UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM PLANO DE MÉTRICAS PARA MONITORAMENTO DE PROJETOS SCRUM

EDUARDO HENRIQUE SPIES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifica Universidade Católica do Rio Grande do Sul

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz

Porto Alegre
2013

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP)

S755p Spies, Eduardo Henrique
Um plano de métricas para monitoramento de projetos scrum / Eduardo Henrique Spies. – Porto Alegre, 2013.
84 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz.

1. Informática. 2. Engenharia de Software. 3. Administração de Projetos. I. Ruiz, Duncan Dubugras Alcobas. II. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Um Plano de Métricas para Monitoramento de Projetos SCRUM" apresentada por Eduardo Henrique Spies como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Engenharia de Software e Banco de Dados, aprovada em 15/03/2013 pela Comissão Examinadora:

Prof. Dr. Duncan Dubugras Alcoba Ruiz – Orientador PPGCC/PUCRS

Prof. Dr. Rafael Prikladnicki – PPGCC/PUCRS

Profa. Dra. Sabrina dos Santos Marczak – FACIN/PUCRS

Homologada em...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

Gostaria de agradecer em primeiro lugar a minha família que sempre me apoiou em todos os momentos e não deixaram de acreditar em mim em momento algum. Meu pai Geraldo Spies pelo carinho e compreensão, minha mãe Jeani Spies pelo incentivo e confiança, minha irmã pela parceria e amizade em todos os momentos em que precisei, foram todos fundamentais nessa jornada.

Ao meu orientador Duncan Ruiz pela paciência e ensinamentos sempre auxiliando a buscar o melhor caminho e mostrando o sempre pode ser melhorado.

Aos meus amigos de infância que sempre estiveram e continuam estando presentes em todos os passos da minha vida provendo apoio e incentivo em todos os aspectos da minha vida, muito obrigado Ricardo Rohden, Lucas Hilgert e William Schneider por sempre estarem comigo durante esse percurso!

Meus grandes amigos da Housing que foram importantíssimos ao meu crescimento durante esses dois anos com os quais eu aprendi muita coisa e continuo aprendendo. Muito obrigado Renata Bartolette, Daniela Chies, Rogério Martins, Liana Giselle Padilla, Luizy Andrade, Luciano T.Filho e todos os outros que me ajudaram a chegar a este ponto.

Os amigos e colegas do mestrado pelas horas de diversão e estudo e divisão de preocupações com prazos e entregas, também foram muito importantes para mim. O grupo de pesquisa GPIN e todos os colegas.

Obrigado a todos por tudo!!

Um Plano de Métricas Para monitoramento de projetos Scrum

Resumo

Métodos ágeis já consolidaram o seu espaço tanto na indústria como na academia, sendo cada vez mais utilizados. Com o foco em retornos frequentes aos clientes, estes métodos têm dificuldades para obter controle e manter comunicação eficiente, especialmente em projetos de maior porte e com grande quantidade de pessoas envolvidas. Técnicas de engenharia de software têm se mostrado de grande valia para aumentar a previsibilidade e dar mais disciplina deste tipo de projetos. Neste trabalho é apresentado um programa de métricas para SCRUM e uma extensão de um ambiente de Data Warehousing para o monitoramento de projetos. Desta forma, é provido um repositório consistente que pode ser utilizado como referencial histórico de projetos e para a visualização de métricas em diferentes dimensões, facilitando o controle sobre todos os aspectos do progresso de um projeto.

Palavras chave: Scrum, Métodos ágeis, *Data Warehouse*, métricas, monitoramento de projetos.

A Metrics Plan for Monitoring Scrum Projects

Abstract

Agile methods have earned their space both in industry and in academia, being increasingly used. With the focus on frequent returns to customers, these methods have difficulties to gain control and maintain efficient communication, especially in larger projects with several collaborators. Software engineering techniques have proved of great value to increase predictability and provide more discipline to this kind of projects. In this paper we present a metrics program for SCRUM and an extension of a Data Warehousing environment for monitoring projects. Thus, we provide a consistent repository that can be used as a historical reference of projects and for exploring metrics in different dimensions, easing control over all aspects of the progress of a project.

Keywords: Data Warehouse, Agile Methods, Scrum, metrics, project monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CMMi – *Capability Maturity Model Integrated*

DSA – *Data Staging Area*

DW – *Data Warehouse*

ETC – Extração Transformação e Carga

EVM – *Earned Value Management*

GQM – *Goal – Question – Metric*

MPS.br – Modelo de Processo de Software brasileiro

OLAP – *On-Line Analytical Process*

TR – Trabalho Restante

XP – *eXtreme Programming*

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos métodos ágeis vêm se tornando um tema bastante recorrente, tanto na indústria [Williams, Brown, Meltzer, & Nagappan, 2011] quanto em trabalhos acadêmicos [Dingsøyr, 2012] Estes métodos utilizam conhecimentos empíricos para melhor gerenciar projetos complexos, de requisitos instáveis e para melhor responder às mudanças. Em geral estes métodos seguem princípios definidos no manifesto ágil e compartilham algumas características como o desenvolvimento através de pequenas iterações, entrega frequente de software ao cliente e aumento do poder do time de desenvolvimento de um projeto. No âmbito das publicações, a grande maioria fala sobre *Extreme Programming* (XP), apesar do Scrum ser mais dominante na indústria [Dingsøyr, 2012]. Contudo, nos últimos anos, o Scrum vem ganhando cada vez mais destaque já ultrapassando outras metodologias ágeis, como o XP e o Crystal, em publicações anuais [Dingsøyr, 2012].

Outros autores discutem os benefícios de utilizar práticas de modelos como o CMMI para complementar os métodos ágeis, especialmente quando o porte dos projetos cresce. É esperado conseguir, desta forma, aproveitar o controle e a disciplina destes modelos em conjunto com a agilidade e a rapidez de resposta a mudanças dos métodos ágeis [Jakobsen & Johnson, 2008], [Mahnic & Zabkar, 2008], [Jakobsen & Sutherland, 2009].

1.1 O PROBLEMA

O Scrum possui características que aumentam a adaptabilidade e produtividade em projetos de desenvolvimento de software. Todavia, em projetos de maior porte, o arcabouço Scrum se beneficia da utilização de práticas de engenharia de software que tem por objetivo aumentar a disciplina e controle destes projetos. Existem diversos relatos de sucesso de aplicação do Scrum em grandes companhias [Williams *et al*., 2011] e em projetos de grande porte [Jakobsen & Sutherland, 2009]. Nestes casos de sucesso foi realizada a aplicação de outras práticas de engenharia de software em conjunto com o arcabouço Scrum evitando que aconteça o que é chamado de um *“Flaccid Scrum”* , termo forjado por Martin Fowler [Fowler, 2009] para descrever uma precária implantação do Scrum onde não há aplicação de nenhuma outra prática para dar suporte ao arcabouço.. A ausência de práticas e técnicas no Scrum é proposital, e tem o objetivo dar poder de

decisão ao time de desenvolvimento sobre quais técnicas utilizar, de acordo com as características específicas do projeto [scrummethodology, 2009]. Desta forma, o uso de técnicas para o controle e utilização do arcabouço com disciplina, fornecem aos projetos uma combinação de adaptabilidade e previsibilidade [Jakobsen & Sutherland, 2009] aumentando a possibilidade de sucesso desses projetos. Uma das maneiras de se obter um nível desejável de controle é através de utilização de métricas e mensuração do projeto de desenvolvimento. Buscamos suprir esta necessidade através da apresentação de um plano de métricas que respeite os princípios ágeis de modo a obter controle e agilidade no decorrer destes projetos, juntamente com um ambiente de Data Warehouse para armazenamento e consulta destas métricas de software.

1.2 OBJETIVOS

Esse trabalho apresenta a proposta de um plano de métricas e modelagem dimensional para um ambiente de DW para o monitoramento de projetos Scrum, que respeite os princípios ágeis e os critérios para criação de métricas em neste tipo de projeto [Hartmann & Dymond, 2006]. Este plano foi criado para a utilização no ambiente de gestão de métricas de projeto SPDW+, o qual é baseado em Data warehouse (DW) [Silveira, Becker, & Ruiz, 2010] que tem o intuito de automatizar o processo de coleta de métricas. O plano de métricas e modelagem dimensional do ambiente de DW apresentado são os passos iniciais na adaptação e evolução do ambiente SPDW+, desenvolvido inicialmente para uma abordagem tradicional de desenvolvimento, em cascata, para que o mesmo possa ser utilizado em métodos ágeis. Baseado neste plano de métricas foi criado um modelo dimensional e um cubo de dados para o armazenamento e monitoramento destas métricas em um ambiente de Data *Warehousing*.

