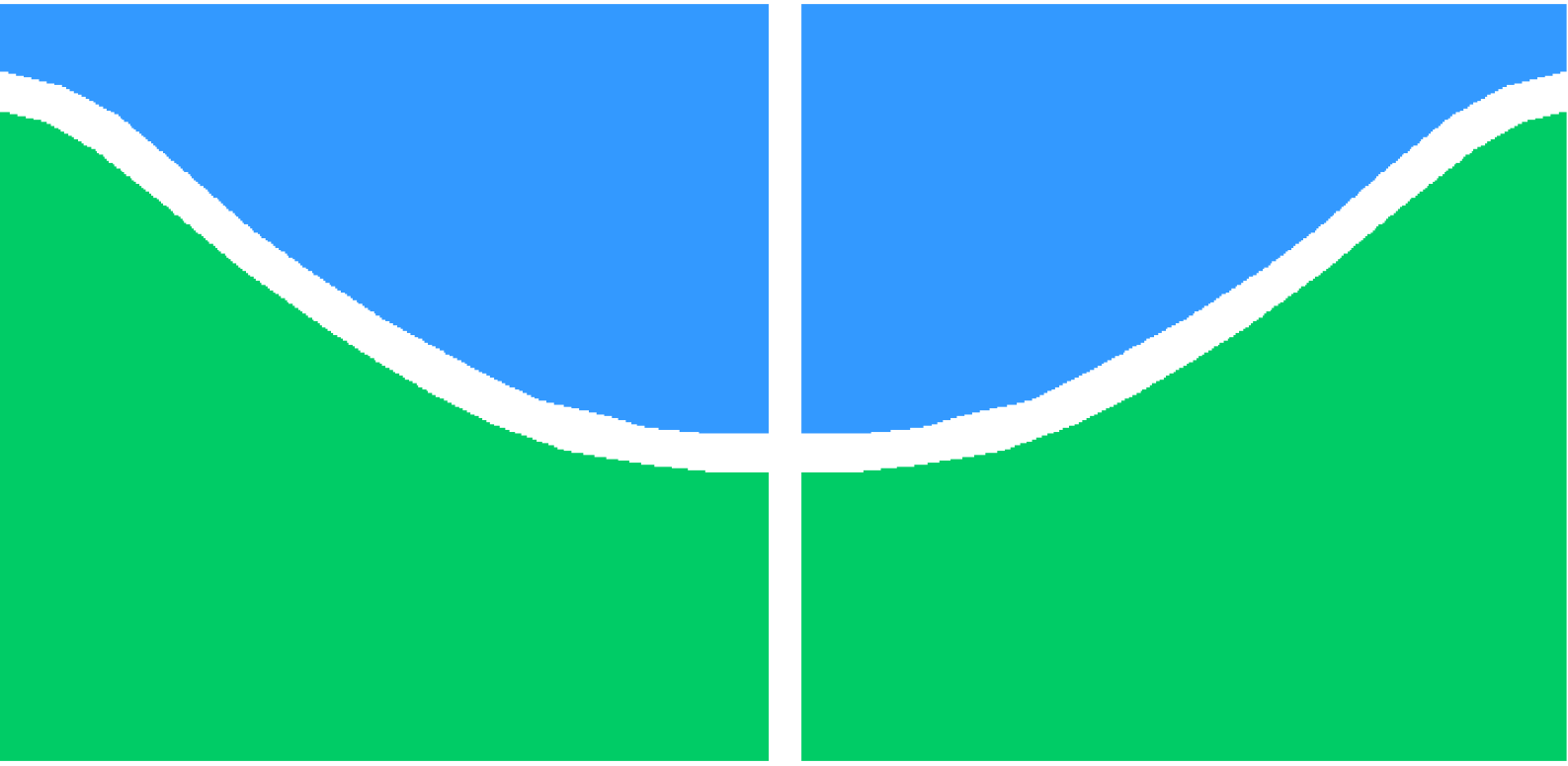

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Requisitos de Software - 201308

