucing, porém sem nenhuma aplicação prática.

3. Proposta de processo

Apresentar um processo, especificado em BPMN, capaz de gerenciar requisitos baseado em *Scrum* em ambientes de *Crowd*.

3.1 Requisitos em *Scrum*

O *Scrum* é um *framework* utilizado para colaborar com o gerenciamento e planejamento de projetos. É muito utilizado no desenvolvimento de softwares. Em *Scrum* os requisitos são tratados de forma incremental, a cada iteração que é chamada de Sprint (SOARES; CABRAL; ALENCAR, 2013). A principal prática do *Scrum* é a *Sprint*, onde são desenvolvidos os itens de trabalho definidos no Product Backlog pelos membros da equipe, que pode durar de uma a quatro semanas, a *Sprint* inclui as fases tradicionais do desenvolvimento de software: requisitos, análise, projeto e entrega.

Esse *framework* é composto por quatro artefatos principais. O *Backlog* do *Produto* que é uma lista composta pelos principais requisitos do projeto. O Backlog da *Sprint* que é uma lista de tarefas para transformar o *Backlog* do Produto, por uma *Sprint*. Um *burndown* é uma medida do *backlog* restante pelo tempo. Um *Burndown* de Versão para Entrega mede o Backlog do Produto restante ao longo do tempo de um plano de entrega. Um Burndown de Sprint mede os itens do Backlog da Sprint restantes ao longo do tempo de uma Sprint.

O *Scrum* defende que a interação entre as pessoas que participam do projeto pode ser um forte auxiliar na etapa de elicitação requisitos e também na análise dos mesmos. A elicitação de requisitos é uma etapa de extrema importância no *Scrum*, pois a partir dela é possível realizar o desenvolvimento do *Product Backlog*, sendo esse o ponto primordial do *Scrum*. O levantamento dos requisitos acontece através das reuniões executadas no *Scrum* com todos os membros do projeto, nas reuniões são apontadas as reais necessidades do negócio e os requisitos a serem desenvolvidos durante a execução do projeto. O *Product Backlog* como citado no capítulo 2.1, é uma lista de atividades que provavelmente serão desenvolvidas durante o projeto (CARVALHO; MELLO, 2009).

3.2 Requisitos em Crowdsourcing

O *Crowdsourcing* está relacionado à sabedoria da multidão, a coleta de informação em massa, para ajudar no desenvolvimento de projetos e na resolução de seus possíveis problemas. Howe denomina o *Crowdsourcing* como uma maneira de terceirizar a resolução de problemas de determinados projetos e trazer a solução deles através da inteligência coletiva (HOWE, 2006). O *Crowdsourcing* é constituído por quatro pilares (HOSSEINI et al., 2014): o *Crowd*, que são os indivíduos que constituem as atividades a serem executadas; o *Crowdsourcer*, que possui demandas a serem executadas; o *Crowdsourcing*, que é a atividade em si; e a plataforma *Crowdsourcing*, que gerencia e auxilia a realização das atividades.

O *Crowdsourcing* pode colaborar com a etapa de elicitação de requisitos no desenvolvimento de um projeto, especialmente em sistemas utilizados por um grande número de indivíduos (HOSSEINI *et al.*, 2015). A partir de estudos é possível afirmar que o *Crowdsourcing* pode auxiliar no desenvolvimento de software, especialmente na elicitação de requisitos, pois à sabedoria coletiva pode ser algo que favorece o que está sendo desenvolvido, e dessa maneira atender as necessidades reais do cliente.

3.3 Processo proposto

O processo proposto neste trabalho é uma compilação das teorias de *Scrum*, *Crowdsourcing* e requisitos. Esse modelo tem-se o nome de r4c. O principal propósito deste trabalho é testar esta intersecção e verificar a viabilidade das etapas propostas nas figuras abaixo:

