UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO EM SISTEMAS E COMPUTAÇÃO

MODELO DE RASTREABILIDADE DE REQUISITOS APLICADA A GESTÃO DE PROJETOS EM MÉTODOS ÁGEIS

Robson Paulo da Silva

Natal-RN
Agosto de 2016

Robson Paulo da Silva

MODELO DE RASTREABILIDADE DE REQUISITOS APLICADA A GESTÃO DE PROJETOS EM MÉTODOS ÁGEIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de Mestre em Sistemas e Computação

Linha de pesquisa:
Engenharia de Software

Orientadora
Profª. Drª. Márcia Jacyntha Nunes Rodrigues Lucena

PPgSC – Programa de Pós-Graduação em Sistemas e Computação
DIMAp – Departamento de Informática e Matemática Aplicada
CCET – Centro de Ciências Exatas e da Terra
UFRN – Universidade Federal do Rio Grande do Norte

Natal-RN
Agosto de 2016

Catalogação da Publicação na Fonte. UFRN / SISBI / Biblioteca Setorial Especializada do Centro de Ciências Exatas e da Terra – CCET.

Silva, Robson Paulo da.
Modelo de rastreabilidade de requisitos aplicada a gestão de projetos em métodos ágeis / Robson Paulo da Silva. – Natal, RN, 2016.
109 f. : il.

Orientadora: Profa. Dra. Márcia Jacyntha Nunes Rodrigues Lucena.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Centro de Ciências Exatas e da Terra. Departamento de Informática e Matemática Aplicada. Programa de Pós-Graduação em Sistemas e Computação.

1. Engenharia de software – Dissertação. 2. Rastreabilidade de requisitos – Dissertação. 3. Métodos ágeis – Dissertação. 4. Projeto de software – Dissertação. 5. Modelos de referência – Dissertação. I. Lucena, Márcia Jacyntha Nunes Rodrigues. II. Título.

RN/UF/BSE-CCET CDU 004.41

Robson Paulo da Silva

MODELO DE RASTREABILIDADE DE REQUISITOS APLICADA A GESTÃO DE PROJETOS EM MÉTODOS ÁGEIS

Esta Dissertação foi julgada adequada para a obtenção do título de mestre em Sistemas e Computação e aprovado em sua forma final pelo Programa de Pós - Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

_____

Profª. Drª. Márcia Jacyntha Nunes Rodrigues Lucena - UFRN
(Presidente)

Banca Examinadora

_____

Prof. Dr. Fernando Figueira Filho - UFRN
(Examinador Interno ao Programa)

_____

Prof.ª Dra. Lyrene Fernandes da Silva - UFRN
(Examinador Interno ao Programa)

_____

Prof. Dr. Gilberto Cysneiros Filho - UFPE
(Examinador Externo ao Programa)

Natal-RN
Agosto de 2016

Dedico este trabalho a minha família, pelo apoio, paciência, incentivo, amor e dedicação.

Agradecimentos

Em primeiro lugar agradeço a Deus por me conceder o dom da vida e a benção de nascer numa boa família, na qual muito aprendi e me fez quem sou hoje. Por, a cada dia me guardar e proteger, estar comigo, guiando meus passos com sua infinita sabedoria, sem sua mão bondosa eu nada seria.

Agradeço em especial a minha mãe, que mesmo viúva e sozinha cuidou de cinco filhos com muito amor e nos ensinou pelo exemplo o valor do trabalho e dos estudos. Não posso me esquecer dos dias de feira, da banca de roupas, e da coragem da mulher batalhadora e determinada, que até hoje me inspira e me motiva.

Agradeço a meu pai (in memoriam), homem simples, batalhador e corajoso, Deus o levou cedo para seu lado, mas sua memória permaneceu conosco, guardado e querido em nossos corações.

Agradeço pela família maravilhosa que tenho hoje, por meus filhos e esposa, cuja paciência e dedicação me deram forças para continuar. Eles enfeitam e alegram minha vida, são minha fortaleza e iluminam meu caminho.

Agradeço a minha orientadora professora Márcia Jacyntha Nunes Rodrigues Lucena, mais que professora, mais que orientadora, foi a voz sensata, competente e amiga que apoiou, instruiu e aconselhou no melhor caminho para o sucesso das pesquisas, sempre presente e ajudando a vencer cada etapa desta empreitada.

Agradeço aos muitos colegas e amigos que fiz durante o tempo deste Mestrado e a todos que de forma direta ou indireta me apoiaram, e contribuíram para o sucesso deste trabalho.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo um alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

(José de Alencar)

Modelo de Rastreabilidade de Requisitos Aplicada a Gestão de Projetos em Métodos Ágeis

Autor: Robson Paulo da Silva.

Orientadora: Profª. Drª. Márcia Jacyntha Nunes Rodrigues Lucena .

RESUMO

Métodos Ágeis tornaram-se uma forma de desenvolvimento predominante, sendo praticados em larga escala na indústria e objeto de grande interesse científico pela comunidade acadêmica. Apesar de qualidades reconhecidas pelos praticantes e pela academia, existem alguns pontos de fraqueza em que os Métodos Ágeis poderiam se beneficiar de soluções mais conservadoras, importadas dos métodos tradicionais. Como, por exemplo, situações em que ocorre a perda da memória coletiva devido à alta rotatividade de pessoal, ou a mudança do desenvolvimento de *onshore* para *offshore*. Uma destas soluções que pode trazer muitos benefícios para os processos ágeis de desenvolvimento é o emprego de Rastreabilidade para apoiar a gestão de projetos.

O objetivo deste trabalho é apresentar um modelo de referência de rastreabilidade no âmbito de gestão de projetos em Métodos Ágeis e mostrar como aplicá-lo em um contexto real. Para construir este modelo foi feita uma revisão da literatura, na qual foram buscadas entidades de referência que representam conceitos, necessidades de informação, reconhecidos pela comunidade científica como importantes para o rastreamento de requisitos, além daqueles relacionados a projetos ágeis. Estes conceitos foram empregados na análise dos dados de projetos de software de uma grande corporação, para identificar problemas tais como: falta de testes; inconsistências na instanciação dos processos; deficiências na gestão dos projetos e; incoerências entre as Histórias de Usuário e seus detalhamentos.

Palavras-chave: Rastreabilidade de requisitos, Métodos Ágeis, Projeto de Software, Modelos de Referência.

ABSTRACT

Agile methods have become a predominant form of development, being practiced on a large scale in the industry and object of great scientific interest in the academic community. Despite its many qualities recognized by practitioners and the academia, there are some points of weakness on what Agile Methods could benefit from more conservative solutions, imported from the traditional methods, for example, situations in which there is loss of collective memory due to high staff turnover, or changing the development of onshore to offshore. One of these solutions that can bring many benefits to the agile development process is the use of traceability to support project management in Agile Methods.

The aim of this paper is to present a traceability reference model for Agile methods project management scope and show how to apply it in a real context. To build this model a literature review was conducted, in which reference entities were sought representing concepts, information needs, recognized by the scientific community as important for traceability requirements in addition to those related to Agile projects. These concepts were used in the construction of a traceability reference model whose usefulness was evaluated in the extraction of software project management information of a large corporation. Used to identify problems such as lack of testing; inconsistencies in the instantiation of processes; deficiencies in the management of projects and; inconsistencies between the User Stories and his detailing.

Key Words: Requirements Traceability, Agile Methods, Software Project, Reference Models.

Lista de figuras

Figura 1: Conceitos Fundamentais de Rastreabilidade Gotel et al., 2012 24

Figura 2: Relações de Rastreamento (*Trace Relation*) Gotel et al., 2012 24

Figura 3: Rastreabilidade Vertical e Horizontal Gotel (1995) 26

Figura 4: Suporte para o ciclo de vida de desenvolvimento (Abrahamsson et al., 2002) 30

Figura 5: Processo XP (Abrahamsson et al. , 2002) 31

Figura 6: Scrum (Abrahamsson et al. , 2002) 34

Figura 7: Scrum (Abrahamsson et al. , 2002) 34

Figura 8: *Crystal Orange* (Abrahamsson et al. , 2002) 37

Figura 9: FDD (Abrahamsson et al. , 2002) 37

Figura 10: DSDM (Abrahamsson et al. , 2002) 39

Figura 11: ASD (Abrahamsson et al. , 2002) 41

Figura 12: Metamodelo de Rastreabilidade Ramesh e Jarke (2001). 45

Figura 13: *Modelo de Rastreabilidade Usuários Básicos*, Ramesh e Jarke (2001). 47

Figura 14: Submodelo de Gerenciamento de Requisitos , ... 48

Figura 15: *Submodelo de Rationale,* Ramesh e Jarke (2001). 49

Figura 16: Submodelo de Alocação de *Design*, Ramesh e Jarke (2001). 50

Figura 17: Submodelo de Verificação de Conformidade, Ramesh e Jarke (2001) 51

Figura 18: Modelo de rastreabilidade, Espinoza e Garbajosa (2011). 54

Figura 19: Modelo de Rastreabilidade para Métodos Ágeis 62

Figura 20: Exemplo de mapeamento de entidades de processo para se tornarem entidades do modelo de rastreamento MRMA. 64

Figura 21: Dimensão de Melhorias e Requisitos. 69

Figura 22: *Link* de Rastreamento 71

Figura 23: Modelo de dados parcial do JIRA 74

Figura 24: Entidade no Esquema de Banco de Dados do Jira que retém o conceito “Histórias de usuário” 75

Figura 25: Entidade no Esquema de Banco de Dados do Jira que retém o conceito “*Feature*” 75

Figura 26: Conceito do Jira que retém o conceito “*Role*” 76

Figura 27: Uso do modelo de rastreabilidade na identificação dos conceitos pertinentes a gestão de projetos em Métodos Ágeis na Ferramenta JIRA 77

Figura 28: Visão das tarefas por responsável 77

Figura 29: Matriz de rastreabilidade associando Histórias de 78

Figura 30: Passo a Passo do Estudo 80

Figura 31: Modelo de Dados do MRMA 82

Figura 32: Taxa de erros nos projetos usando Métodos Ágeis de uma grande companhia no ramo da energia (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro) 89

Figura 33: Percentual de cobertura de testes do projeto 2 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro) 90

Figura 34: Taxa de erros no projeto do sistema 2 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro) 92

Figura 35: Taxa de erros no projeto do sistema 5 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro) 93

Figura 36: Taxa de erros do sistema 4 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro) 95

Figura 37: Quantidade de correções de erros por histórias 96

Figura 38: Gráfico SunBurst – mostrando o detalhamento das histórias – SISTEMA 1 - Identificando problemas nos requisitos 97

Figura 39: Usando o MRMA para identificar falha no processo de requisitos 99

Figura 40: Links quebrados 100

Figura 41: Ligações não validadas 100

Lista de Tabelas

Tabela 1: Comparação Métodos Ágeis x Tradicionais 2011 – 2015, ... 19

Tabela 2: Resolução de projetos 2011 – 2015, acima 10.000 19

Tabela 3: Exemplo de matriz de rastreabilidade 25

Tabela 4: Mapeamento dos artefatos e atividades dos Métodos Ágeis que podem ser de interesse da rastreabilidade, baseado em Abrahamsson et al. (2002). 42

Tabela 5: Comparação *Usuários Iniciantes* versus *Usuários Avançados* 46

Tabela 6:Descrição resumida dos projetos 85

Tabela 7: Tipos de problemas de gestão que podem se respondidos pelo uso do MRMA Gupta at al. (2015). 86

Sumário

1. INTRODUÇÃO 15

1.1 Motivação 16
1.1.1 Gerenciamento do conhecimento – o problema 17
1.1.2 Estagnação dos processos de software 18
1.2 Rastreabilidade – a Proposta 20
1.3 Objetivos Gerais 21
1.4 Organização do Trabalho 22

2 Fundamentação Teórica 22

2.1. Rastreabilidade de Requisitos 22
2.2. Desafios e Benefícios da Rastreabilidade para Métodos Ágeis 26
2.3 Modelos de Processos Ágeis 29

3 Trabalhos Relacionados 44

3.1 Modelo de rastreabilidade proposto por Ramesh e Jarke (2001) 44
3.2 Propostas de rastreabilidade em Métodos Ágeis 52
3.3 Discussão 57

4. Modelo de Rastreabilidade para Gestão de Projetos no Contexto Ágil 58

4.1 Construção do modelo de referência 58
4.2 Mapeando entidades dos processos ágeis para o modelo 59
4.2.1 Mapeando entidades e características de outros modelos de rastreabilidade 64
4.2.2 Descrevendo o modelo 66

5. Aplicando o modelo de Rastreabilidade (Estudo de caso) 73

5.1 Cenário de Uso 1 – Validação das entidades do modelo 73
5.1.1 Discussão 78
5.2 Cenário de Uso 2 – Demonstrar a utilidade do modelo para extração e análise de dados de gestão de desenvolvimento em Métodos Ágeis 79
5.2.1. Instanciar e Popular o Modelo 80
5.2.2 Testar a utilidade do modelo para a gestão de projetos usando Métodos Ágeis 84

6. Conclusões 101

6.1 Contribuições 101

6.1 Limitações.....102
6.3 Trabalhos futuros 103
Referências 105

1. INTRODUÇÃO

Apesar do contínuo avanço no desenvolvimento de ferramentas para apoio e automação das atividades de desenvolvimento de software, estas continuam primordialmente dependentes do ser humano e exigem esforço colaborativo e intenso compartilhamento de conhecimento.

No entanto, há diferenças relevantes na forma como as diversas metodologias de desenvolvimento operacionalizam o compartilhamento de conhecimento. As metodologias mais tradicionais dão grande ênfase na criação e manutenção de documentação, enquanto que o Manifesto Ágil, escrito por Fowler et al. (2001), dá um maior destaque a entrega do software funcionando do que uma documentação abrangente.

Este princípio não significa que Métodos Ágeis não dão valor ao compartilhamento da informação, de fato é justamente o contrário, pois diversas práticas ágeis contribuem fortemente neste sentido, por exemplo: foco no cliente, colaboração regular e; comunicação face a face dentro da equipe.

E, embora elas claramente ajam neste sentido, limitações e diversas barreiras e problemas têm sido reconhecidos pela comunidade científica. Por exemplo, Chan e Thong (2009) destacam que as características de cada indivíduo, tais como ter maior ou menor facilidade para se comunicar, são importantes para alcançar bons resultados em métodos ágeis. Consequentemente, uma equipe ágil que apresente dificuldades neste quesito, pode afetar negativamente o compartilhamento das informações dentro do projeto e para outras equipes que venham, em momento futuro, dar manutenção nos sistemas de software por ela produzidos.

Cockburn(2006), Conboy et al. (2010), Nerur et al. (2005) e Ramesh et al. (2010) apresentam outros obstáculos, por exemplo, pode faltar motivação para o usuário compartilhar conhecimentos de negócio específicos ou para facilitar o compartilhamento do conhecimento tácito entre os membros da equipe.

Na mesma linha Ghobadi e Mathiassen(2014) analisam a presença de várias barreiras ao compartilhamento de conhecimento em Métodos Ágeis, relacionadas a à diversidade da equipe; relativas à percepção da equipe (atitudes e valores

diferentes); além de referentes às habilidades e conhecimento da equipe e referentes à tecnologia.

Além destas, uma outra barreira advém da pressão por cumprir prazos que pode levar as equipes ágeis a não documentarem importantes conhecimentos confiando que eles estão guardados na memória coletiva.

De fato, em situações normais a falta de documentação pode ser compensada pela presença das pessoas que detêm o conhecimento de que se precisa. Mas, equipes podem ser desmanteladas, serviços que eram feitos numa determinada empresa podem passar a ser feitos em outra. E, neste contexto, todo conhecimento que estava somente na cabeça das pessoas, e que não foi registrado de nenhuma outra forma, acaba sendo perdido.

Ao analisar esta problemática, duas perguntas aparecem: (1) quais conhecimentos deveriam ser preservados? E; (2) como preservá-los sem perder a característica ágil dos métodos ágeis? Não pretendemos dar uma resposta plena as duas perguntas neste trabalho, mas ele responde, ao menos em parte, a primeira delas ao identificar e propor quais as entidades que deveriam compor o modelo de rastreabilidade para Métodos Ágeis (MRMA).

Este trabalho postula que muitos conhecimentos importantes já são registrados nas diversas ferramentas de mercado, amplamente conhecidas e usadas para o apoio ao gerenciamento de projetos em Métodos Ágeis, tais como: JIRA; Microsoft Team Foundation Server; Plataforma JAZZ IBM; etc. Assim nosso modelo de rastreabilidade é um instrumento para ajudar a encontrar as entidades de informação e criar as relações semânticas entre elas, diminuindo o esforço de recuperação e análise, documentando conhecimentos que muitas vezes estavam somente nas cabeças dos membros da equipe.

1.1 Motivação

A motivação pode ser mais bem compreendida ao analisarmos a situação vivida por uma grande empresa do ramo da energia que possui várias centenas de aplicações em uso e seu portfólio inclui desde pequenos sistemas a sistemas gigantescos.

Há alguns anos esta empresa adotou Métodos Ágeis no desenvolvimento e manutenção de seus sistemas de software e recentemente, em alinhamento com suas estratégias de negócio, ela realizou uma grande movimentação destes softwares entre suas equipes de desenvolvimento e para fábricas de software. As equipes de desenvolvimento originais foram parcial ou totalmente desmontadas, levando consigo grande quantidade de conhecimento a respeito destes produtos.Este fato trouxe diversos desafios para a gestão dos projetos de software em métodos ágeis, a saber: a ausência dos *stakeholders* iniciais e de documentação resultaram em complicações para a análise de impacto das mudanças. Esta situação tornou difícil determinar se um problema relatado pelos usuários era mesmo um bug ou se tratava de um comportamento esperado para o software; determinar quais eram os requisitos afetados tornou-se uma tarefa mais complicada e sujeita a erros; o mesmo para a identificação das partes do código e da arquitetura que foram impactados pelas mudanças, resultando em prejuízos para as estimativas de custo e prazo, assim como para a garantia de qualidade. O problema está claramente relacionado com gerenciar conhecimento de modo a evitar, ou minimizar, problemas quando ocorrem eventos como os descritos anteriormente. A seguir discorremos mais detalhadamente sobre este assunto.

1.1.1 Gerenciamento do conhecimento – o problema

O cenário descrito na seção anterior expõe um grave problema de gerenciamento do conhecimento ligado à perda de pessoal e/ou a movimentação do serviço de manutenção dos produtos de software de equipes locais para equipes geograficamente distantes.

Este imbróglio é reconhecido em diversos estudos na literatura científica, tais como Chua e Pan (2008) que afirmam que a troca de pessoal devido à mudança do modelo de trabalho localizado (*onshore*) para o distribuído (*offshore*) resulta em perda de conhecimento tácito e do conhecimento dos processos de software que são difíceis de serem transferidos, por exemplo, a heurística para instanciar corretamente o processo considerando a tecnologia a ser adotada, o domínio do problema a ser resolvido e as peculiaridades do projeto, tais como: restrições legais e normativas; tamanho do projeto; restrições de tempo e de prazo; experiência da equipe, entre outros aspectos.

Na mesma linha, Taweel at al. (2009) adicionam o aspecto temporal da entrada ou saída de pessoal no meio ou perto do final dos projetos. Assim, a diferença entre o conhecimento necessário e aquele que efetivamente deve ser possuído exige tempo para ser superado, o que acaba sobrecarregando a equipe em seu esforço para cumprir os prazos dos projetos.

Já Rottman e Lacity (2008) afirmam que na saída de pessoal, eles acabam levando junto o conhecimento dos processos de negócio que são difíceis de serem readquiridos, enquanto Dibbern at al. (2008) argumentam que a perda de pessoal pode ter efeitos disruptivos no gerenciamento de conhecimento.

Métodos Ágeis não estão isentos destas dificuldades, na verdade, diversos trabalhos científicos reconhecem limitações neste sentido. Nerur et al. (2005) afirmam que nos processos ágeis muito dos conhecimentos estão tácitos e residem nas cabeças dos membros das equipes, fato que expõe as organizações a riscos.

De forma semelhante, Nerur et al. (2005) argumentam que os Métodos Ágeis encorajam o pensamento enxuto (*Lean Thinking*), eliminando os excessos, particularmente documentação. Os problemas surgem, segundo Ramesh et al. (2010), quando ocorre uma quebra de comunicação, por exemplo, a excessiva entrada e saída de pessoal (*personnel turnover*), aliada a falta de documentação que podem levar à incapacidade de escalar o software, evoluir ou corrigir a aplicação de forma segura.

Outra dimensão do problema, um pouco mais sutil, mas útil para justificar porque buscar melhorias, é que embora os resultados dos Métodos Ágeis sejam notáveis, eles estagnaram e ainda estão longe de alcançar uma taxa de sucesso dos projetos que se possa considerar no mínimo aceitável, como poderemos ver na próxima subseção.

1.1.2 Estagnação dos processos de software

A Tabela 1 apresenta uma comparação entre a resolução de projetos ágeis em relação aqueles usando a metodologia Tradicional e nela podem ser feitas duas constatações: os Métodos Ágeis alcançaram melhores resultados do que os processos Tradicionais, e; apesar dos números revelarem que os Processos Ágeis

produziram progressos significativos, constata-se que mais de sessenta por cento dos projetos ainda são finalizados fora do prazo, do custo ou sem entregar todas as características e funcionalidades inicialmente especificadas.

Tamanho	Método	Sucesso	Intermediário	Fracasso
			Falha	
Todos os tamanhos de Projetos	Ágil	39%	52%	9%
	Tradicional	11%	60%	29%
Projetos de Grande Porte	Ágil	18%	59%	23%
	Tradicional	3%	55%	42%
Projetos de Médio Porte	Ágil	27%	62%	11%
	Tradicional	7%	68%	25%
Projetos de Pequeno Porte	Ágil	58%	38%	4%
	Tradicional	44%	45%	11%

Tabela 1: Comparação Métodos Ágeis x Tradicionais 2011 – 2015, acima de10.000 projetos em todo o mundo (*Chaos Report* 2015).

A Tabela 2 mostra que entre 2011 e 2015 não houve evolução em termos do alcance de melhores resultados na resolução de projetos. Então, se por um lado a quantidade de projetos que são encerrados sem sucesso continua grande e por outro a resolução praticamente não mudou no período de 2011 a 2015, estes resultados sugerem que há espaço e necessidade de melhorias no desenvolvimento de software.

	2011	2012	2013	2014	2015
Sucesso (Sucess)	29%	27%	31%	28%	29%
Intermediário (Challenged)	49%	56%	50%	55%	52%
Fracasso (Failed)	22%	17%	19%	17%	19%

Tabela 2: Resolução de projetos 2011 – 2015, acima 10.000 projetos em todo o mundo (*Chaos Report* 2015).

Visto isto, a pergunta natural a ser feita é “o que pode ser melhorado?”, embora esta pergunta possa ter muitas respostas diferentes, duas constatações podem nos ajudar a direcionar os esforços. A primeira vem dos resultados da Tabela 1 que mostram os índices de sucesso dos Métodos Ágeis muito superiores aos do processo Tradicional e a segunda, feita por West e Grant (2010), que afirma que os Métodos Ágeis se tornaram *MainStream*. Em resumo, além de alcançarem melhores resultados, sua adoção é significantemente relevante merecendo atenção e esforços de melhoria.

Dado que precisamos melhorar a resolução de projetos e que por sua relevância e resultados os Métodos Ágeis seriam candidatos naturais a este tipo de trabalho, é preciso definir quais seriam estas melhorias.

1.2 Rastreabilidade – a Proposta

Segundo Akgun, Byrne, Keskin e Lynn (2006), as fontes de conhecimento podem estar em repositórios formais (ex., repositórios de código fonte, documentação, ferramentas de gestão de projetos de desenvolvimento, etc..). Outras podem estar em fontes informais, como conhecimento tácito que estão somente na cabeça dos indivíduos da equipe de desenvolvimento, ou de especialistas da organização.

Este conhecimento, “Formal” ou “Informal”, é denominado memória coletiva, de acordo com Akgun, Byrne, Keskin e Lynn (2006), Rottman e Lacity (2008) e Lange e Gillis(2005). Sendo descritos na literatura científica como sistemas de memória transacional (TMS) e definidos como a forma que a organização processa e organiza as informações e também o conhecimento do grupo de como codificar, armazenar e recuperar as informações, para facilitar o compartilhamento e a recuperação do conhecimento.

Na verdade, os sistemas de memória transacional, são uma forma de aprendizado organizacional e um processo emergente da equipe, em que o todo apresenta propriedades não encontradas em nenhuma das partes individuais, conforme definido por Capra e Luisi (2014).

Métodos Ágeis possuem um TMS robusto, em que o compartilhamento de conhecimento é estimulado, por mecanismos como, por exemplo: programação em pares; cliente presente e; código coletivo. Mas, o cenário apresentado na seção “Motivação” mostra que em certas situações, este TMS apresenta problemas e o conhecimento dito “Informal” pode ser perdido em parte ou totalmente como visto na seção 1.1.1.

Por este motivo, Akgun, Byrne, Keskin e Lynn (2006) defendem que é preciso tomar providências para melhorar o acesso a esta memória coletiva. Considerando que nem sempre é possível preservar a memória “Informal”, reforçar a “Formal” para Métodos Ágeis surge como uma opção natural, embora sujeita a oposição, por ser uma proposta para mesclar práticas tradicionais as práticas ágeis.

Lago et al. (2009), Xiaofan Chen (2012), Lee et al. (2003), Espinoza e Garbajosa (2011), Gotel et al. (2014), Cleland e Huang (2012) e Jacobsson (2009) reconheceram esta necessidade e sugeriram a rastreabilidade como um das ferramentas a ser empregada na solução dela.

Portanto, o objetivo deste trabalho é propor um modelo de referência para o rastreamento de requisitos para apoiar a gestão de projetos em Métodos Ágeis, detalharemos este objetivo mais claramente na subseção 1.3.

1.3 Objetivos Gerais

Com base nos argumentos anteriores este trabalho propõe um modelo de rastreabilidade como forma de apoiar a gestão de projetos em Métodos Ágeis, e demonstra sua validade e resultados através do uso do Modelo de Rastreabilidade para Métodos Ágeis(MRMA) em projetos reais de uma grande corporação da área de energia. Para tal, os seguintes objetivos específicos devem ser alcançados:

1 – identificar quais as entidades que um modelo de rastreabilidade para a gestão de projetos em Métodos Ágeis deve conter;

2 – validar as entidades do modelo de rastreabilidade comparando com aquelas presentes nas ferramentas de apoio a gestão, usadas em projetos reais em Métodos Ágeis, para confirmar se as entidades do MRMA são válidas e reconhecidas pela indústria, e;

3 – através de um estudo de caso, obter acesso a dados de projetos reais, aplica-los ao modelo de rastreabilidade e executar análises para validar a sua utilidade para a gestão de projetos em Métodos Ágeis.

O propósito destes objetivos específicos é responder as seguintes perguntas de pesquisa:

1- quais entidades (processo/produto/projeto) um modelo de rastreabilidade para Métodos Ágeis deve conter?

2- o uso do modelo de rastreabilidade fornece informações úteis para a gestão de projetos em Métodos Ágeis?

1.4 Organização do Trabalho

Este trabalho está dividido nas seguintes seções. O capítulo 2 apresenta a fundamentação teórica. No capítulo 3 apresentamos os trabalhos relacionados. No capítulo 4 descrevemos nossa proposta de modelo de rastreabilidade para o contexto ágil e como ele foi construído. No capítulo 5 apresentamos a aplicação deste modelo na gestão de projetos no contexto de Métodos Ágeis e discutimos esta aplicação, e finalmente no capítulo 6 mostramos as conclusões e os trabalhos futuros.

2 Fundamentação Teórica

Nesta seção apresentamos conceitos importantes para o entendimento de rastreabilidade de requisitos, os principais problemas no uso da rastreabilidade no contexto de desenvolvimento de projetos usando Métodos Ágeis, os desafios e benefícios da rastreabilidade para Métodos Ágeis, terminando por discorrer mais profundamente sobre os diversos modelos de processos Ágeis (SCRUM, XP, DSDM, Família Crystal, FDD e ASD).

2.1. Rastreabilidade de Requisitos

Gotel e Finkelstein (1994) definem rastreabilidade como a “habilidade para descrever e seguir a vida de um requerimento, em ambas as direções, para frente e para trás”. Ou seja, seguir o requisito por todo o ciclo de vida, desde sua origem até a implantação e uso. Ramesh e Jarke(2001) compara um sistema de rastreabilidade

a uma rede semântica onde os nós representam objetos (artefatos, *stakeholders* e fontes) e as arestas as ligações de rastreabilidade de diferentes tipos e forças. Para o Coest (2014), *Center of Excellence for Software and Systems Traceability*, a rastreabilidade é a capacidade de estabelecer e manter ao longo do tempo, ligações entre qualquer artefato de software de forma única. Com estas informações em mãos será possível responder perguntas a respeito do projeto, do produto de software e do seu processo de desenvolvimento.

Segundo Gotel et al. (2012), rastreabilidade, no nível mais fundamental é simplesmente o potencial para relacionar dados que já estão armazenados dentro de artefatos de algum tipo, junto com a habilidade para examinar estas relações. Portanto, ela depende da capacidade de criar ligações navegáveis entre dados mantidos nos artefatos que de outra maneira estariam desconectados. Os principais conceitos relacionados a rastreabilidade podem ser resumidos em: (i) Ligação de rastreamento (*Trace Link*), (ii) Entidade de rastreamento (*Trace Artifacts*) e (iii) relação de rastreamento (*Trace relation*).

Entidade de rastreamento é uma unidade da base de dados de rastreamento. As entidades podem ser agrupadas em tipos, que caracterizam entidades de rastreamento com estrutura (sintaxe) e/ou propósito (semântica) igual ou semelhante, por exemplo: estórias de usuário; classes Java; testes de aceitação; *stakeholders*. Enquanto que as ligações de rastreamento são definidas como associações semânticas simples entre duas entidades de rastreamento, uma entidade sendo a fonte e outra entidade sendo o destino. As Ligações de Rastreamento, assim como as Entidades de Rastreamento, possuem atributos, tais como tipos de artefato e descrição, que lhes acrescentam maior valor semântico. A Figura 1 apresenta os dois primeiros conceitos: Ligações de Rastreamento e Entidade de Rastreamento.

