Ricardo Duarte Cardoso

Mapeamento dos processos e artefatos da engenharia de requisitos para o eXtreme Programming

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo

2017

Ricardo Duarte Cardoso

Mapeamento dos processos e artefatos da engenharia de requisitos para o eXtreme Programming

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. Dr. Fábio Levy Siqueira

São Paulo

2017

Catalogação-na-publicação

Cardoso, Ricardo
Mapeamento dos processos e artefatos da engenharia de requisitos para o eXtreme Programming / R. Cardoso -- São Paulo, 2017.
61 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1.Engenharia de requisitos 2.Metodologias ágeis I.Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t.

AGRADECIMENTOS

Ao Professor Fabio Levy, pela orientação, sempre pronto a ensinar, incentivando e tolerando minhas imperfeições.

Aos professores da Escola Politécnica da Universidade de São Paulo com os quais tive contato nos últimos anos, que contribuíram me dando a base de conhecimento necessária para realização de todo esse trabalho.

Ao meu pai, por ser exemplo de pessoa que me ensina que bondade não tem limites; persistência e paciência são essenciais para vencer na vida.

À minha mãe, pela dedicação incansável, sempre prestativa e disposta a ajudar da forma que for possível a qualquer momento e horário.

Ao Gil, pela paciência comigo neste período de ausência, pela força nos momentos de dificuldade, incentivando sempre e apoiando para não me deixar cair.

RESUMO

As metodologias ágeis vêm ganhando cada vez mais espaço nas empresas. Essas metodologias não utilizam as abordagens tradicionais de Engenharia de Requisitos, que baseiam-se em uma execução sequencial das atividades e em documentos formais para comunicação entre os processos de requisitos e o outros dos processos de desenvolvimento. Contudo é importante Engenharia de Requisitos seja aplicada nessas metodologias, pois desempenha um papel fundamental para o sucesso de um projeto. Este estudo analisa o atendimento das práticas de Engenharia de Requisitos por uma metodologia ágil, o eXtreme Programming (XP).

São apresentados o conceito de requisito, os processos e artefatos de uma abordagem formal determinada pela norma ISO 29148, os aspectos que caracterizam as metodologias ágeis e o detalhamento de todos os elementos contidos no XP. Após a apresentação é realizado o mapeamento dos processos e artefatos da norma ISO 29148 para os processos e artefatos do XP, seguindo a norma de avaliação de processos ISO 15504-2. Para realização da análise do atendimento do XP aos processos determinados pela norma ISO 29148 foi examinado qual prática ou etapa do processo do XP cobre cada tarefa do processo analisado. Na análise do atendimento aos artefatos determinados pela norma foi considerado a disponibilidade da informação, não somente a documentação dela, seguindo a filosofia dos métodos ágeis que estabelece que se a informação não precisa estar necessariamente escrita (o importante é ser de conhecimento de todos). Como conclusão da análise são realizadas sugestões de complementação dos processos e artefatos do XP para cobrir as lacunas encontradas. Por fim, são comparadas outras pesquisas, que também tratam da relação entre Engenharia de Requisitos e metodologias ágeis, com esse estudo, demonstrando os pontos comuns e principais diferenças entre eles.

Palavras-chaves: Engenharia de requisitos. ISO 29148. Metodologias ágeis. eXtreme Programming. Estudo de mapeamento

ABSTRACT

Agile methodologies are becoming more and more popular in companies. These methodologies do not use the traditional approaches of Requirements Engineering, which are based on the execution of sequential activities and on formal documents for communication between the processes. However, it is important that Requirements Engineering is applied to these methodologies as it plays a key role in project's success. This study analyze if Requirements Engineering practices are executed in an agile methodology, eXtreme Programming (XP).

It is presented the concept of requirement, the processes and artifacts of a formal approach determined by ISO 29148, the aspects that characterize the agile methodologies and the detailing of all elements contained in XP. After the presentation, the processes and artifacts of ISO 29148 are mapped to XP's processes and artifacts, following ISO 15504-2 process evaluation standard. To analyze the XP coverage to the processes determined by the standard ISO 29148 it was examined which practice or stage of the XP process covers each one of the analyzed process' task. In the analysis to the artifacts fulfillment determined by the norm, the availability of the information was considered, not only the documentation of it, following the philosophy of the agile methods that the information does not have to be necessarily written (the most important is to be known by all). As a conclusion of the analysis, suggestions are made to complement the processes and artifacts of XP to cover the gaps found. Finally, other researches, which also deal with the relationship between requirements engineering and agile methodologies, are compared with this study, presenting common points and key differences between them.

Keywords: Requirements engineering. ISO 29148. Agile methodologies. eXtreme Programming. Mapping study

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ConOps	Concept of operations (Conceito da operação)
ISO	International Organization for Standardization (Organização Internacional de Normalização).
MOP	Measures Of Performance (Medidas de desempenho)
OpsCon	System operational concept (Conceito operacional do sistema)
SRS	Software requirements specification document (Documento de especificação de requisitos de software)
StRS	Stakeholder Requirements Specification (Documento de especificação de requisitos de Stakeholder)
SyRS	System requirements specification document (Documento de especificação dos requisitos de sistema)
TPM	Technical Performance Measure (Medidas técnicas de desempenho)

SUMÁRIO

1. INTRODUÇÃO

1.1. Motivação

As abordagens tradicionais de Engenharia de Requisitos são baseadas em uma execução sequencial das atividades (Bray, 2002). Em tais abordagens, a comunicação entre o processo de Engenharia de Requisitos e o processo de construção de software é baseada principalmente em documentos formais (Bray, 2002). As metodologias ágeis têm crescido de popularidade nos últimos anos (VersionOne, 2016), e um dos valores definidos no Manifesto Ágil (2001) é que o software funcionando é mais importante do que uma documentação abrangente, causando a impressão de que a comunicação entre as etapas do desenvolvimento de software baseada em documentos formais pode não ocorrer corretamente nas metodologias ágeis. Devido a qualidade de uma especificação de requisitos de software e a lacuna existente entre as especificações das necessidades do cliente e os detalhes necessários para produzir a solução estão diretamente ligadas ao sucesso do desenvolvimento de software (Saito et al., 2013) (Abdullah et al., 2011). Isso torna necessária a existência de estudos que auxiliam a aplicação da engenharia de requisitos nas metodologias ágeis.

1.2. Objetivo

O objetivo deste trabalho é analisar como o eXtreme Programming (XP) atende aos processos e artefatos de Engenharia de Requisitos. Para isso são determinadas em quais etapas do processo do XP são realizadas as atividades da Engenharia de Requisitos, e identificando em quais artefatos do XP estão as informações geradas pelos documentos formalmente estipulados para essa disciplina.

1.3. Justificativa

Existem diversos estudos que discutem a relação da Engenharia de Requisitos no contexto de metodologias s (Eberlein; Leite, 2002) (Goetz, 2002) (Paetsch; Eberlein; Maurer, 2003) (Sillitti; Succi, 2005) (Ramesh; Cao; Baskerville, 2007) (Gebhart et al., 2014.) (Lucia; Qusef, 2010), porém nenhum deles analisa cada tarefa dos processos e cada conteúdo dos artefatos das metodologias ágeis, comparando com o esperado para um processo de Engenharia de Requisitos. A maioria desses trabalhos é genérica e abrangente, falando sobre diversas

metodologias ágeis, o que difere deste trabalho, o qual mapeia o processo de Engenharia de Requisitos do XP para a norma ISO 29148 (2011). Essa norma foi usada pois ela descreve em detalhes tanto os processos quanto os artefatos de um processo formal de Engenharia de Requisitos.

1.4. Estrutura do Trabalho

Este estudo é composto por cinco capítulos, estruturados da seguinte forma. No presente capítulo é apresentada uma introdução. Em seguida, o capítulo 2 apresenta o resultado da pesquisa bibliográfica, evidenciando requisitos e seus tipos, o processo de elicitação de requisitos e os artefatos gerados por esse processo definidos na norma ISO 29148 (2011), a definição de metodologias ágeis e o detalhamento da metodologia escolhida para análise nesse trabalho o *eXtreme Programming*. O terceiro capítulo apresenta o mapeamento dos processos e artefatos da ISO 29148 para o eXtreme Programming e uma sugestão para cobertura das lacunas encontradas. O quarto capítulo descreve outras pesquisas relacionadas com esse estudo e as principais diferenças entre eles. Por fim, o quinto capítulo retrata as considerações finais sobre o trabalho, expressando as conclusões e as direções para uma continuidade da pesquisa.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta o conteúdo necessário para compreensão do trabalho. Serão apresentados os requisitos e seus tipos; o processo de elicitação de requisitos e os artefatos gerados por esse processo definidos na norma ISO 29148 (2011); a definição de metodologias ágeis e o detalhamento da metodologia escolhida para análise nesse trabalho, o *eXtreme Programming*.

2.1. Requisitos e tipos de requisitos

A definição de requisitos, segundo a norma ISO 29148 (2011) começa com intenções dos *stakeholders* (chamado em português de partes envolvidas). Elas evoluem para uma declaração mais formal no Conceito da Operação (ConOps), artefato que será detalhado no capitulo 2.3.1, antes de serem decompostas em requisitos dos *stakeholders* válidos. Para a ISO 29148 “Requisito é uma afirmação que traduz ou expressa uma necessidade, e suas restrições e condições associadas” (2011, p. 9). Intenções não servem como requisitos, uma vez que carecem de definição, análise e possivelmente consistência e viabilidade.

Segundo a norma, os requisitos são classificados em:

- **Requisitos de *Stakeholder*** (Nível gerencial e operacional do negócio): É elicitado no início do projeto e descreve as necessidades, vontades, desejos, expectativas e restrições percebidas dos *stakeholders*. Os requisitos são descritos no contexto do ambiente operacional e em condições, concentrando-se na finalidade e no comportamento do sistema.
- **Requisitos de Sistema:** Identificam as especificações técnicas (funções, desempenho, restrições de *design* e atributos) e a interação entre usuário e sistema. São requisitos de alto nível escritos com os termos que o cliente utiliza, juntamente com informações básicas sobre os objetivos globais e seu ambiente operacional.
- **Requisitos de Software:** Especificam a parte do projeto competente ao software, o que pode envolver uma parte do sistema e, neste caso, deve especificar também como o software irá manter as interfaces de comunicação com o restante do sistema.

2.2. Processo de Engenharia de Requisitos

O processo para elicitação de requisitos determinado pela norma ISO 29148 (2011) é apresentado na figura 1, através de um diagrama BPMN. O processo de engenharia de requisitos inicia com a criação do Conceito da Operação (ConOps). Este documento irá nortear todo o processo, pois ele contém as expectativas do cliente com o projeto. Com base neste documento o Processo de definição dos requisitos de *stakeholders* é realizado e gera como resultado o Documento de especificação de requisitos de *stakeholder*. Neste momento são conhecidos todos os objetivos que o projeto tem que ter para atender aos *stakeholders*; esses objetivos serão analisados e transformados em requisitos de sistema pelo Processo de análise de requisitos. Neste processo será produzido o Documento de especificação dos requisitos de sistema (SyRS), detalhando as necessidades sistêmicas, e também será produzido o Conceito operacional do sistema (OpsCon), que é voltado apenas para as necessidades do usuário do sistema. Após esse processo ser executado no nível de sistema, ele é executado, quando necessário, para cada software ou subsistema que compõe o sistema. Quando executado o processo para detalhar o software do sistema, é gerado outro documento chamado Documento de especificação de requisitos de software (SRS). Em paralelo a todos esses processos é executado o gerenciamento de requisitos, o qual que auxilia os outros processos.

Os processos serão detalhados nas subseções sequentes e os artefatos serão aprofundados na Seção 2.3. Essas Seções usam como referência a norma ISO 29148 (2011), a qual não será referenciada para simplificar o texto.

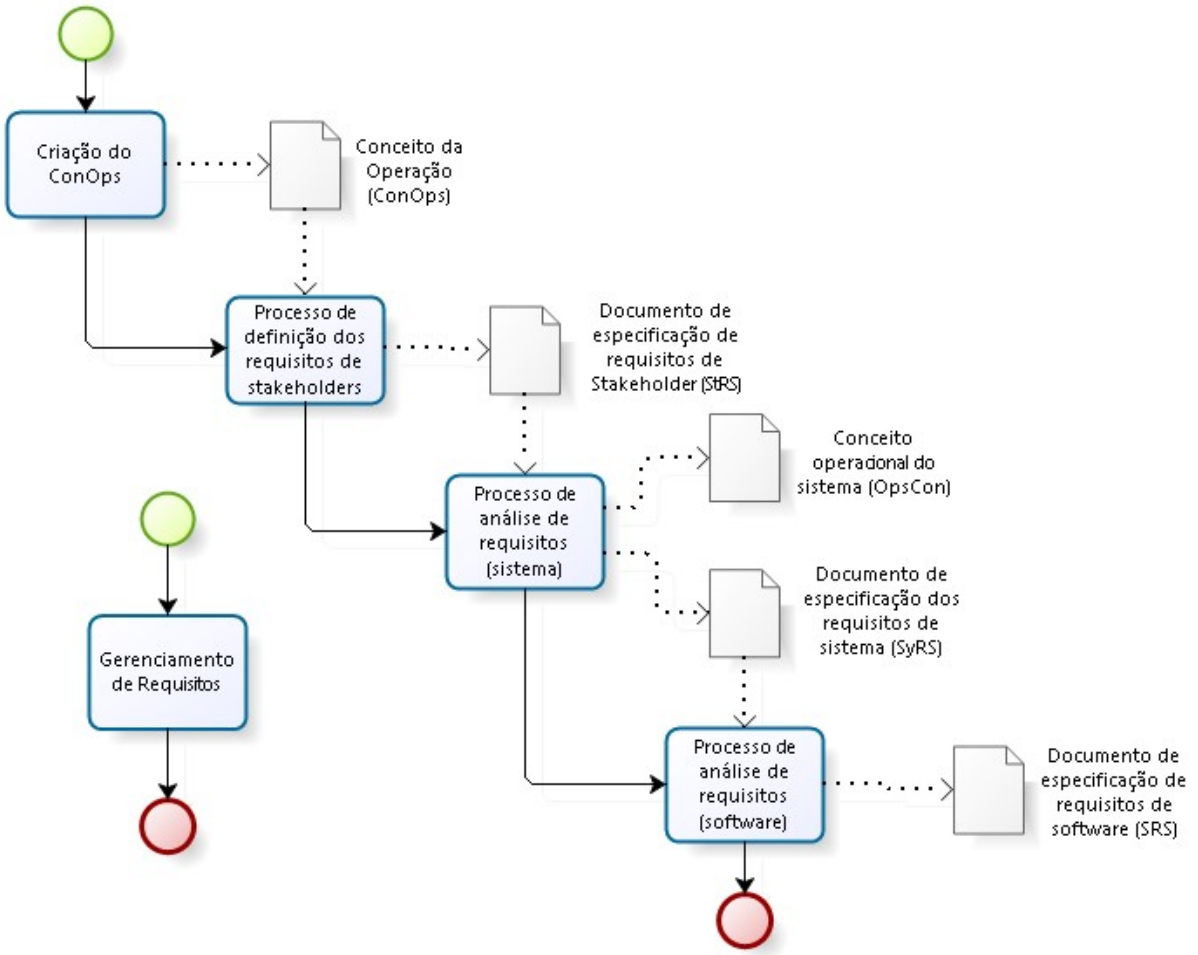

Figura 1 - Sequência de processos e artefatos de requisitos. Fonte: adaptado da ISO 29148 (2011, p.18).

2.2.1. Processo de definição dos requisitos de *stakeholders*

Este processo identifica os *stakeholders* e/ou conjuntos de *stakeholder*, suas necessidades, expectativas e desejos. Essas informações são analisadas e transformadas em um conjunto de requisitos de *stakeholders* que expressam a intenção dos envolvidos com o sistema e como cada função será validada. Como resultado, este processo gera o Documento de especificação de requisitos de *Stakeholder* (StRS).

As atividades que compõem o processo estão representadas graficamente na figura 2, que representa, utilizando um diagrama BPMN, a sequência em que as atividades acontecem durante o processo de definição de requisitos dos *stakeholders.* As Seções seguintes detalham cada uma dessas atividades.