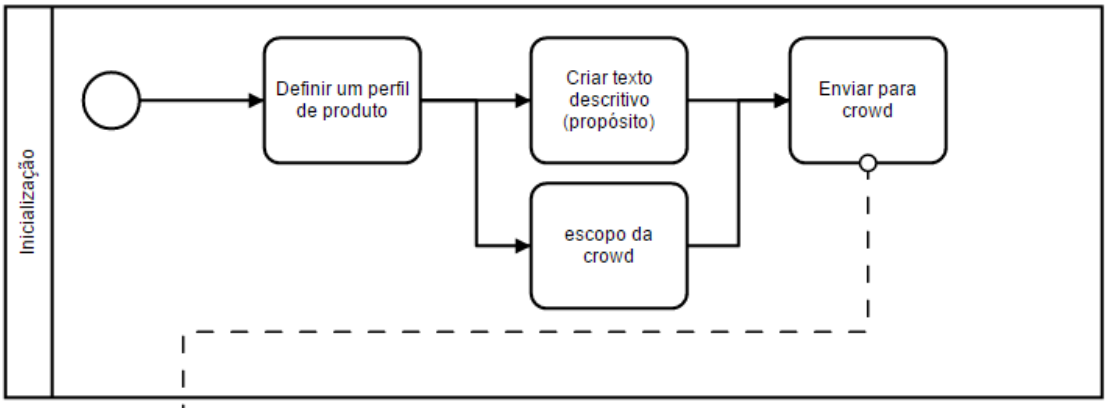

Figura 4 – Etapa inicialização

Fonte: Autoria própria

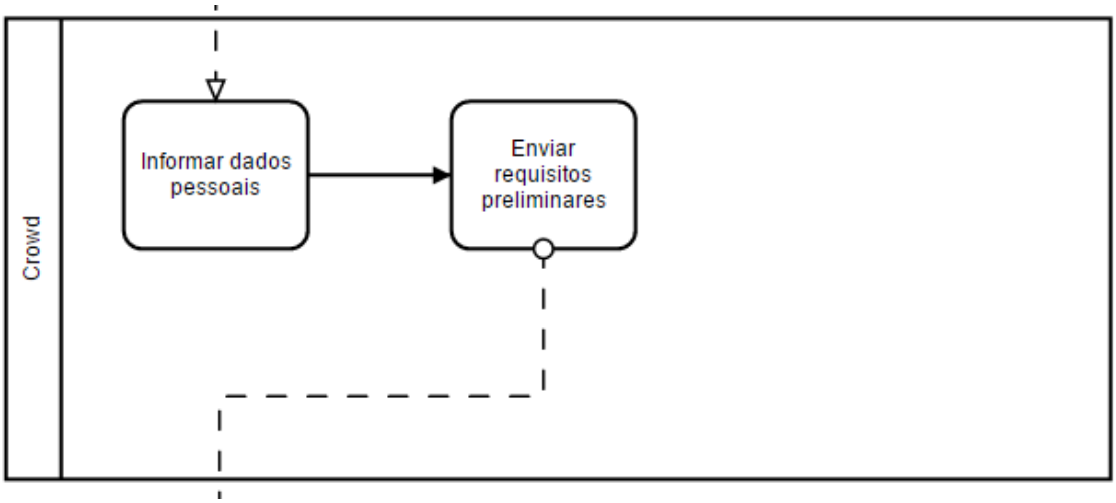

Figura 5 – Etapa *crowd*

Fonte: Autoria própria

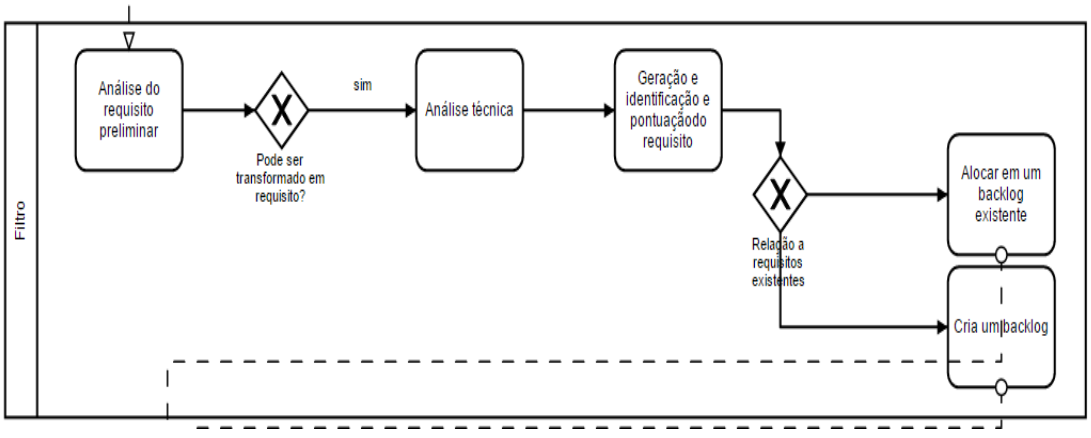

Figura 6 – Etapa de filtroFonte: Autoria própria

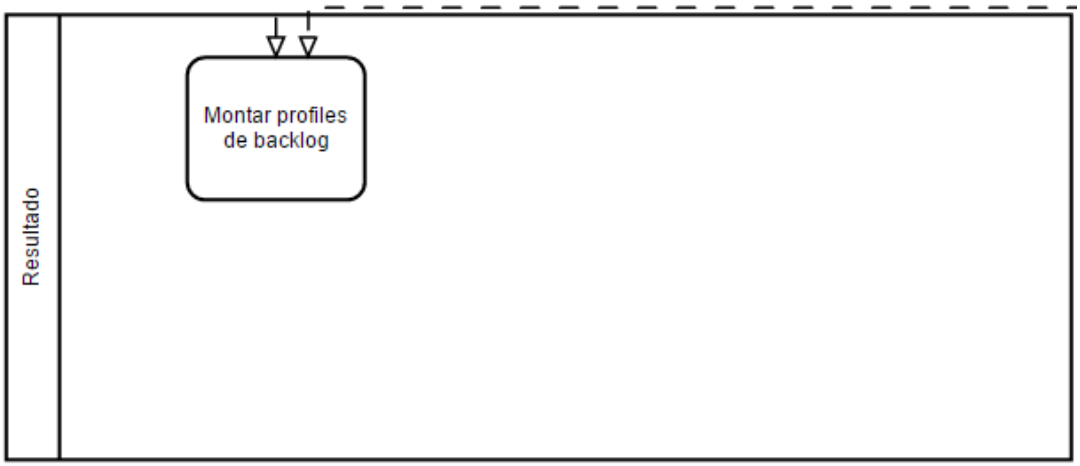

Figura 7 – Resultado

Fonte – Autoria própria

O processo acima está composto por 4 etapas, sendo elas: inicialização, crowd, filtros e resultado.

1. Inicialização

A etapa de inicialização ocorre de maneira independente do *Crowd*, ou seja, neste momento o *Crowd* não e envolvido. Embora as referências citadas neste trabalho não atentam para um procedimento inicial, o r4c, contextualiza um material antes de iniciar o processo de recrutamento da multidão. Neste sentido, espera-se que quando a multidão for constatada, os propósitos e resultados esperados já estejam definidos. Definir o perfil do produto é importante pois, trata-se do procedimento inicial para que a multidão possa iniciar suas atividades, a definição ocorreu por meio de discussões, *braimstorm*.

1.1 Criação de material descritivo

Criar um material específico, para quando começar a recrutar pessoas elas tenham ideia do propósito esperado.

1.2 Escopo do crowd

É importante definir o escopo do Crowd, para saber quem especificamente vai poder trabalhar dentro do projeto.

1.3 Enviar para crowd

Após definido o perfil do produto, feita a criação do material descritivo e definido o escopo do Crowd então será necessário utilizar de alguma ferramenta para que a pessoa entre no sistema e consiga disparar os requisitos.

2. Crowd

2.1 Informar dados pessoais

É importante que cada elemento que constitui o projeto informe seus dados pessoais. Para que assim seja possível identificar o perfil de cada participante.

2.2 Enviar os requisitos preliminares

