UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA

ESPECIALIZAÇÃO EM QUALIDADE DE SOFTWARE

Daline Zat

# A VALIDAÇÃO DE REQUISITOS COM FOCO NA MELHORIA EM PROJETOS DE SOFTWARE

Porto Alegre

2016

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA

ESPECIALIZAÇÃO EM QUALIDADE DE SOFTWARE

Daline Zat

# A VALIDAÇÃO DE REQUISITOS COM FOCO NA MELHORIA EM PROJETOS DE SOFTWARE

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Especialista em Qualidade de Software, pelo curso de Pós-Graduação Lato Sensu em Qualidade de Software da Universidade do Vale do Rio dos Sinos – UNISINOS.
Orientador: Prof. Dr. Sílvio César Cazella

Porto Alegre

2016

# A Validação de Requisitos com Foco na Melhoria em Projetos de Software

**Daline Zat**[1]

[1]Universidade do Vale do Rio dos Sinos (Unisinos) – São Leopoldo – RS – Brasil

dalinezat@gmail.com

***Abstract.*** *This paper introduces the result of an action research, applied in an organization of technology information, in which it applied the validation of software requirements using a checklist. The aim of this study was to develop a tool to improve software projects using a checklist focused on the organization's needs, besides, to implement the validation of requirements in the development process and finally analyze and present the results obtained after implantation the checklist.*

***Resumo.*** *Este artigo apresenta o resultado de uma pesquisa-ação, realizado em uma organização do setor de tecnologia da informação, na qual se aplicou a validação dos requisitos de software através de um checklist. O objetivo deste trabalho foi desenvolver uma ferramenta para a melhoria de projetos de software utilizando um checklist focado nas necessidades da organização, além de, implantar a atividade de validação de requisitos no processo de desenvolvimento e, por fim, analisar e apresentar os resultados obtidos, após a implantação do checklist.*

## 1. Introdução

O software tornou-se um diferencial competitivo para as organizações com aplicação em diversas áreas, no entanto, desenvolve-lo é uma tarefa complexa devido à abstração e a intangibilidade dos requisitos, sendo possível mais de uma solução para o mesmo cenário, isto é, cada sistema exige uma abordagem com processos, métodos e ferramentas diferentes [Sommerville 2011].

A engenharia de software auxilia os profissionais nesta atividade, pois o seu foco envolve todos os aspectos da produção do software desde as fases inicias de especificação até a manutenção, tendo quatro atividades principais: especificação, desenvolvimento, validação e evolução. A especificação estabelece as funcionalidades requeridas pelo sistema e as restrições sobre a operação e o desenvolvimento do sistema [Pressman 2011; Sommerville 2011]. Segundo Sommerville (2011), esta atividade é denominada de engenharia de requisito, sendo responsável por envolver todas as atividades para a criação e manutenção dos requisitos do sistema.

Os requisitos de um software são a base para o desenvolvimento de um sistema, pois eles são as descrições dos principais recursos de um produto de software, ou seja, são as funcionalidades e restrições que o software deve possuir. É a principal atividade da engenharia de requisitos [Souza 2011].

A engenharia de requisitos de software possui o seu processo dividido em quatro atividades principais: estudo de viabilidade, elicitação e análise, especificação e

validação de requisitos. O estudo de viabilidade identifica a possibilidade de construir o sistema do ponto de vista do negócio visando à rentabilidade; a elicitação e análise identificam as necessidades do sistema que será construído; a especificação descreve as informações obtidas na etapa de elicitação; e a validação verifica os requisitos quanto ao realismo, consistência, completude, validade e verificabilidade [Sommerville 2003; Sommerville 2011].

A elicitação dos requisitos é considerada uma das fases mais complexas e critica dentro da engenharia de requisitos, pois ela é a base para todas as atividades posteriores e a sua elaboração influencia no produto desenvolvido, sendo um indicador importante para o sucesso ou fracasso dos projetos de software [Barbosa et al 2009]. Nesta atividade os analistas trabalham junto com clientes e usuários para entender os problemas que devem ser resolvidos, as funcionalidades que o sistema deve conter, os requisitos de desempenho, as restrições de hardware, entre outras necessidades que irão compor o sistema.

Requisitos equivocadamente identificados ou inadequadamente interpretados geram retrabalho, custo e prazos extras nos projetos de software, isto foi citado por Barbosa et al (2009) e vivenciado pela pesquisadora na organização em que atua. Como exemplo, pode-se citar os requisitos conflitantes que podem ocasionar atrasos nos projetos devido uma incompatibilidade entre a especificação e o sistema existente, necessitando uma nova analise de negócio. Consequentemente o tempo desta análise impactará no prazo de entrega do projeto.

Para amenizar as dificuldades desta atividade existem diversas técnicas de elicitação, entretanto, somente elas não são o suficiente para garantir a qualidade dos requisitos, por isso a atividade de validação é tão importante no processo de desenvolvimento, pois busca identificar inconsistências existentes nos requisitos visando melhorar os projetos de software [Sommerville 2011].

Quanto mais cedo forem identificadas as inconsistências nos requisitos, menor será o custo e o tempo para ajustar o software [Sommerville 2011]. A razão para isto é que, uma inconsistência identificada na especificação possui um tempo de correção relativamente simples, porém se um erro é encontrado na arquitetura em decorrência de uma especificação, deve-se corrigir não só arquitetura, mas também a especificação, no entanto, se o problema é identificado em produção deve-se fazer a correção em todas as etapas da construção do software gerando assim um alto custo de correção.

Este trabalho de pesquisa tem como motivação melhorar a qualidade dos projetos software através da inclusão da atividade de validação de requisitos no processo de desenvolvimento de uma organização especifica, com o intuito de identificar o quanto antes as possíveis inconsistências existentes nos requisitos, melhorando assim as entregas dos projetos. A pesquisa busca responder a seguinte questão de pesquisa: A validação de requisitos propicia melhora na qualidade em projetos de software?

Desta forma, o objetivo geral desta pesquisa é conceber um *checklist* para atender a atividade de validação de requisitos e avaliar as entregas dos projetos de software. Visando atingir o objetivo geral, foram definidos os seguintes objetivos específicos: identificar os pontos críticos dos requisitos na organização, criar e implantar o *checklist* e por fim analisar os resultados obtidos através da inclusão da atividade de

validação de requisitos no processo de desenvolvimento da organização. A pesquisa foi aplicada em uma equipe de desenvolvimento de software, comparando os resultados entre o pré e a pós implantação do *checklist*, em uma organização do setor de tecnologia da informação (TI).

Este artigo está organizado em seis seções, incluindo Introdução e Conclusão. A Seção 2 apresenta o referencial teórico utilizado no desenvolvimento do trabalho descrevendo os conceitos de validação de requisitos, inspeção de software e técnicas de revisão baseadas em leitura. A Seção 3 apresenta a metodologia utilizada na construção do estudo. A Seção 4 apresenta o estudo elaborado na pesquisa. Na Seção 5 é descrita a análise e os resultados da pesquisa.

## 2. Referencial Teórico

Nesta seção será abordado o conceito de validação de requisitos e os seus critérios de verificação, juntamente com a inspeção de software e algumas das técnicas de revisão baseada em leitura, além dos princípios utilizados no desenvolvimento do *checklist* e trabalhos relacionados com o estudo.

### 2.1. Validação de Requisitos

A validação de requisitos consiste em verificar se o requisito está de acordo com as necessidades do cliente, além de identificar problemas nas definições através de cinco de critérios de verificação. Sommerville (2011) definiu os critérios da seguinte forma:

- **Validade:** Busca identificar junto aos *stakeholders* a real necessidade de desenvolver determinada funcionalidade no software, isto é, no decorrer da analise após um maior detalhamento dos requisitos podem surgir novas funções ou modificações nas funções existentes. Essas modificações precisam ser aprovadas e validadas pelos *stakeholders*, por isso, o contato com os usuários é fundamental para este critério de avaliação, principalmente quando há diversos *stakeholders* envolvidos na criação do software.
- **Consistência:** Observa há existência de informações contraditórias nos requisitos, ou seja, um requisito não pode conflitar com outro e uma restrição não pode sobrepor outra dentro do mesmo requisito.
- **Completude:** Verifica se todas as funções e restrições da funcionalidade estão descritas de forma clara e objetiva com um nível de detalhamento onde todos possam compreender o que se espera com o desenvolvimento do requisito, além de observar todos os requisitos já definidos que irão compor o software.
- **Realismo:** Dentre as tecnologias utilizadas na construção de um software especifico, os requisitos são verificados com o intuito de assegurar que eles podem ser desenvolvidos quanto à tecnologia escolhida, orçamento e cronograma.
- **Verificabilidade:** Observa se os requisitos do sistema podem ser verificados, isto é, se pode ser escrito um conjunto de testes demostrando que o sistema atenderá a cada requisito especificado.