1.3 DESENHO DE PESQUISA

A o desenho de pesquisa utilizado para este trabalho. Neste desenho estão contempladas as principais fases realizadas em ordem de cumprir o objetivo do trabalho.

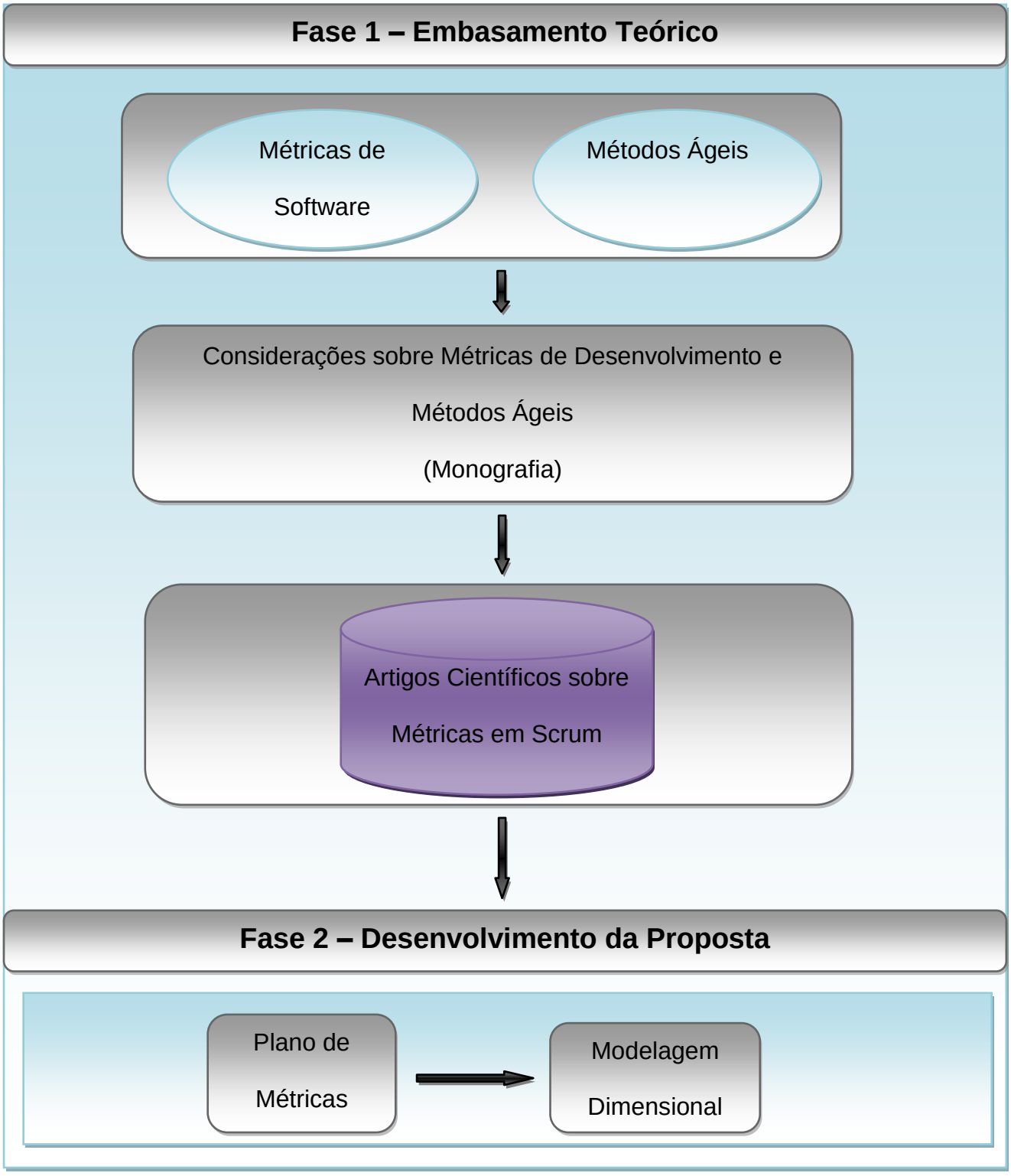

Figura 1 - Desenho de Pesquisa

1.3.1 Fase 1 – Embasamento teórico

Nesta primeira fase foi realizado o embasamento teórico necessário para uma melhor compreensão da área sobre o qual o trabalho seria realizado. Desta forma foram estudados de maneira geral e abrangente conteúdos relacionados a área de métricas em um contexto de desenvolvimento de software, assim como a área de métodos ágeis de desenvolvimento. Nesta etapa o principal objetivo foi buscar a compreensão dos conteúdos e práticas consideradas básicas nessas duas áreas de modo que um denominador comum entre as mesmas pudesse ser encontrado. O resultado desta fase de estudo sobre as duas áreas foi uma monografia identificando as principais métricas e técnicas de coleta de métricas disponíveis na literatura e a compatibilidade dessas técnicas com métodos ágeis de desenvolvimento. Outra fonte de conhecimento para o embasamento teórico e realização deste trabalho vem da vivência do estágio na HP *Enterprise Services* onde foi possível visualizar na prática as técnicas de coletas de métricas utilizadas em ambientes de projetos reais e também os diversos materiais didáticos disponíveis a consulta sobre as práticas organizacionais relacionadas tanto à área de coleta de métricas quanto métodos ágeis de desenvolvimento. Dentro do âmbito de métricas por questão de tempo foi escolhido trabalhar com métricas de acompanhamento e monitoramento de projetos.

Com o embasamento teórico sobre as duas áreas abordadas, iniciou-se a busca de trabalhos científicos relevantes englobando estas duas áreas em conjunto. Nesta fase pesquisas foram realizadas nos diversos repositórios científicos disponíveis buscando por esforços já realizados na área de coleta de métricas e técnicas de estimativas em conjunto com métodos ágeis de desenvolvimento. Essa pesquisa foi realizada através de *strings* de busca com termos chaves e também através do referencial teórico dos trabalhos encontrados. Um dos resultados desta fase é a seção de trabalhos relacionados onde foram identificados os diferentes esforços na área e agrupados de acordo com a classificação de Kan [Kan, 2002] como está apresentado na Ágeis na Seção 2.5. Nesta fase também houve a escolha por um aprofundamento no arcabouço Scrum na área de métodos ágeis e também a limitação à análise e monitoramento de métricas de projeto devido a restrições de tempo e de literatura disponível. A escolha pelo Scrum foi realizada devido à alta popularidade do método na indústria, bem como o foco do mesmo na gerência e monitoramento de projetos que favorece o estudo da utilização de métricas para este propósito.

1.3.2 Fase 2 – Desenvolvimento da proposta

Nesta terceira fase começaram os esforços no desenvolvimento direto dos objetivos deste trabalho de pesquisa. Nesta fase, baseado nos trabalhos encontrados em diferentes trabalhos científicos já disponíveis, foi desenvolvido um plano de métricas para análise e monitoramente das métricas de projetos utilizadores do arcabouço Scrum. Em conjunto com este plano de métricas também foi realizada a proposta de uma modelagem dimensional para o armazenamento destas métricas em um repositório central onde as mesmas possam ser acessadas de maneira interativa, facilitando as práticas de monitoramente e acompanhamento do projeto. Essa interatividade na manipulação de métricas através da utilização de uma modelagem dimensional pode facilitar o trabalho dos gestores de projetos na tomada de decisão baseada nas informações resultantes das métricas dos projetos. Como artefatos nesta fase há o plano de métricas [Spies & Ruiz, 2012], que foi submetido e aceito em um artigo no workshop acadêmico de Métodos Ágeis brasileiro, e também a modelagem dimensional das métricas deste plano conforme descrito na seção 4.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 6 capítulos. Na sequência o capítulo 2 apresenta o referencial teórico da pesquisa, demonstrando os principais conceitos e áreas de estudo relacionados a este trabalho, sendo elas: modelos de processo de software, métodos ágeis, métricas de software, data warehouse e os principais trabalhos relacionados.

No capítulo 3, apresenta-se o plano de métricas para Scrum. Este plano contém as métricas para a realização do acompanhamento e monitoramento das métricas de projeto de organizações que utilizam Scrum. São justificadas neste capítulo as escolhas das métricas e juntamente com suas fontes e formulas.

No capítulo 4 é apresentada a modelagem dimensional utilizada para o armazenamento deste plano de métricas. São explicadas as diferentes tabelas fatos, dimensões e as relações entre elas, bem como os pontos de coleta e as diferentes perspectivas de análise.

O capítulo 5 apresenta a conclusão deste trabalho apresentando as principais considerações, as limitações e possibilidades de trabalho futuro.