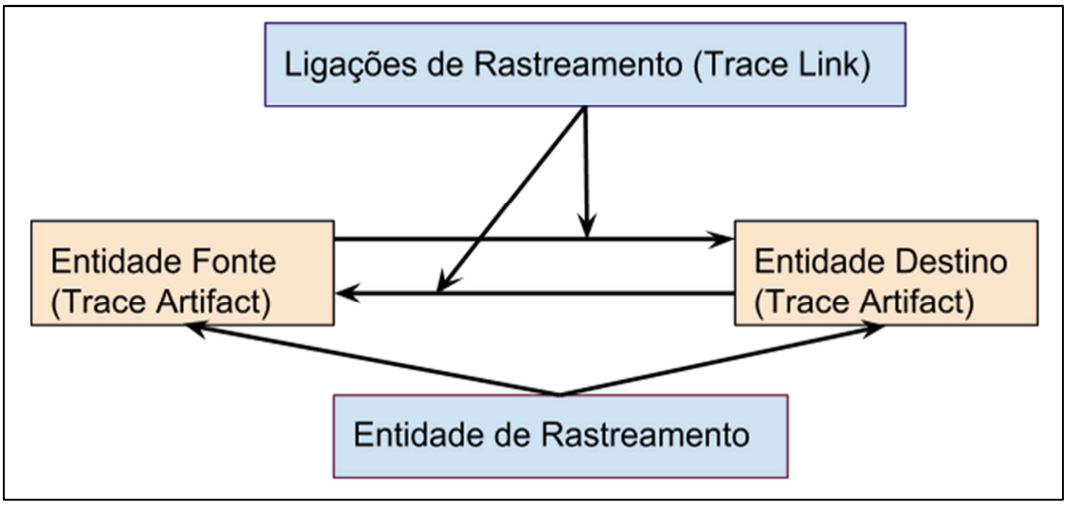


Figura 1: Conceitos Fundamentais de Rastreabilidade Gotel et al., 2012

O conceito de relação de rastreamento (*Trace relation*), conforme vemos na Figura 2, está relacionado a todas as ligações de rastreamento criadas entre dois conjuntos de tipos específicos de artefatos de rastreamento. Uma relação de rastreamento (*Trace relation*) é registrada, por exemplo, em uma matriz de rastreabilidade, que é uma tabela que liga os artefatos(entidades) fonte a seus respectivos artefatos (entidades) destino. A Tabela 3 a seguir exibe um exemplo de matriz de rastreabilidade, observe que ela relaciona as Histórias de Usuário(entidades fonte) as Regras de Negócio correspondentes (entidades destino)

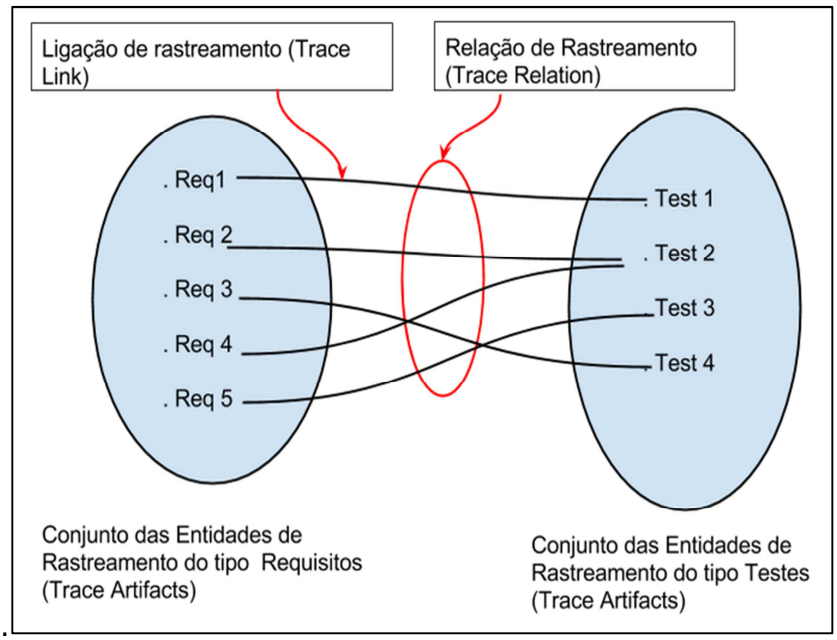


Figura 2: Relações de Rastreamento (*Trace Relation*) Gotel et al., 2012

		Regra de Negócio 1: O cartão deve pertencer ao cliente contratante	Regra de Negócio 2: O apartamento deve estar disponível	Regra de Negócio 3: Se pertencer a pacote promocional aplicar desconto.
Histórias	Pagamento com cartão de crédito	X	X	
	Reserva de apartamento			X

Tabela 3: Exemplo de matriz de rastreabilidade

Outro conceito interessante de rastreabilidade apresentado por Gotel (1995) é o de rastreabilidade Horizontal e Vertical, como é apresentado na Figura 3. A rastreabilidade Horizontal refere-se a rastreabilidade entre modelos, por exemplo, requisitos desdobrando-se em modelos UML, posteriormente em código fonte. Enquanto a rastreabilidade vertical, associa versões do mesmo artefato, a medida que este evolui ao longo do tempo.

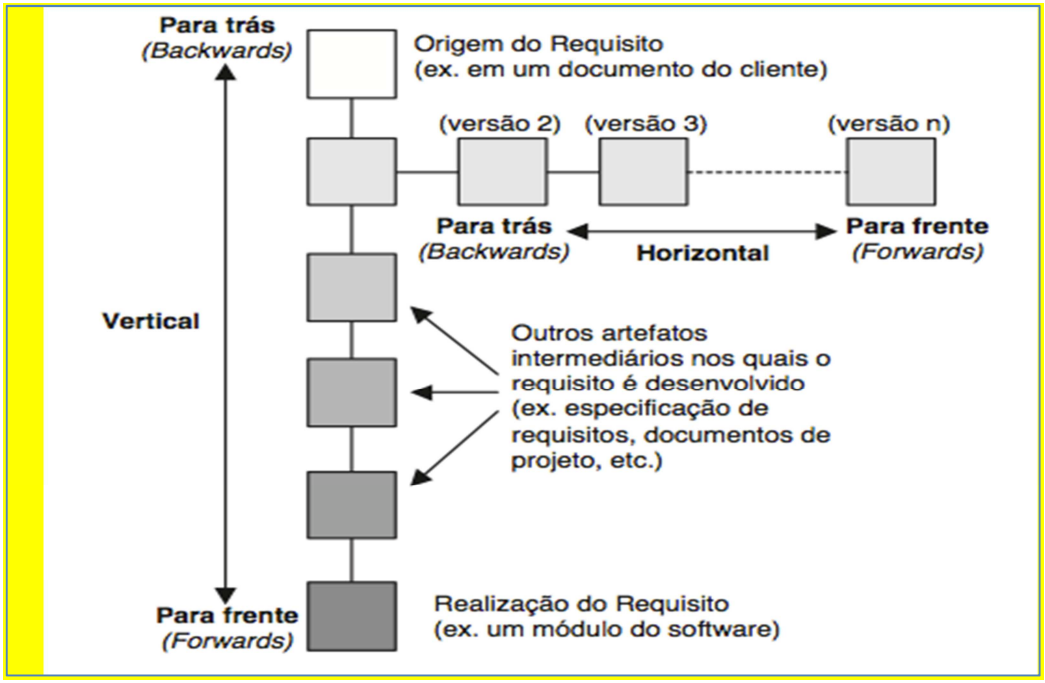


Figura 3: Rastreabilidade Vertical e Horizontal Gotel (1995)

2.2. Desafios e Benefícios da Rastreabilidade para Métodos Ágeis

Diversos problemas de rastreabilidade afetam a gestão de projetos em Métodos Ágeis. Uma situação comum ocorre na gestão dos impactos das mudanças. Segundo Delater et al. (2012), o suporte das ferramentas de rastreabilidade é limitado para coletar e exibir informações que atendam às necessidades dos *stakeholders*. Este suporte limitado é agravado por características que tornam os Métodos Ágeis diferentes dos processos de desenvolvimento de software tradicionais.

Espinoza e Garbajosa (2011) apresentam diversas diferenças existentes nos Métodos Ágeis comparado com os processos tradicionais de desenvolvimento, por exemplo: especificações de requisitos formais podem ser omitidas, ou não existirem na maior parte do desenvolvimento; estórias de usuário possuem maior granularidade e são mais ambíguas que especificações de requisitos; critérios de aceitação podem agir como prática de engenharia de requisitos, de design e de teste.

Estas diferenças afetam a semântica das entidades e das ligações de

rastreamento, e precisam ser consideradas ao se elaborar um modelo de referência de rastreabilidade a ser empregado em Métodos Ágeis.

Outra diferença é o princípio que estimula a propriedade coletiva do código. Em outras palavras, o conhecimento do código, suas relações com os requisitos, e de uma parte para outra, entre outras informações relevantes, são guardados na memória da equipe.

Este princípio é defendido por muitos como um dos pontos fortes dos métodos ágeis, mas pode se tornar uma das suas maiores fraquezas, se, por qualquer razão, esta memória coletiva não estiver mais disponível, por exemplo, se houver grande rotatividade da equipe.

Com o suporte limitado das ferramentas de rastreabilidade, e sem poder lançar mão da memória coletiva, as informações fundamentais para os processos decisórios de gestão podem não estar disponíveis quando necessário, como, por exemplo, para determinar a extensão dos impactos da mudança de um requisito. Esta situação pode comprometer a tomada de decisão, com potencial de causar problemas que podem impactar os prazos e custos dos projetos, afetar a qualidade do produto, e o atendimento das necessidades dos clientes.

Fora estes desafios mais específicos dos Métodos Ágeis: (1) suporte limitado das ferramentas de rastreabilidade e; (2) diferenças dos Métodos Ágeis para os tradicionais. Ainda há aqueles que serão experimentados em qualquer processo, independente de serem Ágeis ou tradicionais.

Gotel et al.(2012) discutem alguns deles, tais como: (i) identificar o que deve ser rastreado e as relações que devem ser estabelecidas; (ii) manter as relações de rastreabilidade sempre atualizadas; (iii) combinar abordagens humanas e automáticas que suportem a rastreabilidade.

Devido a estes diversos desafios, para justificar seu uso, dado o esforço necessário para mantê-la, é preciso que a rastreabilidade apresente benefícios palpáveis e dê apoio efetivo a gestão do desenvolvimento de software. Assim se faz necessário fornecer apoio para a análise e melhoria dos processos e suporte, segundo Mäder e Egyed(2012), para as tarefas de desenvolvimento de software, a

fim de que elas se tornem mais rápidas, baratas e produzam resultados de maior qualidade.

Neste sentido, Mäder e Egyed(2012) realizaram uma pesquisa que demonstra que o uso da rastreabilidade resulta na execução 21% mais rápida das tarefas e num índice 60% maior de soluções corretas, evidenciando que a rastreabilidade pode produzir melhoras significativas no desenvolvimento de software.

A rastreabilidade pode ajudar a gestão do desenvolvimento de software de várias maneiras, aqui destacamos alguns benefícios:

- Identificar requisitos que não foram alocados ou implementados, ou que não foram testados, avaliando a cobertura dos testes;
- Previsão de custos e prazos já que estimativas mais precisas podem ser obtidas ao ser possível identificar as partes, tais como os componentes, fragmentos de código que serão afetadas pelas mudanças;
- Permitir mapear a verdadeira extensão de um erro, que indo além do código, pode se estender para outros artefatos, por exemplo, casos de testes, diagramas UML, etc.

Cleland e Huang (2012) reforçam esta linha de pensamento e apresentam outros benefícios do uso da rastreabilidade:

- Conformidade do produto – Para garantir que o produto entregue atende as necessidades do cliente;
- Responsabilização no projeto – para garantir que todas as mudanças atendem a uma característica que foi requisitada;
- Reproduzir a linha de base – para suportar a gerência de configuração de modo a permitir que as diferentes versões possam ser reproduzidas;
- Para o aprendizado organizacional – para documentar os *rationales* (raciocínio) por trás das decisões críticas, de modo a permitir a transferência de conhecimento, por exemplo, para novos membros da equipe.

Em complemento aos argumentos significativos que já foram apresentados, a análise de sistemas complexos nos traz mais razões para fortalecer a proposição de

que o uso da rastreabilidade nos Métodos Ágeis resultará em benefícios relevantes.

Capra e Luisi (2014) definem sistemas complexos como aqueles em que as propriedades que emergem no todo não podem ser explicadas apenas pela análise de cada parte individualmente. Neles, a configuração de seus componentes e os seus relacionamentos são dois aspectos importantes, e é necessária uma mudança de perspectiva das partes para o todo, o que, por sua vez, é também uma mudança de uma perspectiva de análise baseada nos componentes (ou objetos) para outra baseada nas relações.

O desenvolvimento de software é uma atividade complexa, composta de muitas partes que operam em conjunto numa rede de múltiplas relações entre componentes de software, clientes, equipes de desenvolvimento, ferramentas, linguagens de programação, passos de processos, etc. Deste modo, a capacidade da rastreabilidade, falando de forma genérica, de mapear as diversas entidades envolvidas no desenvolvimento de software, expor e dar significado as suas relações, pode se refletir num benefício para os Métodos Ágeis ao auxiliar na descoberta de ineficiências e inconsistências dos processos, estabelecer correlações entre as métricas de processo e as de produto, extrair padrões gerais de processo para estimativas de esforço e auxiliar na análise de resolução de problemas, conforme argumentado por Gupta at al. (2015).

2.3 Modelos de Processos Ágeis

Para criar o modelo de referência de rastreabilidade é necessário conhecimento dos artefatos e processos envolvidos, a seguir será apresentado os modelos de processos de Métodos Ágeis, fundamentais para entendimento da proposta apresentada neste trabalho.

Algumas das abordagens ágeis focam mais em gerenciamento de projetos, tais como ASD, SCRUM e DSDM, enquanto outras focam mais na engenharia de software, práticas de desenvolvimento, como XP, segundo argumentado por Abrahamsson et al. (2002).

Abrahamsson et al. (2002) apresentam a Figura 4 mostrando que quase todos os processos ágeis possuem algum suporte para o gerenciamento de projetos, embora com grandes diferenças quanto ao grau de cobertura.

O suporte ao ciclo de vida é diferente entre os diversos processos ágeis, mas considerando a possibilidade de complementação combinando partes de mais de

um processo, por exemplo SCRUM com XP, é possível obter a cobertura de todo o ciclo de vida desde a concepção até que o sistema esteja em uso, incluindo Requisitos, Design, Codificação e Testes.

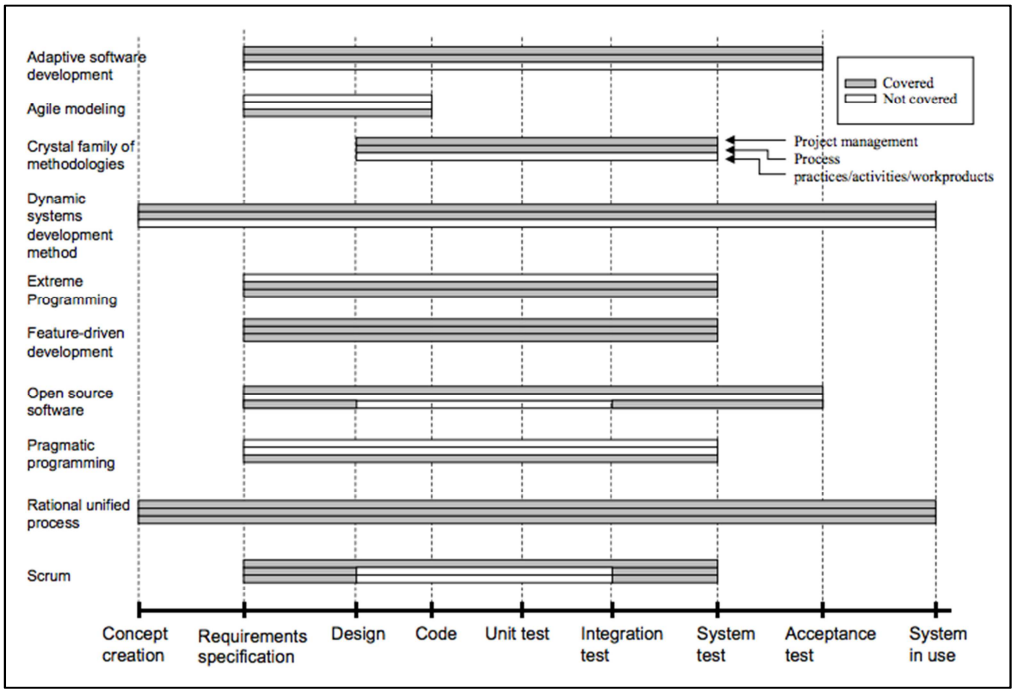


Figura 4: Suporte para o ciclo de vida de desenvolvimento (Abrahamsson et al., 2002)

As diversas disciplinas da engenharia de software e do gerenciamento de software são cobertas, em maior ou menor grau, pelos Métodos Ágeis, como exibido na Tabela 4 e nas Figuras 5 a 11.

Processo XP

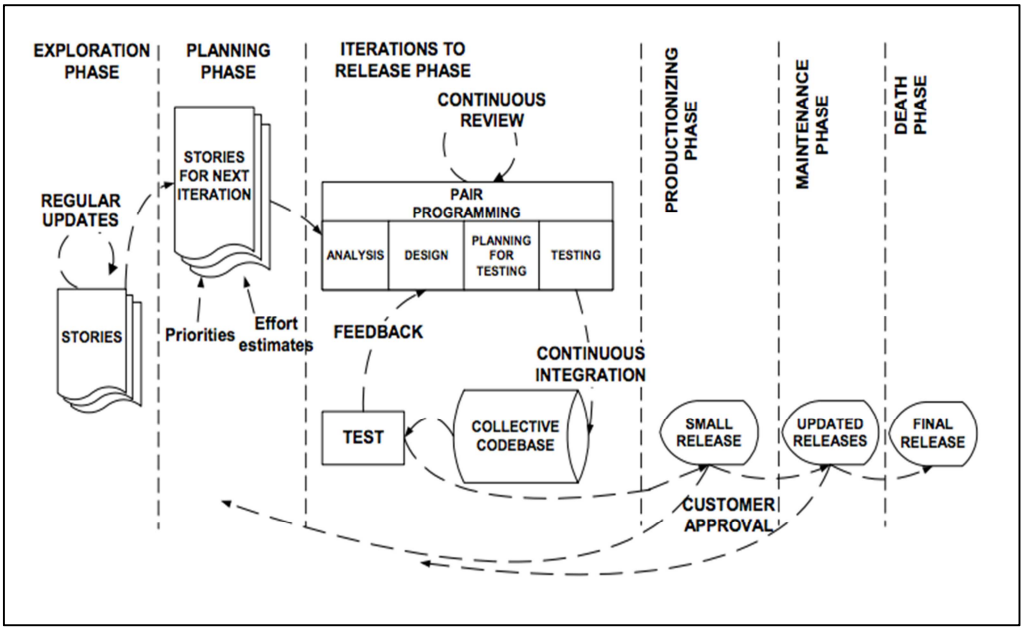


Figura 5: Processo XP (Abrahamsson et al. , 2002)

Na figura 5 acima vemos o processo XP. A primeira fase do processo é a fase de exploração. Nela os requisitos do produto são expressos por meio das Histórias de Usuário em que as *features* a serem adicionadas na primeira Iteração são descritas com mais detalhes (Abrahamsson et al. , 2002).

Na fase de planejamento chega-se a um acordo quanto a ordem de prioridade das histórias e o conteúdo da entrega, deploy, da primeira iteração (Abrahamsson et al. , 2002)..

A construção do produto de software é realizada na fase de construção, *Interation to release*, em iterações de tamanhos fixos de uma a quatro semanas em média. Testes funcionais criados pelo cliente são executados ao final de cada iteração. Na última iteração o produto deve estar pronto para ser disponibilizado em produção (Abrahamsson et al. , 2002).

Encerrada a fase de construção o produto deve ser movido para a produção e testes adicionais podem ser feitos, tais como: testes de performance e de carga, o que pode indicar a necessidade de ajustes finais do produto(Abrahamsson et al. , 2002).

No início de cada iteração há uma fase de planejamento onde as Histórias são revisadas, atualizadas e priorizadas. Em seguida são feitas as estimativas de

esforço para construção das Histórias priorizadas e selecionado o sub-conjunto que será construído na próxima iteração.

A construção do software é iterativa e passa por quatro fases: Análise; Design; Planejamento dos Testes e; Testes. Entre os testes, a Integração contínua ganha destaque, sendo uma prática onde os membros da equipe integram seu código frequentemente, para garantir que o repositório de código fonte permaneça consistente.

Ao fim de cada iteração, produto funcionando deve ser entregue, havendo portanto um conjunto de pequenos releases que vão, gradualmente, agregando funcionalidades ao produto final. Aprovações do cliente realimentam a fase de planejamento e nas iterações de construção, permitindo que os rumos possam ser corrigidos conforme necessário e o mais rápido possível. Então o produto passa para a fase de manutenção, ali permanecendo até que não seja necessário, quando então será desativado (Abrahamsson et al. , 2002).

Os diversos stakeholders assumem papéis de acordo com as tarefas e propósitos a serem alcançados, a seguir citamos apenas alguns deles: programadores; clientes e; testadores (Abrahamsson et al. , 2002).

Muitas práticas são empregadas no XP, entre elas citamos diversas a seguir (Abrahamsson et al. , 2002):

- *Planning game*: Planejamento conjunto feito com a presença dos programadores e clientes. Os primeiros estimam o esforço para construção enquanto os últimos decidem sobre o escopo e o melhor momento para os releases.
- *Releases* curtas;
- Metáforas: são pequenas histórias que guiam todo o trabalho de desenvolvimento;
- *Design* Simples: simplesmente o princípio da simplicidade, deve ser desenhada a solução mais simples possível;
- Testes: os testes guiam o desenvolvimento. Testes unitários são construídos antes do código e devem ser executados continuamente;

- Reestruturação: a estrutura do software deve ser frequentemente melhorada, através da remoção código sujo, simplificação e flexibilização do código;
- Programação em pares;
- Propriedade coletiva: o conhecimento a respeito do código e de suas partes é compartilhado por todos dentro da equipe;
- Integração contínua: cada nova parte do código é integrada ao código base assim que estiver pronta;
- Cliente presente;
- Padrões de código.

Do ponto de vista da rastreabilidade, diversos elementos deste processo podem ser de interesse: o conjunto das Histórias de Usuário do produto; o sub-conjunto das Histórias para a próxima Iteração; a organização em unidades de tempo fixas as iterações; as estimativas de esforço; os testes; o código; o repositório de código fonte; os stakeholders e seus diversos papéis; a integração contínua e; os releases (Abrahamsson et al. , 2002).

Processo SCRUM

O processo SCRUM, descrito nas Figuras 6 e 7, é composto de três fases: pré-jogo; desenvolvimento e; pós-jogo. A fase pré-jogo possui duas sub-fases: planejamento e arquitetura e design de alto nível. Na fase de planejamento é feita a definição do produto a ser construído e sua saída é uma lista contendo todos os requisitos do produto atualmente conhecidos. Esta lista é conhecida como *Product Backlog*. Os requisitos são priorizados e o esforço necessário para sua construção é estimado. Na fase de arquitetura o desenho de alto nível do produto, incluindo sua arquitetura, são planejados tomando como base os itens atuais do *Product Backlog* (Abrahamsson et al. , 2002).

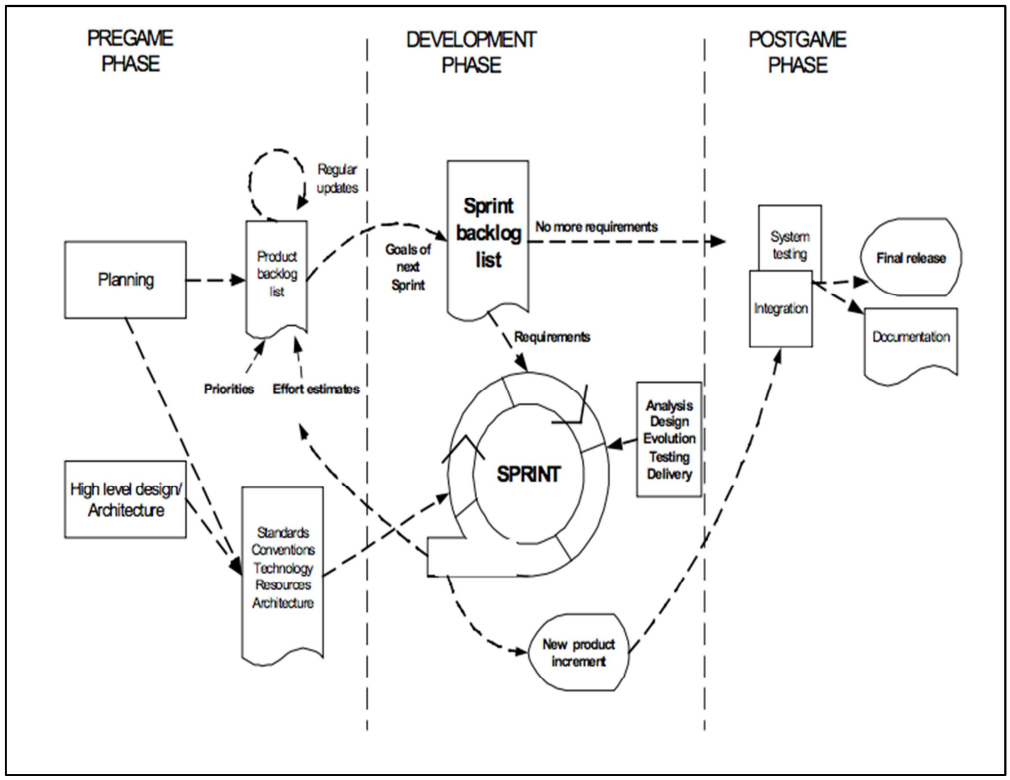


Figura 6: Scrum (Abrahamsson et al. , 2002)

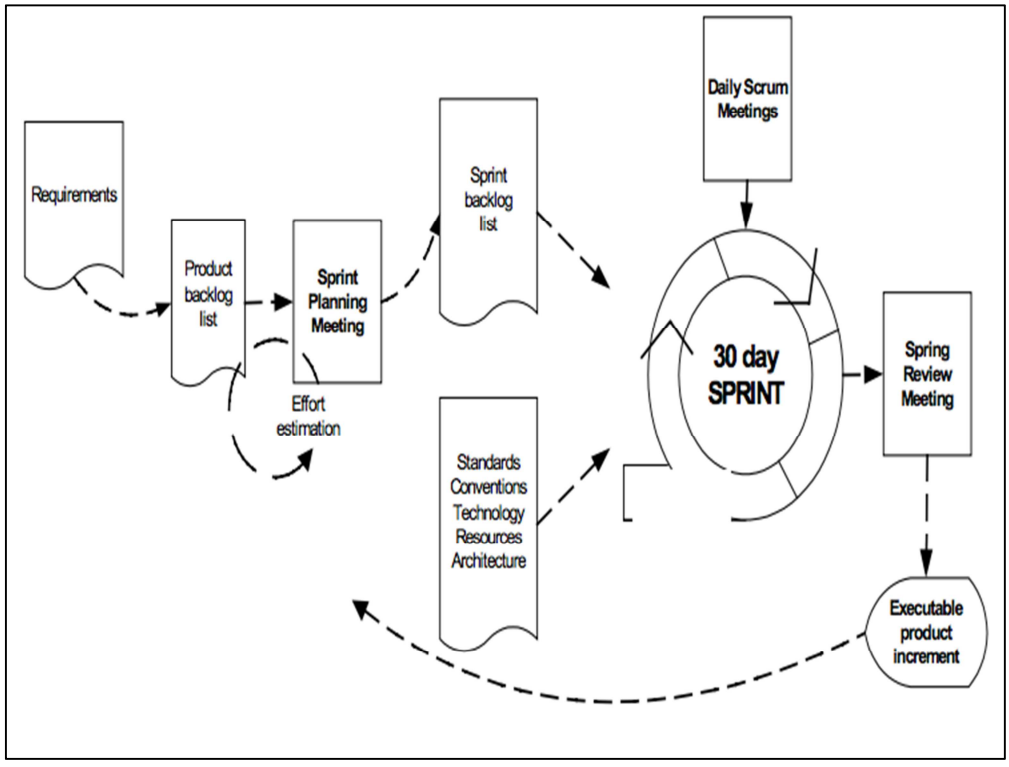


Figura 7: Scrum (Abrahamsson et al. , 2002)

Na fase de desenvolvimento o software é desenvolvido em ciclos iterativos conhecidos como *Sprints*. Cada Sprint inclui as fases tradicionais da engenharia de software: requerimento; análise; design; evolução e; promoção para a produção (*delivery*). A arquitetura e o design são evoluídos durantes os *Sprints* (Abrahamsson et al. , 2002).

Inicia-se a fase pós-jogo quando se chega a conclusão que o produto está pronto para ser implantado em produção. Neste momento algumas tarefas importantes são executadas, tais como: integração, testes do sistema e documentação (Abrahamsson et al. , 2002).

Há seis perfis que os *stakeholders* podem assumir no *Scrum*:

- *Scrum master: responsável por garantir que sejam seguidos as práticas, valores e regras;*
- *Product Owner* : responsável por gerenciar o *Product Backlog,* na estimativa de esforço e converter os itens do backlog em *features* a serem desenvolvidas;
- Equipe *scrum:* é a equipe do projeto;
- Cliente e;
- Gerente.

Do ponto de vista da rastreabilidade, diversos elementos deste processo podem ser de interesse: *Product Backlog*; *Sprint Backlog*; *Sprint*; *release*; testes de integração e de sistema; stakeholders e seus papéis; documentação e; arquitetura (Abrahamsson et al. , 2002).

Família *Crystal*

Na figura 8 vemos o processo *Crystal Orange*. Este processo faz parte da família *Crystal*, que é um conjunto de processos de desenvolvimento ágeis. A proposta é permitir selecionar a metodologia mais adequada para o projeto baseado no tamanho e na criticidade. Por isto são organizados numa escala que vai desde dos mais leves até os mais pesados, respectivamente: *Crystal Clear* (mais leve); *Crystal Yellow; Crystal Orange; Crystal Orange Web; Crystal Red; Crystal Maroon; Crystal Diamond* e; *Crystal Sapphire* (Abrahamsson et al. , 2002). No entanto, apenas *Crystal Orange* e *Crystal Clear* foram de fato instanciados.

A família *Crystal* possui algumas características, regras e valores que são comuns a todos eles. Todos eles usam desenvolvimento iterativo incremental com ciclos fixos que vão de um a quatro meses. Ênfase em comunicação e cooperação entre as pessoas. Não determina nenhuma prática de desenvolvimento específica, ferramenta, ou produto de trabalho, permitindo, por exemplo, o uso de XP ou Scrum (Abrahamsson et al. , 2002).

A seguir veremos algumas das políticas padronizadas destas metodologias (Abrahamsson et al., 2002): Entrega incremental numa base regular; Acompanhamento do progresso do projeto baseado nas entregas de software e decisões relevantes no lugar de documentos escritos; envolvimento direto dos usuários; teste funcional automático regressivo; validações de dois usuários por liberação; *workshop* para ajuste do produto e do processo no meio de cada incremento.

Os produtos de trabalho comuns ao *Crystal Clear* e *Crystal Orange* são os seguintes (Abrahamsson et al. , 2002): sequência de liberação (*release*); modelos dos objetos comuns; manual do usuário; casos de teste e; código de migração. Ademais, *Crystal Clear* inclui casos de uso ou *features*, enquanto *Crystal Orange* requer documentos de especificação. A documentação no *Crystal Clear* é mais leve: desenhos das telas e esquemas do design. Enquanto o *Crystal Orange* usa desenhos da interface de usuário (*UI Design*).

Do ponto de vista da rastreabilidade a família *Crystal* apresenta vários conceitos de interesse: Iteração, liberações incrementais testadas e funcionando (releases); projeto de software; *stakeholders* (usuários e equipe de desenvolvimento); testes funcionais automáticos regressivos; código fonte; modelos de objetos; requisitos, *features* e casos de uso; desenhos de telas e; interfaces UI (Abrahamsson et al. , 2002).