Os requisitos podem ser validados por diversas técnicas, entre elas pode-se citar: revisão de requisitos, prototipação e geração de casos de teste, sendo aplicadas de forma individual ou em conjunto. A revisão de requisitos analisa sistematicamente os requisitos buscando identificar erros e inconsistências. A prototipação valida os requisitos apresentando um modelo executável do sistema para os usuários finais, onde eles verificam se o modelo atende as suas necessidades. A geração de casos de teste parte da afirmação que os requisitos devem ser testáveis, isto é, os testes são criados e utilizados no processo de validação na busca de possíveis problemas nos requisitos [Sommerville 2011].

Para Pressman (2011), a revisão de requisitos deve responder algumas questões, tais como:

- Cada um dos requisitos é consistente com os objetivos globais para o sistema ou produto?
- Todos os requisitos foram especificados no nível de abstração apropriado?
- O requisito é realmente necessário ou representa um recurso adicional que talvez não seja essencial para o objetivo do sistema?
- Cada um dos requisitos é limitado e sem ambiguidades?
- Algum dos requisitos conflita com outros requisitos?
- Cada um dos requisitos é atingível no ambiente técnico que irá abrigar o sistema ou produto?
- Cada um dos requisitos pode ser testado, uma vez implementado?

Responder essas e outras questões sobre os requisitos ajuda a garantir que as necessidades dos clientes serão atendidas, além de fornecer uma base consistente para o projeto. No entanto, os problemas envolvendo os requisitos e a sua validação não devem ser subestimados, pois mostrar para os usuários que um conjunto de requisitos atende as suas necessidades não é algo trivial devido à abstração e a intangibilidade do software [Pressman 2011; Sommerville 2011]. Então, especificar os requisitos de uma forma clara e objetiva para que todas as pessoas envolvidas no projeto compreendam as necessidades do software, é uma atividade complexa e suscetível a erros, por isso os requisitos devem ser inspecionados através de técnicas de revisão.

## 2.2. Inspeção de Software

A inspeção de software foi introduzida na área computacional por Fagan (1976), sendo um tipo particular de revisão que contribui para garantir a qualidade do produto de software. O seu objetivo é encontrar defeitos existentes no artefato inspecionado. Estes defeitos podem surgir ao decorrer de qualquer uma das etapas do desenvolvimento do software e identificados por uma inspeção [Coelho et al 2013].

A definição do processo de inspeção proposta por Fagan (1976) foi questionada por diversos trabalhos na busca de aprimorar o seu desempenho e estrutura. Pode-se citar como exemplo, Votta (1993) com o seguinte argumento, que ao evitar reuniões de inspeção de forma síncrona, custos e conflitos com a alocação de pessoal podem ser reduzidos sem prejudicar a eficiência da inspeção. A partir deste e outros estudos, Sauer

(2000) propôs uma reestruturação no processo de inspeção, sendo ela composta por seis etapas apresentadas abaixo [Kalinowski et al 2004; Coelho et al 2013]:

- **Planejamento:** Define o contexto da inspeção (descrição da inspeção, técnica utilizada na detecção de defeitos, documentos a serem inspecionados, autor do documento, entre outros), seleciona os inspetores e distribui o material a ser inspecionado. Toda a organização desta etapa é desempenhada por um usuário cujo seu papel é chamando de moderador da inspeção.
- **Detecção de Defeitos:** Consiste na execução da inspeção dos documentos entregues aos inspetores selecionados pelo moderador, onde é realizada a tarefa de encontrar possíveis defeitos nos artefatos e assim produzir uma lista de discrepâncias ao fim da inspeção.
- **Coleção de Defeitos:** O moderador agrupa as listas de discrepâncias dos inspetores, eliminando os defeitos repetidos, ou seja, mantendo só um registro para cada defeito.
- **Discriminação de Defeitos:** Visa classificar os itens da lista de discrepância como defeito de fato ou falso positivo, ou seja, o moderador, o autor do documento e os inspetores discutem as discrepâncias encontradas na inspeção e após a classificação eliminam os falsos positivos, permanecendo assim só os defeitos reais que serão registrados em uma lista de defeitos.
- **Retrabalho:** Os defeitos pertencentes à lista são corrigidos pelo autor do documento e, ao fim das correções, é gerado um relatório de correções de defeitos.
- **Continuação:** O moderador decide se uma nova inspeção deve ser realizada ou não, com base no relatório produzido na correção dos defeitos.

A inspeção pode ser aplicada em todos os artefatos de software através de técnicas que possuem grande influência no resultado da revisão. Uma solução utilizada na revisão de software é a técnica baseada em leitura que contribui para o aumento da compressão dos artefatos [Coelho et al 2013].

### 2.3. Técnicas de Revisão Baseada em Leitura

As técnicas de leitura consistem em uma série de passos que auxiliam os inspetores na análise individual dos artefatos, buscando identificar informações que permitam o entendimento para a execução de uma atividade e por consequência, na identificação de defeitos [Basili et al 1996]. O termo leitura foi escolhido, pois enfatiza a similaridade com o processo mental que as pessoas utilizam, quando visam entender o significado de um texto [Mafra et al 2005]. Shull et al (2000) identificou três componentes importantes na definição da técnica de leitura:

- **Uma série de passos:** a técnica deve fornecer orientações aos inspetores sobre como a leitura deve ser conduzida. O tipo de orientação varia conforme o objetivo da tarefa, podendo ser desde um procedimento passo a passo a um conjunto de perguntas formuladas para manter a concentração na leitura.

- **Análise individual:** a técnica deve se preocupar com o processo de compreensão individual, isto é, como cada inspetor interpreta o documento que está revisando. Esta técnica pode ser inserida em métodos que necessitem de trabalhos em equipe, como por exemplo, uma reunião para discutir a inspeção. No entanto, é importante ressaltar que a compreensão de certos aspectos do artefato deve ser feita individualmente.
- **Entendimento necessário para executar uma tarefa específica:** a técnica deve fornecer apoio na obtenção da compreensão de alguns aspectos (ou todos) do documento inspecionado.

Além de possuir os três componentes citados acima as técnicas de leitura necessitam ser dependentes do contexto, bem definidas e orientas a objetivos [Basili 1997]. Dentre as diversas técnicas de leitura existentes podem-se citar algumas como:

- ***Defect-Based Reading (DBR)* ou Defeitos:** é focada na detecção de tipos específicos de defeitos durante a inspeção do documento de requisitos, tais como: inconsistência de tipos de dados, funcionalidades incorretas e ambiguidades. Para isto, é definido um conjunto de cenários que apoiam o inspetor na revisão [Thelin et al 2003].
- ***Scenario-Based Reading (SBR)* ou Cenários:** utiliza como base um conjunto de instruções e diretrizes denominadas de cenários que fornecem orientações personalizadas para os inspetores sobre como detectar os defeitos [Coelho et al 2013].
- ***Perspective-Based Reading (PBR)* ou Perspectivas:** baseia-se na ideia que a inspeção dos artefatos deve ocorrer a partir da perspectiva de diferentes *stakeholders*, isto é, elabora-se cenários que contemplem o ponto de vista dos usuários do sistema, do projetista e do testador aprofundando assim a cobertura da inspeção [Coelho et al 2013].
- ***Traceability-Based Reading (TBR)* ou Rastreabilidade:** é uma técnica de leitura adaptada para orientar a verificação da rastreabilidade entre requisitos e artefatos de projetos orientados a objeto. O método é dividido em leitura vertical, onde é verificado se o projeto corresponde aos requisitos e em leitura horizontal que confronta diferentes artefatos de *design* como, por exemplo: diagrama de classe, diagrama de estados, entre outros [Thelin et al 2003].
- ***Usage-Based Reading (UBR)* ou Usabilidade:** busca direcionar o esforço da leitura em detectar os defeitos considerados mais críticos do objeto inspecionado. São considerados os defeitos de impacto mais negativo sobre a percepção de qualidade do usuário. O método consiste em priorizar os casos de uso do sistema na ordem que um usuário ou um grupo de usuários acha mais importante que sejam desenvolvidos [Thelin et al 2003].
- ***Checklist-Based Reading (CBR)* ou *Checklist*:** consiste na utilização de um formulário com uma série de perguntas sobre o artefato que será inspecionado, guiando assim a leitura do inspetor na revisão. [Thelin et al 2003; Coelho et al 2013].

Na literatura, a técnica de inspeção denominada de *checklist* é classificada de forma distinta por diferentes autores. Para Petersson (2002), Sabaliauskaite et al (2003) e Thelin et al (2003) é considerada uma técnica baseada em leitura, no entanto, menos formal que as outras técnicas de leitura, por não ser repetível e depender fortemente do revisor [Laitenberger et al 2001]. Já Porter et al (1995) classificam como uma técnica de leitura não sistemática e Shull et al (2000) classificam como uma técnica de leitura parcialmente sistemática e não focada. Ao longo deste artigo, quando citado inspeção *Checklist* ou revisão baseada em leitura *Checklist,* refere-se à mesma técnica, pois se entende que fazem parte do mesmo contexto, seguindo os autores Petersson (2002), Sabaliauskaite et al (2003), Thelin et al (2003) e Laitenberger et al (2001).