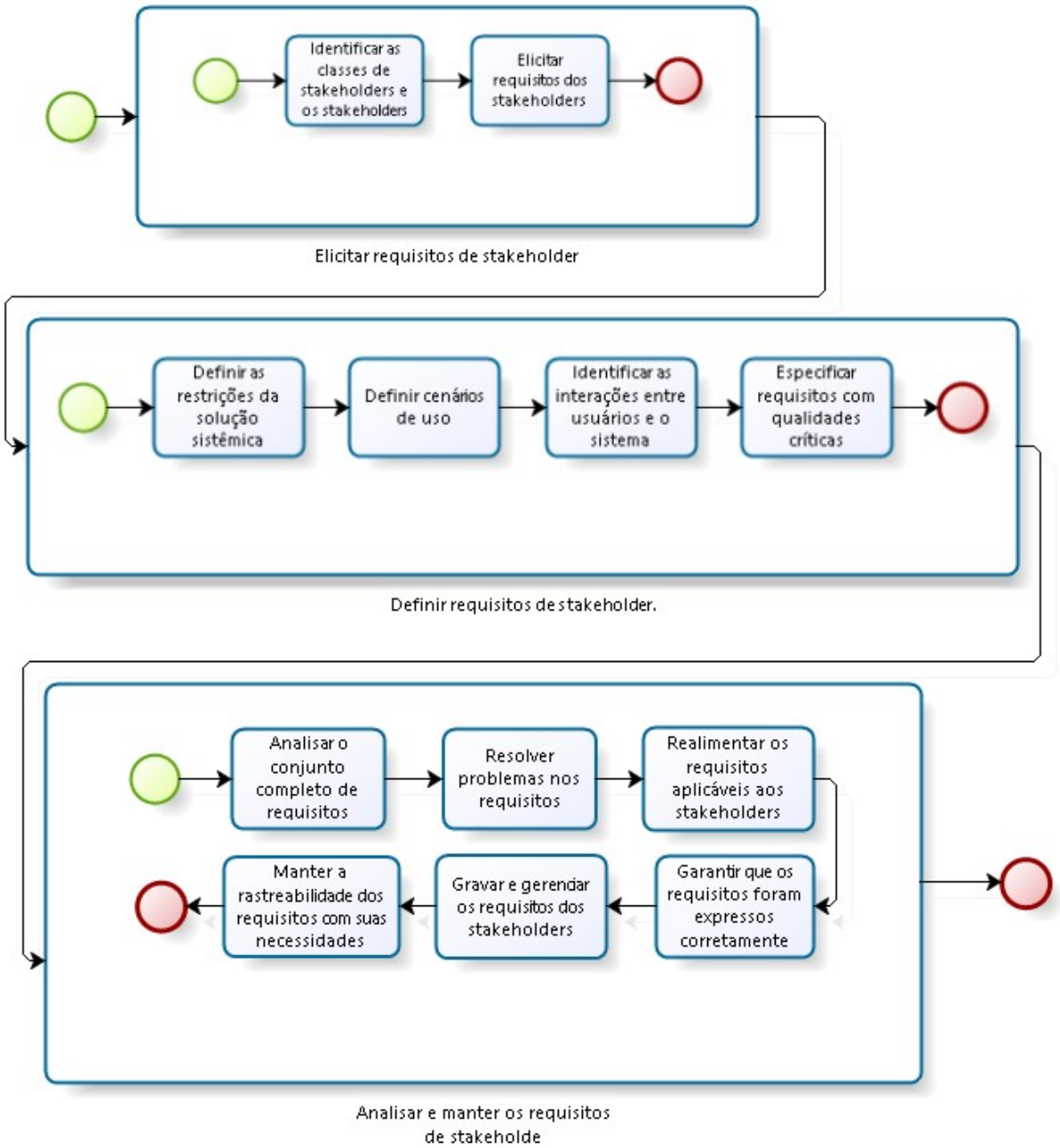

Figura 2 - Sequência de atividades e tarefas do processo de definição dos requisitos de *stakeholders*
Fonte: o autor.

2.2.1.1. Elicitar requisitos de *stakeholder*

Esta atividade inicia com o analista de requisitos identificando as classes de *stakeholders* que têm um papel ou interesse no produto. Em seguida, deve-se identificar os *stakeholders* que têm forte influência sobre metas, estratégias, operações e sistema de destino e, para cada *stakeholder,* deve-se determinar o papel, as responsabilidades e a posição que ele ocupa. Após essa identificação deve-se elicitar requisitos dos *stakeholders*, o que consiste em uma tarefa iterativa cujo objetivo é definir as metas, missão do sistema, cenários operacionais, ambiente

operacional e contexto de uso, implantação operacional, desempenho do sistema, eficácia do sistema, ciclo de vida do sistema, ambiente organizacional, características de usos e operadores do sistema. Para realizar essa tarefa podem ser utilizadas diferentes técnicas, tais como entrevistas, questionários, observação dos padrões de trabalho, análise organizacional, entre outras.

2.2.1.2. Definir requisitos de stakeholder

Após elicitar requisitos de *stakeholder* devem ser definidas as restrições da solução sistêmica. Uma restrição é um requisito, e elas são determinadas por acordos existentes, decisões de gestão ou técnicas, ou limitação de recursos. As restrições podem ser impostas pelos *stakeholders* internos e externos, interação entre sistemas, atividades do ciclo de vida, ou atividades técnicas e medidas de eficácia que refletem a satisfação geral do cliente (por exemplo, requisitos de desempenho, segurança, confiabilidade, disponibilidade, facilidade de manutenção e de carga de trabalho).

Com as informações descritas, deve-se definir os cenários de uso. Para isso são definidos conjuntos de sequências de atividades que apresentam o funcionamento do sistema em seu ambiente de uso. Além do conjunto, deve-se incluir o contexto das atividades, as características relevantes dos usuários, o ambiente físico e os equipamentos utilizados pelo usuário para realização das atividades. É necessário abordar diferentes níveis de abstração ou mecanismos de apresentação para que se consiga abranger todos os cenários dos *stakeholders*.

Após os cenários serem definidos, as interações entre usuários e o sistema são identificadas. A ideia é considerar a integração humano-sistema, promovendo uma abordagem completa do sistema, que inclui os usuários, tecnologia (hardware e software), o contexto operacional e as interfaces, para fazer com que eles trabalhem em harmonia. Esta atividade é centrada em disciplinas humanas tais como recursos humanos, treinamentos, fatores humanos, meio ambiente, saúde, segurança, habitabilidades e capacidade de sobrevivência. Essas interações serão utilizadas para em melhorar o desempenho das tarefas com a utilização do sistema.

Para finalizar deve-se especificar outros requisitos e funções de *stakeholders* que se relacionam com qualidades críticas. As qualidades críticas são aspectos do sistema que são essenciais para garantir a integridade do sistema e do seu ambiente operacional. Nesta atividade são identificados os riscos de segurança e

também especificado todas as áreas aplicáveis, incluindo físicas, processuais, comunicações, computadores, programas e dados. Deve-se levar em consideração as funções que poderiam impactar a segurança do sistema, incluindo o acesso e dano a dados pessoais, de propriedades e informações sensíveis.

2.2.1.3. Analisar e manter os requisitos de *stakeholder*

A última atividade do processo de definição dos requisitos de *stakeholders* se inicia analisando o conjunto completo de requisitos elicitados. Deve-se analisar os requisitos validando a identificação, priorização, se possuem requisitos faltantes, requisitos incompletos, ambíguos e inconsistentes, incoerente ou que não são possíveis testar. Além disso, deve-se validar a viabilidade de utilização de requisitos elicitados como requisitos de reutilização de sistemas legados.

A análise inicial deve resolver problemas nos requisitos, tais como requisitos que não poderão ser realizados ou são impraticáveis. Para isso é importante que haja uma análise e negociação entre os *stakeholders* de quais requisitos devem ser atendidos. Como os interesses entre stakeholders podem ser conflitantes devido a características incompatíveis ou devido a conflitos (tais como de desempenho, restrições orçamento, e cronograma de entrega), é necessário consulta-los para chegar a um consenso sobre um adequado *trade-off* entre os requisitos. Diferentes técnicas de resolução de conflito que podem ser aplicadas para facilitar essa tarefa.

Após os problemas serem resolvidos, o documento deve ser realimentado com os requisitos analisados nas atividades anteriores para garantir que as necessidades e expectativas foram adequadamente capturadas e expressas.

Com a possível alteração do documento é necessário estabelecer com os *stakeholders* que seus requisitos são expressos corretamente. Esta atividade é um ponto formal para a validação dos requisitos pelos *stakeholders*, em que os requisitos são examinados para garantir que eles definem o sistema que os *stakeholders* esperam. O principal objetivo da atividade é identificar eventuais problemas antes que os recursos sejam implementados.

Com o documento finalizado e aprovado deve-se gravar os requisitos dos *stakeholders* de uma forma adequada para o gerenciamento de requisitos durante o ciclo de vida do software. É importante considerar a utilização de uma ferramenta de gerenciamento de requisitos, especialmente para projetos mais complexos. Essa

ferramenta deve ter a capacidade de rastrear ligações entre os requisitos para mostrar as relações entre eles.

É importante ao final do processo manter a rastreabilidade dos requisitos de *stakeholders* com suas necessidades. Esta rastreabilidade entre os requisitos deve ser estabelecida e mantida para documentar formalmente que os objetivos dos *stakeholders* serão atingidos.

2.2.2. Processo de análise de requisitos

Este processo transforma os requisitos que estão em uma visão de negócio em uma visão técnica, determinando os serviços necessários que o produto deve atender. Este processo constrói uma representação do futuro sistema, resultando em requisitos de sistema mensuráveis que especificam, do ponto de vista do construtor do projeto, as características que o sistema deve possuir e o tamanho de cada característica com o fim de satisfazer as necessidades dos *stakeholders*.

Este processo gera o Documento de especificação de requisitos de sistema (SyRS) ou o Documento de especificação de requisitos de software (SRS), pois a norma define o mesmo processo para realizar os dois casos. Inicialmente todos os requisitos e dados obtidos nesse momento estão em nível de sistema. Quando necessário esse processo também deve ser executado para cada subsistema identificado, gerando um SyRS separado. Caso o sistema possua um ou mais softwares, o processo também é executado para cada deles (com os dados em nível de software) e gera um para SRS para cada.

Conforme ilustrado na figura 3, o processo é composto por duas atividades: definir requisitos de sistema e analisar e manter os requisitos de sistema. Na primeira atividade serão definidos o escopo, as fronteiras sistêmicas, as medidas de desempenho e a especificação dos requisitos; na segunda atividade será analisada a integridade dos requisitos, complementado o documento, apresentado a cobertura dos requisitos de sistema e definido como manter esses requisitos.

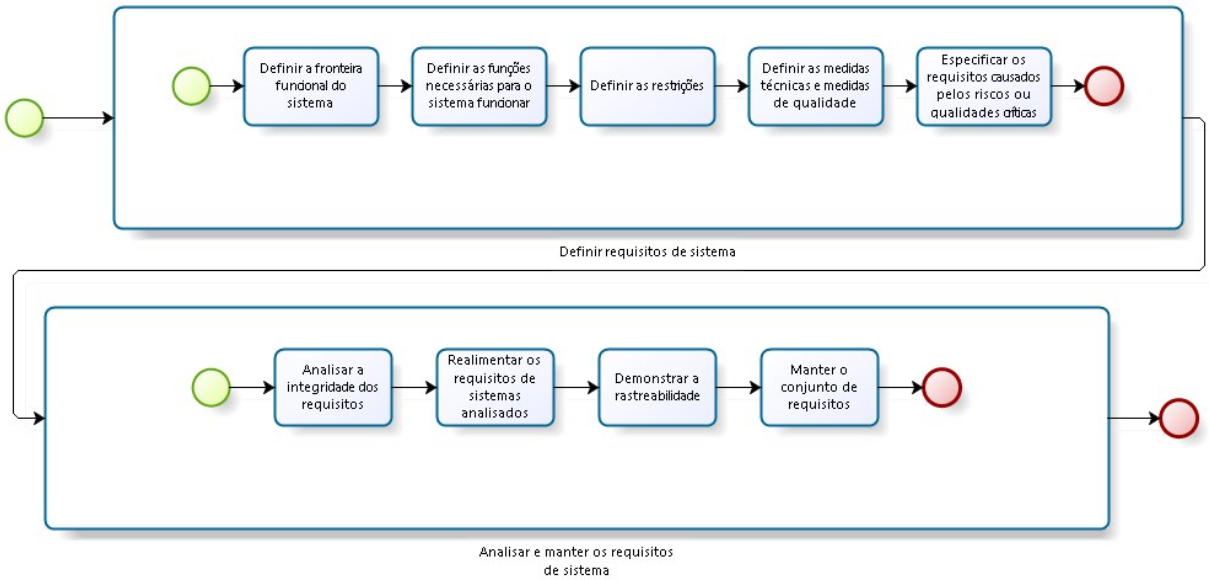

Figura 3 - Sequência de atividades e tarefas do processo de análise de requisitos Fonte: o autor.

2.2.2.1. Definir requisitos de sistema

Inicialmente deve-se definir a fronteira funcional do sistema ao agrupar o comportamento e as propriedades que serão entregues. Com isso, problemas de escopo podem ser minimizados definindo as condições das fronteiras sistêmicas antes de definir os requisitos de sistema.

Em seguida, deve ser definidas quais as funções são necessárias para o sistema funcionar. Deve ser incluso para cada função as condições sob as quais o sistema deve ser capaz de realizá-la (que pode incorporar referência aos estados e modos de operação do sistema), e determinar também as pré-condições e pós-condições do sistema ao executar cada função. Mais uma vez é importante identificar e avaliar a reutilização requisitos previamente existentes.

Com as fronteiras e funções definidas deve-se definir as restrições introduzidas por requisitos dos *stakeholders* ou por limitações inevitáveis da solução. Isso é importante para validar restrições com os *stakeholders* e garantir que as restrições foram totalmente compreendidas, antes de evoluir o conjunto de requisitos de sistema e do projeto arquitetônico.

O próximo passo é definir as medidas técnicas e as medidas de qualidade para atingir os objetivos técnicos. Segundo a norma, “as medidas técnicas são usadas para fornecer informações sobre o andamento dos elementos do sistema para atingir os parâmetros técnicos especificados nos requisitos. Estas medidas incluem medidas de desempenho (MOP) e medidas técnicas de desempenho (TPM) (ISO 29148, 2011, p. 29). A MOP é utilizada para medir uma característica física ou

um atributo funcional durante a operação do sistema. Para que as medidas sejam reais, elas devem ser medidas no ambiente operacional. A MOP é basicamente a determinação de quanto tempo o usuário precisa para realizar uma tarefa pelo sistema. A TPM é uma medida utilizada para avaliar o desempenho do sistema, o desempenho de um requisito especifico e também para avaliar os riscos das tarefas que tem desempenho críticos. As medidas são analisadas para assegurar que o sistema atende às necessidades dos stakeholders com eficiência, produtividade, segurança e satisfação em seu ambiente operacional.

Para finalizar, deve-se especificar os requisitos e funções do sistema causados pela identificação de riscos ou qualidades críticas do sistema. As qualidades críticas do sistema podem ser relacionadas com saúde, segurança, confiabilidade, disponibilidade e capacidade de suporte. Deve-se analisar os requisitos de segurança, incluindo os que são relacionados à proteção de informações confidenciais e utilizar, se necessário, as normas de segurança aplicáveis. A origem e a justificativa para cada requisito precisam ser capturadas mantendo a rastreabilidade, a qual deve ser atualizada e mantida para documentar como os requisitos do sistema satisfazem as exigências e os objetivos dos *stakeholders*.

2.2.2.2. Analisar e manter os requisitos de sistema

Essa atividade se inicia com a análise da integridade dos requisitos do sistema para garantir que cada requisito, ou que o conjunto de requisitos, possui integridade. Com isso, cada requisito de sistema deve ser verificado garantindo que seja único, completo, não ambíguo, consistente com o conjunto de requisitos, implementável e testável. Deve-se identificar requisitos de sistemas existentes que serão reaproveitados e avaliar o uso desses requisitos com base em fatores como: aplicabilidade, viabilidade, disponibilidade, qualidade, custo, valor e moeda.

Com o resultado da análise é necessário realimentar os requisitos de sistema analisados para garantir que estes requisitos refletem adequadamente as necessidades e expectativas dos *stakeholders.* Esta validação de requisitos realizada junto com os *stakeholders* é para garantir que os requisitos dos *stakeholders* foram devidamente transformados em requisitos de sistema. Podem ser utilizadas diversas técnicas para realização dessa tarefa, tais como adicionar anotações dos *stakeholders* no requisito de sistema, prototipagem, modelagem e

simulação, modelagem conceitual e modelagem formal. A técnica a ser utilizada depende do perfil do *stakeholder* que irá fazer a validação.