Relatório de Projeto

Grupo 01
Autores: Dylan, Eduardo, Wilton, Pedro
Orientador: George Marsicano Corrêa, MSc

Brasília, DF
2015

Dylan, Eduardo, Wilton, Pedro

Relatório de Projeto

Relatório referente à disciplina de Requisitos de Software, do curso de Engenharia de Software da Universidade de Brasília.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: George Marsicano Corrêa, MSc

Brasília, DF

2015

Dylan, Eduardo, Wilton, Pedro
Relatório de Projeto/ Dylan, Eduardo, Wilton, Pedro. – Brasília, DF, 2015-
Orientador: George Marsicano Corrêa, MSc

Relatório – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

I. George Marsicano Corrêa, MSc. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Relatório de Projeto

Brasília, DF
2015

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UP	Unified Process
RUP	Rational Unified Process
SAFe	Scaled Agile Framework
ER	Engenharia de Requisitos
PO	Product Owner
PM	Product Manager

Sumário

1 Introdução

Este documento apresenta considerações gerais e preliminares relacionadas à redação de relatórios de Projeto de Graduação da Faculdade UnB Gama (FGA). São abordados os diferentes aspectos sobre a estrutura do trabalho, uso de programas de auxilio a edição, tiragem de cópias, encadernação, etc.

1.1 Visão Geral do Relatório

1.2 Seções do Relatório

2 Contexto

Contexto do Ministério das Comunicações

3 Escolha da Abordagem

No desenvolvimento de um software é imprescindível a utilização de alguma abordagem que conduza seu processo produtivo.

Uma abordagem se refere a uma metodologia que será usada para estruturar, planejar, e controlar o processo de desenvolvimento do sistema (CMS, 2005).

Logo, neste capítulo será justificada a escolha de uma determinada abordagem feita pelo grupo 1 da disciplina Requisitos de Software, responsável por realizar a manutenção de sistemas de software mantidos pelo Ministério das Comunicações. Primeiro será exposto uma visão geral de cada abordagem, explicando suas diferenças e os pontos positivos e negativos presentes em cada uma delas, para que enfim uma justificativa seja apresentada. As abordagens escolhidas para representar cada uma das correntes, tradicional e adaptativa, são o Rational Unified Process (RUP) e o Scaled Agile Framework (SAFe), respectivamente.

3.1 Abordagem Tradicional - RUP

O Processo Unificado surgiu em meados dos anos 90 e tem como proposta reunir diversas práticas e filosofias que se provaram eficientes até então no contexto de desenvolvimento de software (KRUCHTEN, 2003, p. 45).

Valoriza conceitos como arquitetura de software, desenvolvimento iterativo, gerência de requisitos, construção de modelos visuais para descrever o sistema, entre outros (KRUCHTEN, 2003, p. 45). A abordagem é baseada em disciplinas, onde cada disciplina agrupa uma série de atividades relacionadas e cada atividade produz diversos artefatos, que por sua vez contêm informações do processo e são submetidos à controle de versão (são produzidos, modificados e evoluídos durante o ciclo de vida do projeto). As disciplinas representam áreas de interesse presentes no processo de desenvolvimento de software (KRUCHTEN, 2003, p. 45).

O RUP estabelece a divisão do processo em quatro fases, sendo que cada fase possui um marco (objetivo principal) que será almejado durante sua execução. As fases por sua vez são divididas em iterações que consistem em um conjunto de atividades a serem realizadas para se obter um incremento do produto. Argumenta que o planejamento a longo prazo do projeto fornece uma visão à equipe de desenvolvimento do que deve ser feito e quando deve ser feito para que o projeto seja concluído e entregue no prazo. Analisando o processo unificado sob a perspectiva da disciplina de requisitos, é possível observar que o principal recurso para se representar requisitos dentro do processo são

os casos de uso (KRUCHTEN, 2003, p. 45). Um caso de uso descreve uma interação entre uma entidade e o sistema e demonstra uma capacidade do software, assim como as funcionalidades que serão oferecidas ao usuário (KRUCHTEN, 2003, p. 124-125). Para atingir os objetivos da ER no RUP, a disciplina de requisitos descreve como definir uma visão do sistema, traduzir essa visão em um modelo de caso de uso (que, com especificações suplementares, definirá os requisitos detalhados do software), e como usar os atributos dos requisitos para ajudar a gerenciar o escopo do projeto e como mudar os requisitos do sistema (KRUCHTEN, 2003, p. 182-183).