### 2.4. *Checklist*

A inspeção baseada em *Checklist* utiliza um questionário, com diversas perguntas sobre o assunto do artefato que será inspecionado e devem ser respondidas pelos inspetores durante a revisão. Não existe um padrão para a configuração do questionário e, frequentemente, ele é composto por questões cuja resposta é sim ou não, possuindo uma grande flexibilidade na sua elaboração [Laitenberger et al 2001].

O objetivo do *Checklist* é fornecer aos inspetores dicas e recomendações através das perguntas, com o intuito de apoiar a detecção de defeitos, porém, esta técnica possui alguns pontos fracos. O primeiro ponto levantado é sobre as questões pertencentes à lista de verificação, que muitas vezes são genéricas como, por exemplo, "O artefato inspecionado está correto?". Essa questão fornece uma visão geral do que o inspetor deve verificar, mas, não diz de forma precisa, qual o atributo de qualidade está tratando. Desta forma acaba fornecendo pouco apoio para o inspetor entender o artefato inspecionado. O segundo ponto fraco é a ausência de instruções sobre como usar o questionário, ou seja, nem sempre está claro quando e com base em qual informação o inspetor responde cada questão da lista. A terceira fraqueza identificada está relacionada às questões da lista de verificação, que frequentemente são limitadas a detecção de defeitos já conhecidos, isto é, as questões foram baseadas em defeitos encontrados no passado. Assim os inspetores não podem se focar em novos tipos de defeitos [Laitenberger 2002; Akinola 2009].

Para resolver algumas das dificuldades apresentadas, pode-se desenvolver um *Checklist* de acordo com os seguintes princípios [Laitenberger 2002]:

- O tamanho da lista de verificação preferencialmente não deve ultrapassar uma página;
- As questões da lista devem ser escritas de uma forma precisa ou o mais preciso possível;
- A lista de verificação deve ser estruturada da forma onde o atributo de qualidade seja claro para o inspetor e a questão fornecer dicas sobre como garantir a qualidade do atributo.

Os princípios citados acima aprimoram o desenvolvimento do *Checklist*, visando aumentar a detecção de defeitos nos artefatos, no entanto, com o decorrer das inspeções deve-se revisar o *Checklist* e atualiza-lo caso seja necessário com o intuito de aperfeiçoar as próximas inspeções.

### 2.5. Trabalhos Relacionados

Na busca pela garantia da qualidade do produto final de software encontram-se diversas pesquisas que visam desenvolver técnicas de inspeção ou aprimorar as existentes para auxiliar o inspetor na revisão.

Mafra e Travassos (2006) propuseram uma nova técnica de revisão de requisitos chamada OO-PRB (*Object Oriented - Perspective Based Reading*). Está técnica tem como principal objetivo a detecção de defeitos críticos em documentos de requisitos de software descritos em linguagem natural através da construção de modelos iniciais de projetos orientados a objeto com a perspectiva do projetista de software.

Mello et al (2011), apresentaram uma técnica configurável de inspeção individual baseada em *checklist,* desenvolvida para apoiar a identificação de defeitos em diagramas de atividades que compõem a especificação de requisitos em projetos de software, denominada ActCheck. Está técnica é composta por dois *checklists* (A e B) e permite que os itens de avaliação sejam aplicados ou não conforme o contexto de cada projeto.

Macedo e Vilain (2014) abordam a inclusão de requisitos de qualidade durante a especificação de requisitos visando, a transparência do software através de 19 ações elencadas como essenciais para atender os requisitos de qualidade. Essas 19 ações compõem o checkTrans-E utilizado na fase de especificação dos requisitos onde devem ser definidos os requisitos de transparência e em seguida, descreve-se os testes de aceitação utilizando os requisitos de transparência.

## 3. Metodologia

Nesta seção é abordado o delineamento da pesquisa, a definição da população e amostra que irá compor o trabalho, juntamente com as técnicas de coleta e analise dos dados, além de apresentar as limitações do estudo.

### 3.1. Delineamento da Pesquisa

A pesquisa será composta por duas etapas: revisão bibliográfica e uma pesquisa-ação, sendo de natureza aplicada com uma abordagem quantitativa de caráter exploratório e de corte-transversal. A revisão bibliográfica contribuirá para o embasamento teórico-metodológico utilizado na construção do estudo abordando principalmente os conceitos de Validação de Requisitos e Técnica de Revisão Baseada em Leitura *Checklist* [Martins 2008].

Segundo Gil (2010), a escolha pelo método de pesquisa pesquisa-ação é preferencial quando o pesquisador visa diagnosticar um problema específico em uma situação específica e está envolvido no contexto de modo cooperativo e participativo, buscando alcançar algum resultado prático, sendo ele não generalizável.

O estudo analisará quantitativamente a melhora no processo de desenvolvimento da organização, após a implantação de um *checklist* em uma planilha eletrônica para a verificação e validação dos requisitos. A análise quantitativa será através do indicador que mede a quantidade de pontos planejados e entregues em cada projeto disponibilizado pela organização.

### 3.2. Unidade Análise

A unidade de análise escolhida para o estudo é uma equipe de desenvolvimento de uma empresa de TI, onde a pesquisadora atua como Projetista e Desenvolvedora de Software. A equipe é formada por três profissionais, sendo um Analista de Negócio, um Projetista e um Desenvolvedor. A escolha por esta equipe ocorreu pela técnica de amostragem por conveniência ou acessibilidade, pois a pesquisadora possui grande facilidade de acesso aos documentos e pessoas envolvidas no estudo [Vergara 2007]. Desta forma, os resultados interpretados referem-se à unidade de análise adotada neste estudo, não sendo possível generalizá-los.

### 3.3. Coleta de Dados

A coleta de dados abrangerá os projetos de software da equipe de folha de pagamento do produto novo, que foram executados de janeiro a agosto do ano de 2016. Será coletado o resultado do indicador utilizado na organização, que mede a quantidade de pontos planejados versus os pontos entregues em cada projeto, sendo que cada projeto tem duração de um mês. Parte da coleta será realizada pelo uso do instrumento de coleta questionário, conforme o apêndice A e detalha-se na seção 4.3.

### 3.4. Análise de Dados

Após a coleta dos dados foi realizada a análise de forma quantitativa, observando especificamente, o indicador de pontos planejados versus pontos entregues e comparando o seu desempenho antes e depois da aplicação do *Checklist* proposto nesse estudo. Segundo Wainer (2007), uma analise quantitativa baseia-se numa visão dita positivista onde:

- As variáveis a serem observadas são consideradas objetivas.
- Não há desacordo do que é melhor e o que é pior para os valores das variáveis.
- Medições numéricas são consideradas mais ricas que descrições verbais, pois elas se adequam à manipulação estatística.

O indicador possui a meta de entregar no mínimo 80 por cento dos pontos planejados em cada projeto. Visando este objetivo será analisado se após a implantação da atividade de Validação a meta estipulada pela organização foi alcançada com maior frequência, buscando assim estabilizar o processo.

### 3.5. Limitações do Método de Estudo

O estudo contempla uma única equipe do setor de desenvolvimento e foi aplicado em três projetos, devido à limitação de tempo para a realização do levantamento dos dados, a análise dos resultados e conclusão deste trabalho. Desta forma os resultados não são generalizáveis, inclusive, no contexto da organização objeto desse estudo.

## 4. Estudo

Nesta seção, o contexto de melhoria é contextualizado, iniciando pela apresentação da unidade de análise, explicando o processo de desenvolvimento atual da organização, passando para a etapa de Coleta Preliminar de Dados responsável por identificar o cenário atual dos requisitos na organização, utilizada como base para a concepção do

*Checklist*. A subseção 4.4 apresenta o detalhamento do desenvolvimento do *Checklist,* na subseção 4.5 explica-se a proposta do estudo e na subseção 4.6 apresenta-se a aplicação do *Checklist*.

## 4.1. Contextualização da Unidade de Análise

O trabalho foi realizado em uma empresa de tecnologia da informação, que desenvolve soluções para a Gestão de Recursos Humanos desde 1986, além de fornecer um portfólio de serviços que engloba consultoria de processos, assessoria operacional e técnica, suporte operacional e técnico e treinamentos. Atualmente, a empresa atende cerca de 900 clientes de diversos segmentos (indústria metalomecânica, indústria moveleira, indústria calçadista, indústria da construção, varejo, setor de educação, setor da saúde, escritórios de contabilidade, entre outros), distribuídos em todo o território nacional.