Conforme visto após os elementos informarem seus dados pessoais, então é feito o envio dos requisitos preliminares.

3. Filtro

Mecanismo usado para classificar e relacionar os requisitos. Elaborado neste trabalho manualmente, apenas como teste. A análise de requisitos é uma composição de conceitos com linhas tênues entre eles. Tem-se dois tipos de requisitos, os requisitos de software e requisitos de negócio. Os requisitos de software que são instruções que são funções que o usuário precisa realizar para atingir o objetivo do sistema. Os requisitos de negócio ou regra de negócio que é a forma de fazer o negócio, é um conjunto de instruções que os usuários já seguem e que o sistema a ser desenvolvido deve contemplar. Restrições, exceções são exemplos de regras de negócio.

Nessa etapa do modelo proposto verifica-se a compatibilidade dos requisitos, se eles são complementares, não compatíveis ou iguais. Após feita a análise é possível concretizar a execução do *Product Backlog.*

4. Resultado

Após todas as etapas do processo serem executadas obtém-se o resultado, ou seja, a montagem de profiles de backlog.

4. Método e procedimento da pesquisa

O método científico estuda o mundo, sugere a teoria do comportamento, analisa e verifica hipóteses. Esse método é utilizado para entender, o processo, o produto de software e o ambiente. É então uma abordagem sobre construção de distintos modelos. O método de engenharia avalia as soluções já existentes, indicando as soluções mais adequadas, mede e também analisa até que haja alguma melhora em tal modelo. Esse método está relacionado a melhora evolutiva do modelo de processo. O método analítico ou matemático, sugere uma teoria formal, desenvolve a teoria em si. Esse método é dedutivo, não precisa de um projeto experimental no sentido estatístico mas oferece uma base analítica para desenvolvimento de modelos.