2. REFERENCIAL TEÓRICO

Neste capítulo é apresentado o referencial teórico necessário para a compreensão do trabalho desenvolvido. Nele serão discutidos os principais conceitos envolvidos e trabalhos relacionados que deram origem a este trabalho.

2.1 MODELOS DE PROCESSO DE SOFTWARE

Com o advento do mercado de Tecnologia da Informação (TI), empresas desenvolvedoras de software buscam formas de produzir software de maneira produtiva, visando o lucro, qualidade e satisfação do cliente. Estas empresas passaram a se basear em conceitos de outras áreas de produção, como as engenharias, buscando incorporar práticas e processos que melhorassem as chances de realização com sucesso destes projetos de desenvolvimento.

O desenvolvimento de software passou a ser realizado baseado em processos, métodos e ferramentas. E de acordo com os projetos de maior sucesso, as práticas consideradas as mais eficientes [boas práticas da engenharia software] foram sendo descobertas e agrupadas em modelos de processo de desenvolvimento. Sommerville [Sommerville, 2011] define modelos de processos como:

> "*A software process model is a simplified representation of a software process. Each process model represents a process from a particular perspective, and thus provides only partial information about that process.*" [Sommerville, 2011]

Estes modelos de processo de desenvolvimento almejam servir como guias das atividades necessárias para a realização e conclusão com sucesso de um projeto de desenvolvimento de software.

2.2 MÉTODOS ÁGEIS

Ao final dos anos 80 e início dos anos 90 existia uma cultura bastante popular de que a melhor maneira de se obter software de qualidade fosse através de planejamento cuidadoso, garantia de qualidade formalizada, utilização de métodos de análise e *design* através de ferramentas CASE, e através de processos de desenvolvimento de software

rigorosos e controlados. Contudo, este conhecimento era originário de grandes comunidades de engenharia de software que estavam, majoritariamente, associadas a projetos de grande porte como, por exemplo, programa aeroespacial e sistemas governamentais [Sommerville, 2011]. Normalmente este tipo de software era desenvolvido por grandes times distribuídos geograficamente em diversos locais e durante longos períodos de tempo. Nestes ambientes era necessária maior formalização da comunicação através da documentação uma vez que, quanto maior e mais disperso está o time, maiores são as dificuldades para a transferência de informação. Contudo, este método passou a ser utilizado em empresas de pequeno e médio porte, gerando uma sobrecarga tão grande de documentação e controle que estas tarefas dominavam o processo de desenvolvimento de software.

Descontentes com esta realidade, diversos autores dos principais métodos ágeis se reuniram ao final dos anos 90 e propuseram o manifesto ágil de desenvolvimento. Este manifesto reflete a filosofia básica atrás destes métodos de desenvolvimento, demonstrando os princípios, preocupações, valores e objetivos deste grupo. O manifesto proposto foi o seguinte:

"*Nós estamos procurando melhores maneiras de desenvolver software desenvolvendo-o e ajudando outras pessoas a fazerem o mesmo. Através deste trabalho nós começamos a valorizar:*

- *Indivíduos e Interações sobre processos e Ferramentas*
- *Software Funcional sobre documentação compreensiva*
- *Colaboração do Cliente sobre negociação do Contrato*
- *Responder a Mudanças sobre seguir um plano*

Existe valor nos itens presentes à direita das afirmativas, porém, nós valorizamos os itens da esquerda mais [Manifesto for Agile Software Development, 2001]."

Este manifesto ágil de desenvolvimento demonstra as principais prioridades dessa abordagem onde, em cada afirmativa, é priorizado o que está mais a esquerda, mas não quer dizer que os atributos na direita não tenham valor, eles apenas não são o foco da abordagem. Além do manifesto outros doze princípios [Manifesto for Agile Software Development, 2001] são considerados como essenciais aos métodos ágeis. São eles:

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagens das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência a menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar em conjunto por todo o projeto
- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- Software funcionando é a medida primária do progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Contínua atenção à excelência técnica e bom design aumenta a agilidade.
- Simplicidade -- a arte de maximizar o trabalho não realizado -- é essencial.
- As melhores arquiteturas, requisitos e design emergem de equipes auto-organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então ajusta seu comportamento de acordo.

Baseados nestes princípios, métodos ágeis vem ganhando cada vez mais destaque no desenvolvimento de software, especialmente em ambientes coorporativos onde os requisitos são mais instáveis e as mudanças mais freqüentes.

2.2.1 Scrum

O Scrum é um arcabouço adaptativo para a gerência de projetos. Ele consiste em uma abordagem empírica baseada na teoria de controle de processos que tem por objetivo introduzir flexibilidade, adaptabilidade, e produtividade ao desenvolvimento de sistemas [Schwaber & Beedle, 2001]. Este arcabouço busca solucionar o problema da complexidade através de monitoramento frequente e adaptação contínua do processo de desenvolvimento. O fluxo padrão do arcabouço Scrum é descrito na Figura 2 onde pode ser visualizado que o projeto é gerenciado em dois diferentes ciclos, o de *Sprints* e as

reuniões diárias, de modo que a gerência esteja sempre atualizada e adaptada a mudanças de maneira rápida e eficiente.

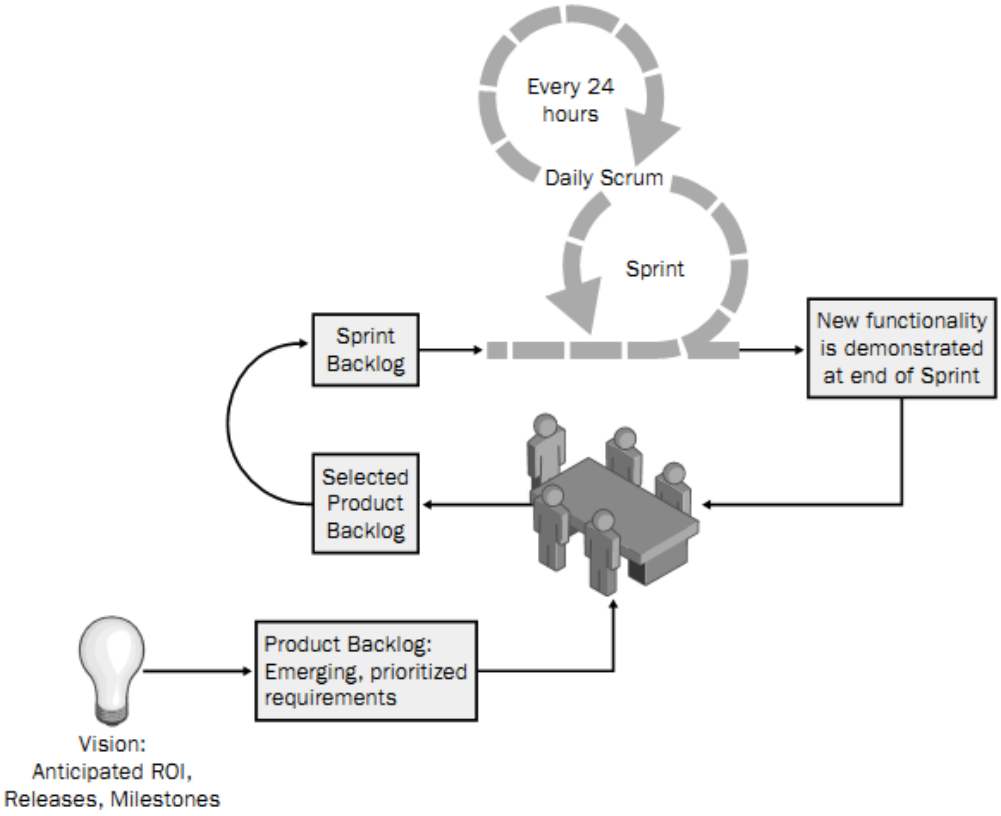

Figura 2 - Arcabouço Scrum [Schwaber, 2004]

Segundo [Schwaber, 2004], um projeto Scrum é iniciado através de uma visão do sistema a ser desenvolvido. Esta visão pode ser vaga inicialmente, ou até mesmo descrita em termos de mercado ao invés de termos de sistemas. Ela vai se tornando mais clara conforme o projeto vai progredindo. A seguir o *Product Owner*, papel típico do Scrum, é responsável pela formulação de um plano que entregue esta visão e, junto com cliente, crie um *Backlog* de Produto. O *Backlog* de Produto é uma lista de todos os requisitos funcionais e não funcionais que, quando transformados em funcionalidades, vão entregar a visão. O *Backlog* de Produto é então priorizado e dividido em diferentes *Sprints*. Esta lista priorizada é um ponto inicial do projeto, e modificações em seu conteúdo, propriedades, e priorização são comuns e refletem a volatilidade no desenvolvimento de sistemas.