A empresa possui três unidades, sendo duas localizadas no estado do Rio Grande do Sul e uma no estado de São Paulo e conta com um total de 103 colaboradores. O setor de desenvolvimento conta com equipes distribuídas nas unidades de Caxias do Sul/RS e Porto Alegre/RS, sendo que a maior concentração de pessoas encontra-se na unidade de Caxias do Sul com um total de 17 pessoas, divididas entre Analistas de Negócio, Projetistas de Software e Desenvolvedores. A unidade de Porto Alegre possui um Projetista de Software e um Desenvolvedor.

As equipes são formadas, no máximo, por quatro pessoas e desenvolvem módulos específicos do Sistema de Gestão de Recursos Humanos. Desde 2011, a organização iniciou o desenvolvimento de um novo sistema de Gestão de Recursos Humanos somente na Web, dividindo o setor de desenvolvimento em duas equipes macro. Uma equipe trabalha na manutenção e atualização do sistema atual e a outra equipe desenvolve o sistema Web, baseado no atual, incluindo novas funcionalidades. As equipes macro são subdivididas por módulos específicos dentro de cada sistema ou produto.

O estudo foi aplicado em uma equipe de desenvolvimento responsável pela construção do módulo de folha de pagamento do produto Web, sendo que dois colaboradores estão alocados na unidade de Caxias do Sul e um na unidade de Porto Alegre.

## 4.2. Processo de Desenvolvimento

O processo de desenvolvimento da organização baseia-se na metodologia de desenvolvimento Ágil ,seguindo como guia o modelo de gerenciamento do *Scrum*. Cada projeto ou s*print* tem duração de quatro semanas. No inicio da *sprint* é realizada a reunião de planejamento com a equipe, onde o analista de negócio apresenta os *Product Backlogs* (requisitos) de maior relevância, que poderão ser trabalhados durante o próximo projeto.

A cada requisito apresentado, a equipe estima o esforço em pontos para desenvolver o requisito através do método *Poker Planing*. Após a definição do tamanho dos requisitos inicia-se a divisão de tarefas. Cada requisito possui três tarefas vinculadas, sendo elas: Análise, Desenvolvimento e Teste, e essas tarefas são atribuídas aos seus responsáveis e devem ser executadas nas próximas quatro semanas.

Durante o planejamento, o gerente de projetos controla a quantidade de requisitos, que devem entrar no projeto pela média de pontos que a equipe costuma entregar. No fim da *sprint* é realizada a reunião de revisão da *sprint,* onde são apresentados os resultados e as dificuldades encontradas na execução do projeto. A Figura 1 representa o processo de desenvolvimento adotado pela organização.

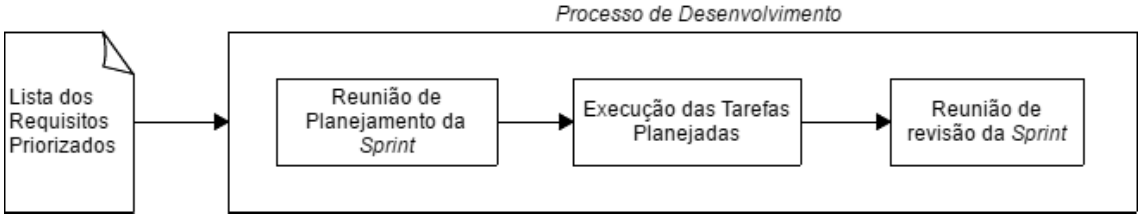


**Figura 1. Fluxo atual de trabalho**

Com o decorrer das s*prints,* na reunião de revisão identificou-se a existência de problemas recorrentes envolvendo a entrega dos requisitos, tais como:

- Mudança na regra de negócio durante o desenvolvimento da funcionalidade devido à falta de informações no requisito;
- Conflito entre as funcionalidades existentes no sistema e a nova funcionalidade requerida;
- Incompatibilidade de dados e regras entre os dois sistemas da organização;
- Estimativa de tamanho errada do requisito;
- Impossibilidade de conclusão do requisito devido à necessidade de implementações na arquitetura do sistema novo;

Um requisito só pode ser entregue quando estiver concluído, isto é, quando todas as tarefas e *Test Cases* (itens que possuem as funções e restrições do requisito) vinculados a ele estiverem como *Done,* caso contrário, o requisito não pode ser entregue na *sprint* atual e passa para a próxima *sprint*.

### 4.3. Coleta Preliminar de Dados

A coleta preliminar de dados buscou identificar o cenário atual da organização quanto aos requisitos, isto é, identificar os pontos fortes e fracos dos requisitos elicitados e especificados. Para isto foi desenvolvido um questionário *on-line* como instrumento de pesquisa, por meio da ferramenta gratuita *Google Drive*, contendo 20 questões objetivas e 1 aberta, que forneceu espaço para sugestões de melhoria quanto aos requisitos, sendo de preenchimento opcional, conforme apêndice A.

O questionário foi elaborado para receber respostas padronizadas, tendo como base a escala *Likert*, contendo 5 pontos: 0 – Discordo Totalmente, 1 – Discordo Parcialmente, 2 – Neutro, 3 – Concordo Parcialmente e 4 – Concordo Totalmente. Organizou-se o questionário dividindo-o em duas etapas: a primeira contempla questões referentes ao perfil do respondente e a segunda contem questões para o levantamento dos pontos críticos sobre os requisitos. As questões foram escritas de forma afirmativa com base nos critérios de verificação citados por Sommerville (2011).

Após concluir o desenvolvimento do questionário, o mesmo passou pela validação do gerente do setor de desenvolvimento da organização, a fim de aprovação, identificação de ajustes e melhorias. Realizou-se a inclusão de uma questão e ajustes nas descrições de algumas afirmativas, com o objetivo de deixá-las mais claras.

Ao término das correções e aprovação do gerente foi disponibilizado o questionário para os Desenvolvedores e Projetistas de Software do setor, através do correio eletrônico oficial da organização. A pesquisa ficou disponível para os participantes entre os dias 28 de abril e 03 de maio de 2016.

A análise dos dados agrupou as questões dentro dos critérios de verificação, citados no referencial teórico e identificou itens que ficaram a desejar nos critérios de: Consistência, Completude e Verificabilidade, demostrando assim a necessidade de implantar a atividade de validação de requisitos na organização e também comprovou a percepção da pesquisadora quanto à deficiência dos requisitos em determinados critérios observados durante as reuniões de revisão das *sprints*.

### 4.4. *Checklist*

A partir das conclusões obtidas na análise da coleta preliminar dos dados iniciou-se o desenvolvimento do *checklist* de validação dos requisitos, com o objetivo de minimizar as deficiências identificadas nos requisitos. O desenvolvimento do *checklist* utilizou como base os cinco critérios de verificação de Sommerville (2011), as questões sugeridas por Pressman (2011) adequando-as para a realidade da organização, além de questões sugeridas pela empresa.

O *checklist* foi implementado em uma planilha eletrônica, contendo 21 questões escritas de forma afirmativa agrupadas nos critérios de verificação, possuindo quatro opções de resposta: Sim (a questão é atendida por completo), Não (a questão não é atendida), Parcialmente (a questão é atendida em partes), Não se Aplica (a questão não faz parte do contexto do requisito). Foram incluídas observações em algumas questões para auxiliar o inspetor na revisão, direcionando assim a inspeção do requisito, além de fornecer um campo no fim do *checklist* para relatar as inconsistências identificadas na inspeção, conforme apêndices B e C.

Após a conclusão da primeira versão do *checklist*, o mesmo foi avaliado pelo gerente do setor do desenvolvimento e os participantes da coleta preliminar de dados, onde foram solicitadas modificações em algumas afirmativas com o intuito de deixa-las claras e objetivas. Ao concluir a construção do *checklist*, este foi disponibilizado para todos os respondentes do questionário via correio eletrônico oficial da organização.

### 4.5. Atividade de Validação de Requisitos

A proposta da pesquisa é incluir a atividade de validação de requisitos antes da atividade de planejamento da *sprint,* isto é, revisar os requisitos de maior prioridade que serão planejados e desenvolvidos na *sprint*. Após a implantação da nova atividade no processo, um requisito só pode ser planejado se atender dois critérios. O primeiro é estar priorizado mantendo a regra atual e o segundo critério é estar valido, sendo assim deve ter sido revisado pela inspeção através do *checklist* e não conter nenhuma questão marcada com Não ou Parcialmente.

A quantidade de requisitos inspecionada em cada s*print* varia de acordo com a complexidade dos requisitos e utiliza-se como base para selecionar o número total de requisitos que serão inspecionados, os projetos executados pela equipe em *sprints* anteriores. No entanto, pode ser que sejam validados alguns requisitos a mais ou a menos, procura-se sempre validar um ou dois requisitos a mais no máximo para cada *sprint*. Quando forem revisados requisitos a mais eles continuam na lista de requisito com o *status* validado na ordem de priorização, se ocorrer qualquer mudança, eles devem ser inspecionados novamente para o próximo planejamento. Caso faltem requisitos validados, não serão inclusos requisito não validos no momento do planejamento, espera-se o andamento da *sprint* para ver a real necessidade de incluir novos requisitos que serão validados posteriormente seguindo a priorização.