No capítulo presente, será descrito o método e procedimento de pesquisa abordado para a realização do presente trabalho.

4.1 Experimentação

O método adotado neste trabalho é o experimental. Segundo (Travassos, 2002), a experimentação é o centro do processo científico, ou seja, somente experimentos verificam teorias e podem explorar os fatores críticos e dar à luz ao fenômeno novo para que as teorias possam ser formuladas e corrigidas. O método experimental inicia-se com o levantamento de um novo modelo, não necessariamente baseado em um modelo existente, e tentar estudar o efeito do processo ou produto sugerido pelo novo modelo.

O autor também afirma que a experimentação oferece um modo sistemático, disciplinado, computável e controlado para avaliação da atividade humana. Novos métodos, técnicas, linguagens processos e ferramentas, segundo ele, não deveriam ser apenas sugeridos, publicados ou apresentados sem experimentação. Portanto, é preciso avaliar novas metodologias em comparação com as já existentes.

Segundo (L'ERARIO, 2009) a ideia central de experimentação é o controle e mensuração. A ideia de experimento também engloba o conceito de simulação. Simulação é um experimento realizado num modelo físico representativo real.

No caso do presente projeto, a escolha do teste experimental se deve ao fato de que não se busca constatar uma hipótese.

4.2 Protocolo de pesquisa adotado

O protocolo adotado deve representar a maneira na qual a pesquisa vai ser conduzida.

4.2.2 Hipóteses

Uma hipótese é uma teoria provisória, uma suposição. Para a análise do teste experimental foram levantadas duas hipóteses:

- **H0:** O processo proposto não atende a proposta e não gerencia requisitos
- **H1:** O processo proposto atende a proposta e gerencia requisitos

4.2.3 Cenário de teste

O cenário de teste é a abordagem que permite testar o processo r4c, neste trabalho. Para tanto, um problema foi conduzido para um conjunto finito de pessoas com o propósito de gerarem requisitos. Os requisitos gerados foram processados seguindo as atividades do r4c.No cenário de teste executei o modelo de processo proposto em BPMN na seção 3.3.

4.2.4 Variáveis

Após a execução do modelo proposto na seção 3.3, as variáveis coletadas e analisadas foram:

R1. Quantidade de requisitos

R2. Tempo para executar o r4c (filtro)

4.2.5 Descrição dos participantes

Uma das etapas que englobam este trabalho foi a escolha dos participantes para colaborar com a execução do experimento. Foi definido que esse teste experimental teria um público restrito, pessoas com conhecimento sobre a área de engenharia de software. Com base nessa ideia, escolheu-se, alunos de graduação dos cursos de Análise e Desenvolvimento de Sistemas e Engenharia Elétrica da Universidade Federal do Paraná do Campus de Cornélio Procópio que estão diretamente ligados a tecnologia.

Os alunos da graduação colaboraram com a etapa de elicitação de requisitos para a então formação de um *Product Backlog* devidamente definido.

4.2.6 Ferramenta

O *Moodle* (*modular object-oriented dynamic learning environment*), ou seja, ambiente de aprendizado modular orientado ao objeto), é um ambiente de aprendizagem. Pode ser utilizado em um ambiente de ensino de educação a distância, mas também pode ser implementado em cursos presenciais, possibilitando uma melhor interação entre aluno e professor. É uma ferramenta de auxílio ao aprendizado, desenvolvido na linguagem de programação PHP, ele faz com que ocorra uma interação entre alunos e professores facilitando a comunicação entre os mesmos.

Os professores podem criar salas de estudo online, disponibilizar todo o material passado em aula e propor tarefas. O *Moodle* ainda é composto por fóruns, bate-papos, avaliações, criação de trabalhos individuais, trabalhos colaborativos, criação de pastas, envio de arquivos, criação de testes. Em síntese, os alunos utilizam esse ambiente para a troca de conhecimentos e envio de arquivos com os professores.

Nesse trabalho, o *Moodle* foi utilizado na etapa da elicitação de requisitos, aonde o professor Dr. Alexandre L'Erário propôs aos alunos de graduação um texto

descritivo sobre o produto em questão e a partir disso cada aluno seria responsável por gerar requisitos. O envio desses requisitos entre os alunos e professor, aconteceu via ferramenta *Moodle*. Em síntese, a internet é uma grande aliada do Crowdsourcing pois traz inúmeras maneiras de realizar a coleta de requisitos.