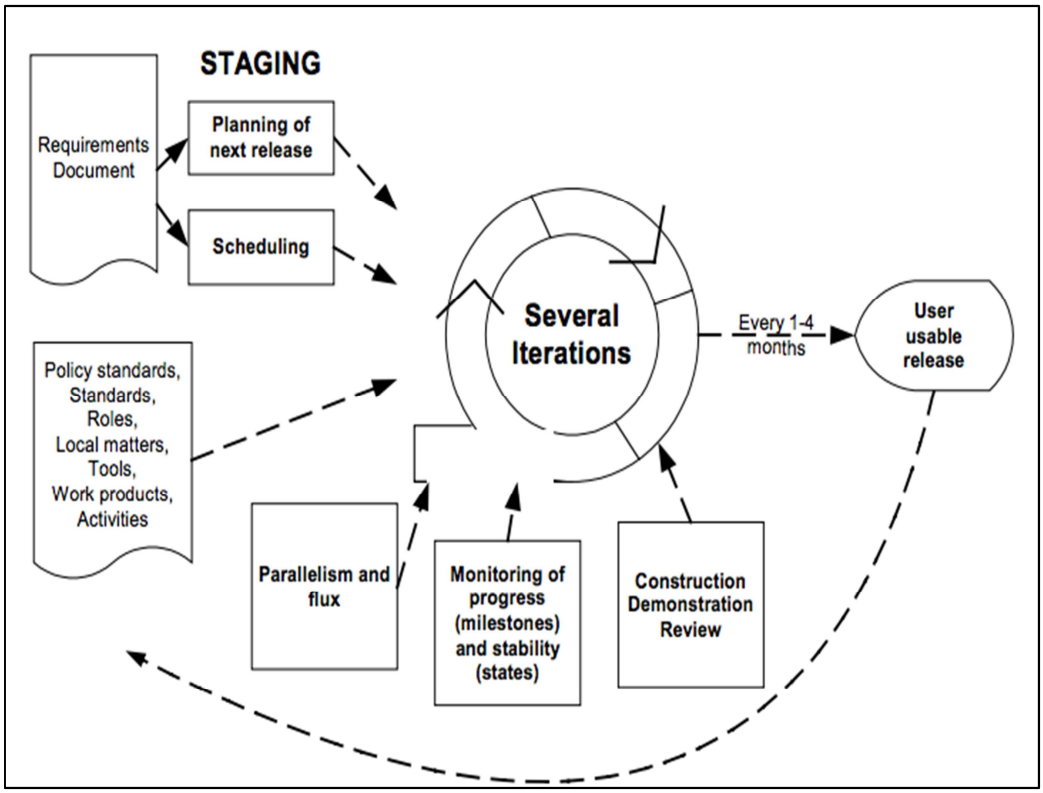


Figura 8: *Crystal Orange* (Abrahamsson et al. , 2002)

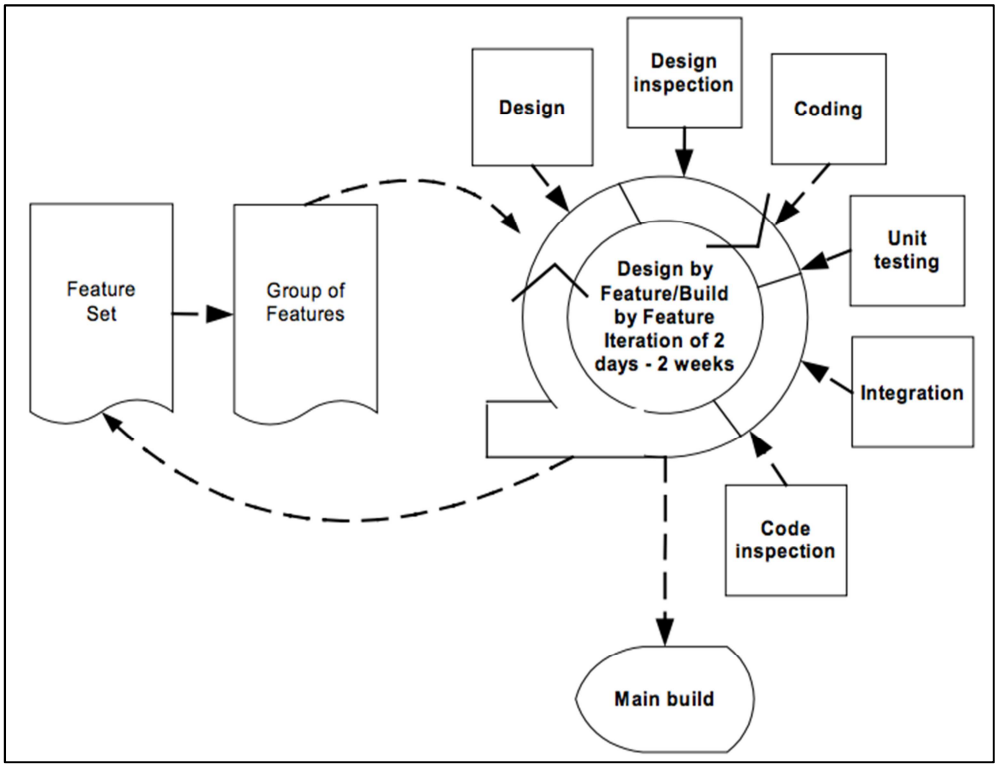


Figura 9: FDD (Abrahamsson et al. , 2002)

FDD – *Feature Driven Design*

Na Figura 9 podemos ver o processo FDD – *Feature Driven Design*. FDD não cobre todo o processo de desenvolvimento, ao contrário foca-se nas fases de *design* e construção. Emprega desenvolvimento iterativo e incremental e inclui entregas frequentes e tangíveis. O FDD inclui cinco fases sequenciais: desenvolvimento do modelo de alto-nível; construção da lista de *features*; planejar pelas *features*; projetar a aplicação utilizando as *features* (*design by features*), (Abrahamsson et al. , 2002).

Na fase de desenvolvimento do modelo de alto-nível os especialistas no domínio da aplicação informam a equipe do projeto e aos arquitetos a descrição de alto nível do sistema. Modelos de objetos são criados com base nos documentos de requisitos existentes, (Abrahamsson et al. , 2002).

Em seguida entramos na fase de construção da lista das *features*, esta lista é validada pelos usuários e patrocinadores e apresentada para a equipe de desenvolvimento, (Abrahamsson et al. , 2002).

Tendo em mãos a lista das *features* a serem construídas, um plano de alto nível é elaborado, aonde as *features* são sequenciadas de acordo com suas prioridades e dependências. Neste momento um cronograma e os principais marcos podem ser criados para os principais conjuntos de *features,* (Abrahamsson et al. , 2002).

Por fim seguimos para o design e a construção, que são baseados nas *features*. A cada iteração, um pequeno subconjunto das *features* é selecionado do conjunto das *features* para design e construção pelas equipes de desenvolvimento. Este processo iterativo é inclui inspeção de design, codificação, teste unitário, integração e inspeção de código. Ao final de uma iteração bem sucedida, as *features* finalizadas são promovidas para comporem o *build* principal, (Abrahamsson et al. , 2002).

Seis papéis são identificados no FDD: gerente de projeto; arquiteto chefe; gerente de desenvolvimento; programador chefe; equipe de desenvolvimento e; especialistas do domínio, (Abrahamsson et al. , 2002).

Do ponto de vista da rastreabilidade, várias entidades podem ser de interesse (Abrahamsson et al. , 2002): iterações; modelos de objetos; arquitetura; conjunto de todas as *features*; subconjunto das *features*; planejamento; projeto; cronograma; *stakeholders* e seus papéis; teste; integração; inspeção e; código.

DSDM – Dynamic Systems Development Method

A Figura 10 exibe o DSDM. O processo consiste de cinco fases: estudo de viabilidade; estudo do processo negócio; interação para modelagem funcional; iteração para *design* e *build* e; implementação (Abrahamsson et al. , 2002).

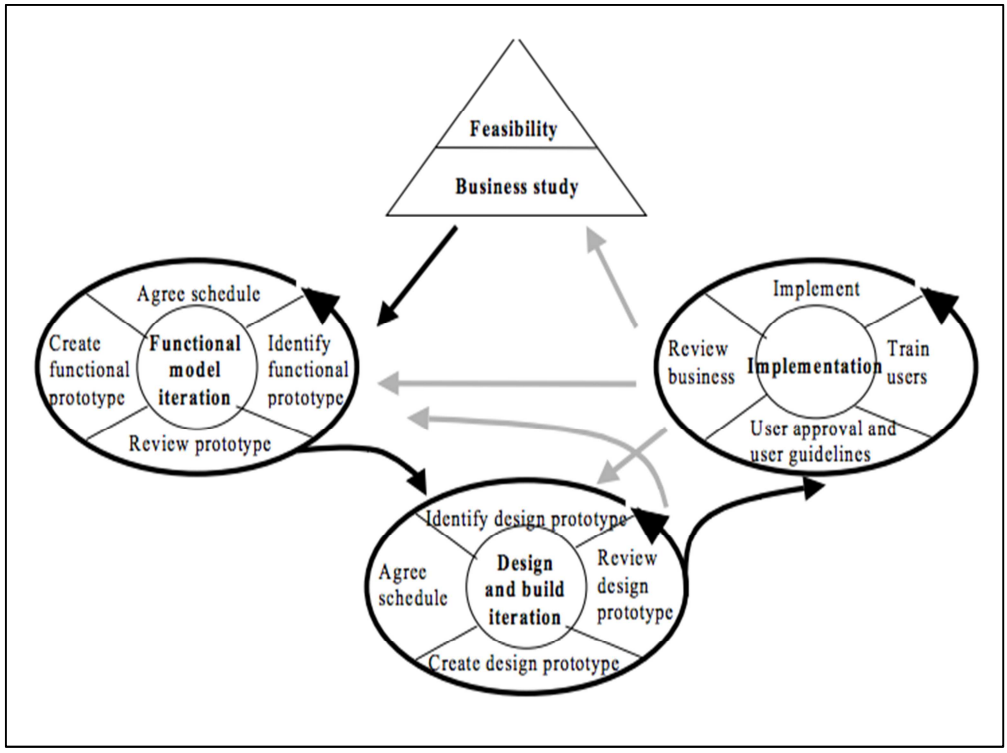


Figura 10: DSDM (Abrahamsson et al. , 2002)

No estudo de viabilidade a sustentabilidade do projeto é avaliada. A fase de viabilidade está preocupada se tecnicamente falando é possível, ou não, continuar com o projeto, e os riscos nele contidos. Dois produtos de trabalho resultam como saída da fase: um relatório de viabilidade e; um plano para desenvolvimento. Opcionalmente, um protótipo rápido pode ser elaborado se o negócio ou a tecnologia não forem bem conhecidos (Abrahamsson et al. , 2002).

Na fase de estudo do processo de negócio as características essenciais do negócio e da tecnologia são analisados. Sendo criadas descrições de alto-nível em artefatos como o diagrama entidade-relacionamento ou modelos de objetos de negócio. Também a definição de alto nível da arquitetura do sistema e um plano de prototipação são criados. O plano de prototipação visa estabelecer uma estratégia para esta atividade nas próximas fases (Abrahamsson et al. , 2002).

A fase seguinte é a iteração de modelagem funcional. Nela um modelo funcional é construído, a medida que cada iteração é concluída, contendo protótipo do código e os modelos de análise. Teste também faz parte desta fase (Abrahamsson et al. , 2002).

São quatro as saídas desta fase: a lista das funções priorizadas; o documento de revisão da prototipagem funcional; os requisitos não funcionais e; uma análise de risco de se fazer mais desenvolvimentos (Abrahamsson et al. , 2002).

Na iteração de design e build o sistema é efetivamente construído. A saída desta fase é um sistema testado que atende no mínimo ao conjunto requisitos acordados (Abrahamsson et al. , 2002).

Na fase implementação o sistema é transferido do ambiente de desenvolvimento para o ambiente de produção. Se o processo envolve muitos usuários e requer muito tempo, ele pode ser feito em iterações. Suas saídas incluem um manual de usuário, e um relatório de revisão do projeto (Abrahamsson et al. , 2002).

Identificamos vários *stakeholders* com diversos papéis: desenvolvedores; desenvolvedores seniores; coordenado técnico; embaixador do usuário, visionário, consultor e; patrocinador (Abrahamsson et al. , 2002).

Os seguintes conceitos podem ser de interesse da rastreabilidade: iteração; código; teste; protótipo; *stakeholders* e seus papéis; requisitos; lista de requisitos; modelos de análise; projeto; relatório de viabilidade e; plano de desenvolvimento.

ASD – *Adaptative Software Development*

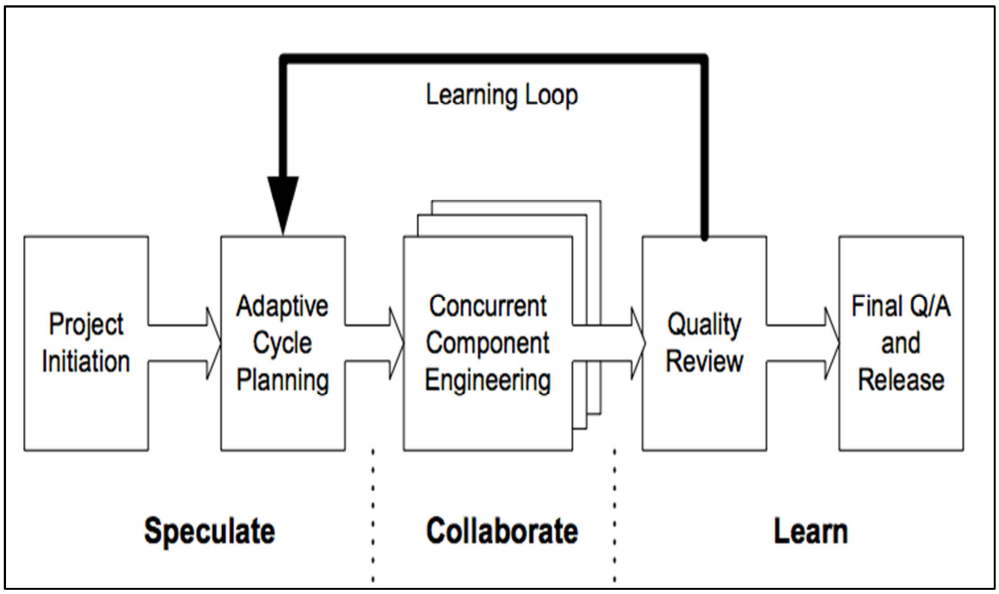


Figura 11: ASD (Abrahamsson et al. , 2002)

A Figura 11 exibe o processo ASD. Ele baseia-se principalmente no problema de desenvolver sistemas grandes e complexos. Este método encoraja desenvolvimento iterativo e incremental com prototipagem constante. Oferece um framework com bastante orientações para evitar projetos de caírem no caos, sem suprimir o senso de emergência e criatividade necessários(Abrahamsson et al. , 2002).

O processo ASD é executado em ciclos de três fases: Especulação; Colaboração e; Aprendizado. O nome Especulação é usado para enfatizar o papel da mudança. Colaboração destaca o papel do trabalho em equipe para desenvolver sistemas que mudam demais. E aprendizado, que é a necessidade de reconhecer e aprender com os erros (Abrahamsson et al. , 2002).

Na fase de iniciação uma noção de alto nível do que será o projeto é adquirida. Define-se uma visão do projeto, uma folha com os dados do projeto e uma especificação do produto. O cronograma global do projeto é fixado. No ASD o controle da execução é explicitamente feito pelos componentes, no sentido, que o que importa é se o componente foi construído na qualidade esperada, ao invés das tarefas e do processo usado para fazê-lo (Abrahamsson et al. , 2002).

O ciclo de aprendizado que aparece na figura 11, destaca que as avaliações de qualidade realimentam o planejamento dos próximos ciclos. Quanto ao ASD não descreve em detalhes a estrutura de suas equipes, portanto poucos perfis são apresentados, por exemplo: Patrocinador executivo; facilitador e; gerente de projeto (Abrahamsson et al. , 2002).

Do ponto de vista da rastreabilidade podemos identificar os seguintes conceitos no ASD: projeto; componentes; *stakeholders* com seus papéis e; iteração.

Disciplina	Mapeamento no MRMA	Artefato/Atividade	XP	SCRUM	FDD	DSDM	ASD	CRYSTAL
Gerenciamento de Projetos	*Project*	*Estudo de Viabilidade*	x			x		
		Protótipos do sistema de alto nível	x			x		
		Visão da Arquitetura em alto nível	x	x		x		
		Levantamento inicial dos requisitos	x			x		
		Estimativa de esforço	x	x	x			
		Ordem de prioridade	x	x	x	x		
		Cronograma			x			
		Retrospectivas	x	x				
		Sprint Reviews		x				
		Reuniões Diárias		x				
		Revisão pelo negócio		x		x		
Engenharia de Software	Product Backlog	Conjunto de todos os requisitos do produto (Ex: Product Backlog)	x	x	x	x	x	x
	Sprint Backlog	Conjunto dos requisitos da iteração (Product Backlog)	x	x	x	x	x	x
	Itens de Backlog	Histórias de Usuários	x	x				
		Features	x	x	x			x
		Casos de Uso	x	x				
		Documentos de Requisitos						x
	Não mapeado	Análise	x	x	x			
		Iteração de modelagem funcional				x		
		Modelo funcional				x		
		Design	x	x	x			
		UI Design				x		x
		Arquitetura	x		x			
		Modelos de objetos			x			x
	Test_Case	Testes de Integração, Unitário, Funcional, etc.	x		x			x
	Não mapeado	Revisão contínua	x					
		Inspeção de Código	x		x			
	Deploy	Automatizado.						
		Repositórios de código fonte.	x	x	x	x	x	x
	Code	Código	x		x			x
	Sprint Backlog	Iteração	x	x	x	x	x	x
	Stakeholder	Stakeholders e seus papéis	x	x	x	x		x

Tabela 4: Mapeamento dos artefatos e atividades dos Métodos Ágeis que podem ser de interesse da rastreabilidade, baseado em Abrahamsson et al. (2002).

Na Tabela 4 mostramos de forma consolidada os diversos artefatos e atividades dos Métodos Ágeis que podem ser de interesse da rastreabilidade. Deve

ser destacado que quando dizemos pode, isso significa que na hora de instanciar, é sempre possível incluir ou excluir conceitos. A escolha de quais serão incluídos, e quais serão excluídos é uma decisão a ser tomada na hora de realizar a instanciação.

Os Métodos Ágeis são compostos de atividades e artefatos de gerenciamento de projetos e da engenharia de software, organizados para permitir a construção de softwares por meio de iterações de tamanho fixo. Estes artefatos e atividades apresentados na Tabela 4 e nas Figuras de 5 a 11, são as entidades que podem ser de interesse da rastreabilidade, por exemplo: Histórias de Usuários e Testes, e serão considerados no capítulo 4.

3 Trabalhos Relacionados

Este capítulo tem como objetivo apresentar uma série de trabalhos a respeito de Modelos de Rastreabilidade, rastreabilidade em Métodos Ágeis e seus problemas e benefícios. A análise deles mostra grandes avanços na busca de uma solução para os problemas associados a rastreabilidade em Métodos Ágeis, e fornecem uma boa visão do estado da arte neste tema.

Martins et al. (2006) e Sommerville e Kotonia (1998) defendem a rastreabilidade de requisitos como um fator importante para o desenvolvimento de software. E, argumentam que embora Métodos Ágeis abordem algumas descrições e práticas da engenharia de software, eles não focalizam na rastreabilidade de requisitos (Zemont, 2005).

Um grande desafio para muitos é implementar a rastreabilidade em Métodos Ágeis, percebendo que a pouca documentação dificulta a realização de tal atividade. Foram realizadas pesquisas que retratassem problemas enfrentados por Métodos Ágeis para alcançar a rastreabilidade de requisitos e também trabalhos que apresentassem modelo de rastreabilidade. A seguir serão comentados alguns dos trabalhos encontrados.

3.1 Modelo de rastreabilidade proposto por Ramesh e Jarke (2001)

Ramesh e Jarke (2001) sugerem que seja feito o uso de um framework de modelagem para realizar as ligações entre requisitos de forma organizada. Em seu trabalho eles realizaram entrevistas com 26 grandes organizações de desenvolvimento de software onde demonstraram uma vasta gama de práticas de rastreabilidade. A partir disto, Ramesh e Jarke (2001) sintetizaram modelos de rastreabilidade que mapeiam diversas tarefas e entidades de desenvolvimento, tais como aqueles apresentados na Figura 13: requisitos; procedimentos de verificação de conformidade e; componentes do sistema. Foram feitos estudos de caso para validar o modelo e em seguida implantá-lo em ferramentas de rastreabilidade.

Ramesh e Jarke (2001) apresentam um metamodelo, adequado para

representar em alto nível o que deve conter qualquer modelo de rastreabilidade. Nele podem ser identificadas três entidades *Stakeholder*, *Object* e *Source*. *Object* é descrita como uma entidade que representa as entradas e saídas do processo de desenvolvimento, exemplos de *Objects* incluem requisitos, premissas, desenhos, *rationale*, alternativas, fatores críticos de sucesso, etc. Eles são os principais objetos conceituais entre os quais a rastreabilidade é mantida durante os vários estágios do ciclo de vida.

Stakeholders representam as pessoas que executam as atividades do ciclo de vida de manutenção e desenvolvimento de software e que agem em diferentes papéis na criação e uso dos diversos objetos conceituais e ligações de rastreabilidade.

Source retrata as fontes ou locais em que as informações encontram-se armazenados, que podem ser meios físicos como documentos, ou intangíveis tais como referências a pessoas ou políticas não documentadas e procedimentos. Um exemplo são os documentos de especificação de requisitos.

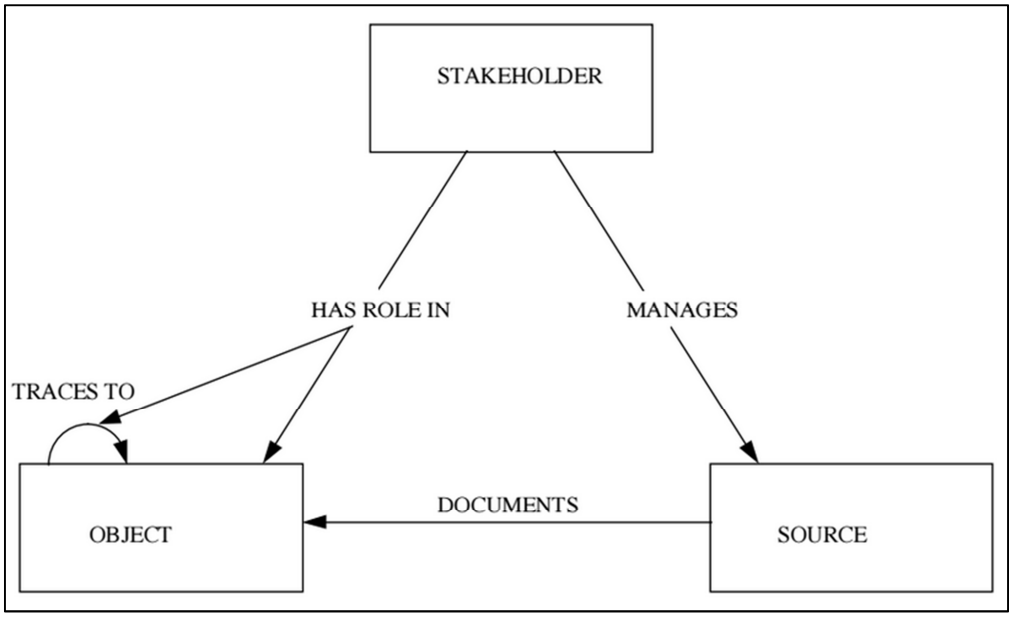


Figura 12: Metamodelo de Rastreabilidade Ramesh e Jarke (2001).

A partir do metamodelo da Figura 12 Ramesh e Jarke (2001) avançaram e identificaram dois tipos de usuários, *Low End User* (Usuário Iniciante) e *High End User (Usuário Avançado)*. E elaborou modelos mais detalhados, adaptados para as necessidades destes usuários. A Tabela 5 faz uma comparação entre eles e mostra

algumas diferenças relevantes na forma e no propósito pelo qual eles utilizam a rastreabilidade, percebe-se que os primeiros estão focados no produto a ser entregue, enquanto os últimos preocupam-se também com o processo e os projetos em todo o ciclo de vida do produto. Considerando estas definições, eles apresentaram modelos de rastreabilidade adaptados para estas duas categorias de usuários.

Característica	***Usuário Iniciante***	***Usuário Avançado***
Definição de rastreabilidade pelo usuário	Rastrear a transformação dos documentos dos requisitos até o design	Forma para aumentar a probabilidade de produzir um sistema que atende a todos os requisitos do cliente e irá ser fácil de ser mantido.
Principal aplicação da rastreabilidade	Rastrear a decomposição dos requisitos; a alocação dos requisitos; a verificação de conformidade e; o controle de mudanças.	Total cobertura do ciclo de vida; inclusão de usuários e clientes; capturar itens discutidos; decisões e *rationales*; capturar rastros através das dimensões de produtos e processos.

Tabela 5: Comparação *Usuários Iniciantes* versus *Usuários Avançados*

Segundo Ramesh e Jarke (2001) os Usuários Iniciantes usam a rastreabilidade para modelar as dependências entre os requisitos, alocação dos requisitos aos subsistemas, verificação de conformidade e controle da mudança. Conforme pode ser visto na Figura 13.

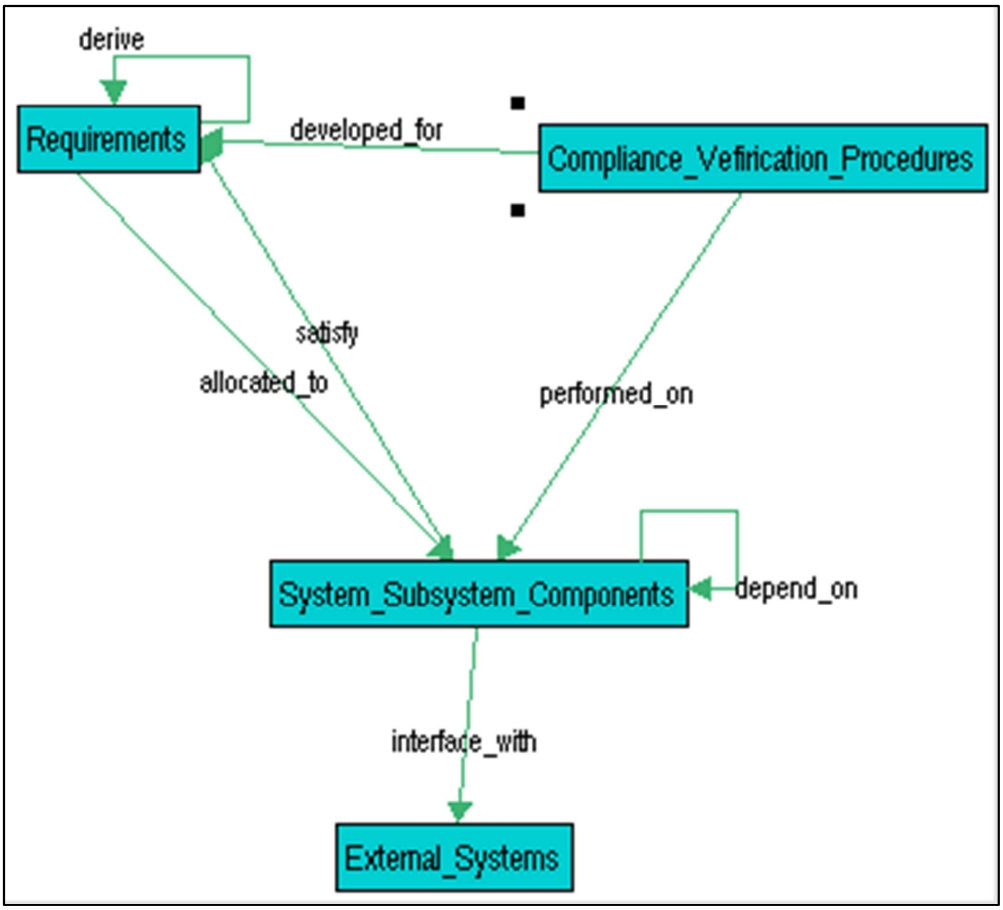


Figura 13: *Modelo de Rastreabilidade Usuários Básicos*, Ramesh e Jarke (2001).

Por outro lado, os Usuários Avançados fazem uso de esquemas de rastreabilidade muito mais ricos, por conta disso Ramesh e Jarke (2001) segregaram o modelo de rastreabilidade em quatro partes: gerenciamento de requisitos; verificação da conformidade; *rationale* e; alocação ao *design*.

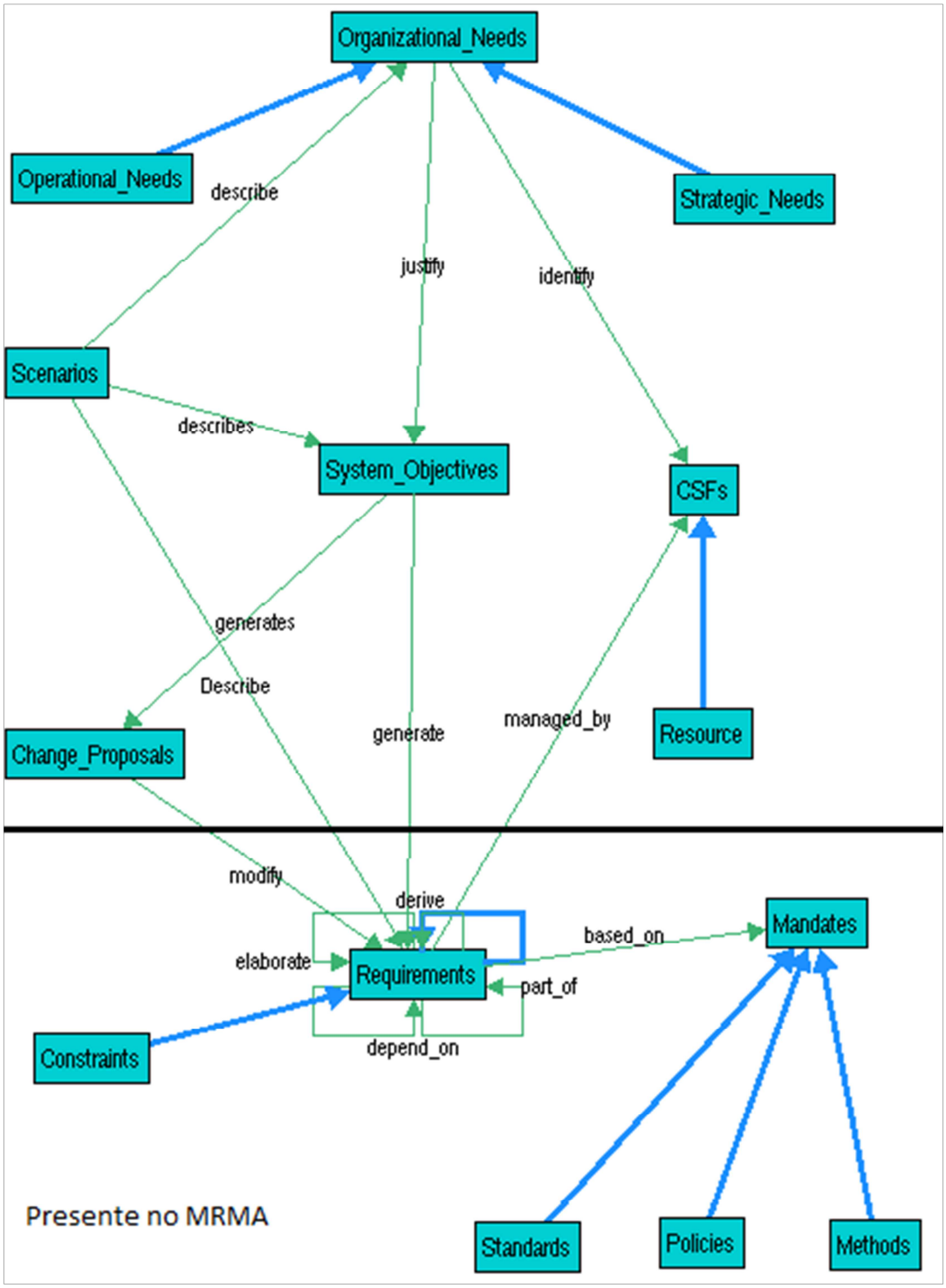


Figura 14: Submodelo de Gerenciamento de Requisitos ,

Ramesh e Jarke (2001).