A definição por não revisar todos os requisitos especificados ocorreu, pois as prioridades podem mudar no decorrer das *sprints,* juntamente com as definições de negócio. Consequentemente seria necessário revisar novamente todos os requisitos antes do próximo planejamento, então se optou por incluir menos requisitos no planejamento do que revisar toda a lista de requisitos a cada mudança de regra ou prioridade. A Figura 2 apresenta a modificação sugerida pelo estudo no processo de desenvolvimento da organização.

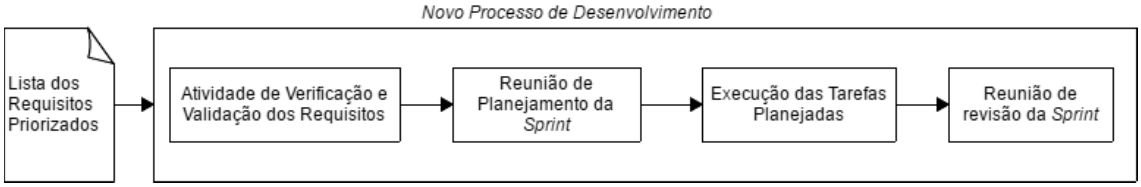


**Figura 2. Novo fluxo de trabalho proposto**

A revisão dos requisitos deve ser realizada por um inspetor cujo seu conhecimento abranja a área técnica e de negócio do sistema, preferencialmente, um projetista de software ou um desenvolvedor com experiência, pois será necessário cruzar as informações para realizar a inspeção. Optou-se inicialmente pela escolha do projetista de software de cada equipe, sendo que cada projetista revisa somente os requisitos pertencentes ao(s) módulo(s) da equipe que atua.

A inspeção inicia com uma breve reunião entre o analista de negócio e o inspetor. O analista de negócio é o responsável pela lista de requisitos, pois é ele quem elícita, especifica e prioriza os requisitos. A lista dos requisitos se encontra na ferramenta Visual Studio e esta disponível para todo o setor de desenvolvimento. Nesta reunião o analista de negócio fornece para o inspetor uma lista com o número dos p*roduct backlogs* que devem ser revisados.

A partir disto o inspetor pega item a item da lista e inicia a revisão. Para cada item revisado deve-se salvar uma cópia do *Checklist* com o seguinte formato: “Checklist de Validação de Requisitos – Backlog xxxxxx – NRyy”. O código “xxxxxx” significa o número do requisito validado e o código “NRyy” representa quantas vezes o requisito foi revisado, por exemplo, se o final estiver com NR02, este requisito foi inspecionado por duas vezes. Dentro do *Checklist* o revisor preenche o número do requisito, data da revisão e responde as questões do *Checklist*, caso alguma das questões possuírem a resposta Não ou Parcialmente ele deve relatar as inconsistências identificadas.

Ao concluir a revisão de cada requisito, o inspetor é responsável por salvar o *checklist* no requisito quando a inspeção não identificar nenhuma inconsistência ou enviar o *checklist* para o analista de negócio fazer as correções necessárias. Quando o requisito apresentar inconsistências deve-se revisa-lo novamente após as correções efetuadas. Este ciclo é repetido até a inspeção não identificar mais inconsistência no requisito.

### 4.6. Aplicação do *Checklist*

O *checklist* foi aplicado em três *sprints* correspondente aos meses de junho, julho e agosto do ano de 2016. A inspeção referente à primeira *sprint* ocorreu no fim do mês maio entre os dias 30 e 31, no inicio do dia 30 foi realizada a reunião entre o inspetor e o analista de negócio onde foi gerada inicialmente a lista de requisitos com apenas o requisito de número 55316 para inspecionar. Este requisito possuía vários *Test Cases* com as definições das regras de conversão de dados entre os dois sistemas da empresa.

Durante a revisão desse requisito identificou-se os seguintes itens de inconsistência:

- O requisito não descrevia a funcionalidade (questão linha 7);
- O requisito não estava escrito no formato de *User Story* (questão linha 14);
- O requisito provavelmente não seria concluído dentro do cronograma, isto é, não seria entregue ao termino da *sprint* (questão linha 34);
- O requisito não podia ser estimado, pois continha *Test Cases* com grande tempo de desenvolvimento (questão linha 36).

Após a primeira inspeção o requisito de número 55316 foi quebrado em mais quatro requisitos (55687, 55688, 55689, 55691) para atender as questões 34 e 36 do *checklist* além de efetuar as outras correções identificadas. Estes novos requisitos foram inspecionados juntamente com a reinspeção do primeiro requisito. Nesta nova inspeção o requisito de número 55687 apresentou inconsistência nas questões 20 e 24 do *checklist* onde se observa a clareza e objetividade das regras e a consistência das informações descritas verificando se estão adequadas para realizar a análise e o desenvolvimento do requisito. A inspeção foi concluída após a segunda revisão deste requisito.

No decorrer da *sprint* identificou-se um problema durante a análise do requisito de número 55688, sendo necessário remover os requisitos 55688, 55689 e 55691 da *sprint.* A remoção destes requisitos ocorreu devido ao tempo necessário para solucionar o problema encontrado, além da dependência direta entre eles, sendo necessário concluir um requisito para iniciar o outro. Com a remoção desses requisitos a *sprint* ficou incompleta, então foram inclusos os requisitos de número 56081 e 56082 completando assim a *sprint*. Estes novos requisitos passaram por todo o processo de inspeção antes de serem adicionados na *sprint*.

A inspeção referente à segunda *sprint* ocorreu no inicio do mês julho entre os dias 01, 04 e 05, no inicio da tarde do dia 01 foi realizada a reunião entre o inspetor e o analista de negócio onde foi gerada inicialmente uma lista com 15 requisitos. Durante a revisão identificou-se que os requisitos estavam mais detalhados, no entanto, dos 14 requisitos inspecionados 12 apresentaram inconsistência na questão 14 do *checklist* que

verifica se o requisito está escrito no formato de *User Story*. Outra inconsistência levantada foi à existência de conflito entre a regra de negócio descrita e a regra existente no sistema, sendo identificada pela questão 12 do *checklist*. A inspeção foi concluída após a segunda revisão deste requisito.

A inspeção referente à terceira *sprint* ocorreu no fim do mês julho entre os dias 26 e 27, no inicio do dia 26 foi realizada a reunião entre o inspetor e o analista de negócio onde foi selecionada uma lista de 12 requisitos para inspecionar, sendo que somente nove requisitos foram planejados e o restante permaneceu na lista de prioridades como inspecionados. Durante a inspeção se observou que não ocorreram inconsistência na questão 14 do *checklist* referente à forma como é escrito o requisito reduzindo assim a recorrência de inconsistências já conhecidas. O requisito de número 55689 que havia sido planejado em outra *sprint* e removido passou por uma nova revisão onde foi identificada inconsistência na questão 12 do *checklist* onde se observa conflitos entre as regas de negócio do sistema e as descritas. A inspeção foi concluída após a segunda revisão deste requisito. A segunda e terceira *sprint* foram executadas sem imprevistos.

## 5. Apresentação e Análise dos Resultados

Nesta seção são apresentados os resultados obtidos pela organização após implantação da atividade de validação dos requisitos no processo de desenvolvimento. Além de apresentar os resultados pré e pós implantação das revisões dos requisitos, através do *checklist* concebido.

### 5.1. Análise e Resultados pré-*checklist*

A análise dos resultados iniciou observando a quantidade de pontos entregues em relação aos pontos planejados (eixo y) entre os meses de janeiro a maio (eixo x) no ano de 2016, apresentados no Gráfico 1.