5. Operacionalização

Neste capítulo serão apresentados os resultados da execução do processo experimental.

5.1 Teste experimental

O teste experimental foi executado em cima do modelo de processo proposto na seção 3.3 do trabalho.

Os telefones móveis surgiram no Brasil em 1990, a globalização fez com que as evoluções das tecnologias de telecomunicação crescessem exponencialmente nos últimos 25 anos, provendo cada vez mais qualidade, velocidade de conexão e estabilidade para os usuários e também um amplo número de modelos de telefones celulares existentes no mercado e atendendo cada vez mais a necessidade de seus clientes. *International Telecommunications Union* estimou em 2014 que 95% das pessoas terão um dispositivo móvel.

O mercado de telefonia móvel em suma, cresceu explosivamente desde a introdução dos sistemas digitais. O número de assinantes continuará crescendo nos próximos anos, com um aumento bastante significativo no número de usuários e dos minutos de uso. Para que o Brasil acompanhe a evolução mundial dos sistemas usados em comunicações móveis, é necessária uma maior agilidade nas negociações entre Governo e operadoras para o desenvolvimento e implantação de novas tecnologias dando assim acesso aos usuários e melhores serviços.

Visando a popularidade da telefonia móvel e a importância da elicitação de requisitos no desenvolvimento de um projeto, foi escolhido o celular como o produto a ser desenvolvido, para então ocorrer o levantamento de requisitos e o desenvolvimento do *Product Backlog* com o apoio do *Crowdsourcing* para a coleta dos requisitos para o teste experimental desse trabalho.

O teste experimental foi executado para que houvesse uma ideia de como iria efetuar-se o experimento desse projeto.

5.1.1 Público alvo

Com base nessa pesquisa, escolheu-se um grupo para participar de um estudo de caso cujo objetivo é a elicitação de requisitos para o desenvolvimento de um *Product Backlog* devidamente definido do software em questão. Para isso foram escolhidos alunos do curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná do Campus de Cornélio Procópio que estão diretamente ligados a tecnologia e a essa globalização e popularidade da telefonia móvel.

5.1.2 Aplicação do experimento

Para o levantamento dos requisitos no teste experimental foi utilizada a ferramenta *Google Forms*, a partir dela é possível construir pesquisas, votações, preparar testes e também coletar informações de uma maneira simples, prática e intuitiva. Previamente, foi proposto aos alunos da graduação a seguinte questão: “Para você o que um celular ideal deve possuir? ”. A partir de tal questão foi pedido para que cada um dos estudantes gerasse 10 requisitos.

Conforme citado, segue-se abaixo os quadros, no qual apresentam os requisitos levantados pelos estudantes:

Luís Otávio - Engenharia Elétrica
R1 - Sistema operacional simples
R2 - Bateria de boa duração
R3 - Memória Ram Boa
R4 - Bom tamanho de memória interna
R5 - Câmera traseira e frontal boa
R6 - Celular fino
R7 - Custo benefício
R8 - Design moderno e bonito
R9 - Dois chips
R10 - Tela de 5 polegadas

Quadro 5 – Teste experimental 1

Fonte: Autoria própria

Matheus Spirandeli- Engenharia Elétrica
R1 - Memória Ram Boa
R2 - Bom processador
R3 - Boa câmera
R4 - Display de alta qualidade
R5 - Memória de armazenamento grande
R6 - Durabilidade alta da bateria
R7 - Design moderno
R8 - Espessura fina
R9 - Possuir capacidade para dois chips
R10 - Sistema operacional atualizado

Quadro 6 – Teste experimental 2

Fonte: Autoria própria

Ruy Oliveira - Engenharia Elétrica
R1 - Preço bom
R2 - Garantia com mais de dois anos
R3 - Possuir um bom processador
R4 - Boa câmera
R5 - Sistema operacional atualizado
R6 - Bateria de longa duração e carregamento rápido
R7 - Tela de 5 polegadas
R8 - Ser leve
R9 - Capacidade de expansão de memória
R10 - Dois chips

Quadro 7 – Teste experimental 3

Fonte: Autoria própria

Rafaela Dizaró Silveira - Engenharia Elétrica
R1 - Bom processador
R2 - Design moderno
R3 - Câmera boa
R4 - Gravador de vídeos em alta resolução
R5 - Bateria duradoura
R6 - Preço justo
R7 - Tamanho não muito grande
R8 - Memória boa
R9 - Ter agenda eletrônica
R10 - Possuir garantia

Quadro 8 – Teste Experimental 4

Fonte: Autoria própria

A execução do teste experimental teve uma duração de aproxidamente 15 dias. Durante esse período foram coletados os requisitos e então analisados para que então pudesse acontecer a construção do *Product Backlog*. Os requisitos foram denominados como complementares, não compatíveis e iguais.