Na Figura 14 pode ser visto o submodelo de gerenciamento de requisitos, nele as necessidades justificam os objetivos de um sistema, que por sua vez são desdobrados nos requisitos, os quais por sua vez se referenciam aos padrões, políticas e normas que regem os processos de negócio aos quais o sistema deve

apoiar. Os objetivos são descritos em cenários, que também servem para descrever os requisitos e as necessidades organizacionais.

Mudanças nos objetivos podem gerar solicitações de mudanças que resultam em alterações nos requisitos. Estes, por sua vez, podem estar relacionados com outros de várias formas, um requisito pode depender, detalhar, derivar ou ser parte de outro, por esta razão o modelo de Ramesh e Jarke (2001) apresenta *links* recursivos entre requisitos, com diversos tipos de *links* de rastreabilidade entre requisitos, tais como: *depend_on*; *part_of* e *elaborate*.

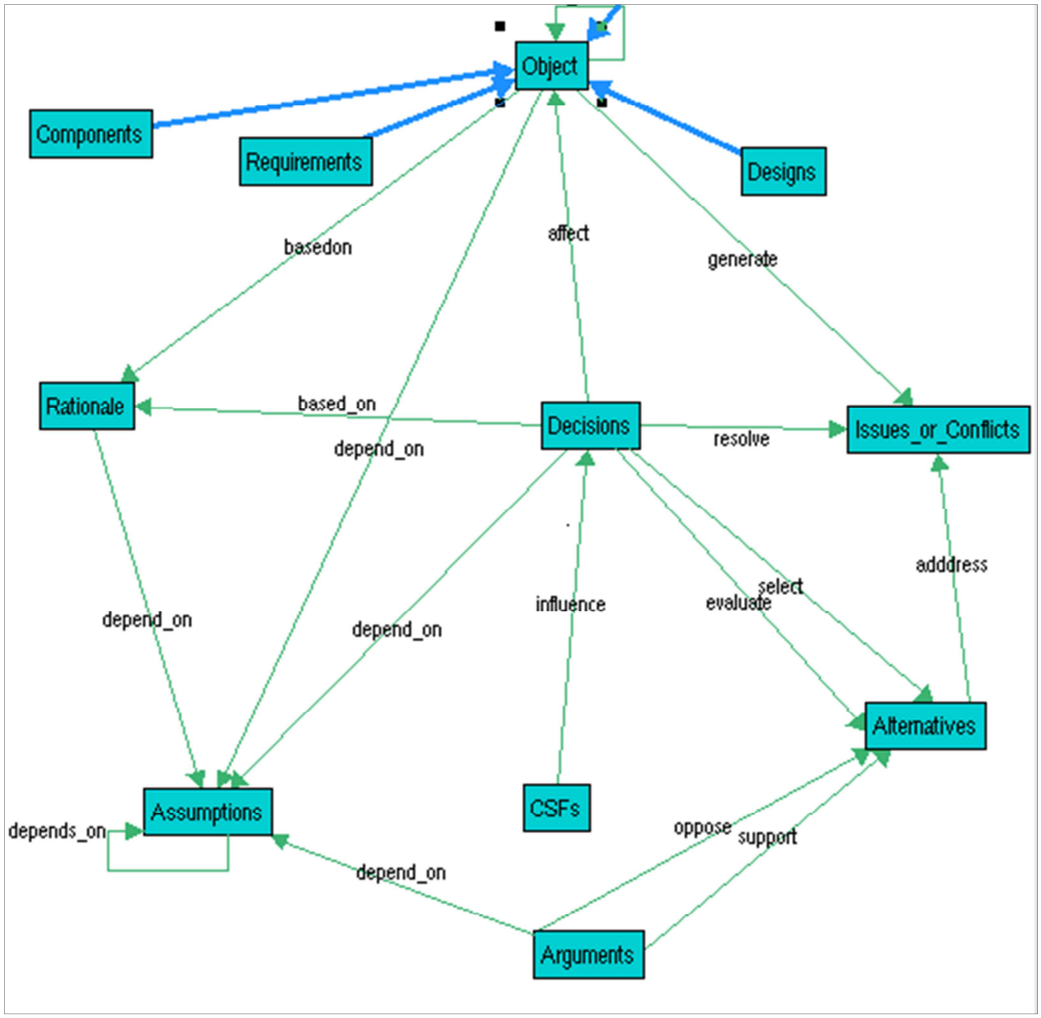


Figura 15: *Submodelo de Rationale,* Ramesh e Jarke (2001).

Durante o ciclo de vida de um software é normal que ocorram divergências devido a diferentes interpretações, interesses, pontos de vistas, premissas, etc. Estas diferenças levam a conflitos que devem ser resolvidos antes que se prossigam as atividades de desenvolvimento de software. Ramesh e Jarke (2001) destacam que as informações de como as decisões foram tomadas para que se resolvesse o

problema precisam ser preservadas, para permitir que os requisitos dos clientes sejam entendidos e satisfeitos e, neste sentido, propõe o modelo da Figura 15.

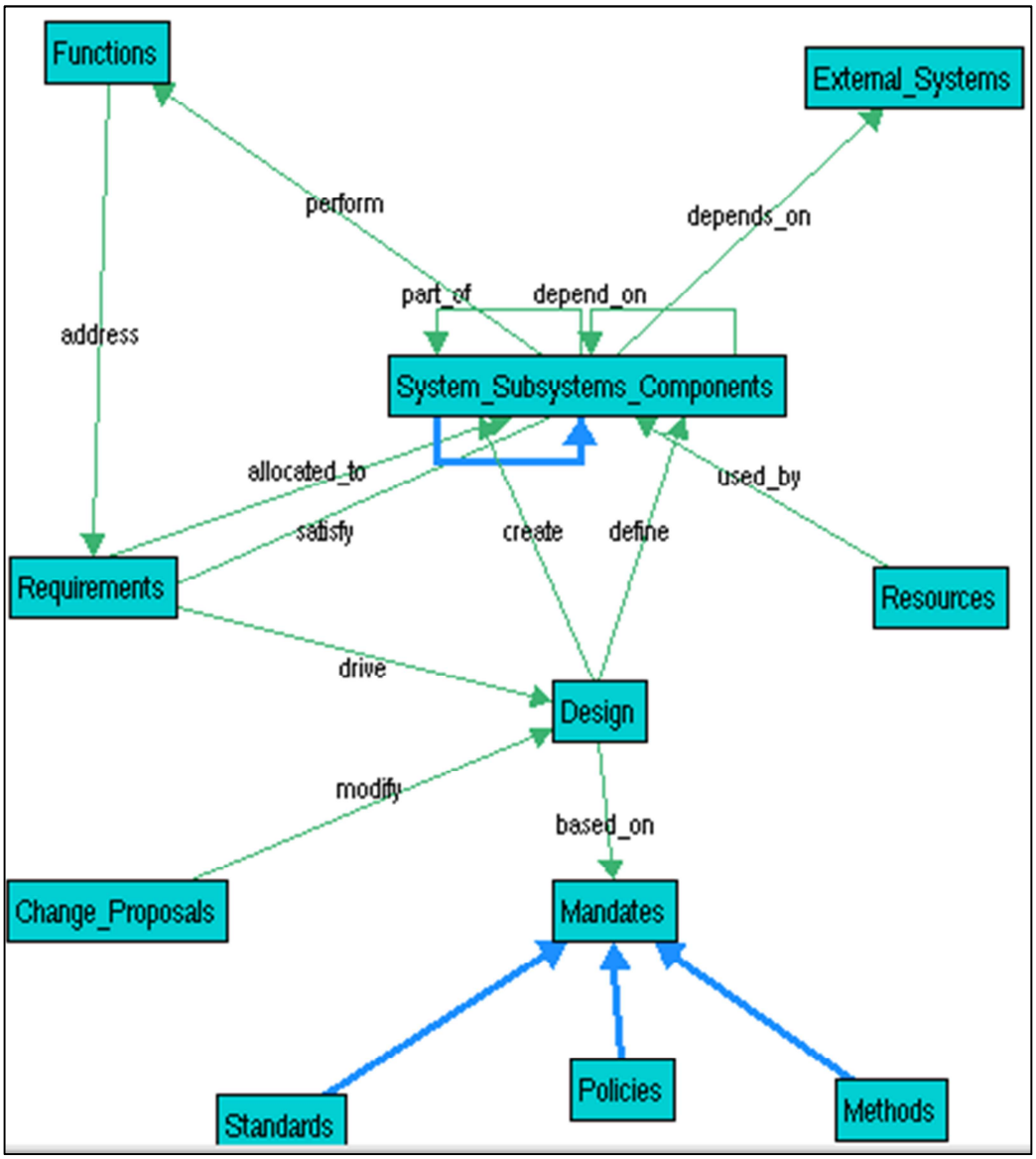


Figura 16: Submodelo de Alocação de *Design*, Ramesh e Jarke (2001).

A Figura 16 mostra que o design toma como base padrões, políticas e normas e são guiados pelos requisitos, que os dirigem e são alocados, no processo de design, aos componentes dos subsistemas. Componentes estes que por sua vez formam os blocos de construção do sistema. Estas associações entre requisitos e design e entre requisitos e componentes ajudam a compreender, por exemplo, se todos os requisitos foram alocados aos componentes do sistema, ou se há componentes no sistema para os quais não se consiga definir os requisitos que eles satisfazem.

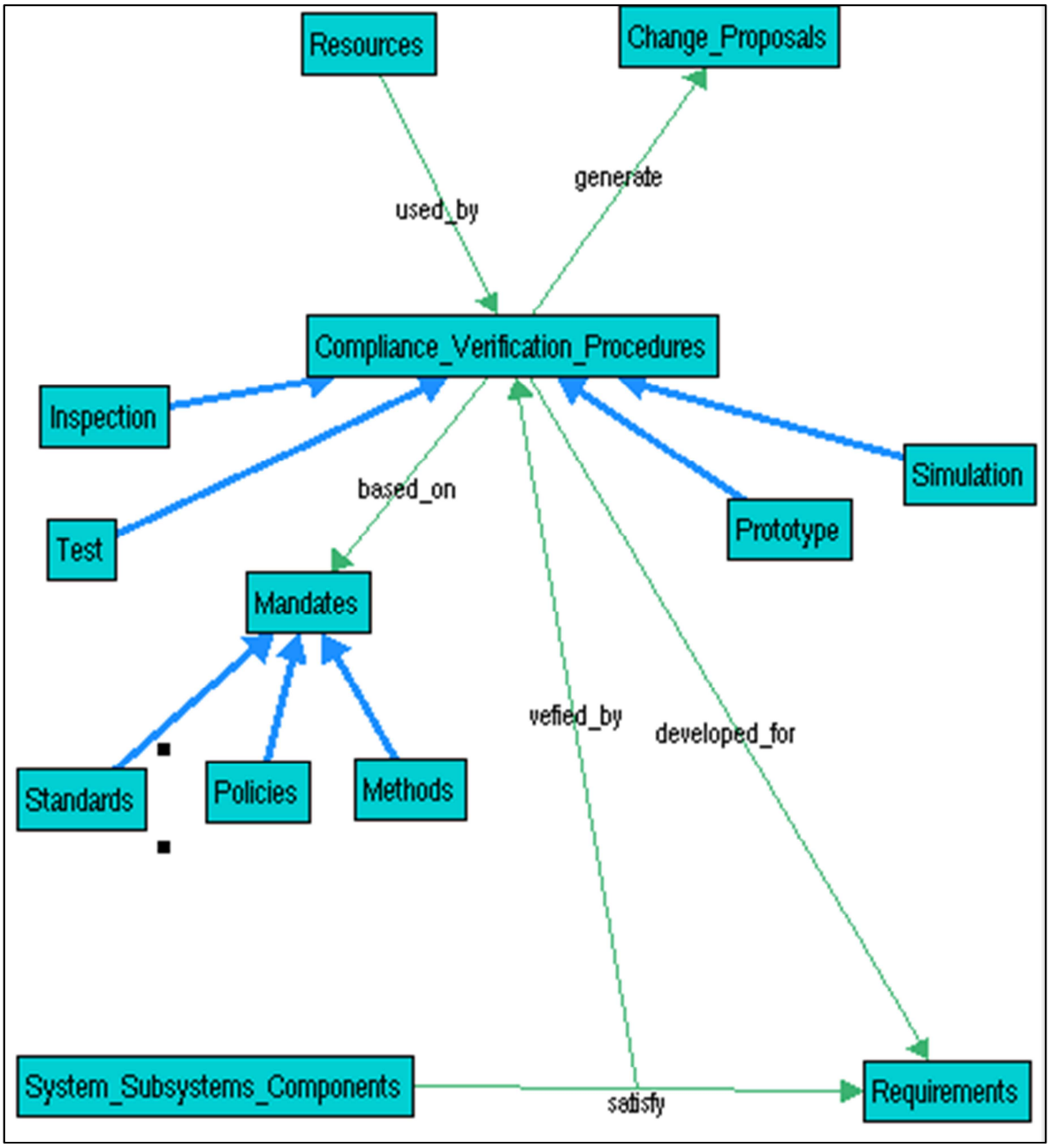


Figura 17: Submodelo de Verificação de Conformidade, Ramesh e Jarke (2001)

Por fim, Ramesh e Jarke (2001) descrevem o submodelo de verificação de conformidade, cuja função é garantir se cada requisito foi adequadamente considerado e tratado durante o desenvolvimento, de modo a permitir que se possa garantir que o sistema atende aos requisitos especificados.

O modelo de Ramesh e Jarke (2001) mapeia quatro conceitos do desenvolvimento de software: Requisitos; *Design*; *Rationale* e; Verificação de Conformidade. Três deles foram diretamente usados no MRMA, a saber, Requisitos, *Rationale* e Verificação de Conformidade. *Design* não foi explicitamente

considerado, pois o modelo foi mais voltado para rastreabilidade de requisitos, mas ele é aberto, e pode ser estendido livremente, permitindo que se acrescente outras entidades, por exemplo, conceitos de *Design*.

Quando comparamos o modelo de requisitos de Ramesh e Jarke (2001) com o MRMA, uma diferença fundamental pode ser percebida. Os diversos Métodos Ágeis: *Scrum*; XP; FDD; *Crystal Orange* e; ASD apresentam pequena cobertura, ou nenhuma em alguns casos, para conceitos no domínio do problema: necessidades estratégicas; Necessidades Operacionais; Cenários; Objetivos de Negócio e recursos. Por esta razão, o MRMA não inclui estas entidades.

No submodelo Ramesh e Jarke (2001) condensam requisitos numa única entidade denominada "*Requirements*". Os requisitos em Métodos Ágeis podem ser trabalhados como Histórias de Usuário, *Features*, Casos de Uso, Critérios de Aceitação atuando como requisitos e, Especificações de requisitos. Neste sentido o MRMA apresenta uma modelagem mais completa que o modelo de Ramesh e Jarke (2001), incluindo estes diversos artefatos de requisitos.

O Submodelo de *Rationale* apresenta muitas entidades, sendo bastante rico. No entanto, adotamos uma abordagem mais simples, incluindo apenas uma entidade associada a este conceito, que pode ser estendida conforme a necessidade do usuário para incluir "Decisões", "Conflitos", "Alternativas"', "Premissas" e "Argumentos".

Além disso, o modelo de Ramesh e Jarke (2001) não mapeia nenhum conceito específico dos Métodos Ágeis, por exemplo, não apresenta entidades como: Histórias de Usuário; *Features*; Histórias de Teste e; *Sprints*. E não apresenta nenhuma forma de se trabalhar com *links* definidos pelo usuário. Algo necessário para Métodos Ágeis, devido a sua característica de apresentar diversos conceitos que possuem semântica diferente daquelas dos Métodos Tradicionais, como explicado por Espinoza e Garbajosa (2011).

3.2 Propostas de rastreabilidade em Métodos Ágeis

Espinoza e Garbajosa (2011) definem três características gerais que os modelos de rastreabilidade devem apoiar: tipos de *links* de rastreabilidade definidos pelos usuários; papéis e; regras de ligações definidos pelos usuários. Trazendo

como principais benefícios: um suporte mais extenso aos Métodos Ágeis, quando comparado às abordagens de rastreabilidade existentes e; permitir que se obtenha um grau de automação mais elevado, com um processo de aquisição e manutenção dos dados de rastreabilidade adaptados para Métodos Ágeis.

Como apresentado na Figura 18, os autores desta abordagem propõe um meta-modelo de rastreabilidade chamado TMM, que pode ser usado como uma referência para a criação de modelos de rastreabilidade específicos para cada projeto, que segundo eles cobre todo o ciclo de vida e pode ser usado tanto para processos ágeis quanto para os tradicionais.

O modelo de Espinoza e Garbajosa (2011), apresenta uma melhoria quando comparado ao modelo Ramesh e Jarke (2001) por reconhecer as diferenças semânticas entre alguns artefatos dos Métodos Ágeis comparado aos Métodos Tradicionais e prever *links* de rastreabilidade definidos pelos usuários. No entanto, ele pode ser entendido mais como um modelo flexível, adaptável a qualquer tipo de processo, do que um modelo específico para Métodos Ágeis. Faltam conceitos específicos dos processos ágeis, tais como: Histórias de Usuário; *Features*; Histórias de Teste; *Backlog* e; *Sprint*.

Além disso, o MRMA pode ser interpretado como uma instanciação do modelo de Espinoza e Garbajosa (2011), em que as *Tasks* e *Sprints* são elementos do tipo *TraceabilityWorkUnitKind*, *Stakeholder* com seu papel (*Role*) são elementos do tipo *TraceabilityProducerKind* e as demais entidades do tipo *TraceabilityWorkProductKind*.

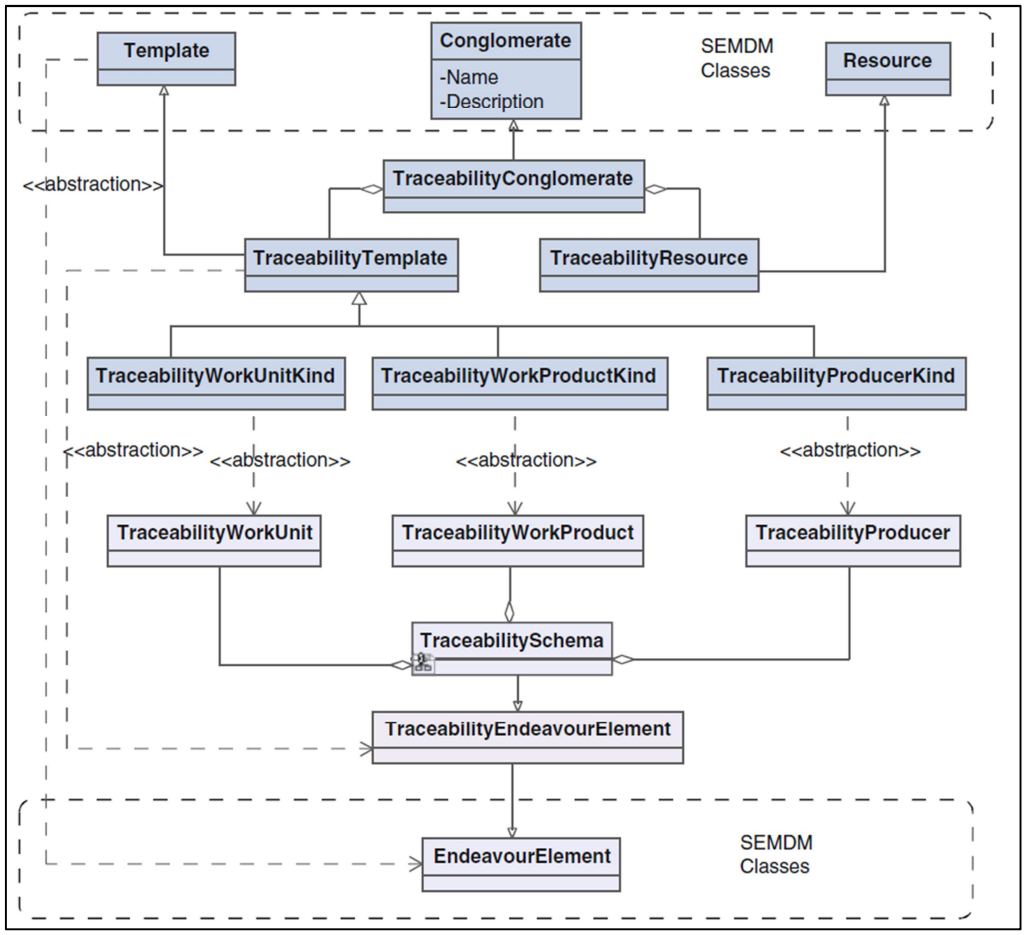


Figura 18: Modelo de rastreabilidade, Espinoza e Garbajosa (2011).

Outro trabalho é o de Taromirad e Paige (2012) que fala da importância da rastreabilidade em projetos que possuem diferentes domínios de engenharia. Nele é proposta uma abordagem baseada em modelo, em que se desenvolve uma solução de rastreabilidade que age em projetos multi-domínio. Também apresenta um protótipo de uma ferramenta que apoia a abordagem implementada.

Taromirad e Paige (2012) propõe uma abordagem de modelagem de rastreabilidade para prover esquemas de rastreabilidade leves e simples, contendo apenas o suficiente para as necessidades de cada domínio. Neste sentido, esta proposta se assemelha muito a abordagem adotada no MRMA, em que um modelo foi proposto especificamente para o domínio dos processos ágeis de desenvolvimento.

Um estudo empírico é realizado por Alaa e Samir (2014) com o objetivo de reunir os tipos de rastreabilidade de requisitos importantes para uma equipe SCRUM, assim como seus motivos com base em informações de diferentes

integrantes com diferentes papéis na equipe. A pesquisa tem o importante papel de fazer uma revisão na literatura onde procura conhecer a taxonomia de rastreabilidade de requisitos no contexto de desenvolvimento ágil, pois segundo eles há um grande número de modelos de rastreabilidade de requisitos e ferramentas sem uma avaliação esquematizada.

Uma dos principais diferenciais do trabalho de Alaa e Samir (2014) foi o de propor um modelo de rastreabilidade já especificamente desenhado para o Scrum. E, a partir deste modelo apresentar um conjunto de tipos de links de rastreabilidade entre suas entidades, organizadas de acordo com as necessidades de cada *stakeholder* (*Product Owner*, Analista de Sistemas).

Quando comparado com o MRMA, a principal limitação deste modelo quando comparamos com o MRMA é que foi modelado apenas para o SCRUM, estando incompleto, contendo apenas uma parte pequena das entidades deste processo.

De forma similar, Jaber et al. (2013), em sua pesquisa, identificaram os tipos de ligações de rastreabilidade que as diferentes partes da equipe estão interessadas. Então, a partir do resultado é criado um modelo de vínculos de rastreabilidade que diferencia os diferentes tipos de rastreabilidade de acordo com os papéis. Ao final, é realizado um estudo que avalia os benefícios da rastreabilidade, resultando no desenvolvimento de um protótipo de ferramenta para rastrear e exibir *links* de rastreabilidade – *TraceLink.* Jaber et al. (2013) concluíram que ligações de rastreabilidade ajudam nas atividades de manutenção.

Nesta dissertação defendemos que o MRMA, um modelo de rastreabilidade para Métodos Ágeis, trará benefícios para estes processos. O trabalho de Jaber et al. (2013) corrobora com esta afirmação, ao demonstrar que o uso da rastreabilidade em manutenções de software faz com que as tarefas sejam até 86,06% mais precisas do que sem ela. Desta maneira reforçando os argumentos desta dissertação a favor de um modelo para Métodos Ágeis.

Ghazarian (2008) apresenta padrões de rastreabilidade como uma abordagem para garantir que o requisito seja rastreável através do código fonte no desenvolvimento ágil, mostrando seu modelo conceitual para a rastreabilidade através da notação UML - *Unified Modeling Language*.

A proposta de Ghazarian (2008) embora válida por apresentar uma forma de

mapear requisitos aos componentes de código fonte, é muito limitada quanto a sua cobertura dos Métodos Ágeis, estando ausentes conceitos importantes mapeados no MRMA, tais como: *Test Cases*, *Sprints*; *Backlog*; *Backlog Itens*; *Project* e; *Deploy*.

Enquanto que Arbain et al. (2014) citam as mudanças constantes em requisitos funcionais que metodologias ágeis enfrentam, e com isso o impacto que pode causar nos requisitos não funcionais. Já em seu trabalho é apresentado um modelo de processo de rastreabilidade que trata do rastreamento de requisitos não funcionais, principalmente em relação a segurança e desempenho.

Comparado com o MRMA o Modelo de Arbain et al. (2014) é incompleto no que se refere a ausência de diversos artefatos específicos dos Métodos Ágeis, exatamente como foi explicado para o trabalho de Ghazariam (2008). Mas apresenta como forte diferencial a presença do conceito “Requisitos Não Funcionais” (NFR). Que não é encontrado em nenhum dos outros modelos para Métodos Ágeis apresentados neste capítulo. Reconhecemos a importância de se rastrear os requisitos não funcionais, portanto os incluímos no MRMA.

Uma ferramenta, UNICASE, é criada por Delater e Paech (2013), que captura de forma semi-automática *links* de rastreabilidade entre os requisitos e o código no decorrer do desenvolvimento. Após seu primeiro trabalho, eles realizam um estudo empírico com base em sua abordagem, onde é aplicado em três projetos de desenvolvimento de diferentes de alunos de graduação, avaliando a sua viabilidade. Enquanto que Omoronyia et al. (2011) cita o desafio que pode ser enfrentado por ferramentas que geram a rastreabilidade de forma automática, sendo ele a geração de muitos relacionamentos de rastreabilidade dos quais alguns são vistos como irrelevantes. Portanto, em seu trabalho, é apresentada duas abordagens para classificar as relações de rastreamento geradas.

Para o sucesso de um modelo de rastreabilidade em Métodos Ágeis, automatizar o processo de captura e geração dos *links* ao máximo possível, é um passo importante. A proposta do MRMA foca no modelo, mas não estabelece uma forma para a captura dos *links*, este é uma ausência que pode ser complementada posteriormente em trabalhos futuros.

A proposta de Delater e Paech (2013) , no que se refere a apresentar uma

ferramenta, UNICASE, e permitir a captura semi-automática de *links* de rastreabilidade, apresenta vantagens quando comparada a este trabalho acadêmico, mas não contempla Métodos Ágeis, sendo deficiente neste quesito.

Um modelo de rastreabilidade para suportar a transição entre os processos de desenvolvimento tradicionais e os Métodos Ágeis é proposto por Furtado e Zisman (2014), fornecendo suporte para rastrear artefatos tais como: Histórias de Usuário para diagramas de classe, diagramas de sequência e diagramas de caso de uso; teste de aceitação para projeto de teste; História de usuário para Personas, entre outras possibilidades.

3.3 Discussão

Os modelos de rastreabilidade apresentados nos trabalhos relacionados, embora apresentem importantes conceitos necessários para implementar rastreabilidade, falham em dar o suporte adequado aos Métodos Ágeis. Isso ocorre por alguns motivos como:, não contêm os conceitos necessários, ou por estarem incompletos, ou por que são Metamodelos e portanto num nível mais genérico que o MRMA, que pode ser entendido como uma instanciação destes Metamodelos.

Apesar disso, eles oferecem contribuições úteis à proposta desta pesquisa de um modelo de rastreabilidade para Métodos Ágeis, tais como, a ideia de tipo de link definido pelo usuário de Espinoza e Garbajosa (2011), que segundo eles permite uma melhor adequação aos tipos de ligação em Métodos Ágeis que são diferentes dos tradicionais.

Também, o fato de deixar claro que em um modelo de rastreabilidade devem existir três tipos de entidades: Unidades de Trabalho (*WorkUnit*); Produto de Trabalho (*WorkProduct*) e Produtor (*Stakeholder),* responsável pelos produtos de trabalho e/ou pelas Unidades de Trabalho.

Outra contribuição advém de que nos modelos de rastreabilidade para Métodos Ágeis, embora incompletos, já aparecem algumas das entidades importantes que contribuem para o nosso modelo.

A definição de Usuário Iniciante e Usuário Avançado (Ramesh e Jarke, 2001), traz a necessidade de que modelos de rastreabilidade possam sofrer movimentos de

especialização/generalização em que sejam acrescentados ou simplificados conforme as necessidades de seus usuários. Característica que deve ser aproveitada em Modelos para processos ágeis.

Ramesh e Jarke(2001) também apresentaram os conceitos de produto, processo e projeto, assim como alguns outros conceitos interessantes, como: *rationale*; requisitos; componentes e; vários exemplos de tipos de *links* de rastreabilidade.

A seguir mostramos como o Modelo de Rastreabilidade para Métodos Ágeis (MRMA) foi concebido a partir dos modelos de processos e das contribuições de alguns dos trabalhos discutidos nesta seção.

4. Modelo de Rastreabilidade para Gestão de Projetos no Contexto Ágil

A criação do MRMA baseou-se nos modelos de processos apresentados no capítulo 2 e nos modelos de rastreabilidade apresentados no capítulo 3 e foi especificamente construído para rastreabilidade em Métodos Ágeis, a seguir será explicado como se chegou a este modelo.

4.1 Construção do modelo de referência

A metodologia para construção do modelo de referência seguiu os seguintes passos:

1. Foi realizada uma revisão de literatura para identificar artefatos, *stakeholders* e aspectos processuais das diversas metodologias ágeis;
2. Após isso uma segunda pesquisa foi conduzida a procura de trabalhos científicos sobre rastreabilidade em métodos ágeis;
3. Uma terceira revisão foi feita para mapear os conceitos, estado da arte e problemas da rastreabilidade de forma geral, e não específica para métodos ágeis;
4. De posse destas informações, e baseado nos modelos processuais de diferentes Métodos Ágeis, das entidades encontradas nos trabalhos

correlatos de rastreabilidade em Métodos Ágeis, no resultado da pesquisa *State of Agile™ Survey* (2015) que exibe um relatório mostrando percentualmente quais são os Métodos Ágeis mais usados. Foram selecionados os conceitos que iriam compor o modelo.

Dois critérios nortearam a escolha dos conceitos. O primeiro deles é que não seriam mapeados conceitos de todas as diferentes formas de Metodologias Ágeis, pois isso traria muita complexidade ao serem incluídas entidades com pouca representatividade no universo dos Métodos Ágeis.

O segundo critério foi que apesar de se tratar de um modelo de referência para Métodos Ágeis, poderiam ser incluídos conceitos das metodologias tradicionais, caso eles pudessem agregar valor ao Modelo. Por esta razão a entidade *Rationale* foi incluída.

Deste modo, foram selecionadas as metodologias Scrum e XP, que são as Metodologias Ágeis mais utilizadas. Para tal usamos como base os resultados da pesquisa *State of Agile™ Survey* (2015). Nesta pesquisa é mostrado que 72% dos praticantes de Métodos Ágeis utilizam Scrum puro, combinado com XP, ou com Kanban. As demais formas de Métodos Ágeis apresentaram percentuais de uso pouco significantes e foram descartadas.