É importante manter o usuário conectado com a solução e para isso é necessário realizar a demonstração da rastreabilidade entre os requisitos do sistema e os requisitos dos *stakeholders*. Cada requisito deve ser rastreado com seus requisitos de nível mais baixo, com o projeto arquitetônico, com os elementos do sistema e com as entidades de validação e teste. Essa rastreabilidade dos requisitos permitirá realizar três análises: (1) a análise de cobertura, que permite validar se todos os requisitos de *stakeholders* estão no projeto e justificar cada requisito de sistema; (2) a análise de conformidade, que documenta que os requisitos de *stakeholders* foram devidamente satisfeitos; e (3) a análise de impacto, que permite avaliar os impactos das mudanças em todas as funcionalidades.

É necessário manter, durante todo ciclo de vida do sistema, o conjunto de requisitos junto com as razões associadas por trás das decisões de *design* e suposições. O conjunto de requisitos deve ter um gerenciamento, permitindo armazenar cada requisito com as informações auxiliares, decisões, suposições, histórico de alterações e a categorização. O uso de uma ferramenta de gerenciamento de requisitos permite que a manutenção da rastreabilidade dos requisitos e controle de configuração não seja trabalhosa e complexa.

2.2.3. Gerenciamento de Requisitos

Os requisitos normalmente não são estáticos e o processo de gerenciamento de requisitos inclui tarefas para gravar, manter os requisitos em evolução, manter o contexto associado a cada requisito e informações históricas do ocorrido durante os processos de engenharia de requisitos. Além disso, a gerência também estabelece procedimentos para definir, controlar e publicar a *baseline* de requisitos. A *baseline* de requisitos é o conjunto de requisitos que será atendido na nova versão do sistema.

O gerenciamento de requisitos é divido em duas tarefas: a gestão de mudança, que irá realizar um controle de cada mudança do requisito, e a medição de requisitos, que irá medir o processo de requisitos.

2.2.3.1. Gestão de Mudança

Os impactos das alterações ou dos novos requisitos devem ser avaliados para assegurar que a intenção inicial da *baseline* de requisitos seja mantida ou que as alterações sejam compreendidas e aceitas pelo adquirente. Em quase todos os casos, os requisitos continuam a evoluir à medida que as atividades do ciclo de vida do projeto caminham.

Qualquer que seja a causa de mudanças de requisitos, é importante reconhecer a inevitabilidade da mudança e adotar medidas para mitigar os efeitos da mudança. A mudança tem de ser gerida, garantindo que as alterações propostas passam por uma avaliação de impacto, revisão e processo de aprovação, aplicando rastreabilidade e gerenciamento de versões.

A gestão da mudança é composta por duas atividades: gerenciamento da configuração e gerenciamento da informação. O objetivo do gerenciamento da configuração é estabelecer e manter a integridade de todas as saídas do projeto e do processo, e tornar essas saídas disponíveis para os *stakeholders* corretos. Já o gerenciamento da informação tem por finalidade fornecer, quando necessário, as informações confidenciais relevantes para a parte designada durante, ou após, o ciclo de vida do sistema.

2.2.3.2. Medição dos requisitos

O processo de medição dos requisitos deve coletar, analisar e relatar os dados relativos aos produtos e processos, apoiando a gestão do processo e a qualidade dos produtos. Essa medição deve ser feita em conformidade com o processo de medição organizacional.

Para atingir os objetivos desse processo é necessário realizar duas atividades: definir o plano de medição e executar a medição. Para definir o plano de medição são determinadas quais características da organização são relevantes para serem medidas, identificadas e priorizadas. Após determinar as características relevantes é determinada a data para coleta, análise e o relatório das medições. Em seguida é executada a medição que deve integrar os procedimentos definidos no plano aos processos necessários, coletando, armazenando e verificando os dados, analisando os dados e desenvolvendo os produtos de informação, e por fim, documentando e comunicando os resultados das medições.

2.2.4. Atividades de engenharia de requisitos em outros processos técnicos

Existem atividades de requisitos em outros processos do ciclo de desenvolvimento de sistema mais especificamente nos processos de arquitetura, verificação e validação de software.

Durante a definição do projeto arquitetônico, as funções identificadas nos processos de requisitos são particionadas e alocados aos elementos arquitetônicos. Além disso, os requisitos são levados em consideração na análise e avaliação da arquitetura, visando uma iteração mais eficaz, eficiente e confiável. As atividades de arquitetura podem gerar requisitos derivados e devem complementar a documentação de requisitos. É importante sempre manter a rastreabilidade entre os requisitos e o projeto arquitetônico.

O propósito da verificação é garantir que os requisitos elicitados são completamente atendidos pelo sistema. A verificação define como, onde e quando o cumprimento de cada requisito será comprovado para a aceitação do adquirente. Isto inclui também os critérios de sucesso e o fechamento do projeto. A rastreabilidade dos requisitos é frequentemente usada como um ponto importante para traçar um caminho da fonte do requisito e avançando através do ciclo de vida do desenvolvimento para avaliar se o requisito foi atendido.

A validação do sistema confirma que o sistema que foi construído satisfaz os requisitos dos *stakeholders,* e é o sistema certo. A validação também pode ser realizada para confirmar que o sistema não só satisfaz todos os requisitos, mas também expressa e às vezes substitui as atitudes e experiências que compõem a satisfação do cliente.

2.3. Artefatos de requisitos

Os artefatos gerados pelo processo de engenharia de requisitos são:

- Conceito da Operação (ConOps)
- Documento de especificação de requisitos de Stakeholder (StRS)
- Documento de especificação de requisitos de Stakeholder (StRS)
- Documento de especificação de requisitos de software (SRS)
- Conceito operacional do sistema (OpsCon)

A seguir cada um desses artefatos será apresentado, enfatizando o seu conteúdo segundo a norma.

2.3.1. Conceito da Operação (ConOps)

O documento, *Concept of Operations* (Conceito da Operação - ConOps) apresenta as intenções das lideranças do projeto em relação à forma como será conduzida a operação da organização, ou seja, será descrito no documento as pretensões ou intenções da empresa no que diz respeito à operação completa da organização ou a um grupo de operações que utilizarão o sistema desenvolvido, sistemas existentes e possíveis sistemas futuros. Este documento é frequentemente incorporado em planos estratégicos de longo prazo e planos operacionais anuais.

Com isso o ConOps serve como base para a organização dirigir as características gerais do futuro empresarial e dos sistemas, e para que o projeto entenda seus fundamentos e direcione a elicitação de requisitos.

Este documento contém:

- Propósito do documento;
- Escopo;
- Plano estratégico;
- Eficácia;
- Contexto da operação global;
- Lista sistemas;
- Unidades organizacionais envolvidas;
- Políticas de governança;
- Unidades responsáveis pela governança;
- Plano de investimento;
- Gestão de ativos de informações;
- Segurança;
- Plano de continuidade de negócios; e
- Políticas de cumprimento do documento.

2.3.2. Documento de especificação de requisitos de *Stakeholder* (StRS)

O documento de especificação de requisitos de Stakeholder (StRS) determina a motivação da organização para alterar ou desenvolver o sistema. Ele define o processo, as políticas e as regras de como o sistema será utilizado no contexto da organização. Além disso, descreve como a organização está buscando novas

oportunidades ou mudando o negócio atual, e como utilizará o sistema como um meio de contribuir para esta mudança.

O StRS contém:

- Descrição do ambiente organizacional;
- Metas e objetivos;
- Modelo de negócio;
- Modelo operacional;
- Modos de operação;
- Qualidade operacional;
- Formação organizacional;
- Concepção do sistema proposto;
- Características necessárias;
- Contexto de utilização das funções;
- Serviços dos produtos;
- Conceitos operacionais;
- Restrições sobre a solução;
- Rastreabilidade do solicitante de cada requisito;
- Quando a necessidade de um *stakeholder* é atingida;
- Requisitos de *stakeholders*; e
- Requisitos de validação dos *stakeholders*.

2.3.3. Documento de especificação dos requisitos de sistema (SyRS)

O objetivo do Documento de especificação dos requisitos de sistema (SyRS) é fornecer uma descrição do que o sistema deve fazer, em relação às interações ou interfaces com seu ambiente externo. Deve descrever completamente todas as entradas, saídas e as relações entre elas. Ele é um documento que comunica os requisitos dos *stakeholders* para a equipe técnica, a qual irá especificar e construir o sistema. Ele pode incluir modelos conceituais para ilustrar o contexto do sistema, os cenários de uso, as principais entidades de domínio, os dados, as informações e fluxos de trabalho.

O SyRS contém:

- Atributos;
- Requisitos funcionais e não funcionais;
- Restrições do projeto;
- Meios para realizar o projeto de arquitetura;
- Integridade e rastreabilidade dos requisitos de sistema; e
- Verificação se os requisitos de sistema foram atendidos.

2.3.4. Documento de especificação de requisitos de software (SRS)

O Documento de especificação de requisitos de software (SRS) define todos os recursos que são aplicáveis e necessários ao software. Ele deve documentar as condições e restrições sob as quais o software tem para ser executado, e as abordagens para verificação dos requisitos. O documento deve concordar e complementar o SyRS e indicar a precedência e criticidade de requisitos.

O SRS contém os mesmos elementos que o SyRS, porém são voltados apenas para a parte do software do sistema. Ou seja, são mais específicos, de menor abstração e abrange apenas informações referentes ao software e não do sistema como um todo conforme o SyRS.

2.3.5. Conceito operacional do sistema (OpsCon)

O Conceito operacional do sistema (*System Operational Concept* - OpsCon) é um documento que descreve o que sistema vai fazer pela perspectiva do usuário. Por ser um documento orientado ao usuário, ele não deve detalhar como fazer ou porque fazer, mas deve determinar as características que o sistema deve ter na visão do usuário.

O OpsCon e o SyRS devem ser produzidos separadamente para que não haja nenhuma influência de elementos, incluindo requisitos do sistema no OpsCon. Assim o OpsCon pode concentrar todas as tarefas de engenharia de requisitos do ponto de vista do usuário. Para atingir um entendimento adequado, o documento deve utilizar de ferramentas, vocabulários, ilustrações que são familiares os conhecimentos e experiências do usuário.

O documento contém:

- Determinação do escopo;
- Documentos de referência;
- Sistema ou situação atual;
- Justificativa e natureza das mudanças;
- Conceitos para o sistema proposto;
- Cenários operacionais;
- Resumo dos impactos; e
- Análise do sistema proposto

O processo de engenharia de requisitos é executado por diversas metodologias, inclusive por metodologias ágeis. No próximo subcapitulo serão descritos os conceitos de metodologia ágil e o detalhamento de uma metodologia, o eXtreme Programming (XP).

2.4. Metodologias Ágeis

As metodologias de desenvolvimento de software utilizam os processos de Engenharia de Requisitos como parte do seu processo global. Um conjunto de metodologias são os métodos ágeis. Segundo o Manifesto Ágil (2001), “desenvolvimento ágil de software é um termo genérico para um conjunto de métodos e práticas com base nos valores e princípios expressos no Manifesto Ágil”. Os valores, definidos no Manifesto Ágil (2001), que norteiam esses métodos determinam que se deve valorizar indivíduos e interações mais que processos e ferramentas; software em funcionamento mais que documentação abrangente; colaboração com o cliente mais que negociação de contratos; responder a mudanças mais que seguir um plano.

Highsmith (2002) diz que as práticas ágeis são aplicáveis a quase qualquer projeto e que a força dos métodos ágeis é em projetos que possuem grandes incertezas: quanto mais volátil os requisitos e mais experimental a tecnologia, as abordagens ágeis têm melhores chances de sucesso. Highsmith (2002) define os métodos ágeis com três características principais: uma perspectiva “caórdica”, valores e princípios de colaboração e apenas o suficiente de metodologia.

Segundo Highsmith (2002) as organizações são “caórdicas”, ou seja, apresentam propriedades tanto do caos e quanto de ordem, o que desafia a gestão

através do planejamento linear, preditivo e práticas de execução. Há impactos da perspectiva “caórdica” nas respostas à mudança, no gerenciamento de equipes, na organização dos projetos e no trabalho do dia-a-dia. Isso cria dois resultados fora de sincronia: metas do produto são realizáveis, mas não são previsíveis; processos podem ajudar a consistência, mas não são repetíveis. Com isso, uma perspectiva caórdica baseia-se em um modelo complexo de sistemas adaptativos, o qual é descentralizado e com agentes independentes que interagem de forma auto gerenciável guiados por um conjunto de regras simples que irão gerar resultados complexos e emergentes.

A segunda característica são os valores e princípios de colaboração que fazem com que as metodologias não reflitam apenas práticas, mas também como as pessoas interagem dentro das metodologias. Essa interação ocorre através da cultura de colaboração. A cultura de colaboração inclui as pessoas, seus clientes, a gestão e parcerias para dentro da equipe de desenvolvimento, tentando capitalizar os pontos fortes e únicos de cada indivíduo, adaptando o processo para as pessoas. Para realizar a colaboração é necessária uma interação face-a-face, facilitando o entendimento comum dos projetos mais complexos, com isso os valores e princípios exaltam a colaboração entre todos envolvidos.

Para Highsmith (2002) uma definição de metodologia é a forma que as pessoas trabalham juntas em um projeto e incluem papéis, atividades, processos, técnicas e ferramentas. As metodologias ágeis são focadas na prática, ou seja, no que realmente acontece na realidade e não em manuais, colocando as pessoas em primeiro lugar, com foco no conhecimento tácito. Com isso a última característica é apenas o suficiente de metodologia, que tenta responder à questão de o quanto de estrutura (metodologia) é necessária. Essa característica define que para ser ágil é necessário equilibrar a flexibilidade e estrutura, e reduzir os custos através da racionalização de estrutura incorporando a perspectiva caótica que a criatividade e a inovação ocorrem em um ambiente não completamente estruturado. O mais importante é definir quais os tipos de capacidades essenciais para a estrutura. As técnicas de engenharia de software serão implantadas dependendo da necessidade da estrutura para o projeto.

Algumas das metodologias ágeis apresentadas por Highsmith (2002) são Extreme Programming (XP), Crystal Methodologies, Adaptive Software Development (ASD), Feature Driven Development (FDD), Dynamic Systems Development Method

(DSDM), Scrum e Lean. Este trabalho trata apenas do Extreme Programming (XP), o qual será discutido a seguir.

2.5. Extreme Programming (XP)

O Extreme Programming (XP) foi criado por Kent Beck e Ron Jeffries, e entrou para o cenário de desenvolvimento de software em 1999. XP surgiu com uma proposta agressiva às metodologias então existentes e ajudou todas as metodologias ágeis a entrarem em evidência.

Para Don Wells (2009) o XP enfatiza o trabalho em equipe fazendo com que gerentes, clientes e desenvolvedores sejam todos iguais, e trabalhem como um time auto organizável para resolver os problemas, implementando um ambiente simples e eficaz que permite às equipes se tornarem produtivas. Para que tudo isso seja possível, o XP determina cinco valores, que regem toda a metodologia, doze práticas, que são guias para que toda a metodologia possa fluir, e o processo, que determina as atividades a serem feitas durante o desenvolvimento.

2.5.1. Valores

Don Wells (2009) defende ainda que o XP não é apenas um conjunto de regras, mas é uma maneira de trabalhar em harmonia com os valores pessoais e corporativos. Embora o XP contenha certas práticas disciplinadas, a sua intenção é a de promover um ambiente que permite a criatividade e comunicação. Com isso, esse ambiente passa a ser movido pelos valores, e o sucesso dos projetos não irá depender das práticas do XP, mas irá depender dos valores do XP alinhados com os princípios da sua organização. Os valores que o XP determina são (Don Wells, 2009):

- Comunicação: A equipe deve se comunicar diariamente e sempre que possível para criar a melhor solução e trabalharem juntos desde as definições até o final do projeto.
- Simplicidade: Fazer apenas o necessário e o mais simples possível, mesmo que futuramente seja necessário retrabalho para contemplar novas funcionalidades. A equipe deve pensar em o que é mais simples a ser feito que poderia funcionar agora.

- *Feedback:* Constantes validações são realizadas durante todo o ciclo, em todos os artefatos produzidos e o mais cedo possível, detectando e realizando as alterações necessárias.
- Coragem: A coragem deve ser aplicada para sempre dizer a verdade sobre os processos, estimativas, falhas encontradas e não procurar desculpas para o ocorrido. Também se deve ter coragem para realizar todas as mudanças que ocorrerão durante o projeto.
- Respeito: Todos os membros da equipe têm o mesmo valor para o projeto e são tratados igualmente. Os desenvolvedores respeitam o conhecimento do cliente e vice e versa.