Os requisitos denominados como complementares são aqueles que um pode conciliar ao outro sem que haja interrupção da implementação de tal requisito.

Os requisitos denominados não compatíveis são requisitos excludentes, ou seja, não podem ser implementados no mesmo produto. Por exemplo: R1 - Quero um celular com tela redonda, - R2: quero um celular com tela quadrada. Não é possível implementar os dois e obter o mesmo resultado.

Os requisitos denonimados iguais, são aqueles que ao implementa-los o resultado será o mesmo.

Tais resultados foram fundamentados de acordo com a entrega da análise ao professor L'Erario (2009), responsável pela disciplina no qual ocorreu os testes experimentais. Durante o teste, o professor foi responsável por supervisionar os participantes e recolher os requisitos coletados.

Após feita a análise dos requisitos, abaixo segue-se o quadro do *Product Backlog* oficial do teste experimental:

Backlog Oficial
R1 - Sistema operacional simples e atualizado
R2 - Bateria duradoura e de rápido carregamento
R3 - Memória RAM boa
R4 - Bom tamanho de memória interna
R5 - Câmera boa
R6 - Espessura fina
R7 - Design moderno
R8 - Dois chips
R9 - Garantia com mais de dois anos
R10 - Bom processador
R11 - Ser leve
R12 - Capacidade de extensão de memória
R13 - Display de alta qualidade
R14 - Gravador de vídeos em alta resolução
R15 - Tamanho do celular não ser muito grande
R16 - Possuir agenda eletrônica

Quadro 9 - *Backlog* do teste experimental

Fonte: Autoria própria

5.2 Experimento

Primeiramente, para a execução do experimento, o professor responsável por ministrar as matérias de programação orientada a objetos e tópicos em computação (Alexandre L'Erário), disponibilizou o conteúdo geral para a execução do experimento onde o sistema proposto a ser desenvolvido trata-se de um ambiente de integração entre clientes e prestadores de serviços (pessoas físicas ou jurídicas). Um cliente pode buscar no sistema um profissional, empresa que presta um determinado serviço, negociar preço, forma de pagamento. Os serviços incluem instalação elétrica, pintura, limpeza, higienização de veículos.

Como já especificado no capítulo 4.2.5, no experimento participaram em torno de 25 alunos, entre 18 e 25 anos, estudantes do segundo e sexto período de Análise e Desenvolvimento de Sistemas e oitavo período do curso de Engenharia Elétrica. Como os alunos no segundo período, não possuíam conhecimento sobre a plataforma *Crowdsourcing* e o *framework Scrum*, foi necessário um breve entendimento sobre o assunto. Os participantes do teste experimental, tiveram as instruções necessárias para que fosse possível gerar os requisitos. Após os estudantes adquirirem conhecimento ao projeto a ser executado e ser todo o

assunto elucidado, então o professor disponibilizou o enunciado do produto a ser desenvolvido via ferramenta Moodle.

Segue abaixo o quadro 10, no qual apresenta uma prévia da matriz dos requisitos coletados pelos estudantes:

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22	R23	R24
R1	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R2	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R3	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R4	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R5	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R6	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R7	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R8	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R9	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C
R10	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C	C	C
R11	C	C	C	C	C	C	C	C	C	C	I	C	C	C	I	C	C	C	C	C	C	C	C	C
R12	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C	C
R13	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C	C
R14	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C	C	C
R15	C	C	C	C	C	C	C	C	C	C	I	C	C	C	I	C	C	C	C	C	C	C	C	C
R16	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C	C
R17	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C	C	C
R18	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I		C	C	C	C	C
R19	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C	C
R20	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C	C
R21	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C	C
R22	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C	C
R23	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I	C
R24	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	I

Quadro 10 – Experimento oficial

Fonte: Autoria própria

A matriz completa dos requisitos está disponível em anexo no final desse trabalho.

5.3 Resultados

Conforme o modelo de processo proposto na seção 3.3, na etapa 4 temos o resultado do teste experimental.

A execução do projeto teve uma duração de aproximadamente 40 dias. Durante esse período os estudantes tiveram as instruções necessárias para transcorrer o levantamento dos requisitos.