Todo o trabalho é executado em *Sprints*. Cada *Sprint* tem uma duração fixa [definida em dias de calendário] e são iniciadas com uma reunião de planejamento, onde o *Product Owner* e o time de desenvolvimento se reúnem para discutir o que será realizado na *Sprint* seguinte. Nesta reunião, o *Product Owner* apresenta os itens de

backlog ao time, que tiram dúvidas quanto ao significado, conteúdo e propósito de cada item, da melhor maneira possível. Ainda na reunião de planejamento, o time avalia a lista de *backlog* e decide o quanto dela é capaz de ser transformado em funcionalidades até o final dessa *Sprint*, e estima as tarefas associadas a cada item as colocando na lista de *backlog* de *Sprint* [Schwaber, 2004].

O restante do monitoramento e acompanhamento do projeto é realizado através de reuniões diárias chamadas de *Daily Scrum*. Nesta reunião cada integrante do time responde a três questões:

1. “O que você fez nesse projeto desde o último *Daily Scrum*?”
2. “O que você pretende fazer até o próximo *Daily Scrum*?”
3. “Quais são os principais impedimentos para que sejam entregues os compromissos para essa *Sprint* e o projeto?”.

As reuniões têm por objetivo monitorar o progresso do time em relação ao objetivo da *Sprint* e avaliar como eles pretendem avançar e trabalhar na direção deste objetivo [Schwaber & Beedle, 2001]. O monitoramento utiliza dois gráficos principais: o *Burndown* de produto e o *Burndown* de *Sprint*. Estes gráficos demonstram a quantidade de trabalho a ser desenvolvida no decorrer do tempo e a capacidade de entrega do time.

2.2.2 Estimativas em Scrum

Dentro de métodos ágeis de desenvolvimento, existe a busca contínua pela relação ideal entre precisão das estimativas e esforço despendido nas mesmas. Desta forma, a maior parte das técnicas utilizadas para a realização das estimativas tenta ser de uma simplicidade suficiente para conseguir estimar dados em um intervalo de precisão adequado reduzindo o esforço nestas atividades. Cohn [Cohn, 2005] apresenta o gráfico da Figura 3 - Relação Precisão X Esforço [Cohn, 2005] sobre a relação da precisão e esforço despendido nas estimativas, onde os métodos ágeis idealizam buscar a parte mais alta da curva.

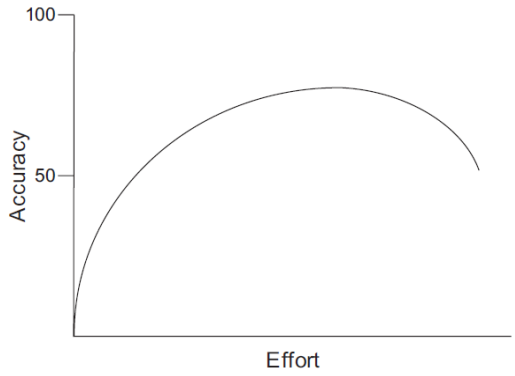

Figura 3 - Relação Precisão X Esforço [Cohn, 2005]

De acordo com [Laird & Brennan, 2007], recomenda-se a utilização de pelo menos três diferentes técnicas para o auxilio na realização das estimativas para realizar a triangulação destas em busca de uma estimativa consistente. Métodos ágeis de desenvolvimento normalmente sugerem (certos métodos exigem), que algumas destas técnicas sejam utilizadas para obtenção de melhores resultados e precisão nestas estimativas. Dentre estas estão: o *planning poker*; a decomposição do sistema; estimativas coletivas; estimativas freqüentes; e a realização das estimativas por quem realiza as tarefas.

Tais exigências por estes métodos e frameworks de desenvolvimento ágeis quanto a utilização de certas práticas e técnicas de estimativas é compatível com a recomendação proposta por [Laird & Brennan, 2007] sobre a utilização de pelo menos três diferentes técnicas para a realização das predições. A utilização destas diversas técnicas é realizada para melhorar a qualidade das estimativas e reduzir o viés de cada uma destas técnicas.

2.2.3 Pequenas *releases* e iterações

Uma das principais características inerentes a métodos ágeis de desenvolvimento é a realização de ciclo de vida iterativo e incremental com o diferencial de que estas iterações sejam de curta duração e tamanho fixo, tendo ao final de cada uma delas um produto passível de utilização do cliente. Esta abordagem tem como objetivo prover um retorno do seu investimento através de funcionalidades entregues iterativamente, e desta forma, suprindo as necessidades funcionais ao longo do progresso do desenvolvimento de software.

Esta prática é considerada por muitos como uma das mais importantes dos métodos ágeis de desenvolvimento uma vez que ela possibilita uma entrega contínua de

valor ao cliente. Esta visibilidade do sistema através da utilização de funcionalidades entregues em curtos intervalos de tempo durante o andamento do projeto possibilita uma melhor tomada de decisão do cliente sobre o que é de fato prioritário, sobre possíveis oportunidades de melhoria do sistema e sobre funcionalidades não mais necessárias de serem implementadas [Cohn, 2005]. Este conhecimento e visibilidade do sistema por parte do cliente melhoram as chances de satisfação do cliente com o sistema uma vez que ele tem maior capacidade de decidir ao longo do tempo o que tem mais valor e ainda provê maiores chances de que o produto entregue alcance as suas expectativas.Pequenas iterações possibilitam uma obtenção mais frequente de *feedback* sobre o andamento do projeto de desenvolvimento tanto pelo lado do cliente como pelo lado do processo. É recomendado, ao se trabalhar com estimativas, que estimem frequentemente e em diferentes granularidades [Laird & Brennan, 2007]. Um dos principais princípios dos métodos ágeis consiste justamente nesta prática. As estimativas são divididas em duas diferentes granularidades, sendo elas: Estimativas de *Release* e estimativas de *Sprint*.

2.2.4 Estimativas de *release*

São estimativas de alto nível que tem por objetivo planejar e estimar as funcionalidades que estarão presentes em determinada *release*. As *releases* têm uma duração sugerida de três a seis meses, onde são estimados os tamanhos das "*User Stories*", através de técnicas como "*Story Points*" ou "*Ideal Days*", é fixado e ajustado o tamanho das iterações a ser desenvolvidas, e é estimada a quantidade de "*User Stories*" a serem desenvolvidas através da velocidade planejada de desenvolvimento.

Estimativas de release têm a sua utilidade em prover predições de alto nível para viabilizar um planejamento de como entregar conteúdo de maior valor em certo período de tempo. Este tipo de estimativa normalmente é bastante relacionado às estimativas de métodos tradicionais nas fases de requisitos de modo a prover uma ideia em nível do usuário das funcionalidades a serem desenvolvidas.

Neste nível são desenvolvidas as atividades de planejamento e estimativas para as funcionalidades a serem desenvolvidas nesta determinada release. Conforme descrito na Figura 4 as etapas do planejamento da *release* são:

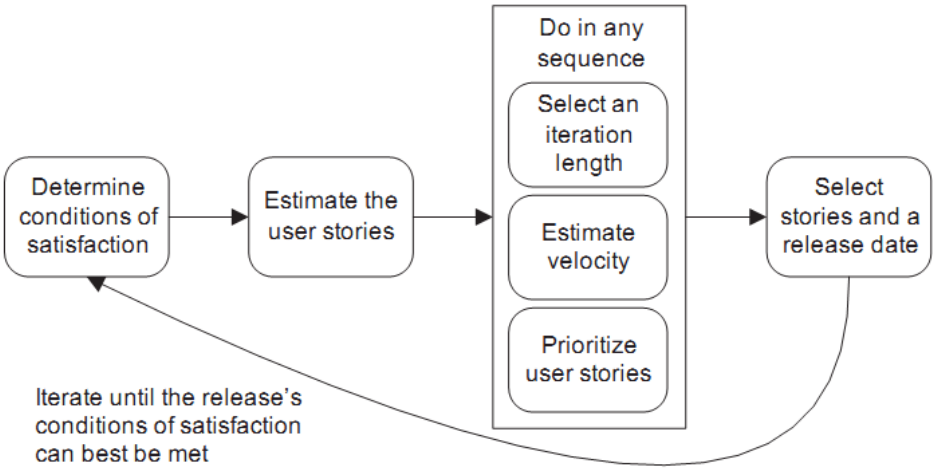

Figura 4 - Planejamento de Release [Cohn, 2005]

- Determinar as condições de satisfação

Esta atividade consiste em estabelecer os critérios pelos quais o projeto será avaliado como um sucesso ou fracasso. Segundo [Cohn, 2005] normalmente os projetos são avaliados pela quantidade de dinheiro gerado ou economizado. Usualmente o cronograma, escopo, recursos são atributos indicadores se estes objetivos estão sendo alcançados ou não. De acordo com estes atributos normalmente os projetos se classificam como orientados a tempo ou a *features*, onde o primeiro tem como necessidade essencial a entrega de um produto em determinado prazo com funcionalidades negociáveis e o segundo tem as funcionalidades como essenciais e o tempo negociável.