Por fim, o RUP é um framework implementado por grandes empresas, mas determina que sua customização é aceita para que ele se adeque à realidade de organizações de menor porte, sendo possível definir as atividades e práticas do RUP que sejam do interesse da empresa em questão, desde que não sejam feridos os seus principais valores. As grandes empresas que o utilizam não o utilizam sempre da mesma maneira: enquanto algumas o utilizam de maneira formal, outras o customizam completamente (KRUCHTEN, 2003, p. 57).

3.2 Abordagem Adaptativa - SAFe

Bem mais novo, o Scaled Agile Framework (SAFe) traz ideias novas, mas tambem aspectos já consolidados na área de metodologias de software (Scrum, Kanban, etc).

Adaptável em diversos pontos, traz a ideia de dividir o projeto em três níveis: Portfolio, Program e Team. Cada nível produz seus artefatos, dispõe de diferentes papeis, tem responsabilidades, e interagem de maneira diferente (LEFFINGWELL, 2010, p. 124-125).

Uma das características mais importantes em relação ao UP, é que, enquanto o UP é dirigido a UC, possuindo assim somente ele como descritor de comportamentos desejáveis para o sistema, no SAFe, existem três níveis de abstração para descrever tais comportamentos: o Epic, a Feature e a User Story (LEFFINGWELL, 2010, p. 182-183). O Epic é bem mais alto nível, a Feature mais baixo nível que o Epic, e a User Story mais baixo nível que ambos (LEFFINGWELL, 2010, p. 182-183).

3.3 Escolha da Abordagem e Justificativa

Levando em consideração os fatos levantados, é evidente que ambas as abordagens apresentam o que há de melhor disponível na escolha de Frameworks/Abordagens de produção de Software. O grupo então optou pela abordagem ágil utilizando o SAFe.

Os principais motivos foram o fato de ser uma abordagem bem mais nova (trazendo assim a oportunidade de se ter um contato inicial com algo completamente novo), e que

ter mais níveis de projeto (Portfolio, Program e Team) e de descritores de comportamento (Epic, Feature e Story) parece ser mais interessante do que como essas coisas são lidadas no UP.

4 Engenharia de Requisitos

Na seção a seguir será dado uma pequena introdução dos três níveis de projeto do SAFe (Portfolio, Program, Team), e será dito um pouco sobre esses níveis no contexto de projeto do grupo 1. Certas adaptações foram feitas com base nas características dos integrantes de ambos os grupos (Requisitos e MPR). Na seção seguinte será falado sobre os papeis principais do SAFe no contexto do grupo. Certos papeis foram retirados (por serem julgados como desnecessários para o contexto do projeto), e, portanto, não serão citados nem explicados. Papeis utilizados serão explicados, e, caso eles sejam alterados para que funcionem de maneira diferente à referida na bibliografia, essas alterações serão explicadas e justificadas.

4.1 Scaled Agile Framework

4.1.1 Nível de Portfólio

Responsável pelos artefatos *epics* e *investiment themes*, este nível de projeto traz os requisitos com maior nível de abstração dentre os três níveis. Contém um tipo de time (*portfolio management team*), tem seu próprio backlog (*portfolio backlog*), e apresenta os conceitos de *portfolio vision* e de *architectural runway* (LEFFINGWELL, 2010, p. 227-228). No contexto do grupo 1 esse nível de projeto não será utilizado, pois julga-se o tamanho do projeto como pequeno, fazendo-se desnecessário a utilização desse nível (ele costuma ser utilizado para grandes projetos, e a não utilização desse nível não prejudicará a qualidade do processo).