2.5.2. Práticas

Beck (1999) explica que as práticas do XP são mais como orientações do que regras (que se tem que seguir obrigatoriamente) sendo, portanto, flexíveis dependendo do caso. Porém as práticas são como um sistema de orientações designados a interagir, contrabalancear e reforçar mutuamente uma a outra, o que dificulta decidir quais práticas usar e quais descartar.

As doze práticas determinadas pelo XP (Beck 1999) são:

- Jogo do planejamento: é um diálogo entre a equipe técnica e a equipe responsável pelo negócio. A equipe que representa o cliente deve decidir: o escopo, determinando o quanto de um problema precisa ser resolvido para que o sistema seja valioso em produção; a prioridade, que determina qual recurso deve ser feito em primeiro lugar; a composição do *release,* definindo o quanto precisa ser feito para que o negócio fique em uma posição melhor do que a atual; e as datas importantes do *release,* que mostram se existe alguma data importante que deva ser respeitada. Essas decisões somente poderão ser tomadas com base nas visões da equipe técnica que irá determinar a estimativa (o tempo necessário para desenvolver um recurso); as consequências (consequências técnicas das decisões estratégicas tomadas); o processo (como será organizado o trabalho e a equipe). Para isso a equipe deve receber a programação detalhada, a qual define qual história deve ser feita primeiramente dentro de um *release*.

Com isso é possível iniciar o desenvolvimento com um plano simples e continuamente ir refinando o planejamento.

- *Releases* pequenos: os *releases* devem ser tão pequenos quanto o possível contendo os recursos mais importantes. Para que isso seja feito é necessário também que o ciclo de implantação seja reduzido o quanto for possível acompanhando os *releases*.
- Metáfora: auxilia a todos do projeto a compreender os elementos básicos do sistema e seus relacionamentos. Cada projeto é guiado por uma única metáfora abrangente; as entidades serão identificadas a partir da metáfora e, conforme o desenvolvimento progride, a metáfora amadurece.
- *Design* simples: para atingir o *design* simples o software tem que executar todos os testes a qualquer momento, não possuir lógica duplicada, e ter o mínimo de classes e métodos possíveis. Cada pedaço do *design* no sistema tem que ser capaz de justificar sua existência dentro dos termos determinados acima e deve-se implementar apenas o necessário para o momento.
- Testes: qualquer recurso do sistema que não possui teste automatizado não existe. O foco dessa prática é aumentar a confiança da equipe no sistema: os testes unitários aumentam a confiança dos programadores na escrita dos códigos e a escrita dos testes funcionais pelo cliente garante o aumento da confiança das funcionalidades que serão entregues. Os testes devem tornar-se parte do próprio sistema e o resultado disso é um sistema com cada vez mais confiança de toda a equipe e capaz de aceitar alterações.
- Refatoração: os programadores devem refazer o código quantas vezes forem necessárias para que o recurso seja atendido com o *design* mais simples possível. Mesmo que seja gasto um tempo maior, a orientação é que o programador sempre busque um *design* mais simples. Pode ser necessário fazer mudanças radicais na concepção do sistema, porém deve ser feita em passos pequenos e com baixo risco.
- Programação em pares: definida por Beck (1999) como “um dialogo entre duas pessoas tentando programar (analisar, projetar e testar) e

entender juntas como programar melhor”. Em cada par uma das pessoas é quem utiliza o teclado e o mouse e tem por meta definir a melhor forma de implementar o método; o outro pensa estrategicamente se a abordagem utilizada irá realmente funcionar, em quais casos de teste pode não funcionar, e se é possível refatorar e ainda assim resolver o problema. A disposição em pares é dinâmica. Quando um desenvolvedor não está familiarizado com a tarefa, ele deve formar um par com alguém que tenha experiência.

- Propriedade coletiva: todos assumem a responsabilidade pela totalidade do sistema. Sendo assim, qualquer um que vê uma oportunidade de agregar valor em qualquer parte do código deve fazer, mesmo que nem todos saibam todas as partes igualmente. Se um par está trabalhando e vê uma oportunidade para melhorar o código, ele deve melhorá-lo. O que garante a qualidade do todo é a integração contínua.
- Integração contínua: o código é integrado e testado no máximo ao final de cada dia, garantindo a resolução de conflitos e que o código continua funcionando. Para que essa prática seja atendida é necessária uma máquina dedicada à integração. Cada dupla que integrar seu código irá rodar os testes nessa máquina dedicada e, caso algum teste falhe, a dupla sabe que o seu código que está com erro, pois o último a realizar integração deixou o sistema funcionando e testado.
- 40 horas semanais: para o XP 40 horais semanais são o ideal para que a equipe sinta-se descansada e criativa, permitindo melhor ritmo de trabalho e satisfação.
- Cliente participa da equipe: estar ao lado do cliente significa ter um cliente real sentado com a equipe, disponível para responder a perguntas, resolver discussões e definir prioridades. O cliente deve ser o usuário final, ou seja, quem realmente irá utilizar o sistema quando ele estiver em produção. A grande objeção a essa prática é que normalmente o usuário final é muito valioso para integrar o time durante todo o projeto, sendo assim os gestores devem decidir entre entregar um sistema mais cedo e com uma garantia de satisfação ou

ter uma pessoa trabalhando. Porém, ter um cliente integrando a equipe não significa que ele irá trabalhar 40 horas semanais no projeto. Apesar de o funcionário não estar fisicamente no mesmo local da empresa, é possível que ele realize grande parte das atividades diárias normais. Com o cliente próximo é possível que ele produza valor para o projeto escrevendo os testes funcionais e realizando as decisões de escopo e esclarecendo dúvidas com os programadores.

- Padrões de codificação: prática vital para o funcionamento do XP, considerando a refatoração, programação em pares e a autoria coletiva. O padrão de codificação escolhido pela equipe não pode onerar o trabalho do desenvolvedor e deve seguir uma única regra: não pode existir código duplicado. O padrão também deve enfatizar a comunicação e tem que ser adotado voluntariamente por toda a equipe.

Essas práticas não são inovadoras; a maioria delas é antiga e foi substituída quando seus pontos fracos foram expostos por práticas mais complexas. No XP os pontos fortes de uma prática cobrem as fraquezas de outra prática fazendo com elas se completem não permitindo que suas fraquezas fiquem expostas (Beck 1999).

2.5.3. Processo

A figura 4 adaptada de Wells (2009) mostra graficamente, através de um diagrama BPMN, como ocorre o processo do XP. Ele possui uma fase inicial em que as histórias são escritas e com base nelas é realizado o planejamento da *release*. *Release* é o conjunto de artefatos que serão entregues ao final do desenvolvimento de uma versão do sistema. Após essa etapa inicial são realizadas as iterações, durante as quais ocorre o planejamento da iteração e o desenvolvimento. As iterações são executadas até que a *release* seja toda desenvolvida. Ao completar a *release* ocorre uma reunião de *Feedback* para que possa analisar o que ocorreu de certo e errado durante o desenvolvimento e os possíveis pontos de melhoria no processo e no sistema. A nova versão é implantada no ambiente produtivo do cliente e retorna ao Planejamento da *release* para que possa iniciar uma nova *release* até que o sistema esteja completo.

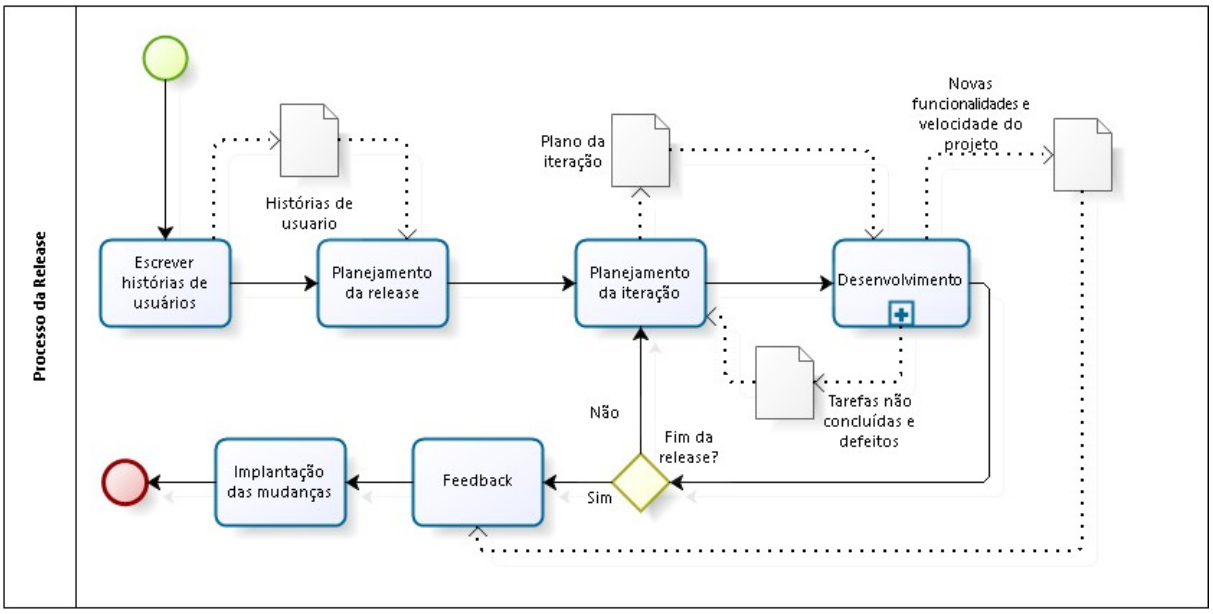

Figura 4 - Processo do XP. Fonte: adaptado de Wells (1999).

2.5.3.1. Escrever histórias de usuários

Na primeira fase do projeto são escritas as histórias de usuário, que são utilizadas para realizar as estimativas, direcionar os desenvolvedores e conduzir a criação dos testes. As histórias são escritas pelos clientes e contém o que eles precisam que o sistema faça por eles. A principal diferença entre as histórias e um documento formal de requisitos é o nível de detalhe: deve ser descrito apenas o suficiente para realizar a estimativa. Quando a história for implementada, os detalhes são passados para o desenvolvedor face a face pelo cliente. A história deve manter o foco apenas na necessidade do usuário e nos benefícios do sistema, evitando detalhes de tecnologia, base de dados e algoritmos. As histórias são escritas em cartões, com o nome da história e um parágrafo curto que descreve seu propósito.

2.5.3.2. Planejamento da release

Com as histórias escritas é realizado planejamento da *release*, através do "jogo de planejamento". O objetivo do jogo é maximizar o valor do software que será desenvolvido, usando como estratégia investir o mínimo possível para adicionar a funcionalidade mais valiosa em produção o mais rápido possível. As peças do jogo de planejamento são os cartões com as histórias escritas e os jogadores são as pessoas de desenvolvimento e de negócios. O jogo de planejamento consiste em três fases (Beck, 1999): (1) exploração, fase que descobre as novas funcionalidades para o software; (2) compromisso, quando é decido quais funcionalidades serão

implementadas na *release*; (3) e conduzir, que irá guiar o desenvolvimento pelas realidades do plano.

Na fase de exploração o cliente, que participa da equipe, descreve a história e o time de desenvolvimento estima a história considerando como prazo máximo três semanas. Caso o time de desenvolvimento não consiga estimar, ele pede para o cliente detalhar ou dividir a história. Quando uma história não pode ser estimada, pois daria mais de três semanas, o cliente tem que dividir a história em duas ou mais. Na fase de compromisso são realizadas quatro etapas:

- O cliente classifica as histórias pelo valor que ela possui para ele;
- O time de desenvolvimento classifica cada história pelo risco;
- É definida a velocidade do projeto: o time de desenvolvimento informa ao cliente o quanto consegue produzir por mês; e
- O cliente define o escopo: O cliente irá decidir o conjunto de histórias que será implementada na *release*, e estabelecer as datas e marcos do projeto de acordo com as estimativas e velocidades já apresentadas.

A última fase do jogo ocorre durante todo o desenvolvimento do projeto. O objetivo dela é atualizar o plano com base nos aprendizados e andamento. Essa fase também possui quatro partes:

- Planejamento da iteração: no início de cada iteração o cliente determina a história que deve ser implementada nessa iteração e decompõe essa história em tarefas;
- Recuperação: o time de desenvolvimento pode ter superestimado sua velocidade, então alerta o cliente e o atualiza da nova velocidade;
- Nova história: ocorre quando o cliente identifica a necessidade de uma história mais valiosa que não está na *release*, então o cliente escreve a história e ela é estimada e substitui uma história que está na release e possui o mesmo tempo de desenvolvimento; e
- Reestimativa: quando o time de desenvolvimento percebe que as estimativas não estão de acordo com a realidade eles devem reestimar as histórias e definir a sua velocidade com base na nova realidade.

Ao final do planejamento do *release*, a equipe terá as histórias estimadas, classificadas e divididas em iterações. As iterações consistem em ciclos de planejamento da iteração e desenvolvimento que têm por objetivo realizar a entrega

das histórias planejadas para ela. No planejamento da iteração as histórias que serão desenvolvidas são decompostas em tarefas e atribuídas para as duplas e o desenvolvimento irá realizar as tarefas planejadas. Cada iteração deve ter duração de uma a três semanas somando todas as histórias e correções com maior valor para o negócio.

2.5.3.3. Planejamento da iteração

O planejamento da iteração se inicia com a decomposição das histórias que foram definidas durante o planejamento da *release* em tarefas para os desenvolvedores. Assim como as histórias, as tarefas são escritas em cartões, porém elas estão na linguagem do desenvolvedor e possuem todos os dados necessários para que o desenvolvedor possa estimar e realizar sua tarefa.

Para que as histórias sejam decompostas em tarefas é realizada uma reunião em que o cliente explica ao time de desenvolvimento tudo que a funcionalidade deve conter. As tarefas devem ser escritas enquanto a explicação acontece. Nesta reunião todas as dúvidas do desenvolvedor para iniciar a tarefa são esclarecidas e o cliente fica com a visão de como foi particionado o desenvolvimento.

Quem executa a tarefa deve estimar o tempo para realiza-la. Este tempo deve ser entre um e três dias; caso dure mais, a tarefa deve ser dividida, caso dure menos, a tarefa deve ser agrupada com outras. Caso o tempo das tarefas somado ao tempo total da iteração passe de três semanas então o cliente escolhe qual tarefa deve ser deixada para a próxima iteração. Caso o tempo seja menor, o cliente deve escolher outra história para adicionar à iteração. Uma vez que essa estimativa em dias das tarefas é mais precisa, ela substitui a estimativa em semanas das histórias, realizada no planejamento do release.

2.5.3.4. Desenvolvimento

O desenvolvimento é onde se emprega a maioria das práticas do XP. Com as tarefas já determinadas e estimadas durante o planejamento da iteração, cada desenvolvedor sabe quais atividades irá realizar. O processo é mais parecido com um conjunto de práticas para que o código seja bem construído do que de fato uma sequência de atividades.

Diariamente durante todo o desenvolvimento é realizada uma reunião de manhã em que todos os participantes devem estar em pé, para evitar longas

discussões. Cada participante deve relatar três coisas: o que foi feito ontem, o que será feito hoje e os problemas que estão causando atrasos. Essa reunião tem os seguintes objetivos:

- Dar a equipe visibilidade do andamento do projeto;
- Auxiliar as duplas que encontram problemas, pois o time pensa junto em uma solução e pode inclusive remanejar as duplas, conforme determina a atividade de mover as pessoas ao redor; e
- Reorganizar as atividades, determinando as próximas tarefas para as duplas que já terminam, avaliando assim os atrasos e indicando se foi alocado muito para ser feito durante a iteração.

O mover as pessoas ao redor é uma da atividade para remanejar os desenvolvedores e fazer com que todos conheçam todas as funcionalidades do sistema, auxiliando a prática de autoria coletiva. As duplas são trocadas e as atividades distribuídas devem ser de pontos diferentes do sistema para que todos possam conviver com diferentes membros da equipe e consigam conhecer todas as partes do código. Assim, quando existirem mudanças ou correções do sistema não é necessário alocar sempre a mesma pessoa para uma determinada funcionalidade, causando um possível gargalo no projeto ou perda de conhecimento caso alguém decida não fazer mais parte da equipe. O mover as pessoas ao redor também auxilia a acelerar atividades: quem tem ideias para a solução de problemas expostos durante a reunião em pé pode ser realocado, trocando a dupla. Isso não precisa ser feito somente durante a reunião em pé; a troca de dupla pode ser feita a qualquer momento, caso a dupla identifique a necessidade.