Outra informação importante para entendimento do modelo, é que para a maioria dos conceitos presentes em nossa proposta, optamos por utilizar os termos de uso mais comum pelos praticantes dos Métodos Ágeis. Desta forma, embora pudesse ter sido usado o termo iteração, optamos por usar Sprint em seu lugar.

4.2 Mapeando entidades dos processos ágeis para o modelo

Na Figura 4, extraída do trabalho de Abrahamsson et al. (2002), os autores fazem uma análise dos Métodos Ágeis por todo o ciclo de vida, comparando-os em termos da disciplina Gerenciamento de Projetos, do processo e, por último, das práticas, atividades e produtos de trabalho.

O gráfico demonstra que a maioria dos Métodos Ágeis possui no escopo de seus processos suporte para o gerenciamento de projetos, por este motivo uma entidade para captura de informações de projetos, "*Project*", foi incluída na

dimensão “*Process*” no modelo da Figura 19.

Métodos Ágeis são processos de desenvolvimento de software, característica que é explicitada na Figura 4 pela presença do suporte de processo ao ciclo de vida de software.

Para facilitar o entendimento, organizamos os conceitos no Modelo de Rastreabilidade para Métodos Ágeis num total de cinco agrupamentos denominados dimensões, respectivamente: *Transversal_Dimension; Requirements_And_Enhancements_Dimension*; *Process_Dimension*; *Product_Dimension e; Quality_Dimension.*

As atividades de desenvolvimento acontecem por meio de iterações que pressupõe que o que vai ser construído esteja organizado em listas de itens, de modo a serem executados por atividades de desenvolvimento organizadas dentro de janelas de tempo de tamanho fixo, pré-definido no início do projeto. No modelo da Figura 19 as iterações, janelas de tempo de tamanho fixo, são mapeadas para a entidade "*Sprint*" e as atividades para "*Task*", ambos os conceitos foram alocados na dimensão “*Process*”.

Analisando o processo de desenvolvimento ágil *Extreme Programming* (XP), percebemos que as Estórias de Usuários contêm os requisitos a serem desenvolvidos. No XP identificamos um portfólio geral de requisitos e subconjuntos deste portfólio, que devem ser implementados a cada iteração. Esta organização dos requisitos numa visão geral dos requisitos do produto e em subconjuntos para implementação, também acontece no SCRUM, FDD, *Crystal Orange*, DSDM e ASD.

Uma observação interessante é que segundo Abrahamsson et al. (2002), o RUP pode ser adaptado para uma versão mínima capaz de atender aos Princípios Ágeis, para que assim possa ser considerado Método Ágil. Ajustado desta maneira o RUP apresenta a mesma forma de organização dos requisitos em portfólio geral e subgrupos por iteração dos demais Métodos Ágeis.

Para representar estes dois conceitos: portfólio geral de requisitos e; subconjuntos deste portfólio. Foram incluídas na Dimensão “*Requirements and Enhancements*” do nosso modelo, respectivamente as entidades “*Product Backlog*”

e “*Sprint Backlog*” . Relembrando que estamos seguindo o princípio de nomearmos as entidades pelo nome mais significativo para os praticantes de métodos ágeis, como foi explicado.

Outro conceito extraído do XP e acrescentado no modelo, foi o de testes, representado pela entidade “*Test Cases*” na dimensão “*Quality*”. Semelhante ao que foi explicado anteriormente para requisitos, este conceito é válido para SCRUM, FDD, *Crystal Orange*, DSDM e ASD, como pode ser visto no trabalho de Abrahamsson et al. (2002), nas Figuras 5 a 11, assim como na Tabela 3.

O conceito de *release* ou *deploy*, presente no XP, pode ser encontrado em todos os Métodos Ágeis, como pode ser visto no trabalho de Abrahamsson et al. (2002) e nos gráficos das Figuras 5 a 11. Em nosso modelo ele foi acrescentado na dimensão “Product”.

O Scrum prevê que aos itens do “*Backlog*” sejam atribuídas prioridade e estimativa de esforço, estes atributos foram acrescentados à entidade “*Backlog Item*”.

Em nosso modelo os itens de “*Backlog*”, entidade “*Backlog Item*”, são elementos que representam as variadas formas de armazenar os requisitos observados nos diversos tipos de processos ágeis, por exemplo: Scrum, XP, ASD, Crystal, etc. Desta forma, eles agregam as Estórias de Usuário do XP e do Scrum, os casos de uso do RUP, as *Features* do FDD, os requisitos da Família Crystal e assim por diante. Esta entidade foi acrescentada à dimensão *“Requirements and Enhancements*”.

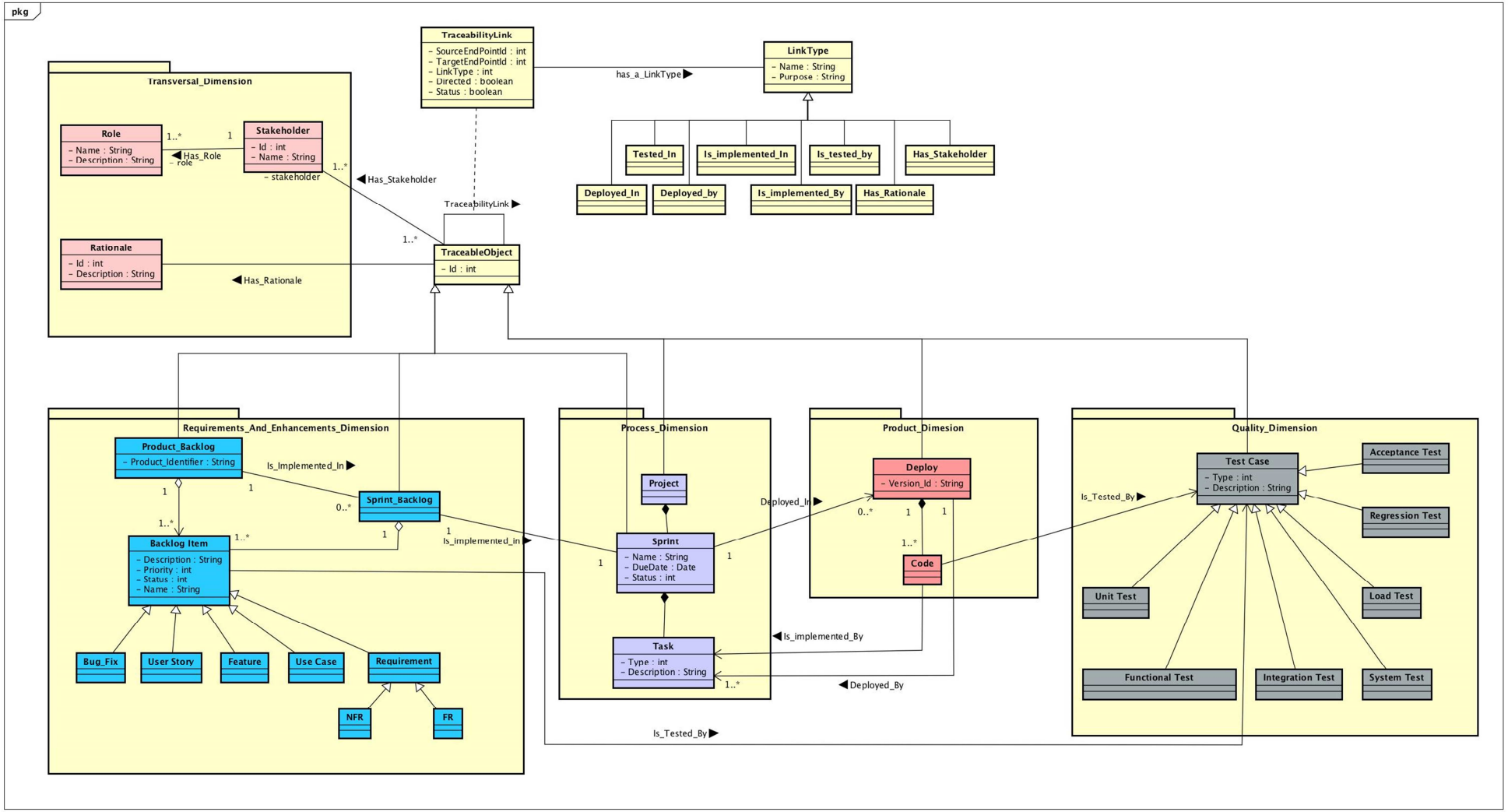


Figura 19: Modelo de Rastreabilidade para Métodos Ágeis

O conceito de Iteração, mapeada na entidade “*Sprint*” da dimensão “*Process*”, aponta a necessidade de armazenar informações sobre a data de entrega de cada iteração “*Sprint*” e controle do status dos *Sprints*. Esta necessidade pode ser identificada nos diversos tipos de Métodos Ágeis, como pode ser visto em Abrahamsson et al. (2002) e nas Figuras 5 a 11. Por exemplo, no processo *Crystal* Orange, nele podemos ver que há uma atividade de monitoramento do progresso do status dos serviços, esta foi a motivação para a inclusão dos atributos “*DueDate*” e “*Status*” na entidade “*Sprint*”.

Todos os Métodos Ágeis trabalham sobre o princípio de entregar código testado e funcionando. No desenvolvimento de software, o código fonte é uma entidade essencial, pois todos os requisitos devem ser alocados as suas respectivas unidades de código fonte, e sempre que alterações são necessárias, uma das tarefas essenciais é identificar os elementos do código fonte que serão alterados pela mudança.

O FDD, exibido na Figura 9, deixa clara a importância de se rastrear a entidade “*Code*”, diversas atividades acontecem em torno do código fonte, como os testes unitários, a codificação, as atividades de integração e a inspeção de código. A correta execução delas durante o ciclo de vida do software depende muitas vezes de se poder determinar corretamente as relações entre os artefatos, tais como: é detalhado por; é testado por; é dependente de, etc.

Deste modo, rastrear o código e suas inter-relações com outras entidades, quando possível, é de extrema utilidade, em especial para as manutenções futuras do sistema dentro do ciclo de vida do produto, apoiando a gestão e a avaliação do impacto da mudança. Todos estes argumentos justificaram a inclusão do conceito código, “*Code*”, na dimensão “*Product*” do modelo de referência (MRMA).

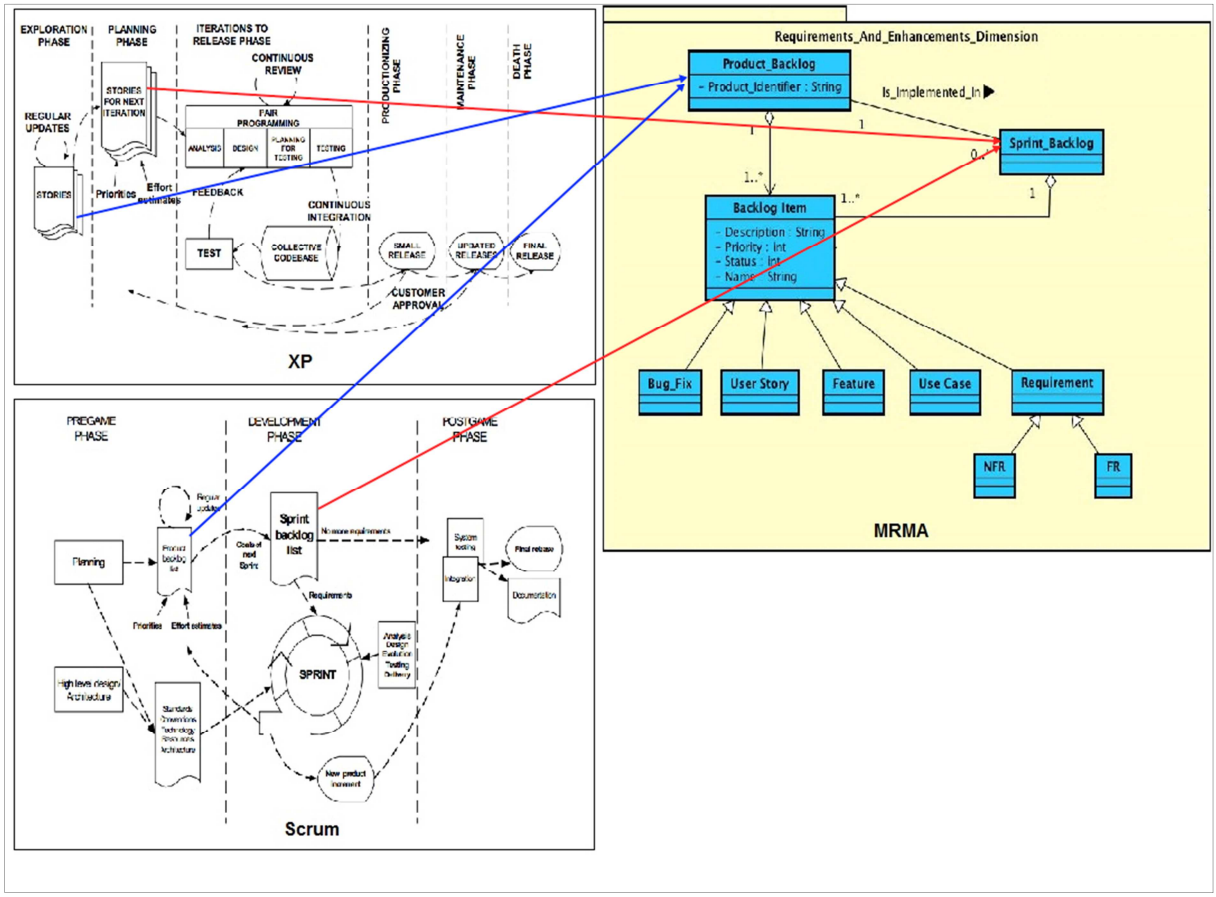


Figura 20: Exemplo de mapeamento de entidades de processo para se tornarem entidades do modelo de rastreamento MRMA.

A Figura 20 mostra um exemplo de mapeamento, onde as histórias a serem trabalhadas na próxima iteração do XP e a lista do Scrum contendo os itens de *Backlog* do *Sprint* foram identificadas e levadas para fazerem parte do modelo MRMA como as entidades *Product Backlog* e *Sprint Backlog*.

4.2.1 Mapeando entidades e características de outros modelos de rastreabilidade

A criação do MRMA foi baseada nas metodologias de processos de desenvolvimento ágeis. Elementos destes processos, tais como: atividades; artefatos e; pessoas e papéis, foram identificados e colocados no MRMA. Além das próprias metodologias de desenvolvimento, algumas entidades de outras propostas de modelos de rastreabilidade foram incluídas. A seguir descreveremos com mais detalhes estas entidades.

Do Metamodelo de Rastreabilidade de Ramesh e Jarke (2001) da Figura 12, foram extraídos o conceito *Stakeholder* com a entidade *Role*, para representar as pessoas que são responsáveis por atividades ou por artefatos nos processos de desenvolvimento ágeis e suas diversas funções num projeto.

O conceito *Object* da Figura 12 já está largamente representado no MRMA, são elementos como *Sprint*, *Backlog e Backlog Item*. Enquanto que a entidade *Source*, não foi explicitamente representada no modelo, para manter o MRMA simples, mas ao ser instanciado este conceito pode ser incluído como uma entidade a parte ou como um atributo das demais entidades, por exemplo, um atributo da entidade *BackLog Item*.

Outro conceito trazido de Ramesh e Jarke (2001), foi o elemento *Rationale*, que no MRMA agrupa, condensa, os vários vieses deste tipo de informação, tais como: decisões, premissas; alternativas, ou qualquer insumo sobre os quais se basearam as decisões tomadas durante o processo de desenvolvimento.

O MRMA não trouxe nenhum elemento que mapeie o domínio do problema ou seus objetivos, conceitos que, por exemplo, aparecem no modelo de Ramesh e Jarke (2001), Figura 14, pois as metodologias ágeis não incluem estas atividades explicitamente. No entanto, há propostas para enriquecer os Métodos Ágeis, como pode ser visto em Jaqueira (2013), e a inclusão deste conceitos pode ser reconsiderada em trabalhos futuros.

Espinoza e Garbajosa (2011) destacam três características que um modelo de rastreabilidade para Métodos Ágeis deve dar suporte para ser menos dependente de um processo de rastreabilidade específico: *links* de rastreabilidade definidos pelos usuários; papéis e; regras de ligações definidas pelos usuários.

O modelo de rastreabilidade de Espinoza e Garbajosa (2011), apresenta três elementos constituintes: *TraceabilityWorkUnit*; *TraceabilityWorkProduct* e; *TraceabilityProducer*. Eles representam os conceitos de rastreabilidade mais comumente empregados nos modelos de rastreabilidade e representam três aspectos a serem considerados ao se definir uma metodologia de rastreabilidade específica: os artefatos a serem usados e produzidos; as pessoas envolvidas e; o processo a ser seguido.

No MRMA, a entidade *Stakeholder* representa as pessoas, com a entidade *Role* agregando a característica de se ter papéis definidos pelo usuário, conforme defendido por Espinoza e Garbajosa (2011), enquanto as entidades das dimensões *Process Dimension* denotam o processo a ser seguido e os demais elementos mapeiam os artefatos a serem usados ou produzidos.

4.2.2 Descrevendo o modelo

O modelo apresentado na figura 19 é composto por cinco dimensões: "*Transversal*"; "*Requirements and Enhancements*"; "*Process*"; "*Product*" e; "*Quality*". Esta forma de organizar foi uma escolha nossa para facilitar o entendimento e agrupar entidades conceitualmente relacionadas. Por exemplo, na dimensão "Transversal", encontramos todas as entidades que se ligam a conceitos de mais de uma dimensão.

4.2.2.1 Dimensão Transversal

A dimensão Transversal foi criada para manter as entidades que podem se relacionar a dois ou mais conceitos nas outras dimensões do modelo. A escolha desta abordagem foi para evitar poluir o modelo de referência com um número grande de *links*. Para entender esta problemática, basta considerar que a entidade "*Stakeholder*" possui associações múltiplas, "*Stakeholder*" podem ser, ao mesmo tempo, responsáveis por definir Estórias de Usuário, por uma "*Task*" de testes, e o proprietário (*owner*) do projeto. Estes perfis (*roles*) diferentes para o mesmo "*Stakeholder*" resultam em múltiplas relações.

Na dimensão transversal temos três conceitos: "*Stakeholder*", "*Role*" e "*Rationale*". O propósito da entidade "*Stakeholder*" é manter a identidade das pessoas cujo papel (*Role*) as tornam responsáveis por artefatos ou tarefas. Esta

entidade também foi proposta no modelo de Ramesh e Jarke (2001) e pode ser muito importante em métodos ágeis, pois na possível ausência de documentação, identificar o "*Stakeholder*" pode ser de fundamental importância para a obtenção de informações a respeito de artefatos ou tarefas.

Ramesh e Jarke (2001) mostraram em seu modelo de referência, que uma das ligações entre *Stakeholders* e Objetos (Tarefas e Artefatos) é a ligação "Possui papel em", essa ligação deixa explícito que um mesmo "*Stakeholder*" pode ter vários perfis, e que para executar determinada tarefa, ou para produzir determinado artefato é preciso possuir certos perfis. Por esta razão, incluímos a entidade "*ROLE*" no modelo de referência.

Em Métodos Ágeis a comunicação entre as pessoas é valorizada acima da documentação, por isso busca-se que as informações sejam compartilhadas pela equipe. Mas num cenário como o desenhado no início deste trabalho, isso pode ser um verdadeiro problema. Neste sentido, implementar "*Rationale*" no modelo de referência justifica-se por reconhecer a importância de se resgatar e reter os porquês que fundamentam as escolhas e decisões, num horizonte de tempo que cubra todo o ciclo de vida do produto, conforme defendido em Lago et al. (2009) e Ramesh e Jarke (2001).

4.2.2.2 Dimensão Process

A dimensão "Process" ("*Process Dimension*") contempla a necessidade de rastreabilidade de informações de processo, apresentada por Lago et al. (2009) e Ramesh e Jarke (2001). Há muitas definições de processos, entre elas, uma que define processos como um conjunto de atividades para se fazer algo, ou atingir um objetivo, que recebe um conjunto de insumos como entrada, executa uma série de atividades de forma organizada e entrega um produto ou serviço como sua saída.

Levando isso em consideração, em nosso modelo as dimensões "*Transversal*", "*Requirements* and *Enhancements*" e "*Quality*" fornecem as entradas para a dimensão "*Process*". Nela a construção do produto é dividida num conjunto de *Sprints*, que são as iterações para construção do software. Em cada "*Sprint*", o trabalho é dividido em um conjunto de "*Tasks*", tarefas para entregar o produto do projeto (Entidade "Project"). A saída deste processo é a entrega a cada "*Sprint*" de

código funcionando. Esta saída é representada no modelo pela dimensão "*Product*".

Portanto, a entidade "*Project*" armazena informações de projeto, por exemplo, o nome do projeto, sua data de criação, seu status (por exemplo: "Em andamento"; "Suspenso"; "Cancelado" e "Concluído"). Enquanto o conceito "*Sprint*" representa as iterações para a construção do software e a entidade "*Task*" representa as tarefas cuja execução constroem o código. As entidades "*Sprint*" e "*Task*" foram incluídas em referência ao trabalho de Jacobsson (2009) e Gotel (1995).

4.2.2.3 Dimensão Requirements and Enhancements

O desenvolvimento ou a manutenção de software requer que se estabeleça um escopo do trabalho a ser feito. Este escopo define as capacidades e comportamentos do produto a ser produzido. Em nosso modelo, esta lista se encontra na dimensão "*Requirements* and *Enhancements*" nos conceitos "*Product Backlog*" e *"Sprint Backlog"*. Conforme explicado anteriormente, estas nomenclaturas foram extraídas do Scrum, e usadas por serem de uso mais comum, conforme *State of Agile™ Survey* (2015).

Neste modelo "*Product Backlog*" representa um conceito que armazena uma lista das funcionalidades, comportamentos, capacidades e requisitos do novo produto. Um "*Product Backlog*" é um super-conjunto formado por conjuntos menores de requisitos conhecidos por "*Sprint Backlog*", os quais, por sua vez contém uma lista de itens chamados "*Backlog Item*". Métodos Ágeis não definem o que um "*Backlog Item*" representa, por esta razão, é bastante comum na indústria, apareceram elementos como: *Bugs_Fixes*; *User Stories*; *Requirement*; *Features e*; *Use Cases*.

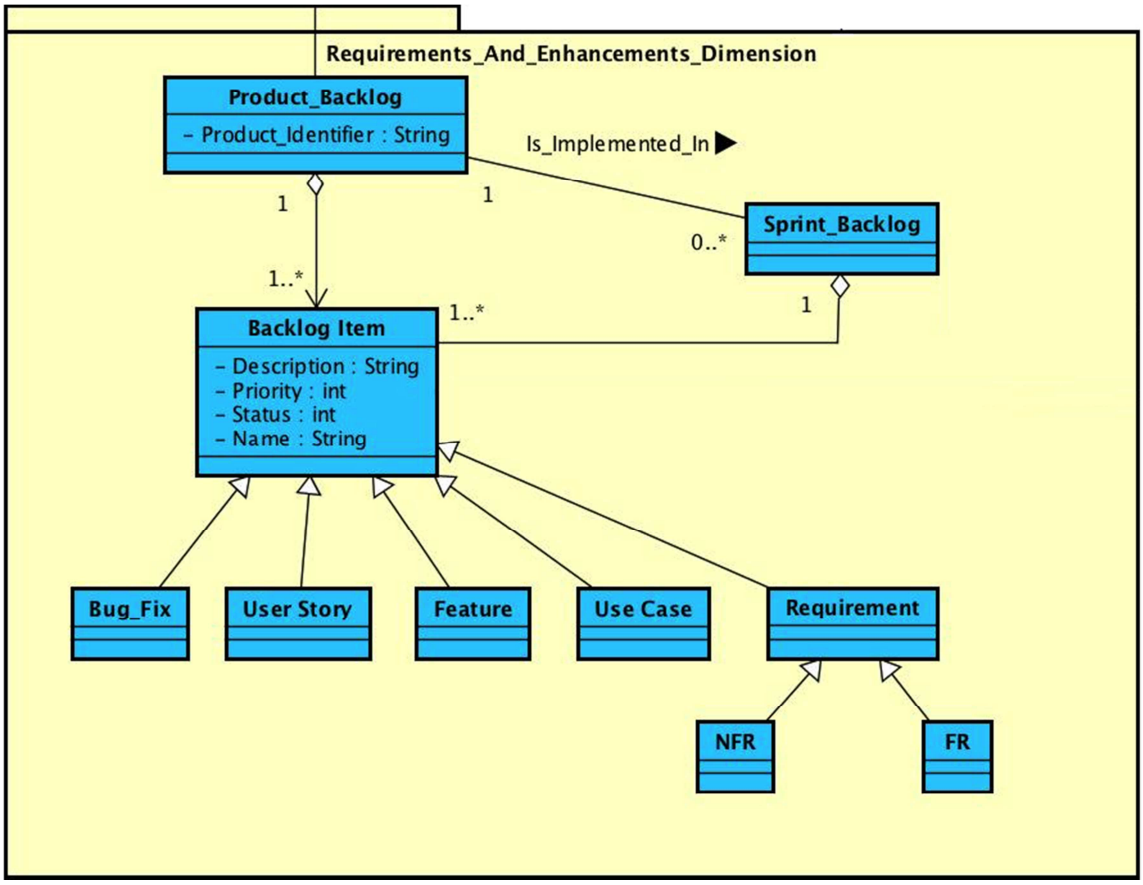


Figura 21: Dimensão de Melhorias e Requisitos.

Deve ser destacado que deixamos explícito no modelo de referência os requisitos não funcionais (NFR), cuja importância foi ressaltada por Arbain et al. (2014), que argumenta que requisitos não funcionais muitas vezes são parcamente trabalhados em Métodos Ágeis, mas que a necessidade de considerá-los explicitamente é cada vez mais reconhecida entre seus praticantes. Do ponto de vista de gestão, determinar se e quais requisitos não funcionais foram considerados num projeto é essencial para a qualidade do produto, ajudando para que ações sejam tomadas a fim de garantir o alcance das metas de custo, prazo e qualidade.

As entidades "*Backlog Item*" podem se relacionar com outras da mesma natureza. Os relacionamentos entre "*Backlog Item*" podem ser de muitos tipos, por exemplo: "*Details*", "*Refinement*" e "*Depends*". Ou seja, uma História de Usuário pode ser o detalhamento de outra História, ou um refinamento, tornando mais clara ou compreensível outra História de Usuário. Pode haver relações de dependências, onde mudanças numa Histórias de Usuário podem levar a alterações nas suas Histórias dependentes.

4.2.2.4 Dimensão Quality

Na dimensão "*Quality*" foram incluídos os conceitos "*Test Case*" que aglutina "testes unitários", de "integração", "testes de sistema", "de carga", "de regressão" e "testes de aceitação". Diversas informações de gestão essenciais são capturadas ao se rastrear testes, tais como: quais "*User Story*" foram testadas; quem executou o teste; entre outras. Esta entidade pode ser encontrada em Mäder e Cleland-Huang (2013) e Arbain et al. (2014).

4.2.2.5 Dimensão Product

Um dos princípios fundamentais dos Métodos Ágeis é entregar código funcionando o mais cedo possível, isto em geral acontece no final de cada *Sprint*. Neste momento é realizada uma entrega, conhecida em termos técnicos como "*Release*" ou "*Deploy*". Cada "*Deploy*" é composto de conjuntos de códigos fontes, representados pela entidade "*Code*". Em última instância, "*Deploy*" e "*Code*" representam produtos sendo entregues, por esta razão foram mapeados na dimensão "*Product*".

4.2.2.6 Links de Rastreabilidade

Em um modelo de rastreabilidade ligações(*links*) de rastreamento são elementos de primeira classe, eles materializam as ligações entre os artefatos de processo e produto. Sua semântica dá significado às ligações, e são, portanto, elementos essenciais num modelo de referência de rastreabilidade. Este conceito é encontrado em diversas pesquisas acadêmicas, por exemplo, em Delater et al. (2011) assim como em Taromirad (2014) e é representado no Modelo de Rastreabilidade para Métodos Ágeis (MRMA) conforme exibido na Figura 22.

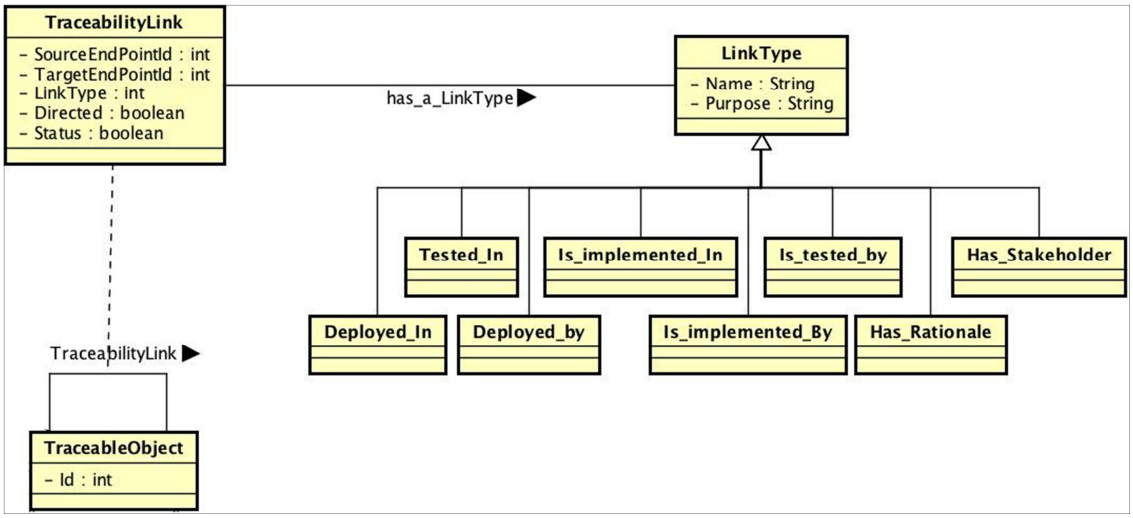


Figura 22: *Link* de Rastreamento

Taromirad (2014) captura outro elemento essencial de um *link* de rastreamento, a saber, tipo de *link*. A literatura apresenta farta variedade de tipos, por exemplo, no trabalho de Lago et al. (2009) encontramos "*drive*", "*modify*", "*define*", "*depend on*", "*implemented in*", "*implemented by*" e "*influence*". Enquanto Marcus et al. (2005) apresenta: "*leads to*"; "*implies*" e "*create*".

O modelo contempla tipos no campo "*Type*", mas como as possibilidades de tipos são muitas, e a semântica dos *links* de rastreabilidade para os Métodos Ágeis são diferentes daquelas usadas para os métodos Tradicionais, Espinoza e Garbajosa (2011) propuseram o uso de tipos de *links* definidos pelo usuário, o modelo incluiu a tabela auxiliar "*LinkType*" com esta finalidade.

Alguns tipos de rastreabilidade são direcionais por natureza, por exemplo, o tipo "*details*", para denotar esta situação foi incluído no modelo o atributo "*Directed*".

Links de Rastreabilidade podem ser capturados pelo uso de diversas técnicas, algumas delas deixam a critério do usuário definir se são *links* válidos ou não, portanto *links* podem ser válidos ou não, por esta razão incluímos o atributo "*Status*".

Neste capítulo foi mostrado o processo de elaboração do MRMA, na seção 5 aplicamos o MRMA num Estudo de Caso em que validamos os conceitos contidos neste modelo e verificamos se eles de fato são encontrados nos processos ágeis e nas ferramentas que dão apoio a gestão de projetos em Métodos Ágeis.

As informações importadas de projetos reais em Métodos Ágeis para o MRMA serão analisados, com o propósito de verificar se podem ser usadas para a gestão de projetos em Métodos Ágeis.