- Estimar as *user stories*

São realizadas as estimativas de tamanho das *features* que tem possibilidade de serem inclusas no próximo release. Estas estimativas normalmente são realizadas em conjunto e possivelmente através de *planning poker*.

- Selecionar uma duração para as iterações

Métodos ágeis de desenvolvimento buscam iterações que fiquem em um intervalo de uma a quatro semanas de duração. A duração das iterações é decidida considerando fatores como a necessidade de *feedback*, risco inerente ao projeto, o tamanho da *release*

que se está trabalhando, quanto tempo as prioridades podem se manter estáticas, a sobrecarga inerente às iterações e ainda o desenvolvimento de um senso de urgência.

- Estimar uma velocidade

A velocidade em métodos ágeis de desenvolvimento é a quantidade de pontos de histórias completados por *sprint*. Ela é o principal motor das estimativas de duração e esforço das *releases* e iterações. A velocidade é estimada valendo-se de dados históricos, rodando iterações e avaliando a velocidade no seu desenrolar, ou ainda através de um *forecast* [previsão de alto nível baseado apenas na experiência dos estimadores].

- Priorizar *user stories*

Priorizar as histórias de usuários consiste na prática de estabelecer prioridades às "*User Stories*" de modo a determinar o que será desenvolvido imediatamente. A priorização pode ser tanto por análises financeiras, riscos eliminados pelo desenvolvimento destas *features*, quão essenciais às mesmas são, dependências em relação a outras *features*, entre outros.

- Selecionar as *user stories* e uma data de entrega

Se o projeto for orientado a data de entrega, uma abordagem é verificar a data no calendário, ver quantas iterações se encaixam neste período de tempo e multiplicá-las pela velocidade estimada do time. Desta maneira, se terá uma estimativa de quantos "*Story Points*" podem ser completados neste intervalo e se selecionam as "*User Stories*" de acordo com sua prioridade, de modo a conseguir que seu somatório de pontos seja igual, ou parecido com o estimado.

Quando os projetos forem orientados a *features*, onde certo conjunto de funcionalidades é considerado como essencial, a abordagem para a realização das estimativas é diferente. São somados todos os "*Story Points*" das "*User Stories*" consideradas como essenciais e estes pontos são divididos pela velocidade estimada, de modo a se obter a quantidade de iterações necessárias para o cumprimento das mesmas. Depois, basta ver a duração desta quantidade de iterações em um calendário para achar

a data de entrega.

Os planejamentos das atividades relacionadas às iterações podem ser desenvolvidos nesta etapa, ou em nível de planejamento de iteração. Planejar na etapa de release pode vir a ser mais interessante ao se trabalhar com múltiplos times de modo a conseguir uma melhor divisão e coordenação de trabalho. Planejar em nível de iteração tira proveito máximo do *feedback* de iterações para planejamento mais confiável; entretanto, dificulta a divisão de trabalho ao se lidar com mais de um time de desenvolvimento. As duas abordagens possuem seus benefícios e desvantagens e os projetos devem decidir o quão a frente planejar de modo a melhor adequar a sua realidade de desenvolvimento.

- Revisitar o plano de *release*

Uma vez planejado o conteúdo da *release* é importante que o mesmo não seja esquecido ou abandonado. Uma prática sugerida é que a cada iteração seja revisitado o plano de modo a controlar o seu progresso viabilizando a tomada de ações corretivas quando desvios significativos são encontrados.

2.2.5 Estimativas de iteração

Planos de iteração têm por objetivo serem especificações mais detalhadas do trabalho a ser desenvolvido. Estimativas de iteração já se assemelham mais às estimativas realizadas nas fases de *design* de métodos tradicionais onde se tem um maior detalhamento das atividades necessárias para entregar ao cliente as funcionalidades requeridas.

Nesta etapa as *User Stories* são quebradas em atividades e cada uma delas é estimada em horas ideais de trabalho durante a reunião de iteração que acontece a cada início de iteração. Nesta etapa, assim como em todas as outras, as estimativas são realizadas por todos os desenvolvedores do time e de diferentes papéis. As atividades nesta etapa não são atribuídas a ninguém: os indivíduos responsáveis pelo desenvolvimento das atividades são controlados diariamente através de pequenas reuniões, onde cada pessoa dá o *feedback* de suas atividades e escolhe novas para desenvolver.

Uma abordagem sugerida por [Cohn, 2005] para o planejamento das iterações é demonstrado na Figura 5, onde é descrito o fluxo de atividades relacionadas a esta etapa

do planejamento. Dentro deste fluxo as atividades compreendidas são:

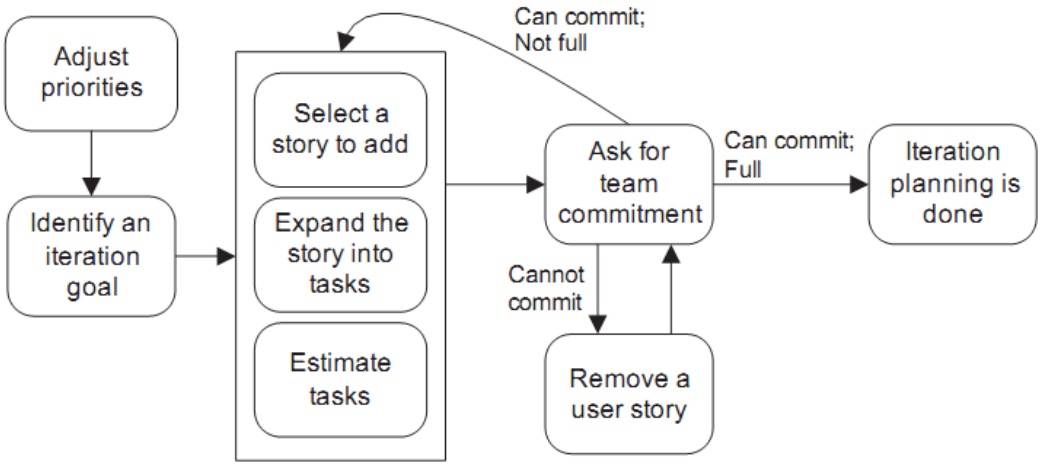

Figura 5 - Planejamento de Iteração [Cohn, 2005]

- Ajustar prioridades

Nesta etapa usualmente é revisitada a pilha de histórias a serem desenvolvidas em casos de mudanças sobre as prioridades de execução. Por exemplo, durante o planejamento da *release* é dado um valor de prioridade a cada história, de modo a definir uma provável ordem de execução das mesmas. Contudo, de acordo com o andamento do projeto, certas mudanças de prioridade podem vir a ocorrer, devido a diversos acontecimentos possíveis. Desta forma, nesta etapa do planejamento da iteração, é revisitada a pilha de histórias existindo a necessidade de alterar a ordem de execução das histórias.

- Identificar o objetivo da iteração

Consiste em identificar um objetivo bastante genérico do que se quer ser alcançar nesta iteração. Alguns exemplos básicos de objetivos de iterações são: trabalhar na segurança do sistema, fazer progresso na geração de relatórios, terminar relatórios de tempo, entre outros.

- Escolher uma história para ser adicionada

É escolhida uma das histórias que tem uma maior prioridade e que seja compatível

com o objetivo definido para a iteração. Esta história então passa a fazer parte do backlog de histórias que fundamentalmente irá compor uma das funcionalidades previstas para a iteração atual.

- Expandir a história em tarefas

Identificar todas as tarefas relacionadas à realização e resolução da funcionalidade desejada, compreendendo atividades de todos os diferentes papéis envolvidos de modo que a história seja finalizada em todos os seus aspectos, desde a documentação necessária ao cliente, até os testes a serem executados sobre ela.

- Estimar as tarefas

Assim como as estimativas de tamanho das histórias são realizadas pelo time [através do *planning poker* dependendo da metodologia adotada], as estimativas de tamanho de tarefas também as são. A cada tarefa é atribuído um tamanho em horas ideais de trabalho, que são horas necessárias para completar a tarefa desconsiderando outras atividades. As tarefas devem idealmente ser divididas em um nível que cada atividade leve em média um dia de trabalho para ser completada, caso certas atividades sejam maiores pode ser que a mesma possa ser dividida em outras atividades menores.