4.1.2 Nível de Programa

Esse nível tem como principais objetivos manter a visão (Documento de Visão), gerenciar a *release*, gerenciar a qualidade (integrando os resultados obtidos pelos times, e garantindo que os padrões de qualidade, performance, entre outros, estão sendo assegurados), fazer o *deploy* do sistema, gerenciar recursos (ajustando prazo e gastos), e eliminar possíveis impedimentos (serão os facilitadores) (LEFFINGWELL, 2010, p. 63-64). No contexto do grupo 1, os integrantes de Requisitos desenvolverão as *features* e o Documento de Visão, que só surgem graças aos inputs fornecidos pelos integrantes de MPR.

4.1.3 Nível de Time

A unidade básica de trabalho para este nível é a *estória de usuário*. O objetivo deste nível é definir, construir e testar as estórias de usuário no escopo da iteração, afim de se concluir mais partes do produto final (LEFFINGWELL, 2010, p. 47-48). No contexto do grupo 1, somente os alunos de Requisitos fazem parte do Nível de Time, e farão todo o processo de alimentação do *backlog* de time (com base nas *features* desenvolvidas no Nível de Programa), processo de planejamento da sprint, priorização das estórias com base no WJDF, desenvolverão e testarão as soluções (com base nos critérios de aceitação).

4.2 Papéis

Um papel define comportamentos e responsabilidades de um indivíduo ou de um grupo de indivíduos que trabalham juntos como um time (KRUCHTEN, 2003, p. 61-65). O corportamento é expresso em termos de atividades que o papel pratica, e, cada papel é associado a um conjunto de atividades. No SAFe existem diferentes papéis para os diferentes níveis do sistema. No Nível de Portfólio vão haver, principalmente, papéis que irão interagir com os épicos e temas de investimento, no Nível de Programa, papéis que irão interagir com as *features*, e, no Nível de Time, papéis que irão interagir com estórias de usuário.

Nesta seção será dada uma breve definição de papeis que serão utilizados, e quem representa este papel no contexto do grupo 1.

4.2.1 Papéis no Nível de Programa

4.2.1.1 Product Manager

Responsável por entender as necessidades do cliente, documentar, priorizar e validar requisitos (no nível de *feature*), gerenciar mudanças, entre outras coisas (LEFFINGWELL, 2010, p. 283-287). No contexto do grupo 1, o Gerente do Produto (*product manager*, PM) seriam os alunos de Requisitos.

4.2.1.2 Release Management

Consistindo do Product Manager e do Release Train Engineer, o papel de Release Management é responsavel por comunicar o *status* da release aos *stakeholders*, coordenar com a gerência do produto e gerência de marketing as comunicações internas e externas, validar a qualidade relevante do produto de acordo com critérios, prover a autorização final da *release*, ajudar a adptar e inspecionar o processo de *release*, entre outras coisas (SCALED AGILE INC., 2015). No contexto do grupo 1, o Release Management será feito por quem atua no papel de Product Manager, no caso, os alunos de Requisitos, mas será

um papel flutuante, onde cada semana um aluno do grupo terá o papel de /emphrelease manager.

4.2.1.3 Business Owners

Pequeno grupo de stakeholders de grande confiança, que dispõe de capacidade de administração, responsáveis pela gestão, qualidade, implantação e desenvolvimento de software (SCALED AGILE INC., 2015). No contexto do projeto serão os alunos de Requisitos.

4.2.1.4 Release Train Engineer

4.2.1.5 System Architect

4.2.1.6 UX

4.2.1.7 Shared Resources

4.2.2 Papéis no Nível de Time

Neste nível somente aparecerão integrantes da parte de Requisitos. Os papeis interagem principalmente com as estórias de usuário, e não serão completamente fixos durante o projeto.