5. Aplicando o modelo de Rastreabilidade (Estudo de caso)

Na seção 4 foi explicado o processo de elaboração do modelo de rastreabilidade cuja construção teve o objetivo de fornecer as informações necessárias para gestão em métodos ágeis. Na Figura 19 mostramos o produto deste trabalho.

Para validar a aplicabilidade do modelo foram elaborados dois cenários, no primeiro é demonstrado que as entidades contidas no modelo de referência de rastreabilidade representam conceitos legítimos para os processos ágeis, evidenciando que eles estão presentes numa importante ferramenta de gestão de projetos ágeis, a ferramenta JIRA.

Enquanto que no segundo aplicamos o modelo a projetos reais, todos já encerrados, para demonstrar que ele permite que sejam extraídas informações relevantes para a gestão e melhoria do desenvolvimento de software usando Métodos Ágeis. Foram usados dados de seis projetos reais, dos quais as manutenções de cinco deles foram retiradas de suas equipes originais que foram desmobilizadas e deslocados para novas equipes localizadas geograficamente distantes das equipes originais.

5.1 Cenário de Uso 1 – Validação das entidades do modelo

Para validar o modelo, foram seguidos os seguintes passos: (i) obter o modelo de dados da ferramenta JIRA. Esta ferramenta foi escolhida, por ser usada para a gestão de projetos de desenvolvimento de software na empresa em que este estudo foi conduzido; (ii) identificar as entidades de dados contendo as informações de rastreabilidade que satisfazem o MRMA; (iii) criar um aplicativo para acessar os conceitos de interesse da rastreabilidade, conforme ditados pelo modelo; (iv) acessar dados reais de projetos e exibi-los em consultas e visualizações. Cada passo será detalhado a seguir.

No passo (i) – Foi obtido do site do fabricante o modelo de dados da ferramenta JIRA, vide Figura 23 que exibe as tabelas mais comumente usadas.

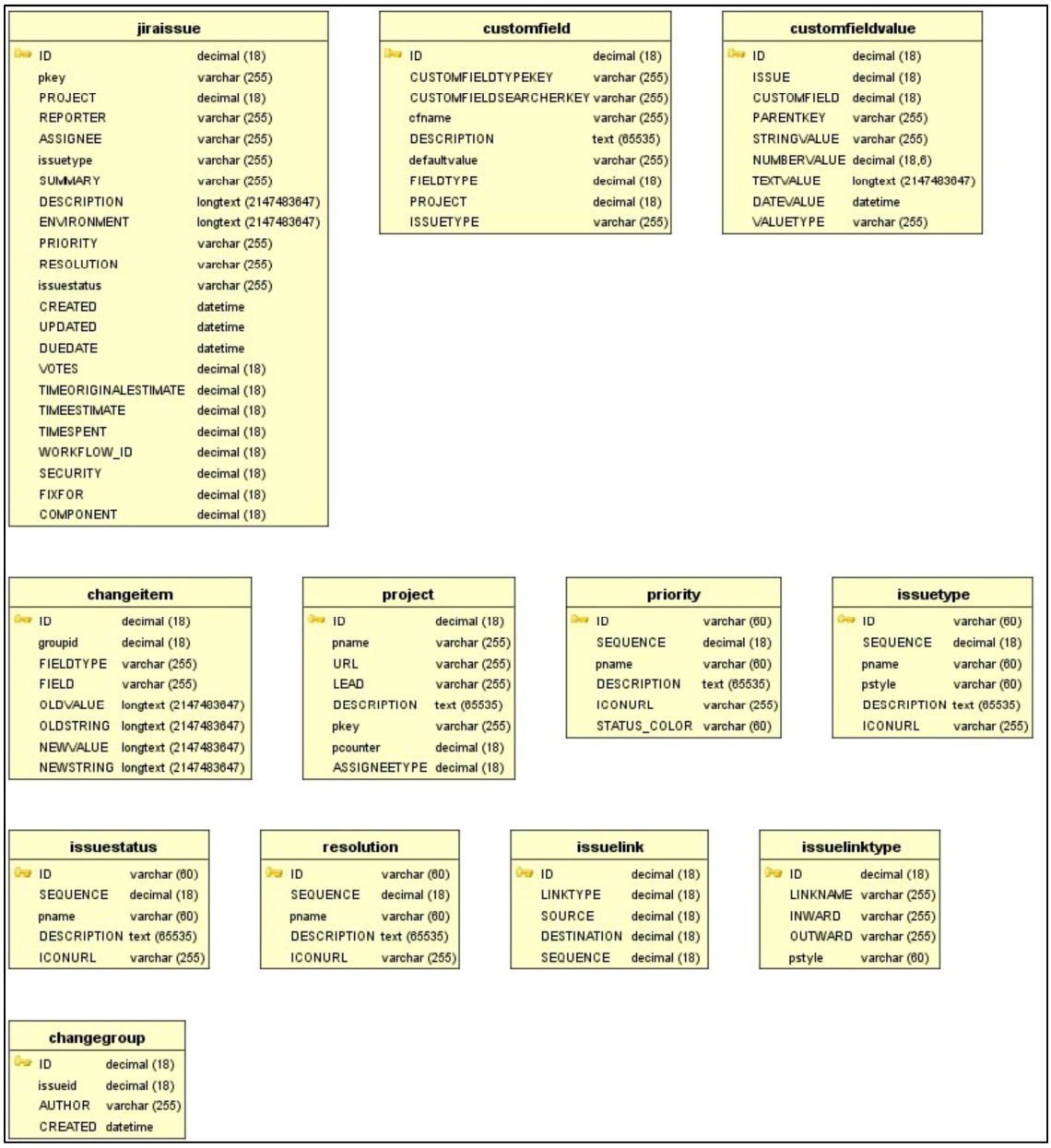
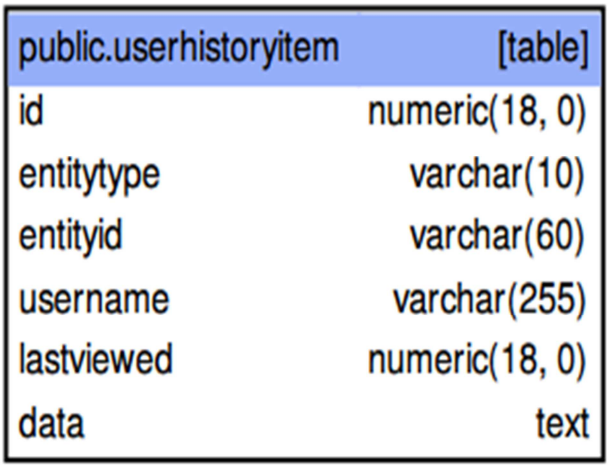


Figura 23: Modelo de dados parcial do JIRA

Passo (ii) - identificar as entidades do modelo de dados do JIRA contendo as informações de rastreabilidade que satisfazem o MRMA. O esquema completo do banco de dados da ferramenta JIRA é bastante complexo com cerca de duzentas entidades, neste sentido o uso do modelo de rastreabilidade trouxe benefícios, pois serviu de referência para facilitar a identificação dos conceitos do banco de dados de interesse para a rastreabilidade.

Por exemplo, na Dimensão de Melhorias e Requisitos do MRMA, identificamos Histórias de Usuário como um dos elementos que podem compor um

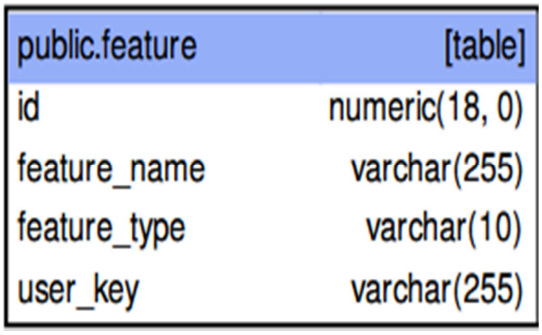
backlog. Com este conceito em mente, analisamos o esquema da base de dados a procura das entidades que representassem este conceito, a Figura 24 mostra o elemento no esquema de banco de dados do Jira que retêm dados a respeito da entidade “Histórias de Usuário”.



public.userhistoryitem	[table]
id	numeric(18, 0)
entitytype	varchar(10)
entityid	varchar(60)
username	varchar(255)
lastviewed	numeric(18, 0)
data	text

Figura 24: Entidade no Esquema de Banco de Dados do Jira que retém o conceito “Histórias de usuário”

Da mesma forma identificamos a entidade “*features*”, que mantém os dados das “*features*”, conforme Figura 25. E a entidade “*projectrole*”, equivalente à entidade “*Role*” em nosso modelo (Figura 26).



public.feature	[table]
id	numeric(18, 0)
feature_name	varchar(255)
feature_type	varchar(10)
user_key	varchar(255)

Figura 25: Entidade no Esquema de Banco de Dados do Jira que retém o conceito “*Feature*”

Figura 26: Conceito do Jira que retém o conceito "*Role*"

Seguindo o modelo mapeamos outras entidades relevantes: "*issuelink*", "*issuelinktype*", "*issuestatus*", "*issuetype*", "*jiraissue*" e "*priority*". No Jira, "*issuelink*" e "*issuelinktype*" equivalem as entidades "*TraceabilityLink*" e "*LinkType*" do MRMA, vide Figura 22, porém mais limitados, uma vez que não deixam claro as ligações que fazem associações com *Stakeholders* e Papéis, enquanto no Modelo de rastreabilidade as "*TraceabilityLink*" e "*LinkType*" podem ser usadas genericamente para qualquer tipo de ligação de rastreabilidade. Por outro lado, conceitos como "*issuestatus*", "*issuetype*", "*jiraissue*" e "*priority*", podem representar outros artefatos, por exemplo, "*Bugs*".

Passo (iii): foi criado um aplicativo para extrair os dados da ferramenta JIRA seguindo o modelo de rastreabilidade e exibi-los em consultas e visualizações. *V*ide Figura 28 e 29, que exibe consultas e visualizações de rastreabilidade nesta ferramenta.

Esquematicamente falando, usamos o MRMA como referência, então fomos na ferramenta JIRA, procuramos pelas entidades de interesse. Uma vez localizadas, construímos rotinas para extrair os dados e exibi-las em visualizações e consultas. Nesta dissertação os dados foram extraídos somente da ferramenta JIRA, mas este trabalho pode ser estendido para outras fontes de dados, conforme mostrado na Figura 27.

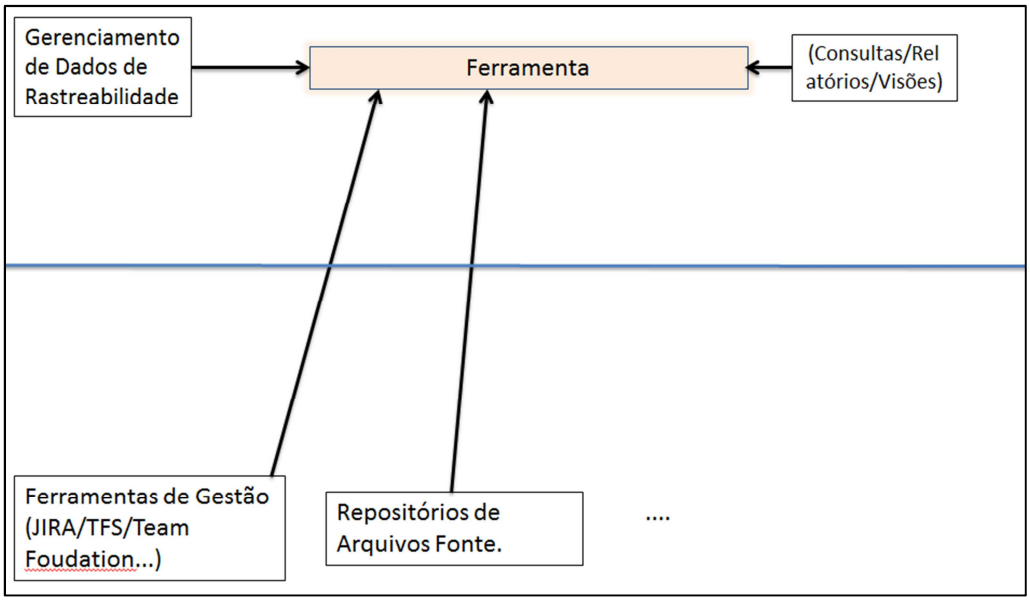


Figura 27: Uso do modelo de rastreabilidade na identificação dos conceitos pertinentes a gestão de projetos em Métodos Ágeis na Ferramenta JIRA

Na Figura 28 vemos uma consulta que exibe as tarefas por responsável, ela usa as relações entre *stakeholders* (responsável) e *tasks*(tarefas), criadas a partir dos dados reais de projetos de uma grande empresa de energia, extraídos das ferramentas JIRA e carregadas numa base de dados construída com base no MRMA, como pode ser visto na Figura 31.

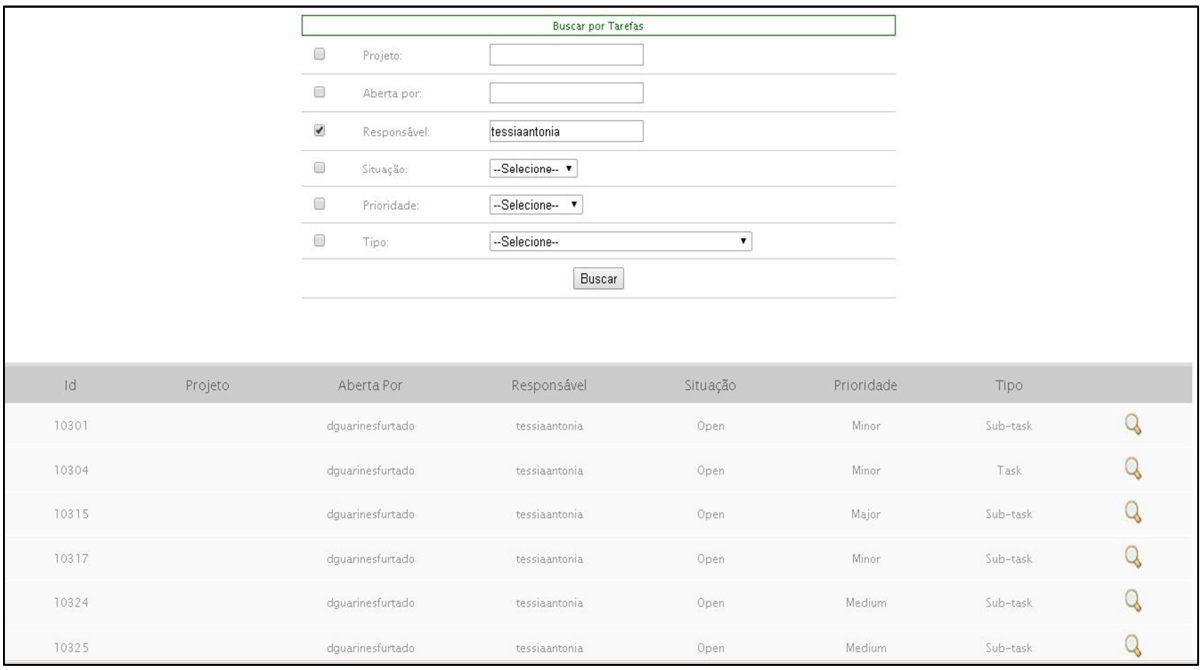


Figura 28: Visão das tarefas por responsável

	Teste criação de questão	REDOCORRCOR-232 Atualizar manual/help da função	REDOCORRCOR-219 Realizar testes funcionais integrados na estória	REDOCORRCOR-213 Ajuste na alteração de turno usado em mão de obra	REDOCORRCOR-200 Ajustar tela de inclusão de turno	REDOCORRCOR-190 Ajustar inclusão de REDOCORR	REDOCORRCOR-198 Atualizar documentação	REDOCORRCOR-164 Atualizar documentação referente à estória
Ajustar cadastro de turno para atender a RN71				✓				
Ajuste no cadastro de turno – ajustar mensagens e tornar mensagem impeditiva em não impeditiva		✓						
Notificar a contratada após a conclusão do REDOCORR pela PXXXXXXOLEO							✓	
Incluir campo desconto no cadastro de turno					✓			
Ajuste no cadastro de turno – inclusão de validação do desconto e da hora final			✓					
INC000021145664 – Corrigir o QUESCOMOBS de resposta obrigatoria						✓		
Exemplo Tarefa Bloqueante	✓							
Tela de Consulta de RELFRENTE								✓

Figura 29: Matriz de rastreabilidade associando Histórias de Usuário a *Tasks* (Tarefas).

A Figura 29 apresenta uma matriz de rastreabilidade, esta é uma das formas mais comuns e práticas de exibir relações de rastreabilidade, por esta razão foi incluída na aplicação como uma das visões disponíveis. Neste estudo, esta visão permite a criação e/ou a visualização das ligações entre as Histórias de Usuário e as tarefas.

5.1.1 Discussão

O objetivo do Cenário 1 era validar se as entidades mapeadas no MRMA eram encontradas em projetos reais usando Métodos Ágeis. A abordagem escolhida para esta validação foi analisar as ferramentas usadas para apoio ao desenvolvimento e gestão de projetos ágeis reais de uma grande empresa de energia.

Quase todas as entidades representadas no modelo puderam ser encontradas ao analisarmos a ferramenta JIRA, usada nesta companhia. Alguns dos conceitos que não foram encontrados, não estavam nela porque ela não dava suporte, por exemplo, não havia nenhuma entidade representando o código fonte, que costuma ser mapeada a partir de repositórios, tais como o *ClearCase* ou *Subversion*.

A impossibilidade de localizar os modelos de algumas ferramentas, por estarem inacessíveis, muitas vezes por razões comerciais (TFS, ClearCase, IBM Team Concert...), prejudicou o trabalho pois era a intenção deste cenário

demonstrar que o modelo é válido porque seus conceitos são reconhecidos e existem nas ferramentas usadas em Métodos Ágeis.

Apesar de não conseguir demonstrar de forma objetiva a validade de todos os conceitos do MRMA, pois não foi possível o acesso nas ferramentas às entidades da dimensão "*Product Dimension*", *Code* e *Deploy*. A maior parte do MRMA foi validada, o que possibilitou avançarmos para o cenário 2.

5.2 Cenário de Uso 2 – Demonstrar a utilidade do modelo para extração e análise de dados de gestão de desenvolvimento em Métodos Ágeis

O objetivo do Cenário 2 é utilizar os dados de projetos reais em Métodos Ágeis, carregados na base de dados do MRMA, para responder as perguntas de gestão de projetos em processos ágeis. A fim de demonstrar a utilidade do modelo, foram seguidos os seguintes passos:

1) **Instanciar e popular o modelo:** (i) criar um modelo lógico de dados segundo o que especificava o MRMA e instanciar este modelo numa base de dados; (ii) selecionar as ferramentas, aplicadas na companhia para a gestão de projetos ágeis, cujos dados serão populados no MRMA; (iii) utilizar as entidades de dados contendo as informações de rastreabilidade identificadas no Cenário de Uso 1; (iv) carregar os dados na base de dados de rastreabilidade; (v) criar os *links* de rastreabilidade; (vi) Fazer análise crítica dos *links*, avaliando, por exemplo, se o *link* proposto é coerente com os tipos de artefatos envolvidos; (vii) elaborar consultas e visualizações para extrair informações relevantes destes dados e analisar do ponto de vista de gestão.

2) **Para testar a utilidade do modelo** para responder as perguntas de gestão de desenvolvimento em Métodos Ágeis, foram seguidos os seguintes passos: (i) definir as perguntas de gestão a serem respondidas; (ii) elaborar consultas SQL com base nestas perguntas; (iii) Exibir as informações por meio de gráficos, visões, tabelas, etc.; (iv) discutir os resultados com foco em gestão do desenvolvimento de software.

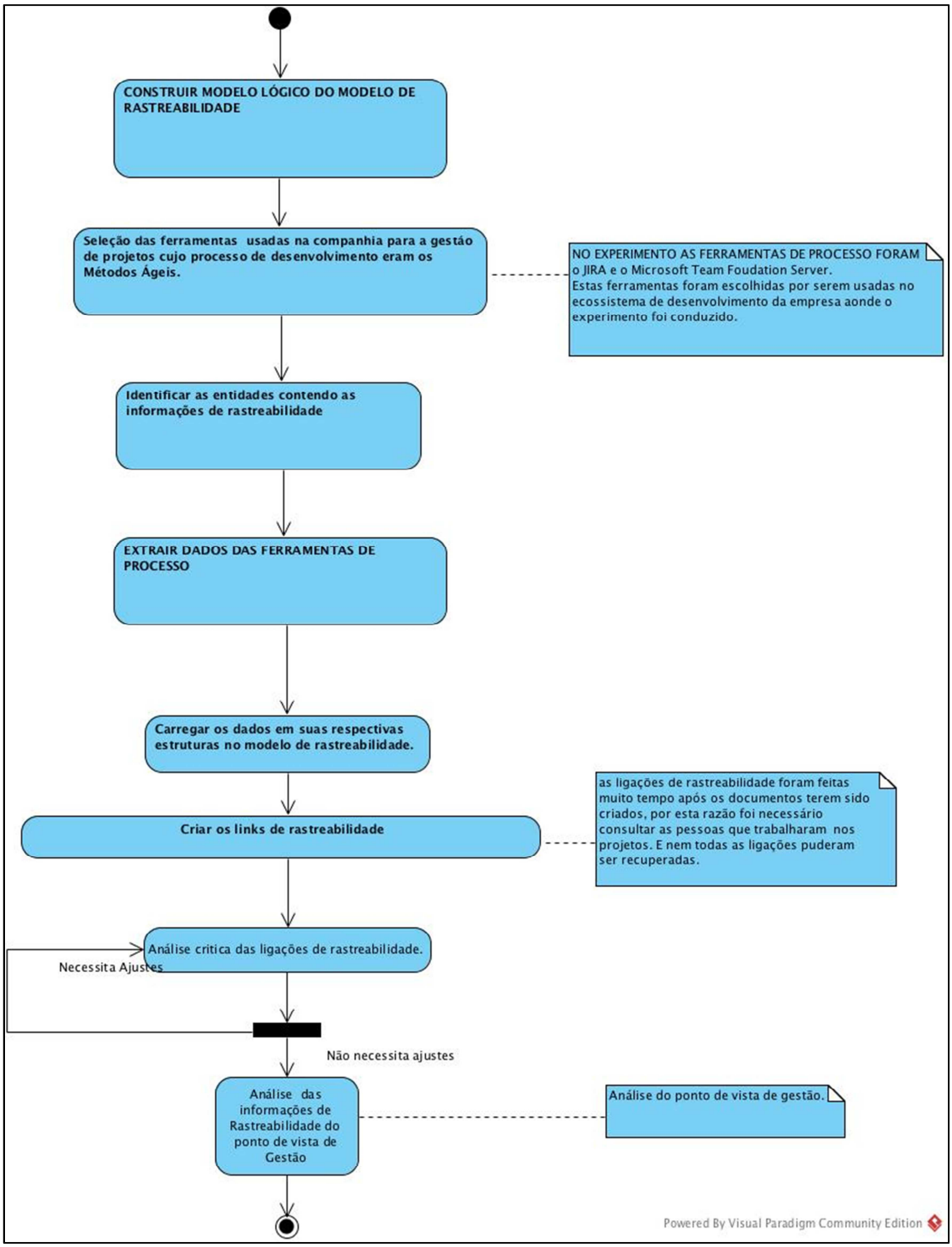


Figura 30: Passo a Passo do Estudo

5.2.1. Instanciar e Popular o Modelo

A seguir descreveremos como o modelo foi instanciado e populado.

5.2.1.1 Instanciando e Populando o Modelo de dados do MRMA

Tomando como base o modelo de rastreabilidade para Métodos Ágeis das Figuras 20 e 22, elaboramos o esquema de banco de dados exibido na Figura 31, que em seguida foi instanciado e carregado com os dados de seis projetos reais de uma grande companhia do ramo da energia, que fez uso de Métodos Ágeis no processo de desenvolvimento de software destes projetos.

Cada elemento neste esquema de banco de dados corresponde a uma entidade no Modelo de Rastreabilidade para Métodos Ágeis, num mapeamento um para um. Deste modo, por exemplo, a entidade *Backlog* elemento do modelo de rastreabilidade, tornou-se o elemento *Backlog_Item* no banco de dados, e assim por diante.

1) **Testar a utilidade do modelo** para responder as perguntas de gestão de desenvolvimento em Métodos Ágeis, para isso foram seguidos os seguintes passos: (i) definir as perguntas de gestão a serem respondidas; (ii) elaborar consultas SQL com base nestas perguntas; (iii) Exibir as informações por meio de gráficos, visões, tabelas, etc.; (iv) discutir os resultados com foco em gestão do desenvolvimento de software.

Para manter o Modelo de Rastreabilidade de Requisitos para Métodos Ágeis enxuto, por razões de simplicidade, certos elementos foram omitidos, mas precisaram ser criados no momento de intanciar o modelo de dados do MRMA, por exemplo, os atributos *Project_Status*, *Backlog_Status, Backlog_Priority, Backlog_Type, Task_Type, Test_Cases_Type* e *Sprint_Status.*

5.2.1.2– Seleção das ferramentas

As principais ferramentas de gestão usadas no desenvolvimento de software em Métodos Ágeis da companhia são: JIRA e Microsoft Team Foundation Server (TFS). Elas foram selecionadas para uso no estudo do Cenário de Uso 2.

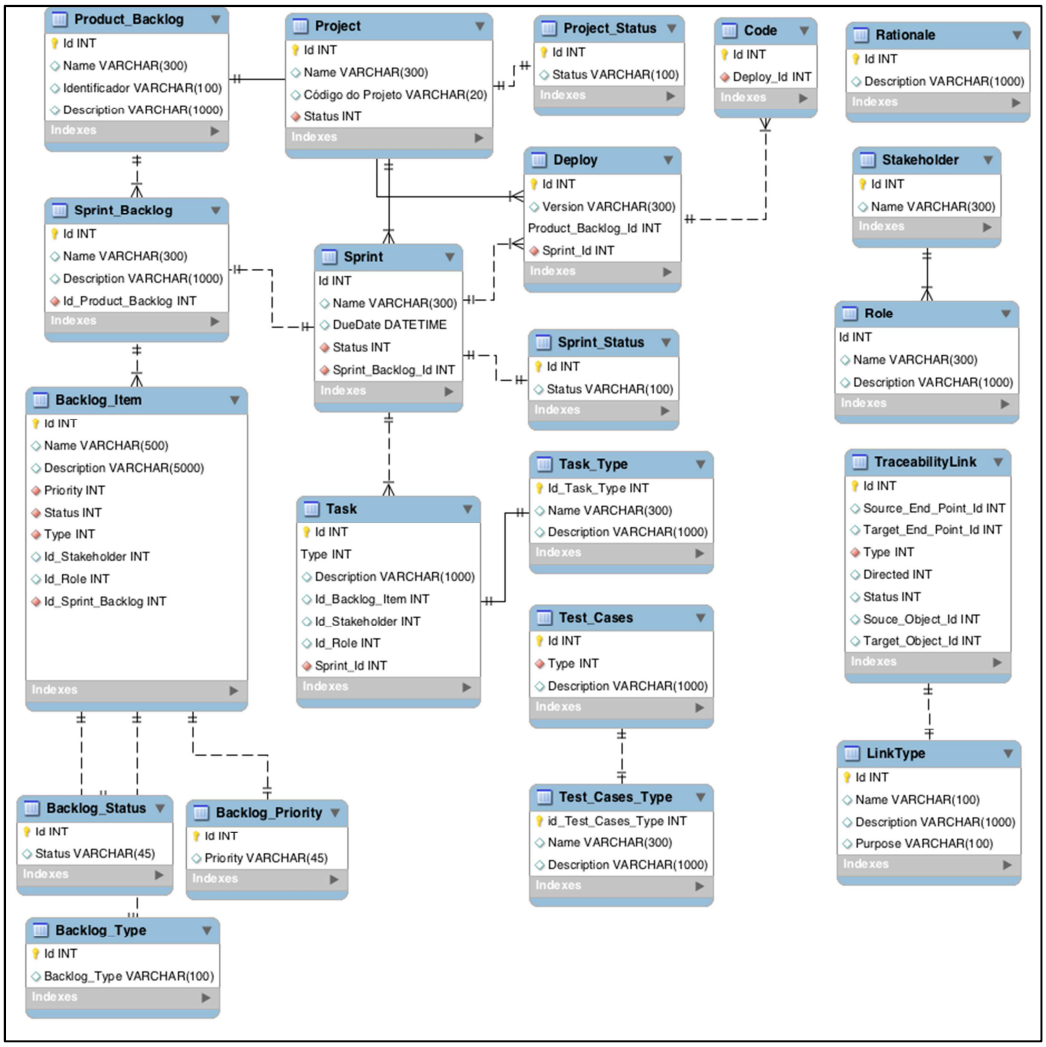


Figura 31: Modelo de Dados do MRMA

5.2.1.3 – Extrair dados reais de projetos

No processo de seleção das informações de rastreabilidade, foi necessário realizar uma filtragem, dado a grande quantidade de projetos usando Métodos Ágeis realizados na empresa, por esta razão foi preciso estabelecer alguns critérios de seleção dos projetos: (1) usarem Métodos Ágeis nos processos de desenvolvimento ou manutenção de software; (2) representarem os principais processos de negócio da empresa, ou seja, serem representativos dos principais tipos de desenvolvimento de software feitos na companhia e; (3) serem projetos concluídos.

Para executar esta tarefa foi necessário o apoio de um especialista que pudesse selecionar os projetos em conformidade com o critério de número dois, tendo sido escolhidos seis projetos de um total de quatorze exportados de uma base de dados contendo mais de cinquenta projetos. Desde de pequenos projetos com

duração de dois a três meses, a grandes, de dois a três anos. Estes projetos são representativos de seis processos de negócio diferentes e importantes da companhia, todos cujo desenvolvimento fez uso de Métodos Ágeis, cinco deles extraídos da ferramenta JIRA e um da ferramenta TFS (Micosoft Team Foudation Studio), todos concluídos antes da carga dos dados.

Os dados da primeira ferramenta foram exportados para planilhas Excell, enquanto o da segunda foram extraídos de forma manual, direto da interface da aplicação, pois a política de segurança da empresa não permitiu acesso direto aos bancos de dados das ferramentas. Um aspecto relevante, que dificultou os trabalhos posteriores, é que a exportação na ferramenta JIRA organizou os dados num formato planar, todos os dados relacionados a cada *issue* (podem ser histórias de usuário, testes, *feature, etc.*) eram apresentados numa única linha na tabela, esta característica aumentou significantemente a dificuldade para execução do passo seguinte.

5.2.1.4 – Carregar os dados no banco de dados do modelo de rastreabilidade

Uma vez que os projetos foram selecionados, devido à característica planar dos dados, foi necessário solicitar o apoio de outro especialista para categorizar, classificar e separar em seus respectivos grupos de registros lógicos presentes na planilha em correspondência ao que especificava o modelo de rastreabilidade para Métodos Ágeis.

5.2.1.5 Criar os links de rastreabilidade

A criação dos *links* de rastreabilidade mostrou-se um desafio a parte, uma vez que se tratavam de projetos que haviam sido encerrados, e este trabalho estava sendo feito a posteriori. Neste sentido o MRMA ajudou, pois orientava quanto aos tipos de entidades que deveriam ser procuradas. Muitas das ligações, por exemplo, entre as histórias de usuário e as *sprints*, foram fáceis e diretas de serem identificadas, pois o próprio JIRA já mantinha estas ligações. Outras, porém, exigiram que fossem feitas consultas a especialistas, para esclarecer como determinadas *issues* deveriam ser interpretadas, um destes casos, foi de uma *issue* que não deixava claro se estava sendo relatado um erro ou uma nova funcionalidade.