- Pedir por confirmação

Após a história ter sido devidamente estimada em tarefas, e suas tarefas terem seu tamanho estimado em horas ideais de trabalho, se obtêm a confirmação do time quanto à capacidade de execução deste conjunto de histórias e tarefas durante esta iteração. O time vai selecionando e estimando iterativamente cada uma das histórias até chegar a um momento em que o time não pode mais se comprometer a aceitar atividades. Uma abordagem naturalmente utilizada para saber até onde um time pode se comprometer é realizar um somatório das atividades e tentar encontrar um número viável que se encaixe com a jornada de trabalho para a iteração.

2.2.6 *Planning poker*

O *Planning poker* é uma técnica que vem sendo utilizada em métodos ágeis para a

realização das estimativas. O *planning poker* combina a opinião de *experts*, analogia, e desagregação (estimativas realizadas por diversas pessoas ao invés de uma só) em uma abordagem que seria supostamente mais “agradável” de estimar, em virtude de metaforizar a prática da realização das estimativas com um jogo de cartas, e que resulte em estimativas rápidas porém confiáveis [Cohn, 2005].

Os participantes no *planning poker* incluem todos os desenvolvedores do time, em times de métodos ágeis o padrão é de 7 a 10 integrantes. Caso o time for superior a isso é melhor separar o grupo em múltiplos times que sigam este padrão. O *Product Owner* (Cliente) participa no *planning poker*, porém não ajuda a estimar. [Cohn, 2005]

O *Planning poker* consiste em entregar a cada integrante da estimativa um baralho contendo cartas que apresentem os valores 0, 1, 2, 3, 5, 8, 13, 20, 40 e 100. O moderador lê a descrição da história de usuário e cada estimador saca uma carta com o valor de tamanho que ele acha melhor representar a descrição da história (pode ser ou em *Story Points*, ou Dias ideais de trabalho, conforme definido no projeto em questão). Nesta etapa, é natural que os valores presentes nas estimativas difiram bastante. Desta forma, os estimadores responsáveis pelo valor mais baixo e pelo valor mais alto explicam às razões pela qual estimaram tal tamanho à história e, após a explicação, seguem para uma nova rodada, onde cada um recolhe as cartas. A descrição é relida e novamente cada um é solicitado que apresente a sua estimativa. Nesta segunda rodada de estimativas, é esperado que os valores comecem a se aproximar, caso não se entre em consenso este passo é repetido mais uma ou duas vezes até que se tenha um valor sobre o qual os estimadores entrem em acordo.

2.2.7 *User stories*

Métodos ágeis realizam as medições do tamanho de software através da atribuição de pesos relativos as suas *User Stories.* As *User Stories* são descrições dos requisitos e funcionalidades que serão implementadas no software. Elas normalmente são descritas em linguagem natural e visam ser bastante simples.

Esta valoração de pesos às *User Stories* através dos *“Story Points” funciona* da mesma maneira que os pontos de função, pontos de caso de uso e *Feature Points.* Consistem em uma medição de *proxy* do tamanho da aplicação sendo construída.

User Stories são descrições bastante simples e abstratas da funcionalidade de software. Elas são criadas pelo *product owner* (que é um papel em métodos ágeis que representa o cliente como um membro da equipe) e representam a visão do que é de

valor, tanto para o usuário como para o comprador do software. De acordo com [Cohn, 2004] as *User stories* são divididas em três diferentes aspectos:

- Uma descrição da *User Story.* Utilizada no planejamento e também como um lembrete.
- Conversas sobre a *User Story.* Acontecem nas reuniões das estimativas e tem o propósito de dar um corpo de detalhes a ela [de acordo com os princípios dos métodos ágeis de priorizar o conhecimento tácito ao explícito].
- Testes sobre a *User Story*. Apresentam um meio de descrever os detalhes e determinar quando a *Story* foi completada.

O processo de coleta das *User Stories* consiste na realização de um *brainstorming*sobre as possíveis *User Stories* a serem implementadas, visando alcançar uma descrição satisfatória das funcionalidades do projeto a ser desenvolvido. Algumas das *User Stories* criadas podem representar a implementação da maior parte do sistema. Essas histórias são classificadas com histórias épicas ou temas e serão subdivididas em histórias menores quando for considerada a sua implementação no planejamento da *release*. Segundo [Cohn, 2004] o tamanho ideal para as *User Stories,* é um que as descreva de tal modo, que elas possam ser completamente desenvolvidas e testadas por um par de programadores, dentro de um período de um dia até duas semanas. Um exemplo de *User Story* é demonstrado na Figura 6.

Figura 6 - User Story [Cohn, 2005]

As *User Stories* são descrições em alto nível das funcionalidades do sistema, e são utilizadas como um parâmetro de entrada para descrever o tamanho do software, tanto através de *Story Points* ou de Ideal *Days of Work.*

- Story points como unidade de tamanho

A utilização de *story points* como unidade de medida, é assim como pontos por

função, uma técnica de *proxy* para estimar o tamanho do software. *Story points* são medidas relativas, ou seja, o seu valor bruto não possui significado nenhum. De acordo com Cohn [Cohn, 2004] as medições de *Story Points* devem apresentar valores relativos de tamanho, de modo que, uma história avaliada com um ponto tenha um esforço previsto de metade de uma que tenha uma avaliação de dois pontos, e que a história de dois pontos apresente 2/3 do esforço em relação a uma história de três pontos e assim sucessivamente.

De maneira geral, a classificação do tamanho utilizando pontos de histórias consiste em valorar as histórias de usuários com um único valor relativo, que compreende considerações sobre aspectos como tamanho, dificuldade de implementação, complexidade e risco, onde o mais importante não é a valoração bruta, e sim a comparação de tamanho relativo entre as *User Stories*.

Segundo Cohn [Cohn, 2005] uma maneira de auxiliar o processo de valoração das histórias nestas escalas consiste em, ao considerar a pilha de histórias a ser implementada nesta iteração, escolher inicialmente três diferentes histórias para serem valoradas antes das outras. A primeira sendo a menor história dentro desta pilha, a segunda uma considerada de tamanho médio entre todas, e a terceira a maior de todas, valorando estas três histórias e depois todas as seguintes comparando relativamente o seu tamanho a essas três iniciais até que todas as histórias sejam estimadas.

- *Ideal days of work*

Outra técnica de medição de tamanho de software em metodologias ágeis de desenvolvimento é a utilização de dias ideais de trabalho. Dias ideais de trabalho consistem em atribuir uma quantidade de tempo que certa história de usuário levaria para ser devidamente codificada e testada, considerando que não existiriam interrupções. A Figura 7 apresenta os fatores que segundo Cohn [Cohn, 2004] afetam o tempo ideal de trabalho.

Tabela 1 - Fatores que afetam o tempo ideal [Cohn, 2004]

Supporting the current release	Training
Sick time	Email
Meetings	Reviews and walk-throughs
Demonstrations	Interviewing candidates
Personnel issues	Task switching
Phone calls	Bug fixing in current releases
Special projects	Management reviews

A ideia de estimar o tamanho utilizando dias ideais de trabalho considera que fatores que afetam as durações ideais devem ser desconsiderados. E estas estimativas devem utilizar esse valor, em dias ideais de trabalho, como uma unidade de tamanho da funcionalidade, da mesma maneira que funcionam os pontos de histórias.

Segundo o Cohn [Cohn, 2004], a mensuração através de dias ideais de trabalho é mais intuitiva e de melhor compreensão. Contudo, não é uma prática ideal, pois uma vez a tarefa estimada em dias ideais de trabalho, pessoas tendem a aproximar cada vez mais os dias ideais com os dias de calendário, o que leva a desenvolvimento errôneo de cronogramas.

2.2.8 *Team velocity*

Métodos ágeis de desenvolvimento realizam suas estimativas e medições de tamanhos em duas unidades relativas, os pontos de histórias e os dias ideais de trabalho. Ambas as medidas são valores relativos de medição que atuam como um *Proxy* para mensurar o tamanho das atividades em métodos ágeis.

A velocidade do time é uma das principais métricas utilizadas em métodos ágeis de desenvolvimento. Ela é utilizada extensamente para derivar diversas outras métricas como, por exemplo, o esforço e a duração.

Basicamente, a velocidade consiste na taxa de progresso do time durante a iteração. Dada uma iteração para a qual foi planejada a realização de cinco diferentes histórias de usuários totalizando 20 pontos a velocidade do time será 20 caso consigam terminar todas as histórias durante esta iteração.