4.2.2.1 Product Owner

Responsável por representar o interesse de todos os envolvidos no projeto, faz parte do time e esta junto no dia-a-dia dos desenvolvedores e testadores, elaborando as estórias de usuário e ajudando o time a alcançar seus objetivos (LEFFINGWELL, 2010, p. 47-48). No contexto do grupo 1, o *product owner* será todos os alunos de Requisitos.

4.2.2.2 Scrum Master

É responsável por facilitar o progresso do time (ajudando assim a se alcançar o objetivo final), liderar os esforços do time, reforçar as regras do processo ágil e eliminar impedimentos (LEFFINGWELL, 2010, p. 47-48). No contexto do grupo 1, o papel de *Scrum Master* irá flutuar durante o projeto, e será escolhido um integrante do grupo de requisitos para ser *Scrum Master* a cada sprint. Essa rotatividade irá fazer com que todos os integrantes aprendam a ser um *Scrum Master*, e, só é necessário um por sprint.

4.2.2.3 Desenvolvedores e Testadores

Responsáveis por desenvolverem e testarem as estórias de usuário. No contexto do grupo 1, todos os integrantes de Requisitos farão parte desse papel, sendo assim responsáveis pelo desenvolvimento da solução final e por testarem as estórias.

5 Técnicas de Elicitação de Requisitos

Descobrir requisitos é um dos grandes desafios do processo de produção de Software. Não ser capaz de completar essa tarefa da maneira certa e com qualidade impossibilitará o projeto de obter sucesso, não importa quão bom sejam as qualidades do time (LEFFINGWELL, 2010, p. 227-228).

Por tal motivo, existe a necessidade de utilização de técnicas que ajudarão a levantar requisitos. Cada técnica apresenta melhor resultado em determinados contextos, e, muitas vezes, elas se completam. Por tal motivo, as técnicas foram escolhidas baseadas nas características do time.

As características que mais influenciaram na escolha das técnicas foram:

- Disponibilidade: técnicas que exigem grande quantidade de encontros são mais difíceis de serem aplicadas pois os horários em que os times poderiam aplicá-las não é suficiente, embora seja flexível.
- Experiência: certas técnicas exigem um certo nível de experiência para apresentar resultados satisfatórios. O grupo não tem experiência com as técnicas ou as metodologias, logo, opta-se por técnicas mais fáceis.

5.1 Workshop de Requisitos

5.2 Brainstorming

5.3 Entrevistas e Questionários

5.4 Mock-Ups

5.5 Product Council

5.6 Análise Competitiva

5.7 Customer change request system

5.8 Modelagem caso-de-uso

6 Tópicos de Gerenciamento de Requisitos

6.1 Rastreabilidade de Requisitos

6.2 Atributos de Requisitos

7 Ferramentas de Gerência de Requisitos

"If the process to select a RM tool seems daunting, the process to implement a tool is an even greater challenge"(BEATTY, 2013).

É mandatório a utilização de uma ferramenta de RM para times que desejam boa organização e qualidade no processo de desenvolvimento. Contudo, essa tarefa é difícil, e como dito na citação acima, mais difícil ainda é convencer um time a utilizá-la. Por tais motivos, pesquisou-se não somente por ferramentas completas: também analisou-se critérios como curva de aprendizagem, adaptação, customização, entre outros.

Assim sendo, foram inspecionadas mais a fundo três ferramentas de gerenciamento de requisitos: *Rally*, *Jama* e *Polarion*. Devido a importância que o grupo dá a realização de um trabalho colaborativo e participativo, foi fator imprescindível a ferramenta ser *web-based*, facilitando o acesso independente da plataforma de trabalho, ou lugar em que a equipe se encontre.

Embora a utilização de uma ferramenta instalável em um servidor pudesse ser uma opção alternativa que forneceria os mesmos benefícios descritos acima, decidiu-se que o esforço e tempo demandado para a instalação e configuração não traria benefícios suficientes, ainda mais considerando o contexto do trabalho.