5.2.1.6 Análise crítica dos links de rastreabilidade

Uma vez que os *links* foram criados, eles passaram por um processo de revisão, para verificar a conformidade das ligações de rastreabilidade, avaliar se as ligações corretas foram criadas, se ficou faltando algum *link*, ou mesmo se ligações necessitavam ser eliminadas, por serem *links* falsos.

Este tipo de análise era necessário, pois o trabalho de criação dos *links* foi feito a posteriori, e em parte baseou-se na memória das pessoas, pois nem todas as ligações podiam ser deduzidas diretamente dos dados coletados a partir da Ferramenta JIRA, deste modo, para aumentar a confiabilidade das informações, uma revisão foi conduzida, procurando por ligações incoerentes, por exemplo, analisar *links* de detalhamento "*is_detailed_by*" para verificar se uma entidade filho era de fato um detalhamento de alguma parte da entidade pai.

Terminados todos os passos anteriores, foi concluído o carregamento do MRMA com os dados de rastreabilidade. Estando pronto para verificarmos se com estes ele permite responder as perguntas de gestão. Veremos isso na seção 5.2.2.

5.2.2 Testar a utilidade do modelo para a gestão de projetos usando Métodos Ágeis

O Cenário de Uso 2 visa testar a utilidade do MRMA para a gestão de projetos de desenvolvimento em Métodos Ágeis.

A tabela 6 exibe uma descrição resumida destes projetos, em todos os casos as tecnologias de desenvolvimento, Java ou Dotnet, e as áreas de negócio são bem dominadas pelas equipes. A seleção destes projetos baseou-se no critério de que

deveriam ser projetos transferidos entre equipes, sendo que, com exceção do "Projeto 1", nos demais as equipes originais foram completamente desmobilizadas.

Ferramenta de Origem dos dados	Tamanho Sistema PPF	Tempo de duração do projeto	Tamanho da equipe	Tecnologia	Domínio da aplicação
Jira	731	6 meses	3 pessoas	Java	Georeferenciamento
Jira	503	12 meses	4 pessoas	Java	Saúde e Segurança do trabalho
Jira	626	18 meses	6 pessoas	Java	Engenharia
Jira	304	9 meses	4 pessoas	Java	Abastecimento e Refino
Jira	1578	10 meses	4 pessoas	Java	Controles empresariais
TFS	1438	7 meses	3 pessoas	Dot Net	Contabilidade e Finanças

Tabela 6:Descrição resumida dos projetos

Como parte do trabalho para testar o MRMA, era preciso definir um conjunto de perguntas cujas respostas são de interesse da gestão de desenvolvimento de software em Métodos Ágeis. A subseção 5.2.2.1 discorrerá a respeito destas perguntas.

5.2.2.1 Perguntas que um modelo de rastreabilidade para Métodos Ágeis ajudam a responder e que são úteis para a gestão do desenvolvimento de software

Diversas destas perguntas vieram a partir da análise do trabalho de Gupta at al. (2015), que identificaram 31 problemas/desafios a serem resolvidos pelo gerenciamento de processos de software, dos quais alguns, que foram classificados como importantes no trabalho de Gupta at al. (2015), são listados na Tabela 7 a seguir.

Descrição do problema
Identificar gargalos e ineficiências que provocam atrasos para tomar medidas corretivas, e ter melhores estimativas no futuro.
Permitir a detecção precoce e prevenção de defeitos em vez de corrigi-los durante a fase posterior por entender padrões de defeitos fugitivos.
Descobrir desvios entre a forma como o processo é usado na prática pela equipe e como foi definido, sua causa, impacto na saída global do desenvolvimento e identificar as pessoas que exibem um maior grau de desvio.

Evitar colocar esforços em atividades menos significativas identificando etapas redundantes ou desnecessárias de processo.
Inspecionar questões reabertas para identificar a causa raiz e recomendar a verificação de problemas futuros com base em aprender com os problemas reabertos no passado.
Melhorar a efetividade do processo de revisão de código e padronização através de revisar as listas de verificação, melhorar e atualizar os analisadores de código usando como base os defeitos relatados durante os testes.
Necessidade de mecanismo de alocação eficiente de tarefas, considerando as habilidades individuais, interesses e conhecimentos, bem como a compatibilidade da equipe para melhor utilização dos recursos.
Mecanismo de evolução contínua de com base nas melhores práticas dos indivíduos que se utilizam do processo. Portanto, melhorar o processo, incentivando aprendizado das pessoas ao invés de depender de designers de processos.
Melhorar a eficácia dos processos de revisão e padronização de código redesenhando a lista de verificação e atualizando analisadores de código com base nos defeitos relatados durante os testes.
Facilitar uma melhor integração entre os diferentes silos através de reconstruir o processo, para reduzir o retrabalho acontecendo devido às diferenças de entendimento.
Analisar o ciclo de vida de revisão código para identificar os desenvolvedores que não estão revisando seu código adequadamente antes de submetê-lo à avaliação externa e as listas de desvios pré-definidas. Ele ajudará a tomar ações corretivas e reduzir defeitos durante os testes de fase.
Possuir uma técnica para rastrear a aderência com os requisitos e adaptar o processo automaticamente com as mudanças nos requisitos.

Tabela 7: Tipos de problemas de gestão que podem se respondidos pelo uso do MRMA Gupta at al. (2015).

Os resultados do uso do MRMA no Cenário 2 mostram que ele permite que se realizem análises capazes de avaliar a presença, nos processos de desenvolvimento de software usando Métodos Ágeis, de diversos dos problemas apontados na Tabela 7, que constituem desafios ao gerenciamento do desenvolvimento de software. Com base nestes problemas foram elaboradas algumas perguntas de gestão, listadas a seguir:

1. Eficiência de Processo (Identificar ineficiências e gargalos):
    1. O processo de desenvolvimento está sendo eficiente, ele produz os produtos no menor tempo possível, ou muito tempo é perdido devido ao retrabalho?

    2. Quais são os principais re-trabalhos por tipo: correção de bugs; *refactoring*; alteração devido a problemas com requisitos?...
    3. Quais são os gargalos(causas da ineficiência/retrabalho)? Falta de Testes? Falta inspeção de código? Equipe mal treinada? Tamanho de equipe insuficiente?…
2. Gestão dos erros (Permitir a detecção precoce e prevenção de defeitos em vez de corrigi-los durante a fase posterior por entender padrões de defeitos fugitivos.)
    1. Como os erros no desenvolvimento estão sendo gerenciados?
    2. Os erros estão sendo detectados e corrigidos dentro da própria iteração, de preferência logo após a sua construção?
    3. Que ações podem ser tomadas para resolvê-los o mais cedo possível, evitando que se propaguem para o final dos projetos, trabalhando no princípio de que quanto mais tarde se corrige um erro, maiores os custos e as dificuldades para eliminá-lo? (Testes automatizados? *Builds* Contínuos?)
3. Uso consistente do processo (maturidade - Descobrir desvios entre a forma como o processo é usado na prática pela equipe e como foi definido)
    1. Qual a maturidade do uso dos Métodos Ágeis no desenvolvimento de software na companhia, todas as equipes sabem instanciar e fazer uso dos processos de forma consistente e semelhante, por exemplo, usando a mesma ferramenta de gestão?
    2. Porque há equipes instanciando os processos de forma tão diferente das demais? Faltou treinamento? Faltou auditoria de processos?
4. Identificando código sujo (Melhorar a efetividade do processo de revisão de código e padronização através de revisar as listas de verificação, melhorar e atualizar os analisadores de código usando como base os defeitos relatados durante os testes.)
    1. Qual a qualidade do código produzido?
    2. A presença de código de baixa qualidade está dificultando a manutenção do software?
5. Gestão de projetos

1. O acompanhamento de projetos está sendo capaz de perceber os desvios ou os problemas só estão emergindo ao final do projeto?

6. Gerenciamento de requisitos(Aderência aos requisitos)
    1. Considerando uma História de Usuário (requisito), quais são os seus desdobramentos?
    2. Qual a quantidade de retrabalho causado por problemas nos requisitos?
    3. O(s) detalhamento(s) das histórias(requisitos) de usuário, de fato realizam a história(requisito) hierarquicamente anterior?

> Já do trabalho de Gotel e Filkenstein (1997) veio outro conjunto de perguntas, desta vez relacionadas ao gerenciamento de requisitos.

1. Quem esteve envolvido na produção deste requisito e como?
2. Quem foi responsável originalmente pela História de Usuário(requisito) e quem é responsável atualmente? Em que pontos de seu ciclo de vida esta responsabilidade mudou de mãos?”
3. Quem precisa ser envolvido, ou informado, de quaisquer mudanças propostas para esta História de Usuário (requisito)?
4. Quais são os desdobramentos, no que diz respeito a possíveis perdas do conhecimento das Histórias de Usuários (requisitos), se um indivíduo ou grupo de indivíduos específico deixa o projeto?
5. Quem usa este requisito e como?
6. Em quais requisitos certos *stakeholders* colaboraram e como?

Ao aplicar o MRMA sobre os dados de projetos reais de desenvolvimento, após a conclusão do trabalho de criação dos *links* de rastreabilidade, o modelo facilitou a elaboração de várias consultas que permitiam extrair informações que ajudavam a responder as seguintes perguntas de gestão apresentadas nas próximas subseções.

Eficiência do processo (Identificar ineficiências e gargalos)

O gráfico da Figura 32, extraído de consultas feitas nos dados de rastreabilidade carregados a partir dos seis projetos anteriormente citados, revela

uma alta taxa de correção de erros, a título de ilustração, no Projeto 2 a cada história implementada dois erros foram corrigidos.

Esta informação ajuda a responder as seguintes perguntas de gestão: “O processo de desenvolvimento está sendo eficiente, ele produz os produtos no menor tempo possível, ou muito tempo é perdido devido ao retrabalho?” e,; “Quais são os principais re-trabalhos por tipo(correção de *bug*; *refactoring*; alteração devido a problemas com requisitos...)?”

E se ela for conjugada com as informações vindas de outras tarefas do processo de desenvolvimento, outras perguntas podem ser respondidas, tais como: “Quais são os gargalos(causas da ineficiência/retrabalho)?”, por exemplo, ao correlacionar a informação da Figura 32, com a apresentada na Figura 33, percebe-se que a cobertura de testes para o Projeto 2, foi muito baixa, o que pode ser uma das causas da alta taxa de erros no projeto.

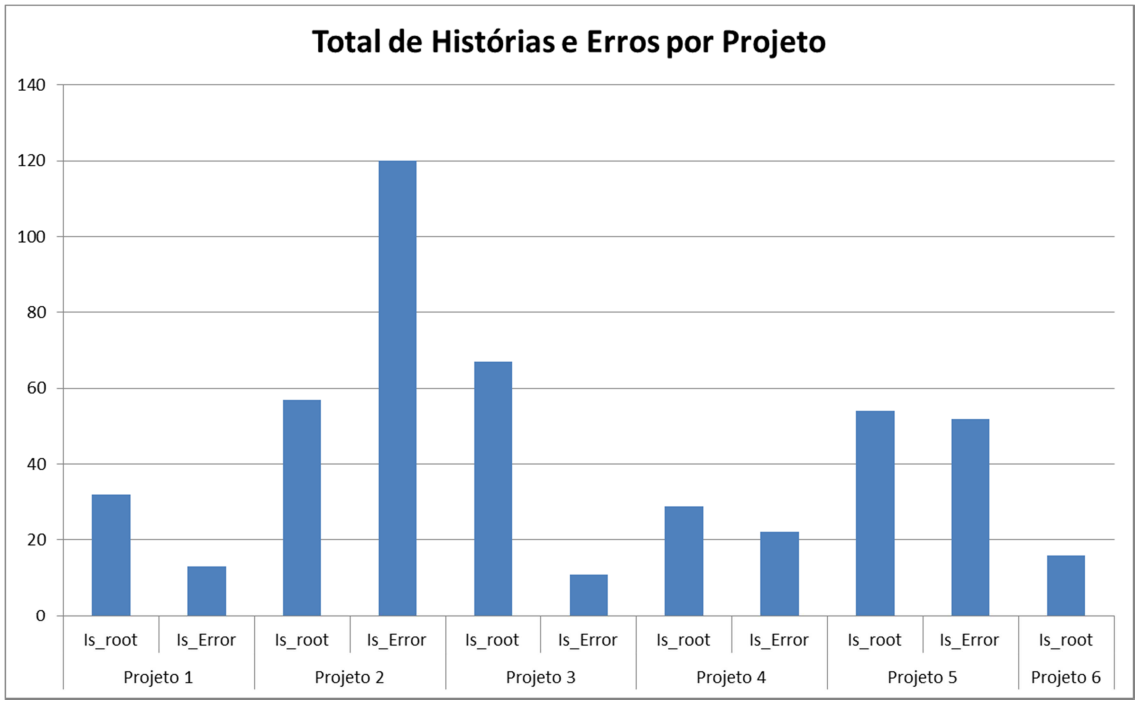


Figura 32: Taxa de erros nos projetos usando Métodos Ágeis de uma grande companhia no ramo da energia (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro)

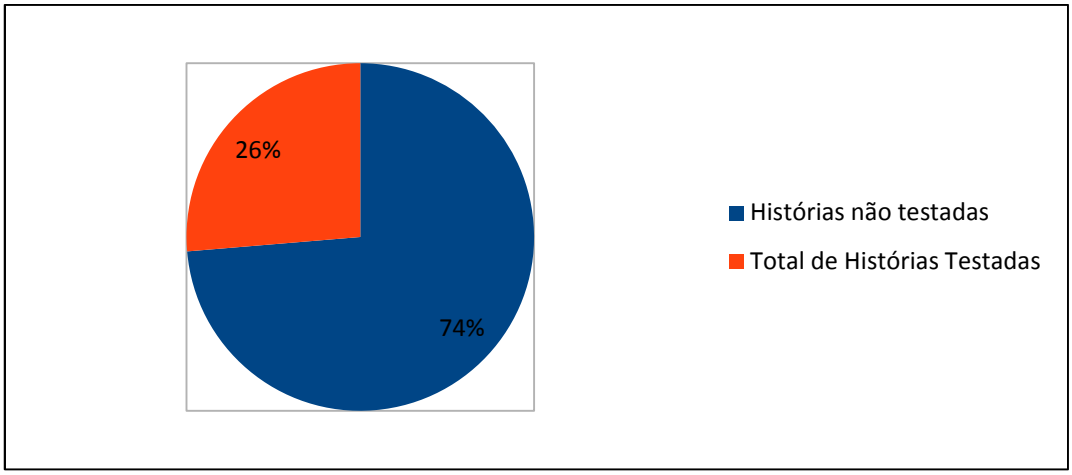


Figura 33: Percentual de cobertura de testes do projeto 2 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro)

Com estas informações em mãos, extraídas do MRMA, análises mais profundas podem ser feitas. Em nosso exemplo, da Figura 32, a elevada taxa de erros, referindo-se aqueles produzidos e corrigidos dentro do próprio projeto, pode afetar o prazo e o custo e muitas vezes a percepção de satisfação dos clientes com o resultado final. Altas taxas de erros sinalizam problemas graves no produto e/ou no processo de desenvolvimento, provocando retrabalho e diminuindo a eficiência do mesmo.

Os problemas no produto podem ser relacionados a uma arquitetura incompatível com os requisitos não funcionais especificados; manutenibilidade difícil com grande quantidade de código sujo; tecnologia obsoleta; modelagem de dados pobre; entre outras possibilidades.

Quanto aos processos podemos ter: erros na instanciação dos processos, em situações em que certas práticas comuns e importantes não foram devidamente implantadas, por exemplo, testes unitários; baixo desempenho de equipes de desenvolvimento; problemas na gestão de conhecimento, um caso típico pode ocorrer ao migrar de um modelo de trabalho *on shore* para *off shore*.

Uma alta taxa de erros não necessariamente provocará a perda dos prazos e custos de um projeto, pois um gerente de projeto experiente conhecerá o comportamento do processo de desenvolvimento em que trabalha e levará isso em conta ao estimar os custos e prazos, ou seja, terminando dentro das metas

planejadas, mas isto a um custo maior e num prazo mais extenso, em virtude da ineficiência do processo.

Indicadores de acompanhamento de projetos, como cumprimento do custo e prazo planejados, podem não ser suficientes para revelar problemas mais sutis como a ineficiência do processo de desenvolvimento, outros meios precisam ser usados.

Para tal é preciso fazer uso de mecanismos mais robustos que mostrem os desperdícios durante o processo de desenvolvimento, e foi exatamente o que o MRMA fez ao mostrar a grande quantidade de trabalho na correção de erros em alguns projetos e ajudar a responder a seguinte pergunta de interesse da gestão do desenvolvimento de software: “o processo de desenvolvimento está sendo eficiente, ele produz os produtos no menor tempo possível, ou muito tempo é perdido devido ao retrabalho?”.

Gestão dos erros (Permitir a detecção precoce e prevenção de defeitos em vez de corrigi-los durante a fase posterior por entender padrões de defeitos fugitivos.)

Embora seja comumente aceito que o desenvolvimento de software inerentemente gere erros, também é reconhecido que quanto mais cedo eles forem encontrados e resolvidos, menor será o impacto que estes erros poderão trazer para o cumprimento do prazo e do custo dos projetos.

Os dados de rastreabilidade coletados no MRMA demonstraram que para os projetos 2 e 5, houve um grande acumulo de erros nos *sprints* finais, na verdade, para o sistema 5, Figura 34, podemos ver que ele completou todas as suas histórias no *Sprint* 10 e precisou estender os trabalhos, acrescentando ao final três *sprints* exclusivos para a correção dos erros.

Um gerente de projetos experiente pode acrescentar folga ao final do projeto a fim de acomodar estas peculiaridades do processo de desenvolvimento mascarando os problemas. A análise dos dados de rastreabilidade feitas a partir do uso do MRMA revelou um problema que poderia não ser percebido por outros meios, como por exemplo, os controles de acompanhamento de projetos.

A análise dos dados reais de projetos armazenados na base de rastreabilidade do MRMA ajudou a responder a duas perguntas de interesse da

gestão: (1) “como os erros no desenvolvimento estão sendo gerenciados? Ações estão sendo tomadas para resolvê-los o mais cedo possível, evitando que se propaguem para o final dos projetos, trabalhando no princípio de que quanto mais tarde se corrige um erro, maiores os custos e as dificuldades para eliminá-lo?”; (2) “O acompanhamento de projetos está sendo capaz de perceber os desvios ou os problemas só estão emergindo ao final do projeto?”.

Para os projetos 2 e 5 a resposta é não, e com esta informação em mãos as pessoas responsáveis pelo gerenciamento do processo de desenvolvimento de software podem usar o MRMA para identificar os atores envolvidos nos projetos, a fim de coletar material para uma análise das causas mais consubstanciada, com emprego de ferramentas como os diagramas de causa e efeito e outras muito úteis e conhecidas empregadas nos trabalhos de melhoria contínua de processos.

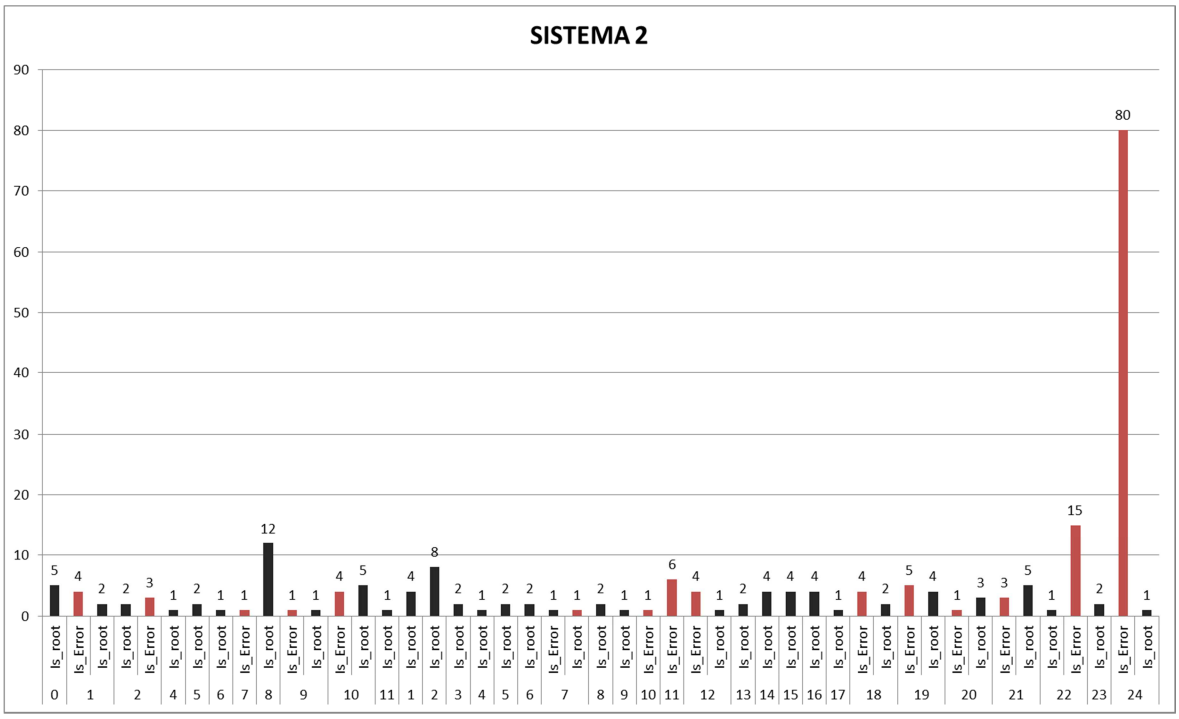


Figura 34: Taxa de erros no projeto do sistema 2 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro)

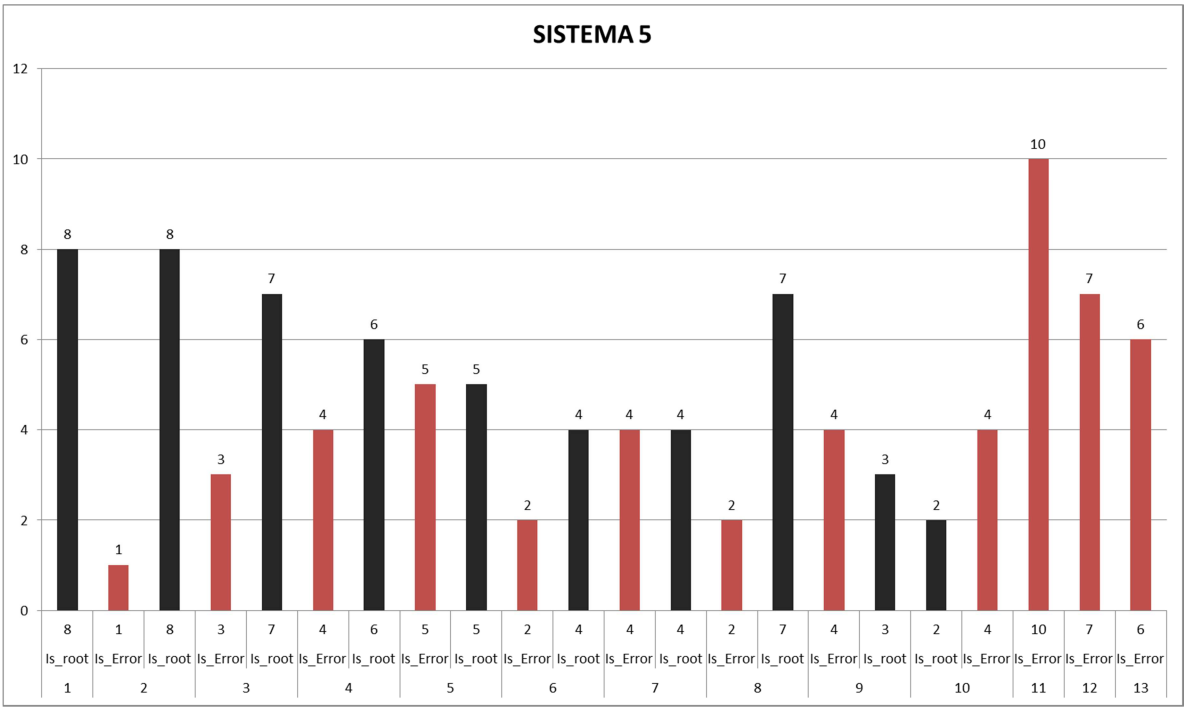


Figura 35: Taxa de erros no projeto do sistema 5 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro)

Uso consistente do processo (maturidade - Descobrir desvios entre a forma como o processo é usado na prática pela equipe e como foi definido)

Outro aspecto que pôde ser analisado a partir dos dados de rastreabilidade é a maturidade do uso do processo de software, ao falar maturidade a ideia é perguntar se existe consistência no uso do processo de desenvolvimento, com considerações como, por exemplo, se todos usam o processo como foi definido, ou se há grupos que executam o processo de forma diferenciada, se há atividades redundantes ou desnecessárias. Neste sentido, o gráfico da Figura 36, comparado com os das Figuras 34 e 35, mostra haver uma diferença relevante entre a forma que este projeto foi acompanhado/registrado nas ferramentas de gestão quando comparada com os demais.

Existe um *backlog* de erros e histórias concluídos, mas não foram registradas informações na ferramenta que permitam saber quantos *sprints* foram realizados, que histórias foram implementadas, ou quem eram os responsáveis por elas ou quais testes foram executados. Este fato não quer dizer que o projeto não foi desenvolvido usando processos ágeis, mas que a ferramenta indicada pela gestão

foi usada de forma diferente daquela adotada nos outros cinco projetos de desenvolvimento usados neste trabalho.

O fato é que a análise dos dados do modelo de referência de rastreabilidade para Métodos Ágeis (MRMA) permitiu que se percebessem indícios de um possível problema quanto a uniformidade do uso do processo ágil na companhia, mostrando que nem todas elas o utilizam da mesma maneira.

Este tipo de problema aparece para a gestão, e passa a ser sua preocupação, quando se percebe uma diferença relevante entre as produtividades médias de equipes que realizam projetos semelhantes e se quer buscar uma explicação para isso, situação que pode levar a que se faça a seguinte pergunta: “qual a maturidade do uso dos Métodos Ágeis no desenvolvimento de software na companhia, todas as equipes sabem instanciar e fazer uso dos processos de forma consistente e semelhante, por exemplo, usando a mesma ferramenta de gestão?”.

Em termos de gestão do desenvolvimento de software, a resposta a esta pergunta pode indicar, a título de exemplo, haver uma necessidade de treinamento para nivelar o conhecimento quanto a forma como o processo ágil adotado pela empresa funciona e deve ser instanciado, ou problemas nas ferramentas de apoio que motivam as equipes a abandoná-las, ou, ainda, a necessidade de maior conscientização das razões pela qual se definiu uma determinada atividade do processo, entre outras possibilidades.

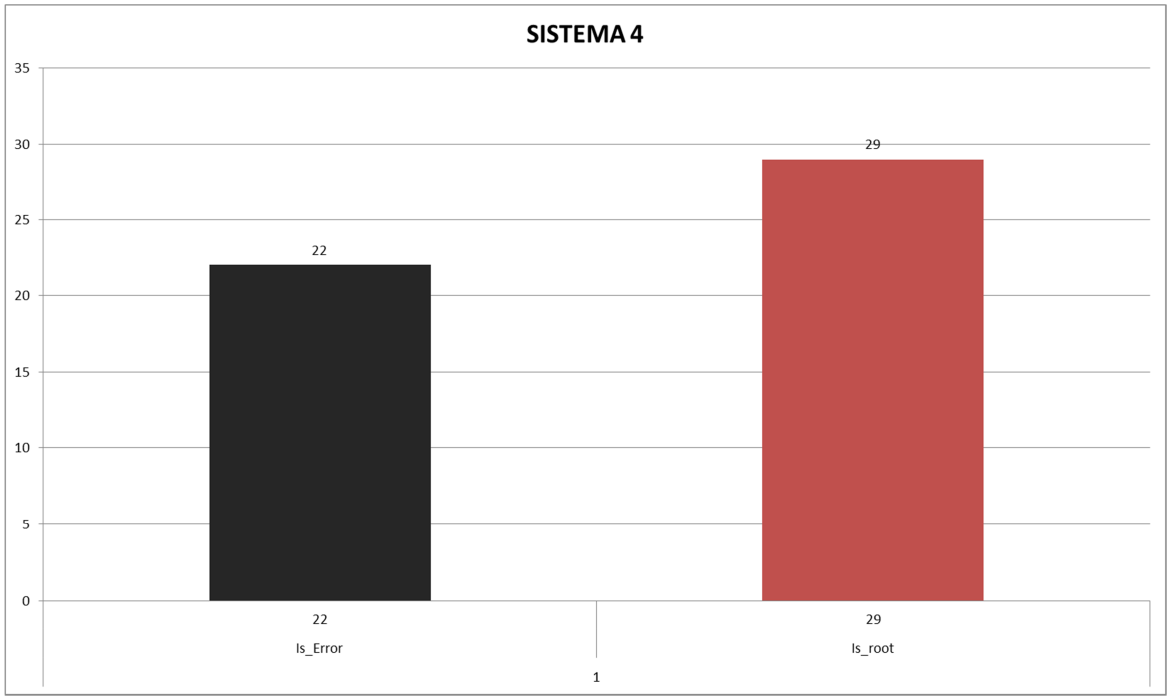


Figura 36: Taxa de erros do sistema 4 (Is_root são as Histórias de Usuário e Is_error são Histórias de Correção de Erro)

Identificando código sujo (Melhorar a efetividade do processo de revisão de código e padronização através de revisar as listas de verificação, melhorar e atualizar os analisadores de código usando como base os defeitos relatados durante os testes.)

Algumas vezes o desenvolvimento de software é realizado por diversas equipes, cada qual trabalhando em trechos diferentes do código, ou mesmo no mesmo trecho em diferentes momentos no tempo, neste processo, problemas que prejudiquem manutenções posteriores podem ser inseridos.

Ao analisar a concentração de erros, como mostrado na Figura 37, a identificação de uma alta taxa de erros acumulados numa determinada funcionalidade, pode ser um sinalizador de que aquela parte do programa é de difícil manutenção, e direcionar para a necessidade de refatoração.

Este tipo de análise só é possível porque o modelo de rastreabilidade permite que se faça uma associação entre Histórias de Usuário e Correção de Erro, na verdade uma ligação de rastreabilidade do tipo “*Is_Corrected_by*” é criada entre os

dois tipos de entidades, e a sua contagem permite saber a quantidade de erros por História.

A detecção deste tipo de comportamento, concentração de erros em determinadas funcionalidades, também pode indicar a necessidade de melhorar as listas de verificação e atualizar as regras dos analisadores de código.

Neste caso o MRMA entrega informações que permitem tanto ao gerenciamento do processo de desenvolvimento ágil tomar medidas corretivas, por exemplo, melhorando a verificação de código ou estimulando mais reúso de código, quanto ao gerenciamento do produto, ao fornecer pistas dos locais mais problemáticos e, portanto, candidatos mais fortes a serem refatorados.