Os cartões de CRC (*Class-responsibility-collaboration*, Classe-responsabilidade-colaboração em tradução livre) são usados para representar objetos. A classe do objeto é escrita no topo, as suas responsabilidades no lado esquerdo, e as classes colaboradoras à direita. Os cartões são escritos em uma sessão onde cada um simula o problema que irá resolver no sistema, falando e movimentando os cartões mostrando como ocorre a troca de mensagens dentro das classes, e assim todos podem participar do *design* do sistema e pensar na forma mais simples de se resolver cada problema. O CRC visa fazer o papel do *design* do sistema, pois simulando os problemas são expostos e alternativas de design podem ser facilmente simuladas.

Outra técnica que o XP utiliza nesta atividade é o TDD (*Test Driven Development*, Desenvolvimento orientado a testes em tradução livre). No TDD o teste é escrito primeiro que o código, então todo código escrito já nasce com um teste unitário, criando um *feedback* imediato do código. Isso também permite dar visibilidade de quando a tarefa está completamente concluída e que os requisitos solicitados estão todos realmente atendidos pelo código. Com isso, o desenvolvedor deve pensar no que ele quer resolver, criar um teste automatizado, executar o teste verificar a falha, e então criar o código mais simples o possível que resolve a falha. Esse ciclo é repetido até que não tenha mais nada para ser testado. Com isso o *design* será pensado em uma maneira fácil de ser testado e será influenciado com o desejo de testar tudo de valor para o cliente.

Além do TDD, para codificar o XP prega que todo código produzido que será enviado para produção deve ser criado por duas pessoas, ou seja, são dois desenvolvedores utilizando o mesmo computador e produzindo um único código. Beck (1999) afirma que em sua experiência a programação em pares é mais produtiva do que a divisão de tarefas entre os dois programadores, mas identifica que essa prática é muitas vezes um ponto de atrito para as pessoas que querem adotar o XP. Segundo ele é uma prática que leva tempo para que consiga dominar. Lembrando que o primeiro valor do XP é a comunicação essa prática é uma forma de valorizar essa comunicação.

Durante a codificação o desenvolvedor pode ter dúvidas sobre os comportamentos da funcionalidade, a aplicação da uma regra ou a melhor forma de utilização da funcionalidade. Essas dúvidas são tiradas no mesmo momento que elas ocorrem com a ajuda do responsável pelo cliente. Isso só é possível, pois o cliente está próximo aos desenvolvedores e assim o cliente pode além de auxiliar os desenvolvedores, validar se os desenvolvedores possuem o entendimento correto sobre a funcionalidade.

Com o código escrito é necessário avaliar se é possível refatorar, para manter o *design* o mais simples o possível, evitar complexidade, manter o código limpo e conciso para fácil compreensão. Durante a refatoração é importante certificar-se de tudo foi expresso apenas uma vez, e validar se o que foi construído não pode ser feito de uma maneira mais simples.

O código que está sendo produzido deve ser integrado sempre que possível. A integração continua do código muitas vezes permite evitar retrabalho, pois se as

duplas estiverem trabalhando com classes que se integram, os códigos liberados podem ser reutilizados e todos podem sempre trabalhar com a versão mais recente do código. Integrações realizadas em códigos obsoletos podem causar problemas quando o novo código for liberado. Dessa forma, cada par de desenvolvedores é responsável por integrar o seu código sempre que finalizar uma tarefa evitando ou detectando cedo possíveis problemas de compatibilidade.

Enquanto os desenvolvedores geram o código, o cliente que participa da equipe deve escrever os testes de aceitação de cada funcionalidade que será entregue na iteração. A pergunta que deve ser respondida com este teste é “O que precisa ser verificado para garantir que essa história foi feita? ”. Cada cenário de teste que surge pela resposta dessa pergunta pode possuir muitas variações com diversos resultados. Essas variações são utilizadas para gerar os testes funcionais do *release*. Normalmente o cliente não possui habilidades para criar seus próprios testes, precisando de auxílio para transcrever as ideias e cenários em testes reais. Por isso é aconselhável que exista um testador dedicado. Ele tem como ponto de partida a ideia de teste do cliente, que habitualmente são superficiais, e cria todas as variações possíveis e propensas a quebrar o software.

Conforme as duplas finalizam as tarefas, elas integram o código e geram uma versão para testes. Se o teste funcional já foi escrito, o cliente irá realizar os testes de aceitação. Este teste não é executado o tempo todo, como ocorre com os testes unitários, mas ao final de cada desenvolvimento da tarefa. Porém a funcionalidade não precisa estar completamente aprovada para finalizar a iteração: os erros encontrados durante os testes funcionais são classificados e entram no planejamento das próximas iterações para que possam ser corrigidos e testados de acordo com a prioridade.

2.5.3.5. Feedback da *release*

Após a última iteração do release ocorre o feedback da *release*, onde a velocidade do projeto é atualizada; os riscos são avaliados e atualizados de acordo com a nova versão do sistema; as experiências aprendidas no projeto são compartilhadas; e deve-se avaliar as práticas e os processos utilizados, se necessário ajustar. Durante o feedback podem ocorrer ideias de melhorias para o sistema, pois o conhecimento da equipe aumenta durante o projeto. Com isso pode

ser feita uma lista, que todos têm acesso, das possíveis melhorias a serem avaliadas pelo cliente e desenvolvidas nos próximos releases.

2.5.3.6. Implantação das mudanças

Para finalizar, o que foi desenvolvido é implantado no ambiente de uso do cliente. Isso deve ser feito o quão cedo for possível e os releases devem ser tão pequenos quanto o possível, mantendo um significado completo para o cliente. Os *releases* devem ser pequenos, pois o XP prega que quando mais cedo se atualize o ambiente de produção, mais cedo será a detecção de erros e o cliente visualizando o sistema em seu ambiente de operacional viabiliza a continuação do projeto.

2.6. Considerações do capítulo

Os tópicos desenvolvidos neste capítulo auxiliam a compreensão do objetivo geral deste estudo, que é avaliar a cobertura do eXtreme Programming a Norma ISO 29148. O método de avaliação, as planilhas de comparação e forma como cada processo ou artefato é atendido são detalhados no capítulo 3.

3. AVALIAÇÃO DO ATENDIMENTO DOS PROCESSOS E ARTEFATOS DA ISO 29148 PELO EXTREME PROGRAMMING

Neste capítulo será apresentado como o eXtreme Programming (XP) atende aos processos e artefatos da norma ISO 29148 (2011).

A norma define o mesmo processo para requisitos de software e requisitos de sistema e o XP aborda apenas sistemas intensivos de software. Segundo a IEEE 1471 (2000, p. 1), sistema intensivo de software é “qualquer sistema em que o software contribua de forma essencial na concepção, construção, implantação e evolução do sistema como um todo”. Sendo assim o processo da ISO que trata de requisitos de sistemas será analisado apenas sob o ponto de vista de requisitos de software. Também por esse motivo o Documento de especificação dos requisitos de sistema (SyRS) não será analisado, já que ele trata de requisitos de sistemas.

Seguindo a norma ISO 15504-2 (2003, p. 10), norma sobre avaliação de processos, o atendimento dos processos e artefatos da norma ISO 29148 pelo XP será avaliado seguindo os seguintes critérios:

- N (Não atende): Não há nenhuma evidência do processo ou artefato da norma ISO 29148 no XP.
- P (Parcialmente atendido): Há pouca evidência do atendimento. Para as atividades, há pouca descrição do que será feito no XP desta atividade da norma; para o artefato, a informação descrita pode ser obtida, mas não é de conhecimento de todos da equipe (e somente será obtida caso alguém da equipe pergunte essa informação).
- L (Atendido em grande parte): Há uma evidência clara do atendimento. Para as atividades é possível identificar exatamente o que tem que ser feito, ainda que falte uma parte da atividade. Para os artefatos, a informação contida neles é de conhecimento de todos da equipe, mas isso não significa que está necessariamente documentada.
- F (Atendimento completo): Há evidência de que tudo que a norma define é coberto pelo XP; não existem pontos relevantes que o XP não atenda. Para os artefatos deve inclusive estar documentado.

Conforme descrito na ISO 15504, mesmo com os critérios de julgamento definidos, a avaliação é subjetiva e baseia-se essencialmente no julgamento do avaliador de acordo com o entendimento das provas obtidas.

Para realização da análise do atendimento do XP aos processos determinados pela norma 29148 (2011) foi examinado qual prática ou etapa do processo do XP cobre cada tarefa do processo analisado. Na análise do atendimento aos artefatos determinados pela norma 29148 (2011) foi considerado a disponibilidade da informação, não somente a documentação dela, seguindo a filosofia dos métodos ágeis que estabelece que se a informação não precisa estar necessariamente escrita (o importante é ser de conhecimento de todos).

Nas seções seguintes será apresentada as analises de atendimento do XP realizadas iniciando pelos processos e depois os artefatos, cada processo ou artefato analisado conterá uma tabela que apresenta as tarefas (para os processos) e capítulos (para artefatos) da norma, qual o correspondente para o XP e o nível de atendimento. Além da tabela também são apresentados os pontos levados em consideração para qualificar o atendimento.

3.1. Processo de definição dos requisitos de stakeholders

Este processo tem foco no cliente e a elicitação de requisito é realizado diretamente com o cliente. Por isso o XP atende o processo quase que totalmente, grande parte com a escrita das histórias, conforme apresentado na Tabela 1.

Na primeira atividade do padrão (“identificar as classes de *stakeholders* e os *stakeholders”*) as duas tarefas são atendidas parcialmente pelo XP, sendo que a primeira tarefa é “identificar as classes de *stakeholders* e os *stakeholders”*. Durante a escrita das histórias os stakeholders são identificados, porém a história não contém as classes de *stakeholders*: são identificados apenas os *stakeholders* que utilizam o sistema, sem descrever as responsabilidades e a posição que eles ocupam. Por isso, apesar de o XP identificar os *stakeholders* o atendimento à norma é parcial. Para “elicitação dos requisitos dos *stakeholders”*, a segunda atividade, a escrita das histórias de usuário atende em grande parte, pois apesar de poder definir todos os elementos que a norma determina, somente será escrito o essencial para que a equipe saiba o que precisa ser feito e possa estimar a história, ou seja, se existirem comportamentos que não impactam a estimativa, mas são relevantes para o cliente, não serão detalhados nesse momento nas histórias.

Tabela 1 – Avaliação do atendimento do processo de definição dos requisitos de *stakeholders* pelo XP

Atividades	Tarefas	XP	Nível
Elicitar requisitos de *stakeholder*	Identificar as classes de *stakeholders* e os *stakeholders*	Escrever histórias de usuário	P
	Elicitar requisitos dos *stakeholders*	Escrever histórias de usuário	L
Definir requisitos de *stakeholder*	Definir as restrições da solução sistêmica	Escrever histórias de usuário	F
	Definir cenários de uso	Escrever histórias de usuário	F
	Identificar as interações entre usuários e o sistema	-	N
	Especificar qualidades críticas (saúde, segurança, meio ambiente)	Escrever histórias de usuário	F
Analisar e manter os requisitos de *stakeholder*	Analisar o conjunto completo de requisitos elicitados	Planejamento da *release*	F
	Resolver problemas nos requisitos	Tirar dúvidas com o cliente	F
	Realimentar os artefatos de requisitos com as alterações realizadas	Tirar dúvidas com o cliente	F
	Estabelecer com os *stakeholders* que seus requisitos são expressos corretamente	Tirar dúvidas com o cliente	F
	Gravar os requisitos adequadamente para o gerenciamento de requisitos	-	N
	Manter os requisitos rastreáveis até as necessidades	Tirar dúvidas com o cliente	P

Na atividade de “definir os requisitos de *stakeholders”*, algumas tarefas são atendidas pela escrita das histórias. A “definição das restrições sistêmicas” é realizada ao escrever as histórias, pois as restrições são elementos importantes que podem influenciar no tempo de desenvolvimento e o XP determina que as informações necessárias, que podem influenciar na estimativa das histórias, devem ser descritas. Sobre a “definição dos cenários de uso”, isso é exatamente o contexto principal da história, no qual na sua descrição é detalhado um cenário de uso que o sistema tem que ter. A diferença é que a história não é limitada apenas a isso e, com isso, a escrita da história atende essa atividade em sua totalidade. A tarefa de “identificar as interações entre usuários e o sistema” não é atendida pelo XP, pois não existe no XP uma tarefa que possui uma abordagem completa do sistema, incluindo os usuários, hardware, software, o contexto operacional e as interfaces. Assim como as restrições sistêmicas, a “definição das qualidades críticas”, quando essas qualidades influenciam o desenvolvimento, é realizada na escrita das histórias

de usuário, pois são elementos que devem ser levados em consideração durante o desenvolvimento podendo influenciar em todo o sistema e as estimativas das histórias.

Em relação à atividade de “analisar e manter os requisitos de *stakeholder”*, a tarefa de “analisar o conjunto completo de requisitos elicitados” é realizada durante o planejamento da *release*. Durante o planejamento é analisado o conjunto completo das histórias, que representam os requisitos de usuário para o XP, e do mesmo modo que a atividade, as possíveis inconsistências nas histórias podem ser localizadas e corrigidas quando necessário. As três próximas tarefas dessa atividade, “resolver problemas nos requisitos”, “realimentar os artefatos de requisitos com as alterações realizadas” e “estabelecer com os *stakeholders* que seus requisitos são expressos corretamente”, são realizadas pelo XP em mais de um momento, quando o desenvolvedor tira dúvidas com o cliente, pois quando o desenvolvedor percebe uma falha na história, durante a codificação ou nas fases de planejamento, irá conversar com o cliente para resolver o possível problema com a história atendendo a atividade de resolver problemas nos requisitos. Na conversa com o cliente pode-se gerar uma nova história, alterar a história ou alterar o escopo e estimativa da história, o que atende as atividades de “realimentar os artefatos de requisitos com as alterações realizadas” e “estabelecer com os *stakeholders* que seus requisitos são expressos corretamente”. A tarefa de “gravar os requisitos adequadamente para o gerenciamento de requisitos” não é atendida pelo XP, pois o XP não define que as histórias e tarefas devam ser armazenadas. Para “manter os requisitos rastreáveis até as necessidades”, o cliente deve estar junto com a equipe de desenvolvimento, ou seja, a qualquer momento ele pode esclarecer as necessidades que motivaram as alterações que serão realizadas no sistema. Porém como não existe uma escrita formal desse rastreamento e não é de conhecimento comum da equipe, toda a tarefa é parcialmente atendida pelo XP.

3.2. Processo de análise de requisitos

Este processo é mais próximo do software e contém dados mais específicos e próximos ao desenvolvimento de software. No XP as tarefas deste processo são atendidas, assim como no processo anterior, principalmente pelas atividades de planejamento, mas também são atendidas pela execução do teste de aceitação,

planejamento do teste de aceitação e tirar dúvidas com o cliente. A Tabela 2 apresenta quais tarefas do XP atendem as tarefas deste processo.

Tabela 2 – Avaliação do atendimento do processo de análise de requisitos pelo XP

Atividades	Tarefas	XP	Nível
Definir requisitos de sistema	Definir a fronteira funcional do sistema	Planejamento *release*	P
	Definir funções necessárias para o sistema funcionar	Planejamento *release*	F
	Definir restrições introduzidas por requisitos dos stakeholders	Planejamento *release*	F
	Definir as medidas técnicas e medidas de qualidade	Execução do teste de aceitação	P
	Especificar requisitos e funções de riscos ou qualidades críticas	Planejamento *release*	F
Analisar e manter os requisitos de sistema	Análise da integridade dos requisitos do sistema	Especificação dos testes de aceitação	F
	Realimentar os requisitos de sistemas analisados	Especificação dos testes de aceitação	F
	Demonstrar rastreabilidade entre os requisitos	-	N
	Manter o conjunto de requisitos e as razões associadas	-	N

A primeira atividade do processo, “definir a fronteira funcional do sistema”, é um agrupamento dos requisitos de acordo com o comportamento e as propriedades comuns, para o XP o agrupamento das histórias desta maneira é realizado durante o planejamento da *release.* Porém XP não é claro em como determinar o ambiente que o sistema está envolvido e por isso essa atividade é atendida parcialmente.

A tarefa seguinte, “definir funções necessárias para o sistema funcionar”, é atendida em sua totalidade pelo planejamento da *release* no jogo de planejamento. Nessa prática, o cliente classifica as histórias em três tipos: aquelas que o sistema não funciona sem; não essenciais, mas que tem um valor comercial significativo; e aquelas que seriam bom ter. Portanto, essa classificação define as funções necessárias para o sistema funcionar.