Como maior parte do planejamento do projeto em métodos ágeis é realizada através da velocidade, estimativas, como a duração, é diretamente proporcional a velocidade do

time. Desta forma, para que seja possível estimar a duração do projeto é antes necessário possuir em mãos as estimativas de tamanho e de velocidade. O tamanho pode ser estimado através do *Planning Poker,* aonde os desenvolvedores vão atribuindo valores relativos de modo a conseguir criar uma unidade de medida do tamanho de cada história. Para a velocidade são propostas três diferentes abordagens para a realização inicial de suas estimativas.

- Utilização de valores históricos

A utilização de valores históricos é uma eficiente abordagem para a realização de estimativas de velocidade, entretanto, a sua eficiência é diretamente proporcional a similaridade do projeto atual com os projetos presentes no repositório de histórico. Quanto maior a similaridade maior a precisão nas estimativas, Mike Cohn [Cohn, 2005] realiza avaliação dos projetos no repositório através das seguintes questões:

- A tecnologia é a mesma?
- O domínio é o mesmo?
- O time é o mesmo?
- O Cliente é o mesmo?
- As ferramentas são as mesmas?
- O ambiente de trabalho é o mesmo?
- As estimativas estão sendo realizadas pelas mesmas pessoas?

Quando trabalhando em diferentes *releases* de um mesmo projeto a resposta destas perguntas normalmente será “sim” o que torna esta abordagem bastante eficiente neste tipo de situação. E mesmo assim, é recomendado expressar a estimativa de velocidade em intervalos, possivelmente variando da menor velocidade obtida na *release* anterior para a maior velocidade obtida nesta mesma release.

- Realizar uma iteração

Ainda segundo Cohn [Cohn, 2005] uma das melhores maneiras para a realização de estimativas de velocidade é a realização de uma iteração (ou duas, ou três) e ir

atualizando as estimativas das iterações futuras com o valor médio das iterações anteriores. Esta é a abordagem de estimativas de velocidade mais recomendada uma vez que a mesma será baseada no *feedback* de cada iteração demonstrando a evolução do time e melhorando a sua precisão a cada iteração. Contudo, esta abordagem pode não ser viável se o cliente exigir uma estimativa de duração inicial para o projeto. Porém, ainda é uma excelente abordagem para as re-estimativas da velocidade assim que o projeto foi iniciado.

Outra prática utilizada para melhorar a precisão das estimativas de velocidade sugerida por Cohn [Cohn, 2005] é acompanhar o cone de incerteza e ir diminuindo o intervalo associado à estimativa, a cada iteração realizada. Esta técnica não possui cálculos matemáticos que a valide. Entretanto, segundo Cohn, é uma abordagem válida uma vez que a incerteza proposta no cone também vai diminuindo de acordo com *feedback* recebido durante a execução do projeto, do mesmo modo funcionaria para métodos ágeis de desenvolvimento. A idéia seria multiplicar a velocidade estimada por 0,6 e 1,6 em um primeiro momento para estimar a variação da velocidade da estimativa inicial, realizar a iteração, e estimar a velocidade média multiplicada por 0,8 e 1,25 na segunda, e assim sucessivamente, conforme os valores demonstrados na tabela seguinte.

Tabela 2 - Intervalos de Incerteza

Iterations Completed	Low Multiplier	High Multiplier
1	0.6	1.60
2	0.8	1.25
3	0.85	1.15
4 or more	0.90	1.10

- Realizar uma previsão

A terceira e última abordagem consiste na realização de uma previsão da velocidade, onde nem se possui valores históricos disponíveis e nem é possível realizar uma iteração para avaliar a sua velocidade. Esta é provavelmente uma das ultimas abordagens a ser utilizada para estimar a velocidade. Entretanto, para casos onde não se possui outra solução, ela é uma abordagem válida.

A realização da previsão da velocidade consiste essencialmente na realização do

planejamento de uma iteração, onde são estimadas todas as histórias de usuário a serem desenvolvidas, e onde as mesmas ainda são quebradas em todas as tarefas que as compõem e assumir que esta iteração com estas histórias foram realizadas com sucesso, e a velocidade seria o número de pontos de histórias supostamente completados durante a iteração.

2.2.9 Gráficos de *burndown*

Os gráficos de *burndown* consistem no método mais utilizado de monitoramento de projeto em métodos ágeis de desenvolvimento. Conforme demonstrado na Figura 8 no eixo vertical é demonstrado o total de pontos de história a serem implementados e no eixo horizontal a quantidade de iterações do projeto. Dada esta estrutura este gráfico é atualizado a cada iteração do projeto de modo que seja possível visualizar graficamente o progresso relativo do projeto de desenvolvimento de software.

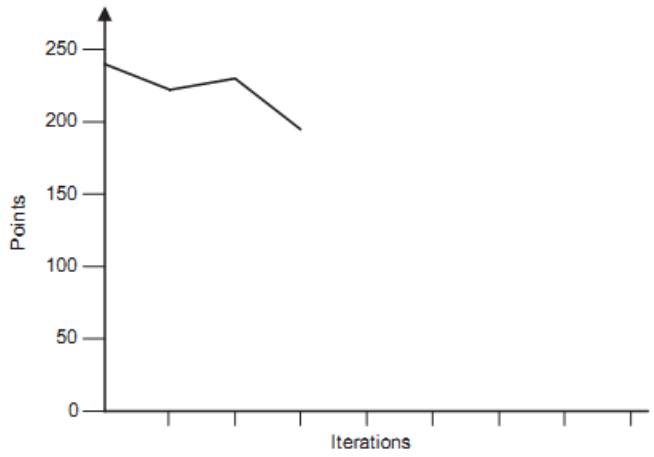

Figura 7 - Gráficos de Burndown [Cohn, 2005]

Em casos como a terceira iteração da Figura 7 em que a quantidade de trabalho aumenta ao invés de diminuir é um indicador ou de que certa atividade foi re-estimada para um valor maior de trabalho, ou ainda, que mais histórias foram adicionadas ao plano de *release* do projeto.

Outro tipo de gráfico de *burndown* utilizado em métodos ágeis é o gráfico de *burndown* com barras. Neste gráfico representado pela Figura 8 cada barra representa uma iteração e as modificações sobre a quantidade de trabalho a ser realizada se tornam mais claras.

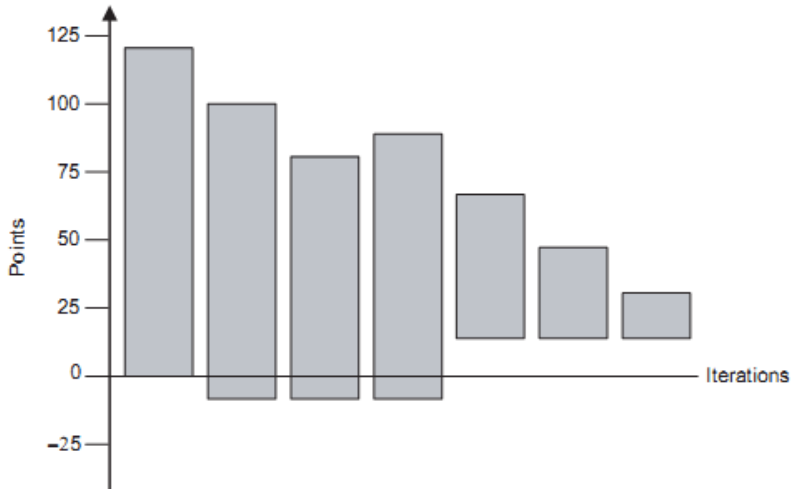

Figura 8 - Gráfico de Burndown de barras [Cohn, 2005]

Neste gráfico de barras a cada vez que a barra cresce abaixo do 0 no eixo horizontal representa que novas histórias de usuário foram adicionadas ao plano de *release*, cada diminuição da parte superior da barra consiste no desenvolvimento com sucesso de certas histórias durante a iteração, e ainda, as diminuições pela parte inferior da barra caracterizam a remoção de histórias de usuário do plano da *release* para o seu desenvolvimento em outra iteração caso seja a vontade do cliente.

Outra utilização dos gráficos de *burndown* em métodos ágeis é para a monitoração da iteração em andamento. Diferente do *burndown* de *release* este gráfico (Figura 10) monitora as horas das tarefas a serem executadas em determinada iteração, e cada dia de trabalho estas horas são subtraídas do total de horas estimado, de modo que ao chegar ao final da iteração todas as horas das tarefas planejadas sejam completadas.