Outro fator fundamental foi a compatibilidade com a metodologia adotada pela equipe.

Sendo assim, as ferramentas testadas serão apresentadas uma a uma, contemplando suas características, e, ao final da apresentação das ferramentas, será mostrada a decisão final e a justificativa.

7.1 Ferramenta 1 - Jama

- Suporte a múltiplos projetos
- Suporte a múltiplas equipes e papéis
- Importação de dados (doc/docx, xls/xlsx, xml, csv, arquivo de exportação do IBM Rational Doors)
- Impressão e exportação dos dados (doc, xls)
- Suporte nativo a: releases, sprints, épicos, estórias de usuário, pontos da estória, cenários de uso, e casos de teste

- Rastreabilidade dos requisitos
- Integração com outros serviços

O Jama possui uma interface agradável, de baixa curva de aprendizagem, e com boas opções de gerenciamento de requisitos voltado para a abordagem adaptativa.

O padrão de criação de requisitos é baseado em campos de formulários e em modelos pré-definidos de documentos de texto que permitem uma flexibilização muito grande em sua criação, sendo bastante útil quando se quer seguir um padrão próprio ou alternativo na criação dos requisitos.

A ferramenta ainda apresenta um boa rastreabilidade dos requisitos, relacionando os épicos com as estórias de usuários e com os casos de testes, funcionando em ambos os sentidos (ida e volta).

A falta de alguns componentes gráficos como *kanban* e *backlog*, que seriam interessantes no contexto adaptativo fazem falta, mas podem ser facilmente substituídos por ferramentas que se integram ao Jama.

A ferramenta apresenta algumas informações gerais sobre o projeto, dentre eles: risco da iteração e gráfico de priorização de estórias. E uma das grandes vantagens da ferramenta é o eficiente gerenciador de revisão, que permite comparar as edições realizadas pelos usuários nas estórias, épicos, e etc, permitindo a restauração dos documentos para uma versão anterior.

7.2 Ferramenta 2 - Rally

- Suporte à múltiplos projetos
- Suporte à múltiplas equipes e papéis
- Interface prática, amigável e organizada
- Rastreabilidade dos requisitos
- Diversas opções de estatísticas em forma de gráfico sobre o projeto
- Suporte nativo a: *releases*, epicos, funcionalidades, estórias de usuário, iteração, pontos da estória, casos de teste, *kanban* e *backlog*

O Rally se destaca por sua interface funcional e prática, sendo bastante informativo e organizada. Pelo fato de possuir um kanban com cinco estágios, diversas opções gráficas para exibir o progresso do projeto, além de muitas informações relevantes para

o gerenciamento do projeto e requisitos, a ferramenta se mostra poderosa e focada em projetos que usem abordagens adaptativas.

A rastreabilidade funciona muito bem quando se vai de épico para estórias de usuários, mas o caminho contrário não é possível, pois há uma ligação apenas para os componentes filhos. No entanto isso pode ser contornado usando algumas das funções de visualização gráfica disponíveis na ferramenta.

Há apenas a opção de exportar os dados para arquivos PDFs, mas de forma individual, o que não se mostra muito útil. A possibilidade de se inserir arquivos anexos e a liberdade de criar estórias de usuário utilizando um editor de texto avançado permite uma flexibilização interessante das descrições das estórias.

Embora o gerenciador de revisão da ferramenta seja de fácil acesso, ela não tem funções importantes como comparação entre diferentes versões de um arquivo, e muito menos a opção de se restaurar o arquivo para uma versão anterior, o que o torna um tanto inútil, servindo apenas como log de alterações.

7.3 Ferramenta 3 - Polarion

- Suporte a múltiplos projetos;
- Suporte a múltiplas equipes e papéis;
- Importação de dados (doc/docx e xls/xlsx)
- Exportação de dados (doc)
- Interface simples
- Rastreabilidade dos requisitos
- Algumas opções de estatistícas em forma gráfica sobre o projeto.

O Polarion se assemelha muito com um repositório de documentos editáveis, o que possui vantagens e desvantagens.