A “definição das restrições introduzidas por requisitos dos *stakeholders* ou por limitações inevitáveis de solução” deve ser feito no XP durante as atividades de planejamento, tanto da release como da iteração. Nesse momento é de reponsabilidade da equipe técnica determinar as consequências e dar visibilidade para o cliente das consequências técnicas das decisões estratégicas tomadas na criação das histórias, por isso o atendimento dessa tarefa pelo XP é total.

A tarefa de “definir as medidas técnicas e medidas de qualidade” possui dois elementos importantes: as medidas de desempenho (MOP) as e medidas técnicas de desempenho (TPM). Apesar de o processo do XP não determinar a medida de nenhuma das duas, como são medidas que estão relacionadas ao desempenho das funções do usuário e do sistema, elas são avaliadas pelo cliente durante os testes de aceitação. Como essa tarefa é depende da percepção do cliente ao realizar o teste de aceitação e não de uma declaração formal do tempo a ser atingido, essa tarefa é parcialmente atendida pelo XP.

A “especificação de requisitos e funções de riscos ou qualidades críticas” é realizada durante o planejamento da iteração, quando as histórias são decompostas em tarefas. As tarefas devem conter todos os elementos necessários para realizar o desenvolvimento, inclusive as qualidades críticas. Caso essa qualidade crítica exija um tempo para ser implementada, de um a três dias, deve ser considerada como uma tarefa a parte.

As tarefas de “análise da integridade dos requisitos do sistema” e “realimentar os requisitos de sistema analisados” são realizadas durante a especificação dos testes de aceitação, quando o próprio cliente valida as tarefas que foram escritas e escreve os termos necessários para que aquela tarefa atenda a suas necessidades. Com isso ele já valida o conjunto e completa com as informações necessárias as tarefas. Por causa disso as duas tarefas são atendidas em sua totalidade

A “demonstração da rastreabilidade entre os requisitos” é uma tarefa para que a equipe de desenvolvimento apresente os requisitos de sistema e como eles foram derivados a partir dos requisitos de *stakeholders* para o cliente. Como o cliente é parte da equipe no processo do XP, essa apresentação não é realizada, pois o próprio cliente ajudou a derivar todos os elementos necessários durante o desenvolvimento. Com isso essa atividade não é atendida pelo XP

Na última tarefa deste processo, “manter o conjunto de requisitos e as razões associadas”, a norma determina que seja armazenado cada requisito de sistema com seu histórico de mudanças, as decisões e sua rastreabilidade durante todo o ciclo de vida do sistema. Para o XP os requisitos de sistema são as histórias e o XP não determina como as histórias devem ser armazenadas durante todo o ciclo de vida do sistema. Elas são escritas nos cartões, porém não é descrito o que deve ser feito com os eles após o desenvolvimento. Ou seja, O XP não determina se os cartões devem ser armazenadas após o ciclo de desenvolvimento.

3.3. Gerenciamento de Requisitos

O processo de gerenciamento de requisitos tem como sua essência controlar a *baseline* de requisitos e medir o processo de requisitos e o XP determina práticas para realizar essas atividades. A Tabela 3 apresenta quais as tarefas do XP correspondem às tarefas da norma.

Tabela 3 – Avaliação do atendimento do processo de gerenciamento de requisitos pelo XP

Atividades	Tarefas	XP	Nível
Gestão de Mudança	Gerenciamento da configuração	Planejamento da release	F
	Gerenciamento da informação	Tirar dúvidas com o cliente e Autoria coletiva	P
Medição dos requisitos	Definir o plano de medição	Planejamento da *release*	P
	Executar a medição	*Feedback*	F

O “gerenciamento da configuração” é atendido pelo XP principalmente pelo planejamento da release onde ocorrem mudanças nas histórias ou novas histórias são adicionadas. É durante o planejamento que será viabilizado em qual release devem ser realizadas as mudanças. O “gerenciamento da informação” o XP atende apenas parcialmente. Quando uma alteração é detectada, isso ocorre ao desenvolvedor tirar dúvidas com o cliente, a autoria coletiva permitirá que todos da equipe saibam das mudanças ocorridas durante o desenvolvimento. Porém cabe ao representante do cliente comunicar aos *stakeholders* externos, se necessário, e o XP não termina uma tarefa especifica para isto.

As duas tarefas que compõem a medição de requisitos são realizadas pelo XP, porém de forma menos abrangente. Durante a “definição do plano de medição” a norma determina que deve ser avaliado o processo, produto e qualidade, e no XP a única medida determinada é a velocidade do desenvolvimento. A velocidade é apresentada durante o planejamento da *release*, apesar de ter processos de qualidade e da qualidade ser aprovada pelo cliente, não existe medição da qualidade, e também não é medido o produto software. Com isso o atendimento dessa tarefa pelo XP é parcial. Já a atividade de “executar a medição” é realizada por completo, pois a única métrica definida (velocidade do projeto) é medida ao final do projeto na tarefa de *feedback* do projeto.

3.4. Artefatos de requisitos

Os artefatos determinados pela norma também possuem referências aos artefatos do XP. Nesta seção serão analisados quais artefatos do XP contém as informações definidas pelos artefatos da norma.

Nos artefatos da norma existem elementos pré e pós-textuais que tem por objetivos facilitar a apresentação, entrega, identificar e fazer síntese do documento, dar acesso às fontes para produção do documento ou complementar o documento com elementos que apareceram durante a produção do documento. Esses elementos não são abordados pelos artefatos do XP, os quais contêm apenas os dados necessários para o desenvolvimento. Como esses elementos não são conteúdos essenciais para o desenvolvimento do software não será levado em consideração para avaliação do atendimento do documento, e serão classificados como “elemento pré ou pós-textual”.

3.4.1. Conceito da Operação (ConOps)

Os capítulos do Conceito da Operação (ConOps) determinados pela norma e como o XP contempla esses capítulos é representado na Tabela 4.

Tabela 4 – Avaliação do atendimento do ConOps pelos artefatos do XP

Capítulo	Subcapítulos	Artefato XP	Nível
1 Objetivo	-	Elemento pré ou pós-textual	
2 Escopo	-	Elemento pré ou pós-textual	
3 Plano estratégico	-	-	N
4 Eficácia	-	-	N
5 Operação global	5.1 Contexto	-	N
	5.2 Sistemas	-	N
	5.3 Unidade organizacional	-	N
6 Governança	6.1 Politicas de governança	-	N
	6.2 Organização	-	N
	6.3 Plano de investimentos	-	N
	6.4 Gestão de ativos informações	-	N
	6.5 Segurança	-	N
	6.6 Plano de continuidade de negócios	-	N
	6.7 Conformidade	-	N

Este documento não é atendido pelo XP. O XP trata do software partindo do princípio que o cliente já tem claramente definido a necessidade de um software. E o

ConOps é um documento que é especificado antes de iniciar o processo de desenvolvimento do software, que permite entender as intenções do cliente.

3.4.2. Documento de especificação de requisitos de Stakeholder (StRS)

Os capítulos do Documento de especificação de requisitos de Stakeholder (StRS) determinados pela norma e como o XP contempla esses capítulos é representado na Tabela 5.

Tabela 5 – Avaliação do atendimento do StRS pelos artefatos do XP

Capítulo	Subcapítulos	Artefato XP	Nível
1. Introdução	1.1 Objetivo comercial	Dúvidas com cliente	P
	1.2 Escopo do negócio	Dúvidas com cliente	P
	1.3 Visão geral do negócio	Dúvidas com cliente	P
	1.4 Definições	Elemento pré ou pós-textual	
	1.5 Stakeholders	Dúvidas com cliente	P
2. Referências	-	Elemento pré ou pós-textual	
3. Requisitos de gerenciamento do negócio	3.1 Ambiente de negócios	História de usuário	F
	3.2 Meta e objetivo	-	N
	3.3 Modelo de negócio	História de usuário	F
	3.4 Ambiente de informação	História de usuário	L
4. Requisitos operacionais do negócio	4.1 Processos de negócios	Dúvidas com cliente	P
	4.2 Políticas e regras operacionais	Dúvidas com cliente	P
	4.3 Restrições operacionais do negócio	História de usuário	F
	4.4 Modos operacionais de negócios	-	N
	4.5 Qualidade operacional do negócio	História de usuário	F
	4.6 Estrutura empresarial	Dúvidas com cliente	P
5. Requisitos de usuários	-	História de usuário	F
6. Conceito de sistema proposto	6.1 Conceito operacional	História de usuário	F
	6.2 Cenário Operacional	Teste de aceitação	F
7 Limitações do Projeto	-	Estimativa das histórias	P
8. Apêndice	8.1 Acrônimos e abreviações	Elemento pré ou pós-textual	

Os “objetivos comerciais” descrevem como o sistema deve atender as metas da empresa. Para o XP, o representante do cliente que conhece as ações da empresa sabe quais metas devem ser atingidas com o novo sistema. O “escopo do negócio” e a “visão geral do negócio” são de conhecimento comum das pessoas que trabalham no cliente, e como existe uma pessoa do cliente participante da equipe, quando necessário o representando do negócio pode expor o domínio de atuação ou as divisões internas e entidades externas da empresa. O subcapítulo de “definições” é apenas para determinar as palavras que tem um significado diferente do dicionário, sendo assim é apenas um auxílio para o leitor tornando esse capítulo um elemento pré-textual. O subcapítulo de “*stakeholders”* contém uma lista com todos os *stakeholders* do software. No XP o representante do negócio que atua na equipe é encarregado de saber quem são estes *stakeholders,* porém como os *stakeholders* podem ser sistemas ou a empresa ser muito grande, há a possibilidade do representante não ter o domínio completo dessa lista e por isso o atendimento desse subcapítulo é parcial.

O capítulo de “referências” é apenas um elemento pré-textual que identifica os documentos referenciados em todo o documento.

O subcapítulo “ambiente de negócio” determina os fatores internos e externos que devem ser levados em consideração para construção do software no XP esses fatores são descritos nas Histórias de usuário. As “metas e objetivos” do sistema são atendidos pelo XP, no XP os artefatos estão direcionados (em sua maioria) para construção do software por isso analise dos objetivos e metas que o cliente quer atingir com o sistema não são detalhados com o consentimento de todos de como a arquitetura deve ser pensada e o que ela precisa atingir. No “modelo de negócio” são descritos os métodos para alcançar as metas de negócios, as histórias de usuário do XP contêm essas descrições. Para o capítulo de “ambiente de informação” deve ser descrito o portfólio de projetos, plano do sistema de longo prazo e as configurações do banco de dados, no XP estas informações são apresentadas pelo representante do negócio durante a estimativa das histórias.

Para os “processos de negócios” o representante do cliente é quem atua nos processos que serão automatizados, por isso ele tem o domínio desses processos, mas os processos não são documentados, e por isso que o atendimento do XP é parcial (a presença do cliente na equipe permite que todos tenham acesso de como o processo é realizado pelo cliente tirando dúvidas). Assim como ocorre com o

processo de negócio, as “políticas e regras operacionais” que regem o processo atual não são documentadas, mas o representante do cliente tem conhecimento delas, e quando necessário disponibiliza essas informações. As restrições impostas pelo negócio, tais como tempo de execução de uma tarefa ou registro de atividades, devem ser descritas no subcapítulo de “estrições operacionais do negócio” e no XP essas restrições são descritas nas histórias, compondo o comportamento da funcionalidade. O subcapítulo “modos operacionais de negócios” não é atendido pelo XP; esse capítulo deve conter descrição de como a operação deve reagir ao passar por situações inesperada como acidente ou desastre natural e o XP não prevê nenhum artefato para realizar essa descrição ou qualquer discussão sobre isso. No subcapítulo de “qualidade operacional do negócio” são detalhados os níveis de qualidade exigidos em uma operação comercial, para o XP as descrições desses níveis devem ser detalhadas durante a escrita da história. Como o cliente faz parte da equipe no processo do XP, a qualquer momento é possível saber sobre a estrutura organizacional, divisões e departamentos, estruturas de funções e responsabilidades, estruturas geográficas e estruturas de partilha de recursos que são as informações determinadas para o subcapítulo “estrutura empresarial”. Porém como essas informações não são documentadas pelo XP o atendimento a este subcapítulo é parcial.

O capítulo de “requisitos de usuários” é determinado os requisitos de usuário, que contém: entradas, seleções ou informações que os usuários precisam para utilizar o sistema; quaisquer resultados que os usuários necessitam do sistema; condição ou restrição que regule o uso do sistema (usabilidade). Para o XP esses requisitos são escritos nas histórias de usuário, ou seja, de acordo com a funcionalidade que será descrita deve conter, quando necessário, os mesmos dados dos requisitos de usuário.

“Conceito operacional” descreve o sistema proposto em alto nível, indicado os recursos que devem ser desenvolvidos. No XP uma história do usuário é exatamente uma descrição de um recurso que será desenvolvido, porém ela contém mais detalhes, logo se juntarmos as histórias teremos o conceito operacional. O “cenário operacional” descreve um exemplo de como o usuário irá interagir com o sistema, é o contexto de uso para realização de uma ou um conjunto de atividades, no XP os cenários operacionais são descritos na forma de teste de aceitação, que são

cenários de uso de uma funcionalidade, porém são executados os cenários de uso ao final do desenvolvimento.

No capítulo “limitações do projeto” deve ser descrito as restrições para a execução do projeto dentro do custo e cronograma. Para o XP as restrições são debatidas e levadas em considerações durante a estimativa das histórias, porém como as restrições não ficam escritas o atendimento deste capítulo pelo XP é parcial.

O último capítulo do documento é o “apêndice”, o qual contém apenas os acrônimos e abreviações utilizadas no documento, ou seja, é um elemento pós textual para auxílio na leitura.

3.4.3. Documento de especificação de requisitos de software (SRS)

Os capítulos do Documento de especificação de requisitos de software (SRS) determinados pela norma e como o XP contempla esses capítulos é representado na tabela 6 - Atendimento do SRS pelos artefatos do XP.

Tabela 6 – Avaliação do atendimento do SRS pelos artefatos do XP

Capítulo	Subcapítulos	Artefato XP	Nível
1. Introdução	1.1 Propósito	Elemento pré ou pós-textual	
	1.2 Escopo	Elemento pré ou pós-textual	
	1.3 Visão geral do produto	Elemento pré ou pós-textual	
	1.4 Definições	Elemento pré ou pós-textual	
2. Referencias	-	Elemento pré ou pós-textual	
3. Requisitos específicos	3.1 Interfaces externas	-	N
	3.2 Funções	Histórias	L
	3.3 Requisitos de usabilidade	Histórias	F
	3.4 Requisitos de desempenho	Histórias	F
	3.5 Requisitos da base de dados lógica	Histórias	P
	3.6 Restrições de design	Histórias	P
	3.7 Cumprimento das normas	Histórias	P
	3.8 Atributos do sistema de software	Histórias	P
	3.9 Informações de suporte	-	N
4. Verificação		Teste de aceitação	L
5. Apêndice	5.1 Suposições e dependências	Tarefas	P
	5.2 Siglas e abreviaturas	Elemento pré ou pós-textual	

O primeiro capítulo é um resumo do que já foi definido para o sistema, sendo assim é um elemento pré-textual para que o leitor do documento possa se situar sobre o que já foi definido. O segundo capítulo também contém apenas as referências do documento e é um elemento pré-textual.

O terceiro capítulo “requisitos específicos”, que contém os requisitos de software, é quase todo atendido pelas histórias. O primeiro subcapítulo, “interfaces externas”, não é atendido pelo XP não existe artefato que descreva as interfaces, elas serão construídas de forma dinâmica durante a codificação. As “funções” são descritas no XP pelas histórias. Cada história contém as funções necessárias para realizar um cenário operacional. Além das funções descritas, conterá os “requisitos de usabilidade” e os “requisitos de desempenho” que pertencem à história, como complementos delas esses subcapítulos são atendidos em grande parte pelo XP. Os “requisitos da base de dados lógica”, “restrições de design”, “cumprimento das normas”, “atributos do sistema de software” também serão descritos nas histórias, porém podem ser que sejam na hora da passagem das tarefas, ou seja, pode ser que o conhecimento fique com o domínio de apenas uma pessoa, fazendo com que o atendimento desses itens seja parcial pelo XP. O último subcapítulo “informações de suporte” contém informações complementares ao software tais como instruções de embalagem e transporte, informações de base para ajudar o leitor do SRS, descrição dos problemas resolvidos e com isso o XP não atende a esse subcapítulo.