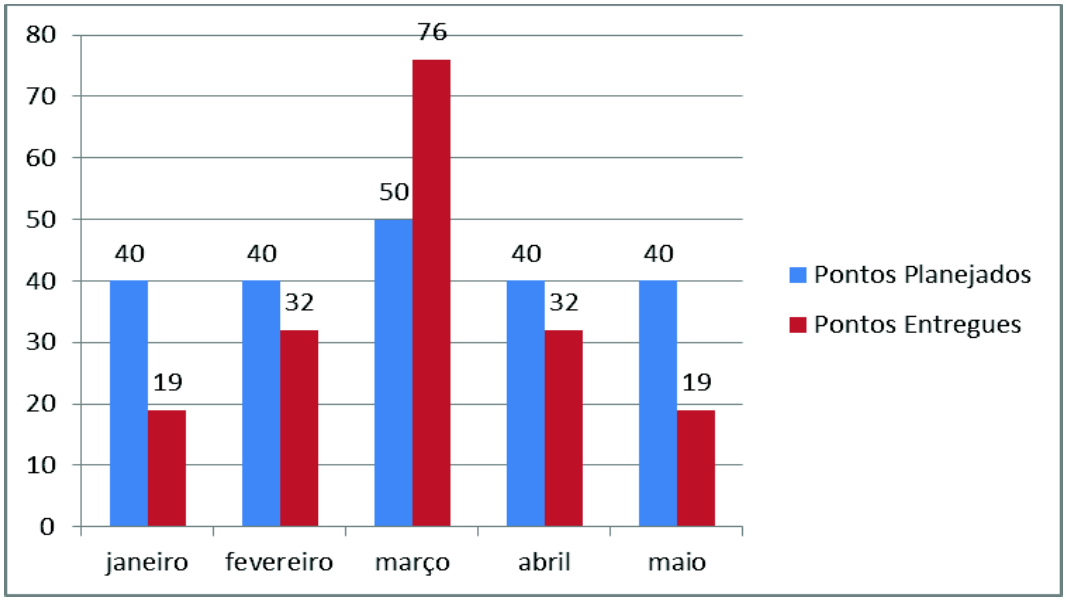


**Gráfico 1. Relação de pontos planejados x entregues antes da aplicação do *checklist***

Através resultados obtidos, identificou-se uma instabilidade no processo de desenvolvimento. Observa-se no Gráfico 1 que existem meses como janeiro e maio onde a quantidade de pontos entregues não atingiu nem a metade do planejamento da S*print*, demostrando que os requisitos foram subestimados, sendo necessário mais tempo para concluir o seu desenvolvimento. Já no mês de março entregou-se 52% a mais do planejado na *sprint*, que também não é considerado um resultado positivo, pois significa que os requisitos foram superestimados e no meio da *sprint* foi necessário incluir novos requisitos. Os percentuais de entrega podem ser observados no Gráfico 2.

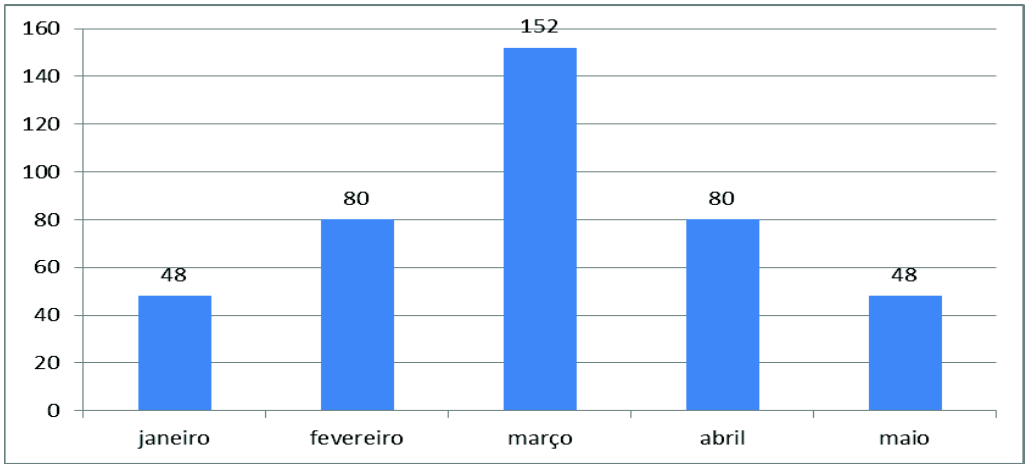


**Gráfico 2. Percentual de pontos entregues antes da aplicação do *checklist***

## 5.2. Apresentação e Análise dos Resultados pós-*checklist*

A análise dos resultados obtidos pelo estudo contemplou os meses de junho, julho e agosto do ano de 2016. Durante este período pode-se observar que a quantidade de pontos entregues se aproximou da quantidade de pontos planejados, sendo apresentados pelo Gráfico 3.

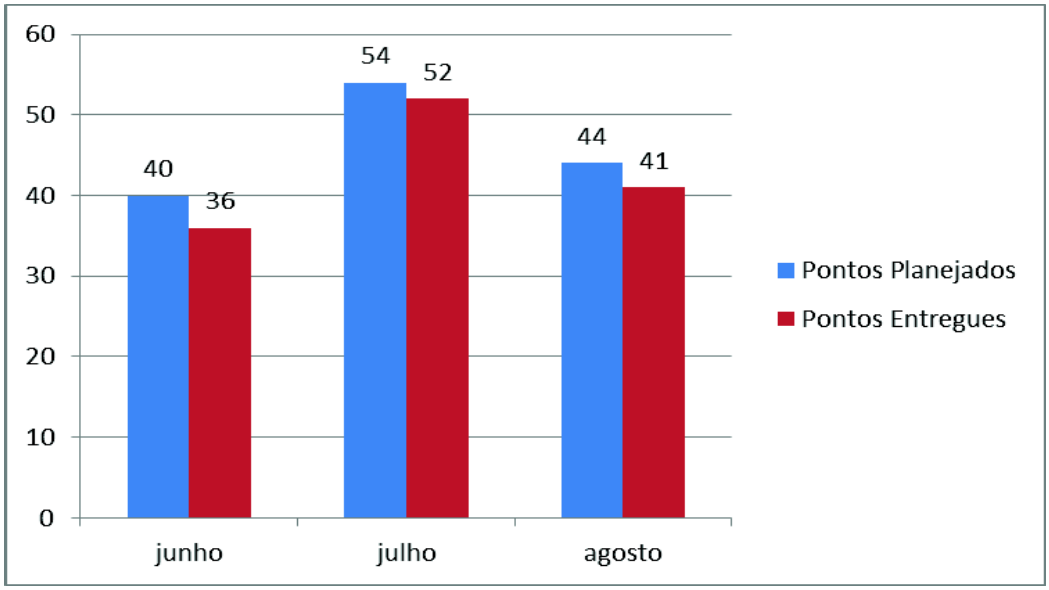

**Gráfico 3. Relação de pontos planejados x entregues durante aplicação do *checklist***

Observa-se no Gráfico 3, que houve uma melhora na estimativa dos pontos para cada requisito, diminuindo a instabilidade dos projetos da equipe alvo do estudo. Os resultados obtidos com a utilização do *checklist* nas revisões dos requisitos superaram a meta estipulada pela organização onde era esperada uma entrega de 80% dos pontos planejados para cada *sprint*. O Gráfico 4 apresenta os percentuais obtidos nos três meses de execução da validação de requisitos na organização.

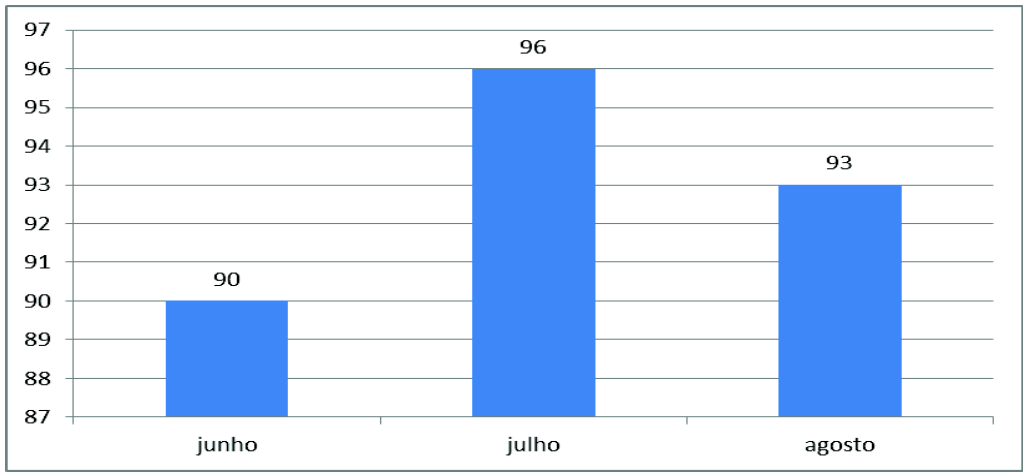


**Gráfico 4. Percentual de pontos entregues durante aplicação do *checklist***

O Gráfico 5 apresenta os percentuais de entrega das *sprints*, realizada de janeiro a agosto do ano de 2016, onde fica visível a melhora na entrega dos projetos após a implantação das inspeções dos requisitos através do *checklist*.