Como desvantagens, o sistema se distancia bastante de um sistema elegante e moderno de gerenciamento de requisitos, e não possui um suporte direto a metodologias ágeis, já que há nativamente apenas requisitos funcionais, requisitos do usuário, casos de testes e *features*.

A vantagem desse sistema é que tudo funciona em forma de *templates*, ou seja, é possível personalizar os itens para assumirem a forma de épicos, mas tudo acontece como se estivesse produzindo um documento de texto. Mas não é por isso que a ferramenta

deixa de apresentar recursos importantes como status, prioridade, rastreabilidade com ligações entre requisitos de forma bidirecional e histórico de versão, inclusive com opções de comparação entre as revisões, mas sem uma opção específica para a restauração do documento para uma versão anterior.

O Polarion é uma ferramenta versátil, que pode coibir os desavisados que ainda temem editar requisitos em forma de documentos de texto, mas basta dar uma olhada mais aprofundada para perceber que a ferramenta cumpre com os requisitos que qualquer gerenciador de requisitos deva possuir.

7.4 Ferramenta Escolhida e Justificativa

De todas as ferramentas testadas era esperado que o Rally fornecesse os componentes necessários para sua adoção como ferramenta de gerenciamento de requisitos, devido a sua forte integração com a abordagem adaptativa, no entanto, dois fatos considerados de grande importância acabaram tirando a ferramenta, inicialmente preferida, da disputa: a impossibilidade de rastrear bidirecionalmente os requisitos de forma fácil, e também a incapacidade de comparar e restaurar as modificações realizadas nos requisitos.

A Polarion mostrou-se muito eficiente nas funções fundamentais de gerenciamento de requisitos, surpassando as expectativas e até mesmo mostrando-se mais indispensável que a ferramenta Rally. No entanto, por ser pouco adaptada aos métodos ágeis, ter uma forma de gerenciamento de requisitos considerada arcaica, e principalmente por não ter características interessantes e úteis presentes nas outras ferramentas, também optou-se por não utilizá-la.

Sendo assim, optou-se pela ferramenta Jama. Poderia-se dizer que foi a escolha que sobrou, mas isso não faria jus ao que a ferramenta demonstrou. Sem dúvida, é a ferramenta mais completa entre as testadas, e embora não tenha uma interface tão agradável ou funcionalidades tão chamativas quanto as que são apresentadas na ferramenta Rally, possui o que é essencial, no que as demais ferramentas sempre deveriam fornecer: fácil e sólida rastreabilidade. A Jama, além de tudo é muito preparada para o contexto ágil e possui integração nativa com ferramentas que extendem suas funcionalidades para outros contextos dentro do projeto de software, não deixando a equipe presa à ferramenta para gerência de outras coisas.

Referências

BEATTY, J. *Winning the Hidden Battle: Requirements Tool Selection and Adoption*. Austin, TX USA, 2013. Citado na página 27.

CENTERS FOR MEDICARE & MEDICAID SERVICES. *Selecting a Development Approach*. [S.l.], 2005. Disponível em: <http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf>. Citado na página 17.

KRUCHTEN, P. *The Rational Unified Process: An Introduction*. [S.l.], 2003. Citado 3 vezes nas páginas 17, 18 e 20.

LEFFINGWELL, D. *Agile Software Requirements*. [S.l.], 2010. Citado 5 vezes nas páginas 18, 19, 20, 21 e 23.

SCALED AGILE INC. *Release Management*. [S.l.], 2015. Disponível em: <http://www.scaledagileframework.com/release-management/>. Citado 2 vezes nas páginas 20 e 21.

Apêndices

APÊNDICE A – Primeiro Apêndice

Texto do primeiro apêndice.

APÊNDICE B – Segundo Apêndice

Texto do segundo apêndice.

Anexos

ANEXO A – Primeiro Anexo

Texto do primeiro anexo.

ANEXO B – Segundo Anexo

Texto do segundo anexo.