A “verificação” que deve ser descrita no capitulo 4 é atendida pelo XP pelos testes de aceitação.

O último capítulo desse documento é o “apêndice” que contém as “suposições e dependências”; e as “siglas e abreviaturas”. Para as “suposições e dependências” nesse documento são consideradas apenas as que podem afetar os requisitos de software. No XP essas suposições e dependências são debatidas na apresentação das histórias e como não são, obrigatoriamente, documentadas e também não são muito mencionadas pelo XP o atendimento a esse subcapítulo é parcial. As “siglas e abreviações” é apenas um elemento pós-textual para auxílio do leitor do documento.

3.4.4. Conceito operacional do sistema (OpsCon)

Os capítulos do Conceito operacional do sistema (OpsCon) determinados pela norma e como o XP contempla esses capítulos é representado na tabela 8 – Avaliação do atendimento do OpsCon pelos artefatos do XP.

Tabela 7 – Avaliação do atendimento do OpsCon pelos artefatos do XP

Capítulo	Subcapítulos	Artefato XP	Nível
1 Escopo	1.1 Identificação	Elemento pré ou pós-textual	
	1.2 Visão geral do documento	Elemento pré ou pós-textual	
	1.3 Visão geral do sistema	Elemento pré ou pós-textual	
2 Documentos Referenciados	-	Elemento pré ou pós-textual	
3 Sistema/ situação atual	3.1 *Background*, objetivos e escopo	Dúvidas com cliente	P
	3.2 Políticas operacionais e restrições	Dúvidas com cliente	P
	3.3 Descrição do sistema/situação atual	Dúvidas com cliente	P
	3.4 Modo de operação do sistema/situação atual	Dúvidas com cliente	P
	3.5 As classes de usuário e outro pessoal envolvido	Dúvidas com cliente	P
	3.6 Ambiente de suporte	Dúvidas com cliente	P
4 Justificativa e natureza das mudanças	4.1 Justificativa para mudanças	-	N
	4.2 Descrição das alterações desejadas	História de usuário	L
	4.3 Prioridades entre as mudanças	Estimativa das histórias	F
	4.4 Alterações consideradas, mas não inclusas	*Backlog*	L
	4.5 Premissas e restrições	História de usuário	P
5 Conceitos para o sistema proposto	5.1 *Background*, objetivos e escopo	-	N
	5.2 Políticas operacionais e restrições	História de usuário	P
	5.3 Descrição do sistema proposto	Metáfora	L
	5.4 Modo de operação	Teste de aceitação	L
	5.5 As classes de usuário e outro pessoal envolvido	Dúvidas com o cliente	P
	5.6 Ambiente de suporte	Dúvidas com o cliente	P
6 Cenários operacionais	-	História de usuário	L
7 Resumo dos impactos	7.1 Impactos operacionais	Teste de aceitação	P
	7.2 Impactos organizacionais	-	N
	7.3 Impactos durante o desenvolvimento	Estimativa das histórias	P
8 Análise do sistema proposto	8.1 Benefícios	-	N
	8.2 Desvantagens e limitações	-	N
	8.3 Alternativas consideradas	-	N
9 Apêndices	-	Elemento pré ou pós-textual	
10 Glossário	-	Elemento pré ou pós-textual	

Os primeiros capítulos do OpsCon, “escopo” e “referencias”, são apenas elementos pré textuais para que o leitor do documento consiga se localizar no documento.

No terceiro capítulo é detalhada a situação ou sistema atual que será atendido com o novo sistema. Como todo esse capítulo é baseado em atividades atuais, e o representante do cliente que está na equipe deve ser quem trabalha diretamente com essa situação atual. Portanto, as dúvidas com o cliente atendem parcialmente este capítulo, pois a qualquer momento é possível solicitar essas informações, mas elas não estão devidamente documentadas.

O foco do quarto capítulo é detalhar as mudanças necessárias. O subcapítulo “justificativa para mudanças” descreve as razões para realizar as mudanças do software ou do processo existente, o XP não possui artefato que descreve as razões, por isso o subcapítulo não é atendido pelo XP. A “descrição das alterações desejadas”, segundo subcapítulo, é atendido pelo XP em grande parte com as histórias de usuário que contém as alterações desejadas. No planejamento da *release* ocorre a estimativa das histórias que prioriza as histórias, determinando qual a ordem de desenvolvimento das histórias, e com isso o XP atende completamente o subcapítulo “prioridades entre as mudanças”. O subcapítulo de “alterações consideradas, mas não inclusas” é atendido pelo XP através do *backlog* que contém todas as histórias não entraram para o desenvolvimento. As “premissas e restrições” para as alterações no XP também são descritas nas histórias de usuário, porém somente serão descritas quando devem ser levadas em consideração para aquela história, por isso o atendimento do XP é parcial.

No quinto capítulo o objetivo é descrever o software que será construído. O subcapítulo “*Background*, objetivos e escopo” apresenta uma visão geral do software que será construído, e o XP não possui um artefato que de uma descrição geral do software que será construído. As “políticas operacionais e restrições”, o segundo subcapítulo, serão descritas nas histórias de usuário conforme for necessário, como não há uma descrição clara do XP o atendimento a esse subcapítulo é parcial. O subcapítulo “descrição do sistema proposto” é realizado no XP pela metáfora que é de conhecimento de toda equipe de como o sistema é, como a metáfora não é documentada, esse subcapítulo é atendido em grande parte pelo XP. O “modo de operação”, quarto subcapítulo, é coberto pelo XP pelo teste de aceitação que também deve conter a forma com que o sistema será usado pelo cliente, com isso o

XP atende em grande parte esse subcapítulo. “Classes de usuário e outro pessoal envolvido” neste subcapítulo são descritas as classes de usuários que são determinadas pela maneira com que os usuários utilizarão o sistema. No XP para ter essa descrição é necessário que seja perguntado ao representante do cliente com isso o atendimento a esse subcapítulo é parcial. O último subcapítulo “ambiente de suporte” contém informações de como será o ambiente depois que o sistema for implantado. No XP para obter esta informação é necessário que a equipe solicite ao representante do negócio. Como essa informação não é documentada o XP atende parcialmente a este subcapítulo.

Os “cenários operacionais” são atendidos pelo XP com a história de usuário, na qual cada cenário operacional é uma história. Por isso esse capitulo é atendido em sua totalidade pelo XP.

O capítulo de “resumo dos impactos”, os “impactos operacionais” são atendidos parcialmente pelo XP com os testes de aceitação. Os testes de aceitação contêm a utilização do sistema pelos usuários, e isso demonstra como a rotina dos usuários será alterada.

O capítulo “análise do sistema proposto” realiza uma avaliação dos benefícios, limitações, desvantagens e alternativas consideradas para o sistema proposto. No XP essa análise não é realizada, pois o XP tem processos com o foco no desenvolvimento do software. Os “impactos organizacionais” não são atendidos pelo XP, pois em nenhum artefato o XP demonstra possíveis alterações na estrutura empresarial. Os “impactos durante o desenvolvimento” são apresentados para o cliente na estimativa das histórias durante o planejamento da *release*, pois são apresentados pela equipe de desenvolvimento para o representante do cliente as consequências técnicas das decisões estratégicas tomadas e isso envolve possíveis impactos que podem ocorre no cliente durante o desenvolvimento, tais como, envolvimento da equipe do cliente, impactos operacionais durantes os testes entre outros.

Os dois últimos capítulos “apêndices” e “glossário” são capítulos pós textuais contendo apenas elementos para auxílio do leitor do documento.

3.5. Conclusão da avaliação

Os processos definidos pela norma são atendidos em quase todas P. Na sua maioria, as tarefas são atendidas por completo (quatorze tarefas atendidas completamente de um total de vinte e cinco). Apenas as atividades que correspondem ao armazenamento e gravação dos requisitos não são atendidas pelo XP. O gráfico apresentado na Figura 5 contém um gráfico que resume todos os processos e suas atividades e do nível de atendimento dela pelo XP. Há uma predominância do “Completamente atendido” (F), algumas partes são parcialmente atendidas (P), e poucas não são atendidas (N) ou são atendidas em grande parte (L).

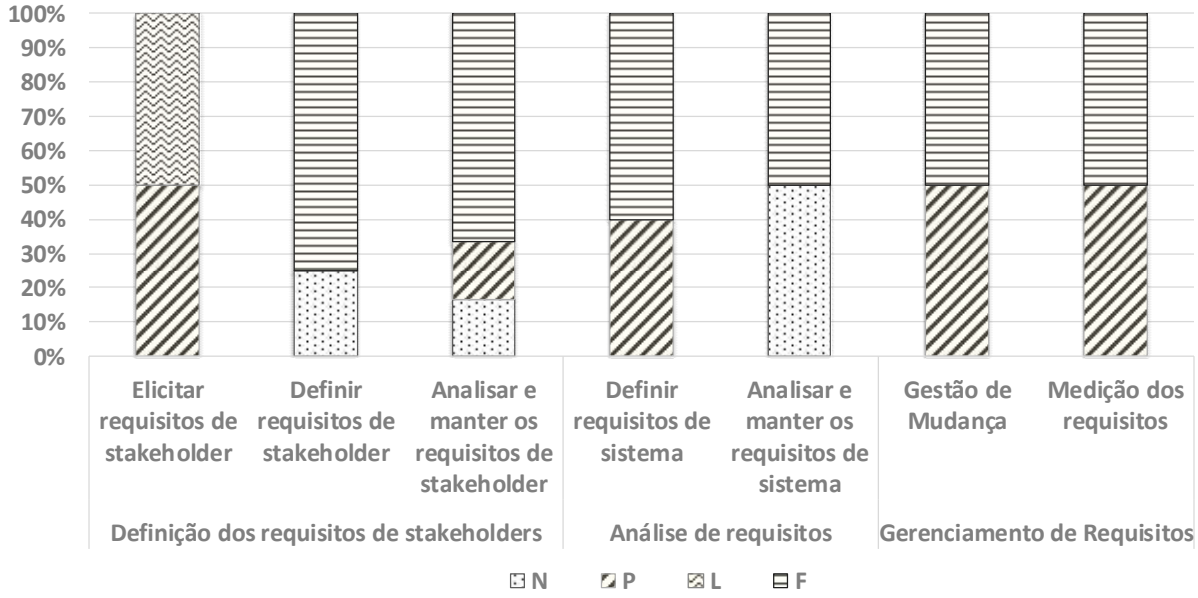

Figura 5 - Gráfico do resumo da avaliação dos processos. Fonte: o autor.

Dos artefatos propostos pela norma, o documento ConOps não é atendido pelo XP devido a uma questão temporal, pois esse documento é produzido para entendimento das necessidades do cliente antes do início do projeto, e o XP atua quando já há o entendimento do projeto. O StRS é atendido completamente na maioria dos subcapítulos. Os documentos SRS e OpsCon tem a maioria dos subcapítulos atendidos parcialmente. Em todos os documentos, os subcapítulos que são atendidos parcialmente são, em sua maioria, a representação de como é a operação atual, porém o representante do cliente que participa da equipe supre essa necessidade, apenas não é documentado formalmente como é essa situação atual. A figura 6 apresenta um gráfico com o resumo da avaliação dos artefatos, sem levar

em consideração o ConOps e os elementos textuais. No gráfico apesar de haver uma distribuição entre F, L, P e N, que representam o nível de atendimento do capítulo pelo XP, a maior parte está em P representando o “Parcialmente atendido”.

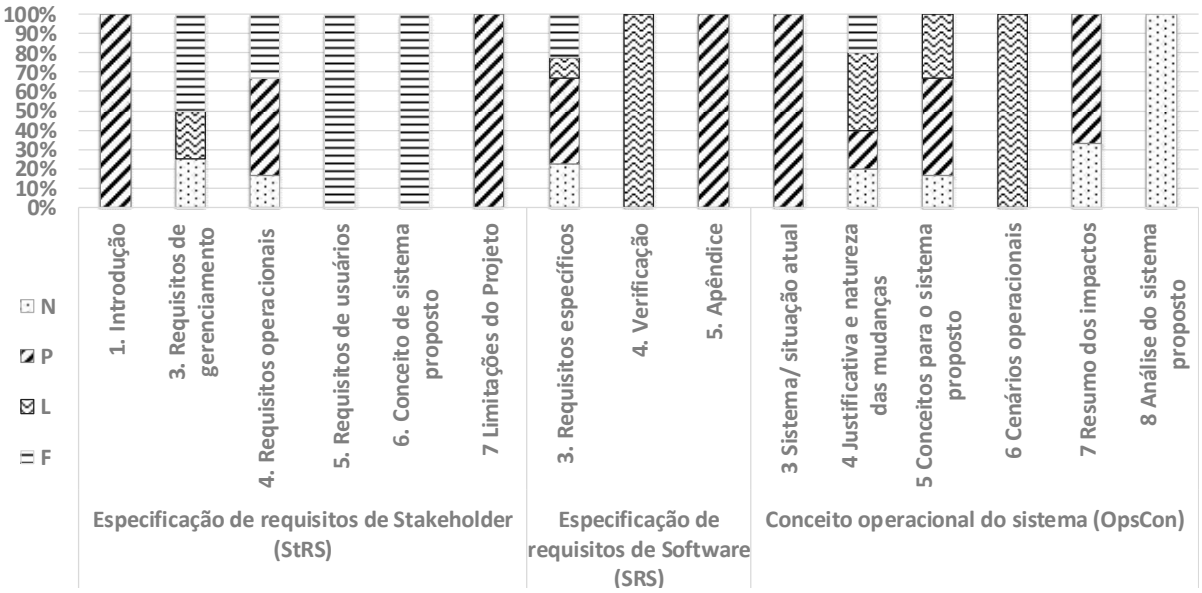

Figura 6 - Gráfico do resumo da avaliação dos artefatos. Fonte: o autor.

Como apenas quatro atividades dos processos e nove subcapítulos dos artefatos não são atendidas o XP supre, em grande parte, a norma ISO 29148.

Para que o XP atenda a todos os itens da norma, a seguir são apresentadas algumas sugestões de melhoria, organizadas por processo / artefato.

Itens não atendidos no processo de definição dos requisitos de *stakeholders*:

- Atividade: Definir requisitos de *stakeholder* - Tarefa: Identificar as interações entre usuários e o sistema. As interações poderiam ser escritas nos cartões de histórias pelo representante do cliente, pois assim completaria o sentido delas mostrando exatamente como o cliente vai utilizar o sistema para concluir essa história. Para realizar isso, existiria um impacto em relação ao trabalho do representante, que seria necessário dedicar mais tempo para isso, porém não impactaria no desenvolvimento. Com isso poderia existir uma etapa após ou durante o planejamento onde o representante do cliente descrevesse as interações entre o usuário e sistema para aquela história
- Atividade: Analisar e manter os requisitos de *stakeholder* - Tarefa: Gravar os requisitos adequadamente para o gerenciamento de requisitos. Para essa atividade Lucia e Qusef (2010) já realizaram uma sugestão para resolver este problema. A sugestão consiste em três

passos. No primeiro, o líder da equipe atribui dois ou três membros para produzir documentação em paralelo ao desenvolvimento. Os membros serão responsáveis pelo manuseio dos requisitos, redação, revisão e manutenção para manter a consistência com o desenvolvimento. No segundo, deve-se utilizar ferramentas eletrônicas para realizar a especificação das histórias, artefato que melhor representa os requisitos. No terceiro, deve-se desenvolver um processo de engenharia reversa, e utilizar esse processo para fazer com que o código produza a documentação de *design*. O impacto para implementação desses passos seria pequeno, afinal o tempo de escrita das histórias é o mesmo mudando apenas o meio, porém seria necessário um tempo para determinar uma ferramenta correta para realizar essa escrita.

Itens não atendidos no Processo de análise de requisitos

- Atividade: Analisar e manter os requisitos de sistema - Tarefa: Demonstrar rastreabilidade entre os requisitos. Para esta tarefa Lucia e Qusef (2010) também fizeram uma sugestão para atendê-la. Para eles as técnicas já concebidas para rastreabilidade não funcionam em um processo de desenvolvimento ágil pois foram concebidas para processos orientados a plano, nos quais não se tem os requisitos, os testes de aceitação, os testes unitários e a alteração de código ao mesmo tempo. Porém eles sugerem que se utilize o TDD como fonte de informação de rastreabilidade, extraindo uma matriz de rastreabilidade obtida por comparação de novos testes com alterações no código. Para isso seria necessário incluir ferramentas de rastreabilidade de requisitos junto com a ferramenta de validação. Em seguida devem ser identificados os links de rastreabilidade no ambiente TDD. Em outras palavras, as ligações de rastreabilidade entre os requisitos, casos de teste e o código relacionado devem ser identificadas e evoluídas para controlar mudanças. Desta forma, uma vez que o código é alterado, a equipe ágil é capaz de reconstruir a matriz de rastreabilidade e determinar quais são os casos de teste alterados e com isso os requisitos que foram alterados por consequência

- Atividade: Analisar e manter os requisitos de sistema - Tarefa: Manter o conjunto de requisitos e as razões associadas. Para o XP o artefato de requisitos é somente as histórias. Com isso, essa etapa seria a mesma do “Gravar os requisitos adequadamente para o gerenciamento de requisitos”.