Conforme citado anteriormente, no experimento participaram em torno de 25 alunos, entre 18 e 25 anos, estudantes do segundo e sexto período de Análise e Desenvolvimento de Sistemas. Foram coletados 65 requisitos ao todo, todos eles foram analisados para que fosse possível executar a montagem da matriz de requisitos. A análise dos requisitos e desenvolvimento da matriz levou em torno de 5 horas e 15 minutos para acontecer.

Na etapa 3 da seção 3.3, o filtro, foi detectado um impasse, a de analisar os requisitos como compatíveis, não compatíveis e iguais.

Conforme a figura do quadro 10, que corresponde a seção 5.2, a matriz possui 4225 comparações no total, a partir das definições para os tipos de requisitos, após a análise é possível detectar 125 comparações definidas como iguais, 0 como não compatíveis e 4100 como complementares.

Houve alguns impasses detectados ao executar o desenvolvimento do teste experimental, pois a realização do filtro é uma atividade abstrata, com forte dependência de especialista, um outro impasse encontrado foi a capacidade de processar um volume de informações e a veracidade das comparações.

Segue abaixo a figura 10 com uma prévia do *Product Backlog* gerado no experimento, o *Backlog* completo está disponível em anexo ao fim do trabalho:

Requisitos
R1 - Nota fiscal eletrônica
R2- Encontrar prestador de serviço mais próximo
R3- Sistema de feedback
R4- Opções de pesquisa e formas de apresentação dos resultados
R5- Serviços da empresa e preços dentro da plataforma
R6 - Manter serviço
R7 - Manter cliente
R8 - Informações sobre a empresa dentro da plataforma
R9 - Meios de compra
R10 - Negociação de preço
R11 - Agendar serviço
R12 - Verificar e validar forma de pagamento
R13 - Indicação de serviços
R14 - software deve ser capaz de gerar descontos caso o cliente seja um usuário que utilize a plataforma frequentemente
R15 - O software deve ser capaz de agendar horários que atendam a disponibilidade dos clientes e da empresa
R16 - O software deve ser capaz de gerar relatórios mensais para a gerência da empresa

Figura 11 – Prévia *Backlog* do experimento

Fonte: Autoria própria

6. Conclusões

Com base nos dados disponibilizados e na análise dos requisitos é possível concluir que o processo atende restritamente a proposta, mas com quantidade limitada de requisitos, ou seja a hipótese H1: que é a de que o processo proposto atende a proposta e gerencia os requisitos é parcialmente verdadeira. Durante a execução do experimento foi executado todos os passos do processo r4c, eles têm aderência, mas o volume de geração de requisitos pode tornar esse processo inviável, uma vez que ele está sendo executado de maneira manual.

Conclui-se ainda que a etapa 3 (filtro) do modelo proposto r4c pode ser melhorada, visando que neste trabalho essa etapa foi executada manualmente, apenas como teste e o resultado foi desfavorável, visando que a análise dos requisitos e montagem da matriz manualmente levou em torno de 5 horas e 15 minutos para concretizar-se, ou seja, quanto maior o volume de dados, maior será o tempo para análise de requisitos.

Portanto, pode-se dizer que o processo proposto atende restritamente a proposta.

6.1 Trabalhos futuros

Um possível trabalho futuro a ser desenvolvido seria propor mecanismos automáticos ou semiautomáticos de analisar requisitos visando a melhoria da etapa 3 do modelo proposto r4c, ou seja, a automatização do filtro e análise dos requisitos. Automatizando esse processo seria possível agilizar a etapa de análise dos requisitos. Conforme visto que no teste experimental e no próprio experimento levou-se um tempo elevado para tal etapa.

Referências

ALAA, G. **A multi-faceted Roadmap of Requirements Traceability Types Adoption in SCRUM** : An Empirical Study. p. 1–9, 2014.

ANEGEKUH, L.; SUN, L.; IFEACHOR, E. **A Screening Methodology for Crowdsourcing Video QoE Evaluation**. *IEEE Globecom conference, Austin, TX USA*, p. 1152–1157, 2014.

ARRUDA, N. D. S. **Engenharia de Requisitos- como Previnir e Reduzir Riscos**. 2011.

BERCZUK, S. **Back to Basics : The Role of Agile Principles in Success with an Distributed Scrum Team All Development is Local**. 2007.

BITENCOURT, C.; MELO, B. DE; BERNARDES, J. R. **Crowdsourcing como uma Ferramenta à Inovação Estratégica Empresarial**. v. 1, n. 1, p. 13–24, 2014.

BOOTLA, P. *et al.* **Necessary Skills and Attitudes for Development Team Members in Scrum** : p. 184–189, 2015.