Ambiente - Exportação de dados	3
Ajustar a parte de Funcionalidades a Perfis(papéis no CAv4)	3
Criar Exportações - Avaliação Física - Aspectos Neuromotores	3
NCY2-43 Implementar Tela de Filtro	3
Telas de Filtro de Exportação Composição Corporal e Exportação Geral não filtram por período.	3
Ajuste dos campos de período	3
Criar Exportações - Avaliação Física - Composição Corporal	3
Avaliação Postural - Consultar	1
Avaliação Postural - Modificar - cabeçalho e aba observação	1
Cadastro de Funcionários - Alteração não trás dados do Nome do funcionário e Matrícula	1
Falhas no relatório Tela Relatório - Nominais - Desempenho - Aderência - Físico	1
Falhas no Relatório - Nominais - Atividades Educativas	1
Falhas em Relatórios » Nominais » Registro de Frequências	1
Criar Exportações - Avaliação Física - Condicionamento Cardiorespiratório	2
Criar Exportações - Avaliação Física - Condicionamento Físico Global	2
Criar Exportações - Avaliação Postural - Geral	2
Criar Exportações - Avaliação Postural - Consolidado	2
Criar Exportações - Avaliação Patologias	2
NCY2-48 Implementar Tela	2
Falhas em Relatórios > Nominais > Desempenho > Abandono	2
NCY2-18 O campo ICP permite edição do valor calculado. Este campo não deveria permitir edição. Correçã	2
NCY2-43 Erro na consulta quando não filtramos nada.	2
Usuario loga sem unidade selecionada	1
NCY2-18 Hint inválido - telas de Aptidão Física e Avaliação postural	2
Após cadastrar um novo ambiente, ocorre um erro ao acessar a tela de academias neste novo ambiente	1
Tela - Cadastro - Grupos de Funcionários . Não foi possível cadastrar um novo grupo - mensagem de valida	1
Tela - Cadastro - Protocolo Esforço . Corrigir rótulo do campo - 'Nomeanamnese' ---	1
Tela - Cadastro - Aptidão Física . Ao tentar gravar sem informar dados - Erro ao tentar inserir uma nova ava	1

Figura 37: Quantidade de correções de erros por histórias

Rastreando os detalhamentos das histórias

O quadro da Figura 37 mostra o desdobramento das Histórias de Usuário (requisitos) em histórias (requisitos) mais detalhada(o)s. Detalhar requisitos (histórias) é uma técnica para aumentar o entendimento e facilitar as estimativas. Porém para ser útil é preciso responder ao menos a duas perguntas, a primeira é “Considerando uma História de Usuário (requisito), quais são os seus desdobramentos?”, ou seja, ser capaz de encontrar e seguir a trilha que leva das

histórias (requisitos) de maior granularidade, para as mais detalhadas. Enquanto que a segunda pergunta, uma vez que a resposta da primeira seja sim, é se “as histórias(requisitos) detalhadas, de fato realizam a história(requisito) hierarquicamente anterior?”.

Através do uso do MRMA foi possível estabelecer uma relação do tipo “*Is_detailed_by*”, a qual permite compreender o processo de evolução dos requisitos por desdobramento partindo das versões de mais alto nível indo na direção de um nível cada vez maior de detalhamento.

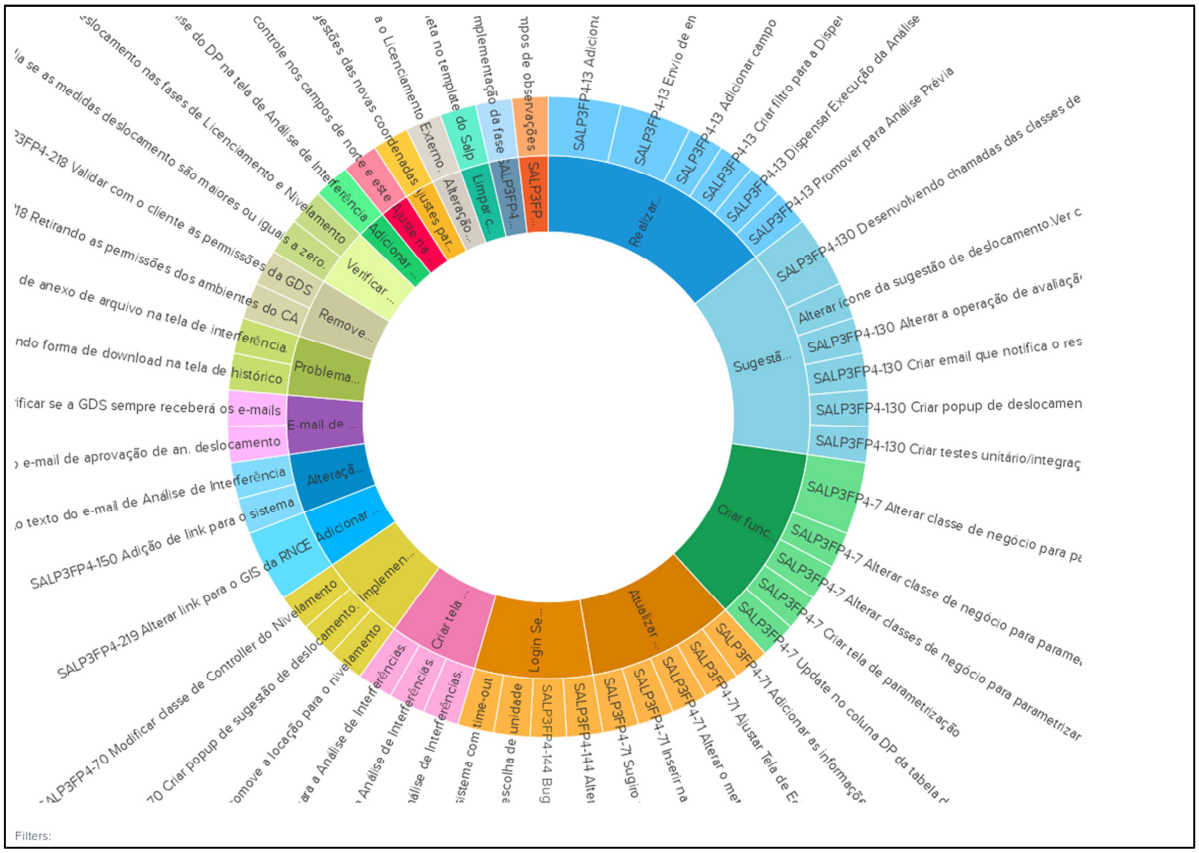


Figura 38: Gráfico SunBurst – mostrando o detalhamento das histórias – SISTEMA 1 - Identificando problemas nos requisitos

Conseguir seguir os requisitos, desde sua versão de mais alto nível em direção as mais detalhadas, e de lá mudar a direção de investigação para identificar as correções de erros, pode ajudar a gestão do desenvolvimento de software a identificar falhas no processo de elicitação dos requisitos, por exemplo, um requisito ausente.

Na Figura 39 vemos um exemplo deste tipo de uso, nele percebe-se que um determinado campo (Freq. Cardiáca) deveria ter sido deixado aberto para edição, mas isso não aconteceu na implementação de quatro Histórias.

As causas deste acontecimento podem ser várias, mas do ponto de vista de gestão, ele significa que há falhas no detalhamento das histórias (requisitos) culminando em incompletudes prejudiciais ao projeto, por exemplo, tornando necessária a correção das funcionalidades após terem sido implementadas, conforme retratado na Figura 39.

Outro aspecto a ser considerado é que as histórias que corrigiam o problema exibido na Figura 38, foram registradas como correção de erros na ferramenta JIRA, e por isso ligadas as Histórias de Usuário correspondentes usando o tipo de ligação *Is_Corrected_by*.

Sabendo disso, de imediato se retorna ao problema da taxa de erros com a pergunta, “quanto dos erros é retrabalho causado por problemas nos requisitos?”. Do ponto de vista da gestão a resposta a esta pergunta é um direcionador que indica que esforços devem ser empreendidos para melhoria nesta importante disciplina da engenharia de software.

Considerando o emprego dos Métodos Ágeis, podemos nos perguntar, pelas causas deste problema, foi a ausência do cliente? Ou os critérios de aceitação não estão suficientemente detalhados? Ou outro motivo? O fato do MRMA identificar e expor o problema, fornece subsídios que fundamentam a realização de análises críticas mais profundas, que instrumentam a gestão do desenvolvimento de software com informações que permitem direcionar mais adequadamente os esforços de melhoria dos processos.

O emprego da rastreabilidade em métodos ágeis fornece meios para análises muito variadas, como podemos ver na Figura 39. Permitindo ao rastreamento sair das histórias de mais alto nível, caminhar para as histórias detalhadas, seguir para as correções de erros, em seguida identificar os testes a que foram submetidas, correlacionando com as pessoas responsáveis. Estes são exemplos de passos possíveis no MRMA, que dão condições de se responder a perguntas interessantes, tais como: “os casos de teste verificam de fato o que a História de Usuário descreve?”; “se os casos de teste e as Histórias de Usuário são coerentes entre si,

porque não correspondiam ao que o usuário esperava?”, entre outras possíveis perguntas.

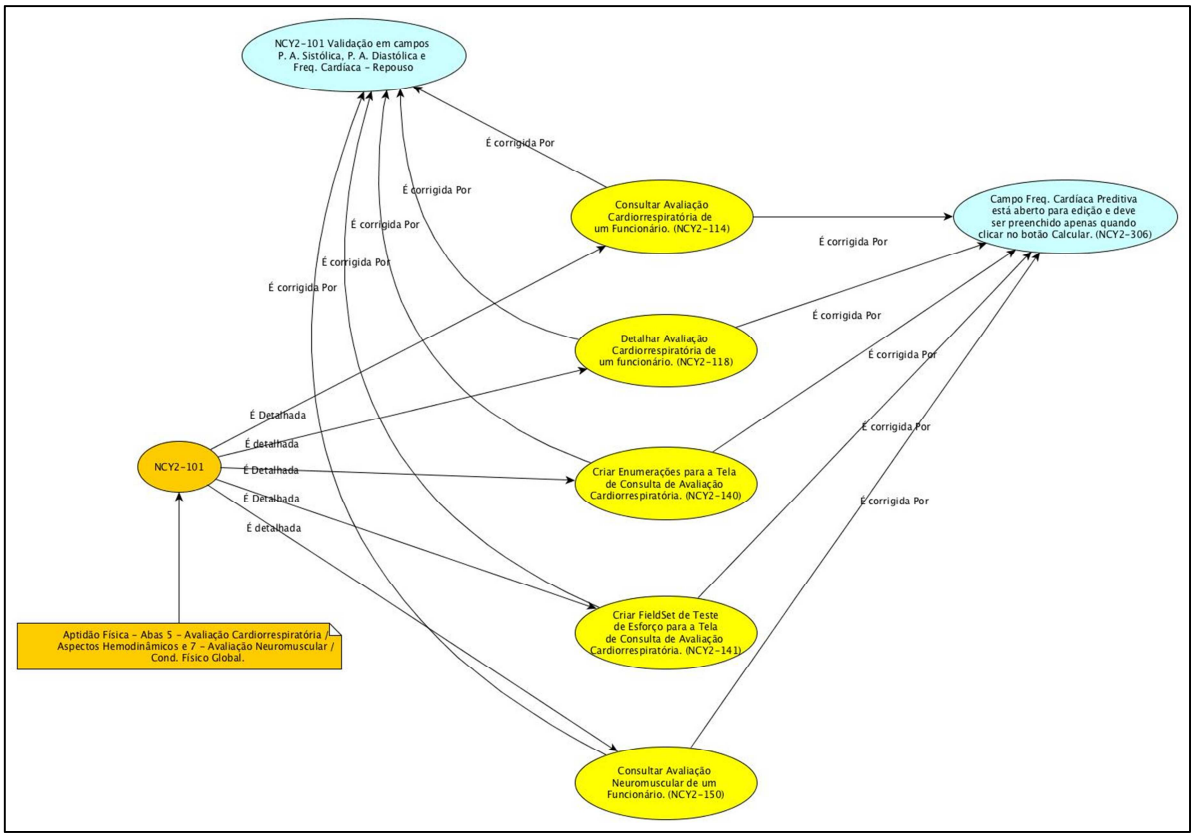


Figura 39: Usando o MRMA para identificar falha no processo de requisitos

Gerenciando as ligações

O MRMA pode ser usado para apoiar a gestão de um processo de software, mas para ele ser útil deve ser possível minimizar a quantidade de ligações quebradas, que são aquelas em que se localiza um item que não participa de nenhuma ligação, nem como fonte e nem como destino.

SourceEndPointID	TargetEndPointId	Name	Description	Name1	ID_Sprint_Backlog
6		Sistema 1	Verificar Tela de . . .	Is_root	0
5		Sistema 1	Exibir botões de fechar no pop de niv	Is_root	0
3		Sistema 1	Validar a inclusão/alteração das . . .	Is_root	0
2		Sistema 1	Quando o usuário do sistema não for	Is_root	0
1		Sistema 1	Mudar Título do Email Enviado na Apr	Is_root	0
16		Sistema 1	Na tela de aprovação do estudo de fu	Is_Error	1
15		Sistema 1	Problema de CSS na exibição dos botõ	Is_Error	1

Figura 40: Links quebrados

Source	Target	Tipo de rastro	Status
Sugestão de Deslocamento ...	Alterar ícone da sugestão de Deslocamento...	Is_Detailed_by	false
CRUD Bairros	Tela - Cadastros - Bairro . Ordenar lista de cidades alfabetica	Is_Detailed_by	false

Figura 41: Ligações não validadas

Embora não tenha sido o caso deste trabalho, analisar a validade dos *links* torna-se ainda mais importante quando se faz uso de técnicas automáticas de recuperação das ligações de rastreabilidade, a exemplo do uso da técnica "*Information Retrieval*".

6. Conclusões

Os Métodos Ágeis podem se beneficiar da rastreabilidade como ferramenta para ajudar a gerência de conhecimento, especialmente em situações de alta rotatividade de pessoal, ou no caso em que o desenvolvimento de um produto de software seja transferido de *on shore* para *off shore, que são s*ituações em que um dos principais pontos fortes dos Métodos Ágeis, a memória coletiva, é perdida.

Embora haja algumas propostas para implementar rastreabilidade em Métodos Ágeis, nenhuma delas mapeia especificamente os conceitos desta família de processos. Neste sentido o Modelo de Rastreabilidade MRMA trouxe os conceitos extraídos diretamente dos modelos de processos ágeis e os validou em projetos reais utilizando duas ferramentas para gestão de projetos em Métodos Ágeis. JIRA e TFS.

Apesar de rastreabilidade ser uma técnica reconhecida como originalmente dos processos tradicionais, os Métodos Ágeis podem também se beneficiar dela. O MRMA demonstrou que é possível usar a rastreabilidade nas ferramentas de gestão de projetos ágeis, e conseguiu fornecer respostas bastante interessantes às perguntas de gestão, por exemplo, conseguiu demonstrar altas taxas de bugs sendo corrigidos em muitos *sprints*. Também conseguiu demonstrar problemas na instanciação dos processos ágeis e que ele pode ser usado para analisar o processo de desdobramento e detalhamento dos requisitos, ou das Histórias de Usuários.

6.1 Contribuições

O objetivo deste trabalho foi propor um Modelo de Rastreabilidade para apoiar a gestão de projetos em Métodos Ágeis, a seguir veremos seus diferenciais e contribuições, quando comparado com outras pesquisas, tais como: Espinoza e Garbajosa (2011); Taromirad e Paige (2012); Lee et al. (2003) e; Alaa e Samir (2011).

As principais contribuições deste trabalho foram:

1. Propor um modelo para Métodos Ágeis, em vez de modelos generalistas, que não mapeiam especificamente os processos ágeis;

2. Construir o MRMA a partir os modelos de processos ágeis, SCRUM, XP, DSDM, Família *Crystal*, ASD e FDD. Levantando conceitos de gerenciamento de projetos e engenharia de software usado em processos ágeis, tornando o modelo de rastreabilidade apto a manipular com artefatos, atividades e papéis próprios destes processos;
3. Aplicar este modelo em dados reais de projetos de uma grande empresa do ramo da energia, validando tanto os conceitos propostos, quanto a utilidade, ao demonstrar sua capacidade de responder perguntas úteis para a melhoria e gestão de projetos em Métodos Ágeis;
4. Demonstrar que os conceitos propostos e aplicados no MRMA são válidos para a rastreabilidade em Métodos Ágeis e estão presentes nas ferramentas que dão apoio a gestão do desenvolvimento em Métodos Ágeis.

Uma contribuição adicional foi a aplicação do MRMA nas bases de dados das ferramentas que apoiam os processos ágeis, podendo ser uma boa alternativa para a rastreabilidade em Métodos Ágeis, uma vez que feita de forma automática, não prejudicará a característica ágil dos Métodos Ágeis.

6.1 Limitações

Devido ao acesso limitado, este trabalho só pôde validar os conceitos do MRMA comparando com a base de dados da ferramenta JIRA, para outras ferramentas como o Microsoft Team Foundation Server, a plataforma JAZZ da IBM ou Mingle não foi possível acessar seus modelos de design ou suas bases de dados, impedindo que se pudesse fazer validações dos conceitos do MRMA nestas ferramentas.

Outra limitação foi a falta de acesso ao código fonte, o que pode ser conseguido através de acesso aos repositórios de código fonte, ou de maior integração às ferramentas de desenvolvimento, por exemplo, integração a ferramenta Eclipse.

Apesar do acesso aos dados reais de projetos em Métodos Ágeis, a quantidade de projetos foi muito limitada, seis projetos no total, devido a restrição de acesso a dados dos projetos por motivos de segurança da informação. Isto limitou

as possibilidades de validação de quais perguntas de gestão o uso do MRMA poderia responder.

Uma observação pertinente, é que os seis projetos, apesar de serem da mesma companhia, foram conduzidos por equipes diferentes, internas e externas à empresa, e que as informações de rastreabilidade não apresentavam similaridade nos tipos de informações presentes. Um exemplo disso é a comparação entre o Sistema 4, Figura 36, e os Sistemas 2 e 5 das figuras 34 e 35. No projeto do sistema 4 não haviam as entidades *Sprint*, *Sprint Backlog* e *Test Case,* enquanto que nos Sistemas 2 e 5 estas entidades estavam presentes.

6.3 Trabalhos futuros

Como trabalho futuro, o modelo deve ser experimentado em outras fontes de dados de rastreabilidade, para validação mais ampla pois apenas uma parte do modelo foi satisfeito, uma vez que o JIRA não apresentava todos os conceitos, os conceitos "*Code*" e "*Deploy*" não existe dentro do esquema de dados do JIRA.

Adicionalmente, agentes podem ser construídos para extrair dados durante o andamento dos projetos para popular a base de dados do MRMA da forma mais automática possível, a fim de evitar ao máximo que a agilidade dos Métodos Ágeis seja prejudicada.

Neste trabalho construímos uma ferramenta para extrair dados do JIRA, mas ela não se integra com as IDE's de desenvolvimento como o Eclipse, onde ficam certas entidades, como, por exemplo, Testes Unitários e Módulos de Código.

Outra proposta é desenvolver e/ou usar ferramentas de visualizações mais avançadas permitindo análises mais complexas sobre os dados de rastreabilidade. Entre as possíveis visualizações, teríamos: *Sunburst view; CityScape view; Bar chart; Net Map view,* entre outras.

Aplicar ferramentas analíticas de *datawharehouse* para análise complexas multidimensional dos dados.

Uma validação da utilidade do modelo pode ser feita, conduzindo um *survey* na indústria, avaliando se os dados de rastreabilidade fornecidos pelo modelo são reconhecidos como úteis para a gestão de projetos em métodos ágeis.

A cobertura do modelo aos Métodos Ágeis pode ser mais trabalhada, por exemplo, não existe nada no modelo que mapeie os requisitos de negócios e estudos de viabilidade previstos no DSDM, e que podem ser importantes no desenvolvimento de software. Da mesma forma, nenhum conceito foi previsto para o mapeamento de entradas como Padrões, Normas, Definições Arquiteturais, entre outras. Estas e outras ausências podem ser estudadas, para avaliar se estes elementos poderiam ser inseridos no modelo.

Referências

[Abrahamsson et al., 2002] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., Agile software development methods, In: Review and analysis. Espoo 2002. VTT Publications 478. 107 p, 2002.

[Alaa e Samir, 2014] Alaa, G., Samir, Z., A multi-faceted roadmap of requirements traceability types adoption in SCRUM: An empirical study, In: Informatics and Systems (INFOS), 9th International Conference on. IEEE, 2014. p. SW-1-SW-9, 2014.

[Alavi e Leidner 2001] Alavi, M., and Leidner, D. E., Knowledge Management and Knowledge Management Systems: Conceptua lFoundations and Research Issues, , pp. 107-136, 2001.

[Alves e Alves, 2000] Alves, S., Alves, A., Engenharia De Requisitos Em Métodologias Ágeis, 2000.

[Akgun, Byrne, Keskin e Lynn, 2006] Akgun, A.E., Byrne, J.C., Keskin, H. and Lynn, G.S., 'Transactive memory system in new product development teams', IEEE Transactions on Engineering Management, Vol. 53, No. 1, pp.95–111.

[Bender e Fish, 2000] Bender, S., Fish, A., "Transfer of knowledge and the retention of expertise: the continuing need for global assignments". Knowledge Management. **4**(2): pp. 125-37, 2000.

[Arbain et al., 2014] Arbain, B., Wan Kadir, W.M.N., Ghani, I., Agile non functional requiremnents (NFR) traceability metamodel, 2014.

[Capra e Luisi, 2014] Capra, F., Luisi, P., The systems view of life: A Unifying Vision, 2014.

[Chan & Thong, 2009] Chan, F.K.Y., Thong, J.Y.L., Acceptance of agile methodologies: A critical review and conceptual framework., In: Decision Support Systems, 46, p. 803-814, 2009.

[Chua e Pan, 2008] Chua, L., Pan, S. L., Knowledge transfer and organizational learning in IS ofshore sourcing., Omega, Volume 36, Issue 2, Pages 267–281, 2008.

[Cleland e Huang, 2012] Cleland, J., Huang, Traceability in Agile Projects, 2012.

[Cockburn, 2006] Cockburn, A. , Agile software development: the cooperative game (agile software development series), Addison-Wesley Professional, Boston, United States, 2006.

[Conboy et al., 2010] Conboy, K., Coyle, S., Wang, X., Pikkarainen, M. , People over process: key people challenges in agile development., In: IEEE Software, 99, p. 47-57, 2010.

[Delater et al., 2011] Delater, A., Juppner, D. , Merten, T., Improved Representation of Traceability Links in Requirements Engineering Knowledge using Sunburst and Netmap Visualizations., In: Fourth International Workshop on Managing Requirements Knowledge (MARK), p. 17 - 21, 2011.

[Delater et. al, 2012] Delater, A., Narayan, N., Paech, B., Tracing requirements and source code during software development., In: ICSEA 2012, The Seventh International Conference on Software Engineering Advances. 2012. p. 274-282, 2012.

[Delater e Paech, 2013] Delater, A., Paech, B., UNICASE Trace Client:(Semi-) Automatic Tracing of Requirements and Code During Development for Small and Medium Enterprises. Softwaretechnik-Trends, v. 33, n. 1, p. 27-28, 2013.

[Dibbern at al., 2008] Dibbern, J., Winkler, J., Heinzl, A., Explaining variations in client extra costs between software projects offshored to India, Management Information Systems Quarterly, Volume 32, Issue 2, Pages 333–366, 2008.

[Espinoza e Garbajosa, 2011] Espinoza, A., Garbajosa, J., A study to support agile methods more effectively through traceability., In: Innovations in Systems and Software Engineering, v. 7, n. 1, p. 53-69, 2011.

[Finkelstein e Emmerich , 2000] Finkelstein, A., Emmerich, W., The future of requirements management tools., In: Information Systems in Public Administration and Law, 2000.

[Fitzgerald et al., 2013] Fitzgerald, B., Stol, K., O'Sullivan, R., O'Brien, D. , Scaling agile methods to regulated environments: An industry case study., In: 35th International Conference on Software Engineering (ICSE-SEIP), 2013.

[Fowler et al., 2001] Fowler, M., Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham,W., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, Jon.,

Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland,J., Thomas, D., Agile Manifesto, 2001.

[Furtado e Zisman, 2014] Furtado, F., Zisman, A., Trace++: A Traceability Approach to Support Transitioning to Agile Software Engineering, 2014.

[Ghazarian , 2008] Ghazarian, A., Traceability patterns: an approach to requirement-component traceability in agile software development., In: Proceedings of the 8th conference on Applied computer scince., World Scientific and Engineering Academy and Society (WSEAS), p. 236-241, 2008.

[Ghobadi e Mathiassen, 2014] Ghobadi, S., Mathiassen, L., Perceived Barriers to Effective Knowledge Sharing in Agile Software Teams, Information Systems Journal (ISJ), DOI: 10.1111/isj.12053, 2014.

[Gotel et al., 2012] Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher , P., Dekhtyar, A., Antoniol ,G. ; Maletic, J.; Mäder, P., Software and Systems Traceability, 2012.

[Gotel et al., 2014] Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Mäder, P., Software traceability: Trends and future directions., In: Proc. of the 36th International Conference on Software Engineering (ICSE), Hyderabad, India , 2014.

[Gotel e Finkelstein, 1994] Gotel, O., Finkelstein, A., An analysis of the requirements traceability problem., In: Requirements Engineering, 1994., Proceedings of the First International Conference on. IEEE, p. 94-101, 1994.

[Gotel, 1995] Gotel, O., Contribution Structures for Requirements Traceability. , In: Imperial College of Science, Technology and Medicine., University of London., 1995.

[Gotel e Finkelstein, 1997] Gotel, O., Finkelstein, A., Extended Requirements Traceability: Results of an Industrial Case Study, In: Requirements Engineering, 1994., Proceedings of Third IEEE, 1997.

[Gregory at al., 2009] Gregory, R., Beck, R., Prifling, M., Breaching the knowledge transfer blockade in IT offshore outsourcing projects - A case from the financial services industry, Proceedings of the 42nd Annual Hawaii International Conference on System Sciences (HICSS'09), 2009.

[Gupta at al., 2015] Gupta, A., Sureka, A. , Padmanabhuni, S., Identifying software process management challenges: survey of practitioners in a large global IT company,In: IEEE Explore, 2015

[Jaber et al., 2013] Jaber, K., Sharif, B., Liu, C., A Study on the Effect of Traceability Links in Software Maintenance., In: Access, IEEE, v. 1, p. 726-741, 2013.

[Jacobsson, 2009] Jacobsson, M., Implementing Traceability In Agile Software Developmen., Lund University, 2009.

[Jaqueira, 2013] Jaqueira, A., Uso de Modelos i* para Enriquecer Requisitos em Métodos Ágeis, 2013.

[Lago et al., 2009] Lago, P., Muccini, H., Van Vliet, H., A scoped approach to traceability management, In: Journal of Systens and Software, v. 82, n. 1, p. 168-182, 2009.

[Lee et al. , 2003] Lee, C. ; Guadagno, L., Jia, X., An agile approach to capturing requirements and traceability., In: Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003). Citeseer., 2003.

[Lange e Gillis, 2005] Lewis, K., Lange, D., Gillis, L., Transactive Memory Systems, Learning, and Learning Transfer, 2005.

[Liyanage,Ballal, Elhag, e Li, 2009] Liyanage, C., Ballal, T., Elhag, T. and Li, Q. (2009) Knowledge communication and translation- A knowledge transfer model. Journal of Knowledge Management, 2009.

[Marcus et al., 2005] Marcus, A., Xie, X. Poshyvanyk, D., When and how to visualize traceability links?. , In: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering. ACM, p. 56-61, 2005.

[Mäder e Cleland-Huang, 2013] Mäder, P., Cleland-Huang, J., A visual language for modeling and executing traceability queries., In: Software & Systems Modeling, v. 12, n. 3, p. 537-553, 2013.

[Mäder e Egyed, 2012] Mäder, P., Egyed, A., Assessing the Effect of Requirements Traceability for Software Maintenance, IEEE, 2012.

[Martins et al., 2006] Martins, A. F., Nardi, J. C., Falbo, R. A., ReqODE: Uma Ferramenta de Apoio à Engenharia de Requisitos Integrada ao Ambiente ODE. XX Simpósio Brasileiro de Engenharia de Software, SBES, p. 31-36, 2006.

[Nerur et al., 2005] Nerur, S., Mahapatra, R.K., Mangalaraj, G. , Challenges of migrating to agile methodologies., In: Communications of the ACM, 48, p. 72-78, 2005.

[Omoronyia et al., 2011] Omoronyia, I., Sindre, G., STÅLHANE, T. , Exploring a Bayesian and linear approach to requirements traceability., In: Information and Software Technology., v. 53, n. 8, p. 851-871, 2011.

[Ramesh, 2008] Ramesh, B. , Agile requirements engineering practices:an empirical study. IEEE Softw 25(1):60–67, 2008

[Ramesh et al., 2010] Ramesh, B., Cao, L., Baskerville, R. , Agile requirements engineering practices and challenges: an empirical study. , In: Information Systems Journal, 20, p. 449-480, 2010.

[Ramesh e Jarke, 2001] Ramesh, B., Jarke, M., Toward reference models for requirements traceability, In: Software Engineering, IEEE Transactions on, v. 27, n. 1, p. 58-93, 2001.

[Ramesh et al., 2012] Ramesh, B., Mohan, K.& Cao, L., Ambidexterity in Agile Distributed Development: An Empirical Investigation. Information System Research., 23, 323-339.

[Richard Torkar at al., 2012] Torkar, R., Nidhra, S., Yanamadala M., Afzal, W., Knowledge transfer challenges and mitigation strategies in global software development – A systematic literature review and industrial validation. , 2012.

[Rottman e Lacity, 2008] Rottman, J., Lacity, M., The impact of outsourcing on client project managers, 2008.

[Salger at al., 2010] Salger, F., Sauer, S., Engels, G., Baumann, A., Knowledge transfer in global software development – Leveraging ontologies, tools and assessments, Proceedings of the 5th IEEE International Conference on Global Software Engineering (ICGSE'10), 2010.

[Singley e Anderson, 1989] Singley, M. K., & Anderson, J. R. *The transfer of cognitive skill*. Cambridge, MA: Harvard Univ. Press, 1989.

[Sommerville e Kotonia, 1998] Sommerville, I., Kotonia, G., Requirements engineering: processes and techniques., In: John Wiley & Sons, Inc., 1998.

[State of Agile™ Survey (2015)] State of Agile™ Survey (2015), In: http://info.versionone.com/state-of-agile-development-survey-ninth.html, 2015.

[Taromirad e Paige, 2012] Taromirad, M., Paige, R., Agile Requirements Traceability Using Domain-Specific Modelling Languages, 2012.

[Taromirad, 2014] Taromirad, M., A Modelling Approach to Multi-Domain Traceability., In: Tese de Doutorado. University of York., 2014.

[Taweel at al., 2009] Taweel, A., Delaney, B., Arvanitis, T., Zhao, L., Communication, knowledge and co-ordination management in globally distributed software development: Informed by a scientific software engineering case study, Proceedings

of the 4th IEEE International Conference on Global Software Engineering (ICGSE'09), 2009.

[Vicente, Delamaro e Maldonado, 2010] Vicente, AA., Delamaro, M., Maldonado, J., Uma Revisão Sistemática sobre a atividade de Teste de Software no contexto de Métodos Ágeis, 2010

[West e Grant, 2010] West, D., Grant, T., Agile Development: Mainstream Adoption has Changed Agility., In: Forrester Research Inc, pp. 1–20, 2010.

[Xiaofan Chen, 2012] Chen, X., Extraction and visualization of traceability relationships between documents and source code., In: ASE '10 Proceedings of the IEEE/ACM international conference on Automated software engineering, p. 505-510, 2012.

[Zemont, 2005] Zemont, G., Towards Value-Based Requirements Traceability, 2005.