Itens não atendidos no Conceito da Operação (ConOps):

- Neste caso o documento todo não é feito pelo XP. Para que o XP incluísse esse artefato, seria necessário que o processo iniciasse uma etapa antes do processo atual. Nessa etapa poderia ser feita uma reunião com as lideranças do projeto e o representante do cliente, onde seria apresentado para toda a equipe de desenvolvimento as intenções da empresa com a utilização do novo software e como ocorre as operações atuais que serão impactadas pelo novo software. Como resultado dessa reunião poder-se-ia gerar uma versão simplificada do ConOps, algo próximo a uma metáfora escrita, onde toda a equipe utilizaria esse artefato para direcionar as etapas tentando alcanças a intensões do cliente. Considero o impacto pequeno ao processo do XP essa nova etapa, pois é apenas uma reunião, e com um benefício de consciência coletiva das melhorias esperadas pelo cliente.

Itens não atendidos no Documento de especificação de requisitos de *Stakeholder* (StRS):

- Capítulo: Requisitos de gerenciamento do negócio – Subcapítulo: Meta e objetivo. O conteúdo deste subcapítulo poderia ser adicionado a cada história, como um tópico de cada história, mostrando cada meta ou objetivo que aquela história irá atender. O impacto para adicionar esse subcapitulo à história é pequeno, pois é um item de descrição rápida se divido para cada história.
- Capítulo: Requisitos operacionais do negócio – Subcapítulo: Modos operacionais de negócio. Para atender esse subcapítulo poderia ser utilizada a etapa descrita para suprir a tarefa “Identificar as interações entre usuários e o sistema”, apresentando junto ao diálogo, o processo que compõem o modo operacional que a história faz parte. O impacto desse subcapitulo pode ser grande, pois a descrição de um modo

operacional pode ser grande e levar um tempo considerável, logo o ideal era que fosse feito apenas para tarefas mais complexas que exigem a necessidade de detalhamento maior para compreensão da equipe de desenvolvimento.

Itens não atendidos no documento de especificação de requisitos de software (SRS):

- Capítulo: Requisitos específicos – Subcapítulo: Interfaces externas. Durante a análise dos cartões CRC, poder-se-ia adicionar aos cartões a comunicação entre as interfaces externas. O impacto dessa mudança seria pequeno, pois apesar de necessário um tempo para escrever as interfaces nos cartões, isso auxiliaria na escrita do código, o que provavelmente iria reduzir o tempo que a dupla leva para realizar a codificação, pois as interfaces já foram pensadas anteriormente.

Itens não atendidos no conceito operacional do sistema (OpsCon):

- Capítulo: Justificativa e natureza das mudanças – Subcapítulo: Justificativa para mudanças. Este subcapítulo também poderia ser adicionado a cada história, igual ao subcapítulo “Meta e objetivo”, como um tópico de cada história. O impacto para adicionar esse subcapitulo à história é pequeno, pois é uma frase que justifique a história.
- Capítulo: Conceitos para o sistema proposto – Subcapítulo: *Background*, objetivos e escopo. Esse subcapítulo poderia ser atendido pela reunião proposta para atender ao artefato ConOps. Os pontos que devem ser detalhados nesse capítulo é o mesmo conteúdo do ConOps, mas nesse subcapítulo é apenas um resumo dos principais tópicos do documento. Com isso o impacto para adicionar esse tópico é pequeno conforme já proposto.
- Capítulo: Resumo dos impactos – Subcapítulo: Impactos organizacionais e capítulo: Análise do sistema proposto - Subcapítulos: Benefícios; Desvantagens e limitações; Alternativas consideradas. Para atender a esse e os próximos subcapítulos poderia ser criado durante a atividade de *feedback* um artefato de liberação da *release* que conteria os impactos, os benefícios, desvantagens e limitações, além das discussões realizadas na reunião de *feedback.* O impacto da adição desse artefato é pequeno, considerando que ocorreria sua

construção durante o *feedback*, e que os pontos analisados para construção do artefato são de conhecimento de todos da equipe, pois irá tratar do que acabou de ser construído.

Com a utilização de todas essas sugestões, o processo do XP pode ficar mais oneroso. Com isso, além das sugestões seria necessário realizar uma avaliação do processo com essas sugestões. A avaliação do processo já faz parte do XP e é realizado durante a atividade de *feedback*, conforme já detalhado.

4. TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos que contém pesquisas realizadas por outros autores com assuntos relacionados a este trabalho. Além disso, é apresentado as principais diferenças entre eles e este trabalho.

4.1. Medeiros et al. (2016)

Medeiros et al. (2016) realizam uma análise da qualidade do SRS (documento de especificação de software) no Desenvolvimento Ágil de software (ASD). O objetivo da pesquisa é contribuir para a compreensão geral de como escrever SRS mais eficaz em projetos ágeis. Foi investigada a noção de qualidade de um SRS no contexto de ASD e foi construído um modelo explicativo que proporciona uma compreensão mais profunda sobre os fatores que afetam a qualidade de SRS neste contexto. Mais especificamente, o objetivo é encontrar respostas para a seguinte pergunta: Como é afetada a qualidade da especificação de requisitos de software no desenvolvimento ágil de software?

A principal diferença entre a pesquisa de Medeiros et al.(2016) e este trabalho é o foco da análise. Apesar de também realizar uma análise entre a norma ISO 29148 e metodologias ágeis, o foco de Medeiros et al. (2016) é a como atingir qualidade necessária do SRS de um projeto de ASD. Além disso, Medeiros et al. (2016) utiliza apenas um dos cinco artefatos citados pela norma e também não considera o processo de Engenharia de Requisitos como parte da análise.

4.2. Heikkilä, et al. (2015)

No estudo de Heikkilä, et al. (2015) é feito uma revisão da literatura considerando a engenharia de requisitos (RE) no desenvolvimento ágil de software (ASD). Foram analisados 28 artigos e os autores concluíram que a definição de requisitos no ASD é vaga. Os benefícios da RE em uma ASD são: custos gerais de processo mais baixos; melhor compreensão de requisitos; uma tendência reduzida para aumentar os recursos de desenvolvimento; capacidade de resposta às mudanças; entrega rápida de valor para o cliente e melhores relações com os clientes. E os malefícios são: uso de representantes de clientes, o formato de requisitos de história de usuário, a priorização de requisitos, a crescente dívida técnica, o conhecimento de necessidades tácitas e a estimativa de esforço

imprecisa. O estudo também apresenta propostas de soluções para os problemas identificados.

Para esse estudo foi realização uma condensação dos dados dos artigos, porém não foi analisado um processo, procurando lacunas dentro do processo ou dos artefatos gerados. Ou seja, diferente deste trabalho, Heikkilä, et al. (2015) busca as falhas de acordo com os estudos analisados, e não realizando uma comparação de processos. Com isso os resultados dos estudos deles são mais abstratos e abrangentes.

4.3. Gebhart et al. (2014)

Gebhart et al. (2014) apresenta a análise de um projeto de construção de um sistema de decisões, o qual foi desenvolvido usando método ágil. Um dos fatores de sucesso do projeto segundo os autores foi o processo de engenharia de requisitos, o qual foi baseado na norma ISO 29148 (2011) e adaptado quando necessário.

As diferenças entre o estudo de Gebhart et al. (2014) e este estudo são duas. A primeira é que Gebhart et al. (2014) adapta a norma para encaixar nas suas necessidades. Ao contrário de Gebhart et al. (2014), neste trabalho está sendo proposto utilizar a norma como prática principal e adaptar o processo do XP para atender a norma. A segundo diferença é que Gebhart et al. (2014) analisou esse processo em um projeto real – o que não é feito neste estudo.

4.4. Lucia e Qusef (2010)

Lucia e Qusef (2010) também falam sobre a relação da engenharia de requisitos com o desenvolvimento ágil. No artigo é abordado como os requisitos podem ser capturados e especificados no contexto de ágil, identificando como as técnicas e processos de engenharia de requisitos podem ser combinados com as práticas ágeis e encontrar soluções para as dificuldades. Além disso, também é discutido como resolver o problema de rastreabilidade entre os artefatos no desenvolvimento de software ágil.

As diferenças entre o artigo de Lucia e Qusef e este estudo foram: neste estudo é comparado o processo e os artefatos; na comparação de Lucia e Qusef usa-se apenas as atividades, não falando de tarefas e artefatos. Além disso, Lucia e Qusef citam diversas metodologias ágeis para realizar a comparação e não apenas uma.

4.5. Conclusão dos trabalhos relacionados.

Os quatro estudos apesar de terem o mesmo foco a relação entre a engenharia de requisitos e os processos ágeis tem abordagens diferentes. Medeiros et al. (2016) têm foco na qualidade dos requisitos, Heikkilä, et al. (2015) fazem um condensação de artigos através de uma revisão da literatura, Gebhart et al. (2014) analisa a utilização da engenharia de requisitos com processos ágeis baseado nos fatores de sucesso de um projeto e Lucia e Qusef (2010) focam em como capturar requisitos nos processos ágeis. O estudo apresentado difere de todos, pois é um mapeamento dos processos e artefatos da norma ISO 29148 (2011) para os processos e artefatos do eXtreme Programming.

Contudo é possível perceber que Medeiros et al.(2016); Lucia e Qusef (2010) e este estudo contém resultados das análises muitos similares, o que confirmam a necessidade de criar soluções alternativas para os problemas encontrados e solidifica a conclusão dos estudos.

5. CONSIDERAÇÕES FINAIS

De acordo com o objetivo do trabalho, que é analisar como o Extreme Programming (XP) atende o processo clássico de Engenharia de Requisitos, foi realizado o mapeamento identificando como os processos e artefatos descritos na norma ISO 29148 (2010) são atendidos pelo XP. Para isso a análise foi realizada na menor parte de cada elemento, ou seja, nas tarefas para os processos e nos capítulos para os artefatos. Na análise determinou-se qual o correspondente para o XP e o nível de atendimento daquela tarefa ou artefato pelo XP.

Foi possível concluir que o XP suporta eficientemente os processos e artefatos da Engenharia de Requisitos definidos pela norma. As tarefas dos processos são quase todas atendidas e em grande parte por completo. Dos artefatos propostos pela norma, o documento ConOps não é atendido pelo XP devido a uma questão temporal. Os outros artefatos também são quase todos atendidos, porém a maioria dos capítulos é atendido parcialmente.

Além disso, foram sugeridas algumas melhorias para poder suprir as quatro atividades dos processos, o ConOps e os nove subcapítulos dos artefatos não atendidos pelo XP. Porém as sugestões incluem novas etapas no processo e alterações nos artefatos gerados, o que pode ocasionar em um processo mais burocrático.

Por fim foi apresentado que, apesar de terem outros estudos que trabalham a relação entre a Engenharia de Requisitos e as metodologias ágeis, o foco muda para cada estudo. E mesmo com o foco diferente as conclusões entre os estudos são próximas, o que consolida a pesquisa realizada.

5.1. Trabalhos futuros

Conforme mencionado, as sugestões de mudanças no XP podem tornar o processo mais burocrático e talvez oneroso. Para analisar se as mudanças são positivas seria necessário realizar a implantação dessas melhorias, medir se o tempo para realizar o processo com as mudanças não aumenta muito e se as mudanças trariam os resultados esperados, tais como: melhor entendimento no inicio do projeto; permitir consultar as funções já implementadas no sistema como base de conhecimento para futuras implementações; rastreabilidade entre os artefatos justificando cada implementação; entre outras. Isso serviria para analisar

se as mudanças são viáveis dentro de uma metodologia ágil, ou seja, se trazem benefícios esperados sem onerar o processo.

Além desse possível estudo, para dar continuidade a esse mapeamento, seria possível realiza-lo com outras metodologias ágeis tais como: Crystal, Adaptive Software Development (ASD), Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), Scrum e Lean.

REFERÊNCIAS

ABDULLAH, B.N.N. et al. Communication patterns of agile requirements engineering. In: Workshop on Agile Requirements Engineering, 1., 2011, Lancaster, **Proceedings**. New York: ACM, 2011.

AGILE MANIFESTO. **Manifesto for Agile Software Development**. [S.l.], 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 15 fev. 2017.

BECK, K. **Extreme Programming Explained: Embrace Change**. EUA. Addison-Wesley Professional, 1999.

BRAY, I. **An introduction to requirements engineering**. Harlow, Reino Unido. Addison-Wesley, 2002.

EBERLEIN, A.; LEITE, J. Agile Requirements Definition: A View from Requirements Engineering. In: International Workshop on Time-Constrained Requirements Engineering, 2, 2002, Alemanha, **Proceedings**, Alemanha: 2002. p. 4-8.

GEBHART, M. et al. Quality-Oriented Requirements Engineering for Agile Development of RESTful Participation Service. In: ICSEA: The Ninth International Conference on Software Engineering Advances, 19, 2014, França, **Proceedings**, França: IARIA, 2014. p. 69-74.

GOETZ, R**.** How Agile Processes Can Help in Time-Constrained Requirements Engineering. In: International Workshop on Time-Constrained Requirements Engineering, 1, 2002, Rio de Janeiro. **Proceedings**. Rio de Janeiro: PUC, 2002. Disponível em:< http://www-di.inf.puc-rio.br/~julio/tcre-site/tcre-2.htm>. Acesso em 19 abr. 2017.

HEIKKILÄ, V. et al. A Mapping Study on Requirements Engineering in Agile Software Development. In 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 41, 2015, Portugal. **Proceedings**. Portugal: IEEE 2015, p.199-207.

HIGHSMITH, J. **Agile Software Development Ecosystems.** USA: Addison-Wesley Professional, Maio 2002.

HIGHSMITH, J. What Is Agile Software Development? **The Journal of Defense Software Engineering** p. 4-9, Out 2002.

IEEE, **IEEE Std 830: Recommended practice for software requirements specifications**. USA, 1998.

IEEE. **IEEE Std 1471: recommended practice for architectural description of software-intensice systems**. USA, 2000.

ISO/IEC. **ISO/IEC 15504-2: Software engineering — Process assessment**. Suiça, 2003.

ISO/IEC/IEEE. **ISO/IEC/IEEE 29148: Systems and software engineering — Life cycle processes — Requirements engineering**. Suiça, 2011.

LUCIA, A.; QUSEF, A. Requirements engineering in agile software development. **Journal of Emerging Technologies in Web Intelligence**, Itália, n. 3., v. 2., p.212-220, ago. 2010.

MEDEIROS et al. Towards a model about quality of software requirements specification in agile projects. In 10th International Conference on the Quality of Information and Communications Technology, 10, 2016. Portugal. **Proceedings**. Portugal: CPS, 2016. p. 236 – 241.

PAETSH, F.; EBERLEIN, A.; MAURER F. Requirements engineering and agile software development. In: WET ICE 2003. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 12, 2003. **Proceedings**., Austria: IEEE, 2003 p 309-313.

RAMESH, B.; CAO, L.; BASKERVILLE, R. Agile requirements engineering practices and challenges: an empirical study. **Information Systems Journal**, [S.l.], Blackwell Publishing Ltd , v. 20, e. 5, p. 449 – 480. Set. 2010

SAITO S. et al. **Requirements clinic: Third party methodology for improving the quality of software requirements specifications** In: IEEE International Requirements Engineering Conference (RE), 21, 2013, Rio de Janeiro, Anais... [S.l.]: IEEE, 2013 p. 290-295.

SILLITTI, A.; SUCCI, G. Requirements Engineering for Agile methods. In: AURUM, A.; WOHLIN, C. (eds). **Engineering and Managing Software Requirements**, Alemanha: Springer, 2005, p. 309-326.

VERSIONONE, **10th Annual State of Agile Survey**. [S.l.], 2016. Disponível em: <http://info.versionone.com/state-of-agile-development-survey-ninth.html> Acesso em 07 fev. 2017.

WELLS, D. **Extreme Programming: A gentle introduction**. [S.l.], 2013 Disponível em: <http://www.extremeprogramming.org/>. Acesso em: 15 fev. 2017.