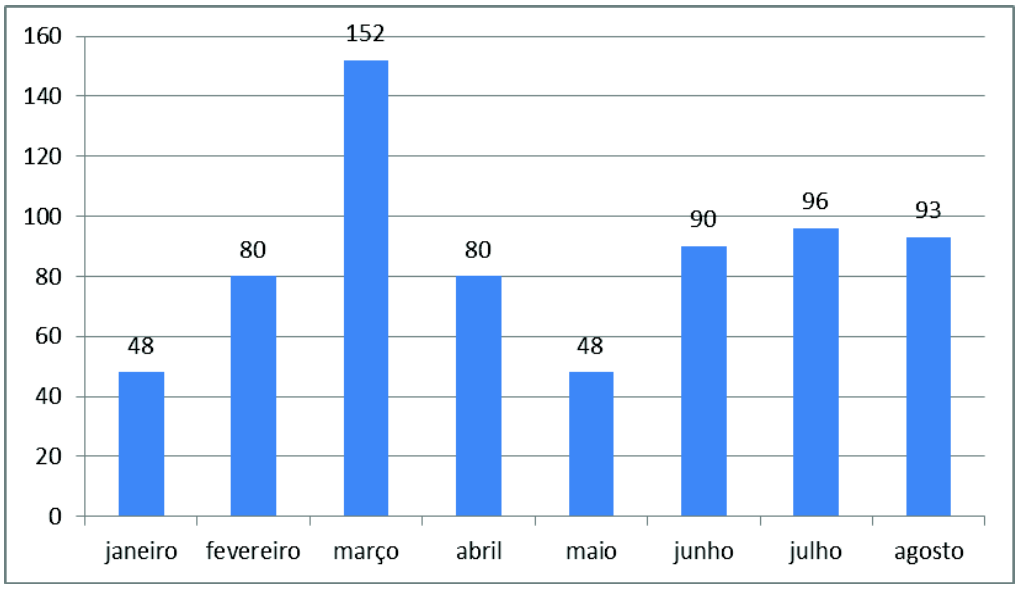


**Gráfico 5. Percentual de pontos entregues pela equipe até agosto**

Durante a aplicação do *checklist* identificou-se que um dos maiores problemas nos requisitos estava no seu tamanho. Os requisitos eram muito grandes dificultando a estimativa e a entrega do mesmo na *sprint* planejada. Observou-se que os requisitos eram iniciados em uma *sprint* e seriam entregues somente em outra *sprint* justificando assim *sprints* com entregas muito baixas e outras muito elevadas. Isto foi identificado na primeira inspeção onde a lista de requisitos partiu de um único requisito e começou a ser divido em mais requisitos, este trabalho continuou nas *sprints* seguintes aprimorando a especificação dos requisitos.

Após a implantação do *checklist,* os projetos passaram de uma variância de entrega de 18,22% para 0,10%, isto é, as entregas dos projetos estão se aproximando do que foi planejado. Como base na análise quantitativa observa-se que a qualidade no gerenciamento dos projetos obteve uma melhor significativa.

## 6. Conclusão

Este trabalho teve como objetivo principal melhorar a qualidade dos projetos de software, respondendo a questão de pesquisa: A validação de requisitos propicia melhora na qualidade em projetos de software? Para que isto fosse possível, desenvolveu-se um *Checklist* através de estudos aprofundados sobre validação de requisitos, critérios de verificação, além de princípios para a composição do *Checklist* juntamente com as técnicas baseadas em leitura. Com o embasamento teórico foi possível implantar as inspeções nos requisitos utilizando o *Checklist* na organização e avaliar as entregas de três projetos nos meses de junho a agosto do ano de 2016.

Sendo assim pode-se concluir que a partir das inspeções realizadas, os projetos apresentaram uma melhora significativa quanto ao percentual de entrega, atingindo o objetivo proposto pelo trabalho e superando a meta estipulada pela organização. No entanto, deve-se destacar perante os resultados obtidos após a implantação do *Checklist* uma limitação quanto a influência da pesquisadora, pois a mesma está incluída no ambiente do estudo, podendo não identificar alguma situação existente. Além de contemplar somente uma equipe como amostra e o curto tempo de aplicação da pesquisa, sendo assim os resultados não podem ser generalizados.

A partir das informações levantadas neste trabalho a organização pode ampliar a aplicação das inspeções através do *checklist* para as demais equipes do setor de desenvolvimento e se considerar interessante institucionalizar a atividade de validação de requisitos, como parte do processo de desenvolvimento da empresa. Vale ressaltar que o processo ainda está em fase inicial e necessita de ajustes para contemplar todo o setor.

Este trabalho apresenta como contribuição para a área de conhecimento, o *checklist* concebido e o modelo de inspeção aplicado nos requisitos, servindo como orientação para outros estudos ou organizações que buscam a melhoria continua dos seus processos. Como trabalhos futuros poderiam ser criados indicadores que medem o retrabalho nos projetos de software pré e pós implantação do *checklist*, visando contabilizar a redução de custo nos projetos de software da organização, com o objetivo de comprovar financeiramente o beneficio da atividade de validação de requisitos.

Por fim, conclui-se que a validação dos requisitos propicia a melhoria na qualidade dos projetos de software respondendo à questão de pesquisa deste trabalho através dos resultados obtidos neste estudo.

## Referências


Akinola, O., Osofisan A. O. (2009), An Empirical Comparative Study of Checklistbased and Ad Hoc Code Reading Techniques in a Distributed Groupware Environment. International Journal of Computer Science ans Information Security, Vol. 5, No. 1.

Basili, V., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., Zelkowitz, M. (1996), The Empirical Investigation of Perspective-Based Reading, Empirical Software Engineering: Na International Journal, 1(2): p. 133-164.

Basili, V. R. (1997), Evolving and Packaging Reading Technologies. Journal of Systems and Software, 38(1).

Barboza, G., Werneck, M., Assis, H., Fernandes, U., Silva, I. (2009), Um processo de elicitação de requisitos com foco na seleção da técnica de elicitação. VIII Simpósio Brasileiro de Qualidade de Software.

Coelho, J. S., Braga, J. L., Ambrósio, B. G. (2013), Otimizando recursos no processo de inspeção de software: uma abordagem utilizando simulação com dinâmica de sistemas, 14th Argentine Symposium on Software Engineering.

Fagan, M. E. (1976), Design and Code Inspection to Reduce Errors in Program Development, IBM Systems Journal, vol. 15, no. 3, p. 182-211.

Gil, A. C. (2010). Como Elaborar Projetos de Pesquisa. 5ª Edição.

Kalinowski, M., Spínola, R. O., Travassos, G. H. (2004), Infra-estrutura Computacional para Apoio ao Processo de Inspeção de Software. III Simpósio Brasileiro de Qualidade de Software.

Laitenberger, O., El Anam, K., Harbich, T. G. (2001), An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective- Based Reading of Code Documents. IEEE Transactions on Software Engineering, vol. 27, No. 5.

Laitenberger O. (2002), A Survey of Software Inspection Technologies. Handbook on Software Engineering and Knowledge Engineering, vol. II.

Macedo, F. F., Vilain, P. (2014), Especializando Ações de Transparência para Qualidade no Desenvolvimento de Software no Setor Público. Anais do XIII SBQS.

Mafra, S. N., Travassos, G. H. (2005), Técnicas de Leitura de Software: Uma Revisão Sistemática. Simpósio Brasileiro de Engenharia de Software, Uberlândia, MG, Brasil, Outubro.

Mafra, S. N., Travassos, G. H. (2006), Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos. Anais do V SBQS.

Martins, G. A. (2008), Estudo de caso: uma estratégia de pesquisa, 2ª Edição.

Mello, R. M., Massollar, J. L., Travassos, G. H. (2011), Técnica de Inspeção Baseada em Checklist para Identificação de Defeitos em Diagramas de Atividades. Anais do X SBQS.

Petersson, H. (2002), Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis. Technical report 147, Department of Communication Systems, Lund Institute of Technology.

Porter, A. A., Votta, L. G., Basili, V. R. (1995), Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. IEEE Transactions on Software Engineering, Vol. 21, No. 6.

Pressman, R. S. (2011), Engenharia de Software: uma abordagem profissional, 7ª Edição.

Sabaliauskaite, G., Matsukawa, F., Kusumoto, S., Inoue, K. (2003) Further investigations of reading techniques for object-oriented design inspection. Information and Software Technology 45, pp 571-585. ElSevier Science B. V.

Shull, F., Rus, I., Basili. V. (2000), How perspective based reading can improve requirements inspections, IEEE Computer, v. 33, n. 7, pp 73-79

Sommerville, I. (2003), Engenharia de Software, 6ª Edição.

Sommerville, I. (2011), Engenharia de Software, 9ª Edição.

Souza, V. C. (2011), Engenharia de Requisito de Software.

Thelin, T., Runeson, P., Wohlin, C. (2003), An Experimental Comparison of Usage-Based and Checklist-Based Reading. IEEE Transactions on Software Engineering, Vol. 29, No. 8.

Vergara, S. C. (2007), Projetos e relatórios de pesquisa em Administração. 9ª Edição.

Votta, L.G. (1993), Does Every Inspectin Need a Meeting?, Proc. First ACM Symp. Foundations of Software Engineering, pp. 107-114.

Wainer, J. (2007), "Métodos de Pesquisa Quantitativa e Qualitativa para a Ciência da Computação".Disponível em: http://www.ic.unicamp.br/~wainer/papers/metod07.pdf

Yin, R. K. (2015), Estudo de caso: planejamento e métodos, 5ª Edição.

## APÊNDICE A – Coleta de Dados Preliminar: Questionário aplicado

### QUESTIONARIO SOBRE REQUISITOS DE SOFTWARE

Está pesquisa está relacionada ao curso de especialização em Qualidade de Software, da Universidade do Vale dos Sinos (Unisinos), sob orientação do professor Silvio Cesar Cazella.