CAMPOS, P. **Engenharia de Requisitos**. *Cee.Uma.Pt*, p. 1–33, 2013. Disponível em: <http://cee.uma.pt/edu/er/slides/ER-ConceitosBase.pdf>.

CARVALHO, B.; MELLO, C. **Revisão, análise e classificação da literatura sobre o método de desenvolvimento de produtos ágil Scrum**. *Anais do XII Simpósio de Administração da Produção, Logística e Operações Internacionais - SIMPOI*, p. 1–16, 2009.

CASTTELS, C. 2006. ***A Sociedade em Rede***.

CHUENE, D.; MTSWENI, J. **The Adoption of Crowdsourcing Platforms in South Africa**. p. 1–9, 2015.

GAUR, V.; SONI, A.; BEDI, P. **An agent-oriented approach to requirements engineering**. *2010 IEEE 2nd International Advance Computing Conference, IACC 2010*, p. 449–454, 2010.

GIOVANELLA, T. **Levantamento de Requisito**s. *Slideshere.Net*, p. 1–33, 2013. Disponível em: <http://www.slideshare.net/tgiovanella1/aula-1-levantamento-de-requisitos>.

HOSSAIN, M. **Users ’ Motivation to Participate in Online Crowdsourcing Platforms**. p. 21–22, 2012.

HOSSEINI, M. *et al.* **Configuring crowdsourcing for requirements elicitation**. *Proceedings - International Conference on Research Challenges in Information Science*, v. 2015–June, n. June, p. 133–138, 2015.

HOSSEINI, M. *et al.* **Model**. 2014.

HOSSEINI, M. *et al.* **Towards Crowdsourcing for Requirements Engineering Crowdsourcing for Requirements Engineering**. 2014.

KNIBERG, H. **Scrum e XP direto das Trincheiras**. 2007.

L’ERARIO, 2009. *Alexandre L ’ Erario **M3DS : um modelo de dinâmica de desenvolvimento distribuído de software** São Paulo Alexandre L ’ Erario M3DS : um modelo de dinâmica de desenvolvimento distribuído de software São Paulo*.

MELOROSE, J.; PERROY, R.; CAREAS, S. **A engenharia de requisitos e o desenvolvimento dirigido a modelos: uma revisão sistemática**. *Statewide Agricultural Land Use Baseline 2015*, v. 1, 2015.

MUNDRA, A.; MISRA, S.; DHAWALE, C. A. **Practical Scrum-Scrum Team : Way to Produce Successful and Quality Software**. 2013.

PENG, X.; BABAR, M. A.; EBERT, C. **Collaborative Software Development Platforms for Crowdsourcing**. n. 740, 2014.

RUOYU, LU, 2011. **On Operating Mechanism of Crowdsourcing Service Innovation**.

SANTOS, R. DOS. **Utilizando Experimentação para Apoiar a Pesquisa em Educação em Engenharia de Software no Brasil**. *… em Engenharia de …*, 2008. Disponível em: <http://fees.inf.puc-rio.br/FEESArtigos/artigos/artigos_FEES08/santos.pdf>.

SLACK, N. 2001. *626546137.Capítulo I-SlackNigel-**ADMINISTRAÇÃODAPRODUÇÃO**.pdf*. [S.l: s.n.], [S.d.].

SOARES, R.; CABRAL, T.; ALENCAR, F. **Gerenciamento de Requisitos em Scrum baseado em Test Driven Development**. *CEUR Workshop Proceedings*, v. 1005, p. 139–164, 2013.

SOMMERVILLE, I. ***Software Engineering***. [S.l: s.n.], 2010.

TECNOLOGIA, P.; UNISANTA, D. I.; GUSTAVO, S. Artigo 5_Prof Marcos. p. 58–71.

TOBERGTE, D. R.; CURTIS, S. **Especificação e implementação de uma ferramenta para elicitação de requisitos de software baseada na teoria da atividade**. *Journal of Chemical Information and Modeling*, v. 53, n. 9, p. 1689–1699, 2013.

TRAVASSOS, G.; GUROV, D.; AMARAL, E. **Introdução à Engenharia de Software Experimental**. *Relatório Técnico ES59002Abril Programa de Engenharia de Sistemas e Computação COPPEUFRJ*, p. 52, 2002.

VLAANDEREN, K. *et al.* **The Agile Requirements Refinery : Applying SCRUM Principles to Software Product Management**. 2010.

HOWE, J. **The Rise of Crowdsourcing. Wired magazine**, n. 14, p. 1–5, 2006.

WHITE, S. A. Using BPMN to Model a BPEL Process. IBM, New York, 2006. Disponível em: Acesso em: 13 set. 2016.