Objetivo Geral da pesquisa: levantar os pontos críticos em relação aos requisitos de software e a partir disto implantar a etapa de validação de requisito no processo de desenvolvimento da organização.

A pesquisa visa identificar pontos de inconsistência nos requisitos (*Product Backlog*) de software elicitados na organização, por isso foi desenvolvido um questionário onde são observados os seguintes itens: Validade, Consistência, Completude, Realismo e Verificabilidade para cada requisito.

Validade: observa a real necessidade de desenvolver a funcionalidade.

Consistência: observa se existem informações contraditórias, onde uma restrição sobrepõe outra.

Completude: observa se todas as funções e restrições da funcionalidade estão descritas.

Realismo: observa se a funcionalidade pode ser desenvolvida levando em consideração a tecnologia existente, orçamento e cronograma.

Verificabilidade: observa se a funcionalidade pode ser testada.

Para responder a pesquisa tome como base a sua experiência em relação aos requisitos analisados ou desenvolvidos nos últimos projetos. Responda pensando nas dificuldades encontradas em relação à interpretação do requisito em um âmbito geral.

Público-alvo: Desenvolvedores e Projetistas de Software da empresa Metadados.

Este questionário está divido em duas etapas:

Etapa 1: Perfil do respondente.

Etapa 2: Levantamento de pontos críticos referente aos requisitos de software.

As informações coletadas serão utilizadas somente para esta pesquisa. É garantido o anonimato ao respondente. O tempo estimado para responder o questionário é de 10 minutos.

### Etapa 1: Perfil do respondente.

**Nível de Formação***

- Ensino Médio Completo
- Ensino Superior Incompleto
- Ensino Superior Completo

- Pós-graduação ou Especialização

**Tempo de empresa***
- Menos de 3 anos
- Entre 3 e 5 anos
- Acima de 5 anos

**Papel na Área de Desenvolvimento***
- Programador
- Projetista

**Equipe de desenvolvimento***
- Folha
- GP/SS
- Frequência
- Framework

**Projeto que está alocado atualmente***
- Sistema atual
- Sistema novo
- Ambos os sistemas

**Etapa 2: Levantamento de pontos críticos referente aos requisitos de software.**

Assinale as afirmativas abaixo conforme a sua percepção, considerando a seguinte escala:

0-Discordo Totalmente, 1- Discordo Parcialmente, 2-Neutro, 3-Concordo Parcialmente, 4-Concordo Totalmente.

1) **Os requisitos contem todas as informações necessárias para serem desenvolvidos, isto é, possuem as funções e restrições da funcionalidade definidas.***
   Exemplo: O que a funcionalidade deve fazer?

   Resposta: 0 1 2 3 4

2) **Os requisitos possuem todas as funções descritas da funcionalidade.***
   Exemplo: Como implementar a função da funcionalidade?

   Resposta: 0 1 2 3 4

3) **Os requisitos possuem todas as restrições descritas da funcionalidade.***

Exemplo: Como implementar a restrição da funcionalidade?

Resposta: 0 1 2 3 4

4) **As informações dos requisitos são ambíguas.** *

Resposta: 0 1 2 3 4

5) **Os requisitos estão escritos de forma clara e objetiva.** *

Resposta: 0 1 2 3 4

6) **As informações contidas no requisito apresentam conflito entre si.** *
Exemplo de conflitos: O conflito pode ser entre *test cases* ou no documento que contem o detalhamento da implementação.

Resposta: 0 1 2 3 4

7) **Os requisitos apresentam conflitos em relação à outra funcionalidade existente no sistema.** *

Resposta: 0 1 2 3 4

8) **Os requisitos podem ser desenvolvidos por completo quanto às regras de negócio, isto é, não existiam regras de negócio que precisam ser discutidas e definidas.** *

Resposta: 0 1 2 3 4

9) **Os requisitos podem ser desenvolvidos por completo quanto à arquitetura do sistema, isto é, não existem empecilhos de framework para eles serem concluídos.** *

Resposta: 0 1 2 3 4

10) **A funcionalidade descrita no requisito considera a tecnologia de desenvolvimento do sistema.** *

Resposta: 0 1 2 3 4

11) **As funcionalidades desenvolvidas são sempre utilizadas.** *

Resposta: 0 1 2 3 4

12)**Os requisitos expressam a real necessidade da funcionalidade que será desenvolvida.** *

Resposta: 0 1 2 3 4

13)**Os requisitos são testáveis, isto é, após o desenvolvimento é possível testar a funcionalidade.** *

Resposta: 0 1 2 3 4

14)**Os requisitos podem ser desenvolvidos dentro das funções estabelecidas.** *

Resposta: 0 1 2 3 4

15)**Os requisitos podem ser desenvolvidos dentro das restrições estabelecidas.** *

Resposta: 0 1 2 3 4

**Espaço para sugestões**

Cite pontos de melhorias que você considere importante para os requisitos.

**APÊNDICE B – *Checklist* de Validação**

# *CheckList* de Validação dos Requisitos

*Checklist* necessário para analisar e validar os requisitos, minimizando assim, a probabilidade de inconsistências identificada nos requisitos após a realização do planejamento da Sprint. Caso alguma questão tenha a resposta "Não" ou "Parcialmente" o problema deve ser relatado no item "Relate com mais detalhes os problemas encontrados..." deste documento.

Versão do Documento: 1.0

| Número do Requisito Validado: | Data da Validação: | | | |
|---|---|---|---|---|
| **Questões** | **Sim** | **Não** | **Parcialmente** | **Não se Aplica** |
| **Validade:** | | | | |
| O requisito descreve a necessidade da funcionalidade? | | | | |
| | | | | |
| | | | | |
| **Consistência:** | | | | |
| Os *Test Case*s do requisito **não** se contradizem? | | | | |
| O requisito **não** está em conflito com outras funcionalidades do sistema? | | | | |
| O requisito não é ambíguo? | | | | |
| O requisito está escrito no formato de *User Story*? | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| **Completude:** | | | | |
| Os *Test Cases* estão escritos de forma clara e objetiva? | | | | |
| As informações relativas ao requisito estão completas? | | | | |
| O requisito possui a descrição das funções da funcionalidade? | | | | |
| O requisito possui a descrição das restrições da funcionalidade? | | | | |
| O requisito fornece as informações adequadas para a análise e desenvolvimento da funcionalidade? | | | | |
| O requisito pode ser desenvolvido considerando a regra de negócio contida nele? | | | | |
| O requisito pode ser desenvolvido considerando a arquitetura atual do Framework? | | | | |
| O requisito considera a conversão dos dados entre o sistema Atual e o Novo? | | | | |
| As regras de negócio contemplam a conversão dos dados entre os dois sistemas? | | | | |
| O protótipo está coerente com a descrição dos *Test Cases*? | | | | |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| **Realismo:** | | | | |
| O desenvolvimento do requisito é viável tecnicamente? | | | | |
| O requisito pode ser desenvolvido dentro do cronograma estabelecido? | | | | |
| Custo x Benefício indica que o requisito pode ser desenvolvido? | | | | |
| O requisito é estimável? | | | | |
| | | | | |
| **Verificabilidade:** | | | | |
| O requisito está priorizado? | | | | |
| O requisito pode ser testado? | | | | |
| | | | | |
| | | | | |
| **Relate com mais detalhes os problemas encontrados nas questões acima, caso existam:** | | | | |
| | | | | |

**APÊNDICE C – *Checklist* de Validação: Exemplo de uma questão com observação**

| | | | | | |
|---|---|---|---|---|---|
| 1 | **CheckList de Validação dos Requisitos** | | | | |
| | Checklist necessário para analisar e validar os requisitos, minimizando assim, a probabilidade de inconsistências identificada nos requisitos após a realização do planejamento da Sprint. Caso alguma questão tenha a resposta "Não" ou "Parcialmente" o problema deve ser relatado no item "Relate com mais detalhes os problemas encontrados..." deste documento. | | | | |
| 2 | | | | | |
| 3 | Versão do Documento: 1.0 | | | | |
| 4 | Numero do Requisito Validado: | Data da Validação: | | | |
| 5 | **Questões** | **Sim** | **Não** | **Parcialmente** | **Não se Aplica** |
| 6 | **Validade:** | | | | |
| 7 | O requisito descreve a necessidade da funcionalide? | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | **Consistência:** | | | | |
| 11 | Os *Test Cases* do requisito **não** se contradizem? | | | | |
| 12 | O requisito **não** está em conflito com outras funcionalidades do sistema? | | | | |
| 13 | O requisito não é ambiguo? | | | | |
| 14 | O requisito está escrito no formato de *User Story*? | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | **Completude:** | | | | |

Deve ser observado nesta questão se a regra descrita em um *Test Case* não sobrepõem outra regra descrita em outro *Test Case*.