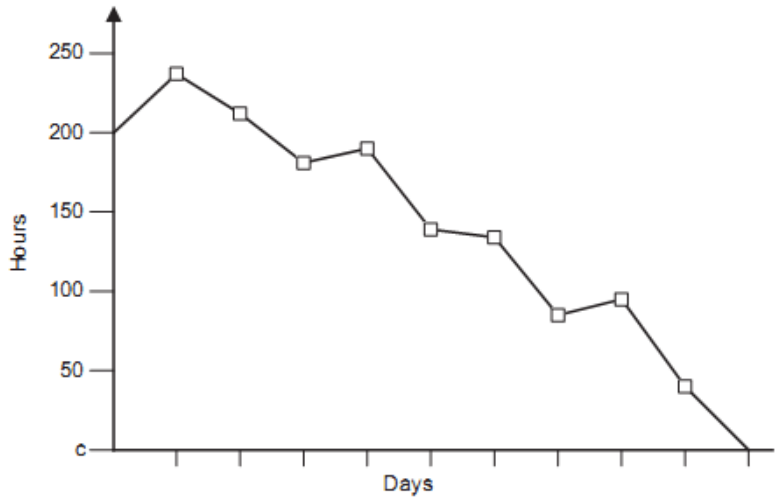

Figura 9 - Burndown de Iteração [Cohn, 2005]

Através deste gráfico e das reuniões diárias pregadas pelos métodos ágeis se possui uma visualização e atualização bastante freqüente do progresso da iteração planejada de modo a melhorar a visualização de possíveis desvios e tomada de ações corretivas.

Outra alternativa que pode ser utilizada é a utilização de gráficos de *Burn up* (Figura 10) que são basicamente equivalentes aos de *burndown* porém representam uma visão de contagem a 100% do trabalho total de maneira a ser mais similar a métodos tradicionais de monitoramento como os gráficos de *Earned Value Management* (EVM).

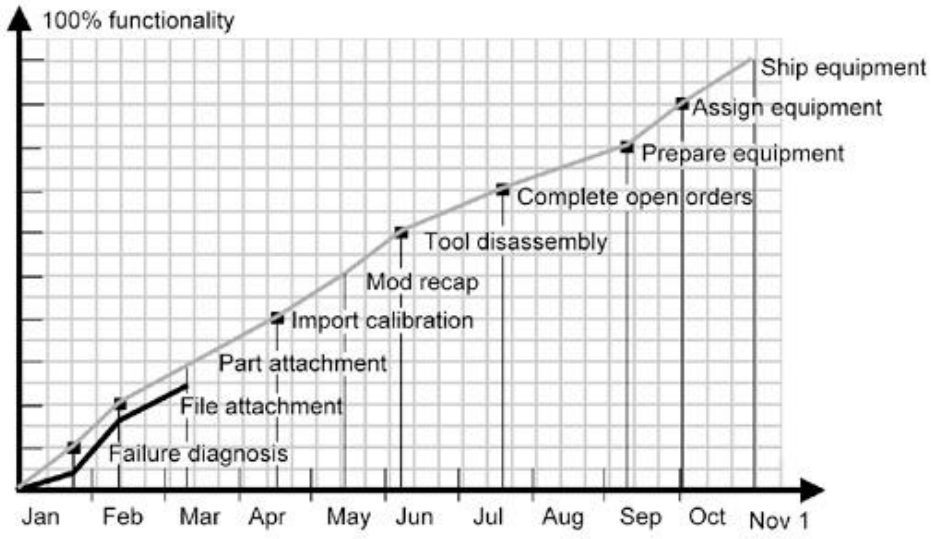

Figura 10 - Gráfico de Burn Up [Cockburn, 2004]

2.3 MÉTRICAS DE SOFTWARE

De acordo com o CMMI [SEI, 2010] o objetivo da mensuração e análise de métricas de software é desenvolver e manter uma capacidade mensuração usada para dar suporte às necessidades da gerência de informação. Este processo de medição e análise envolve:

- Especificar os objetivos de medição e análise, de forma que estejam alinhados com as necessidades de informação e objetivos identificados;
- Especificar medidas, técnicas de análise e mecanismos para coleta e armazenamento de dados, e formas de relato e de *feedback*;
- Implementar coleta, armazenamento, análise e relato de dados;
- Fornecer resultados objetivos que possam ser utilizados em tomadas de decisões bem fundamentadas e na implementação de ações corretivas apropriadas.

A implementação de um processo de coleta e análise de métricas possibilita a realização de estimativas de forma objetiva; acompanhar o desempenho em relação aos planos e objetivos estabelecido; identificar e tratar questões críticas relacionadas a processos e ainda fornece uma base para incorporação futura de medições em outros processos.

De acordo com [Laird & Brennan, 2007] outro fator importante relacionado à prática da realização de mensurações é que o fato de simplesmente medir certos aspectos causa uma mudança comportamental dos envolvidos no processo sendo medido. Se você começar e avaliar a produtividade das pessoas pela quantidade de linhas de código escrita, sistemas cada vez maiores serão criados, se você considerar como produtividade a ausência de defeitos, o número de defeitos será reduzido. Esta característica é chamada de efeito Hawthorne [Laird & Brennan, 2007] que diz que se você prestar atenção em algo e realizar medições sobre isso, melhorias serão alcançadas.

2.3.1 Propriedades das Métricas

Segundo Kan [Kan, 2002] as métricas podem ser divididas em três diferentes categorias:

- Métricas de processo, que compreendem dados que tem respeito à execução do processo e seu controle (e.g. Eficiência da remoção de defeitos durante o desenvolvimento, o padrão de descoberta de defeitos dentro da fase de testes, e tempo de resposta e correção de defeitos);

- Métricas de produto que caracterizam as propriedades inerentes ao produto desenvolvido (e.g. Tamanho, complexidade, qualidade, nível de desempenho);

- Métricas de projeto que compreendem as características intrínsecas ao projeto sendo desenvolvido (e.g. Cronogramas, esforço, equipe, duração). Estes tipos de métricas não são mutuamente exclusivos o que implica em que uma métrica pode pertencer simultaneamente a mais de uma dessas categorias.

Existem diferentes padrões estabelecidos sobre a coleta de métricas de desenvolvimento de software. Um dos padrões mais conhecidos é o modelo de maturidade do SEI [SEI, 2010] que exige que métricas de tamanho de sistema, duração de projetos, níveis de esforço e defeitos de software, entre outros sejam coletadas em ordem para o processo ser considerado de certa maturidade. Laird [Laird & Brennan, 2007] ainda reitera a importância da coleta destas métricas em conjunto com métricas de produtividade como um conjunto mínimo de métricas a serem coletadas em qualquer organização.

Ainda segundo Kan [Kan, 2002] quando falando sobre métricas e mensurações, é necessário a compreensão de três conceitos sobre a natureza da prática da mensuração. Estes conceitos são respectivamente a definição, definição operacional e a medição. A primeira representa o conhecimento em forma de conceito, por exemplo, uma linha é sempre uma linha e um ponto é sempre um ponto, e são características primitivas e bem compreendidas. Definições operacionais são definições que já declaram quais são suas métricas e seus processos de coleta (e.g. uma métrica de peso de uma pessoa vai indicar como o mesmo deve ser coletado, a unidade de medida relacionada, os instrumentos e ferramentas necessárias para medi-lo).

Mais uma característica inerente às métricas é a sua escala. Métricas podem ser apresentadas em escala nominal, ordinal, intervalar, e razão. Definições operacionais compreendem métricas que podem ser medidas em certas escalas; em alguns casos, mais de uma escala pode se aplicar a mesma definição operacional; em outros casos, a natureza da definição já define a escala da métrica.

2.3.2 Considerações

A coleta de métricas é fundamental para a obtenção de conhecimento sobre processos de desenvolvimento de software. As definições das métricas a serem coletadas e analisadas nos possibilitam conseguir dados sobre determinados aspectos e através de

suas medições viabilizam análises de causa e conseqüência sobre este aspecto de modo que seja possível analisar quantitativamente o impacto de certas modificações e procurando outras formas de melhorar a eficiência de tal aspecto.

2.4 *DATA WAREHOUSING*

Com o advento da TI (Tecnologia da Informação) volumes cada vez maiores de dados estão se tornando disponíveis para diversos tipos de empresas e negócios. Baseado nesse volume de dados uma abordagem mais eficiente para a consulta destes dados precisou ser criada de modo que os tomadores de decisões pudessem se valer desta grande quantidade de dados disponíveis de maneira rápida e eficiente para auxiliar na tomada de decisão. Baseado nessa demanda, ambientes de DW (*Data Warehouse*) passaram a ser utilizados em cada vez mais empresas de diferentes portes [Kimball & Ross, 2002].

Um *Data Mart* consiste em uma base de dados para armazenar um único processo de negócio ou um grupo de processos de negócio voltados a alcançar determinado objetivo. Nesse sentido, um DW é definido como a união de todos os *Data Marts* de uma organização [Kimball & Ross, 2002].

Kimball descreve os principais objetivos de um ambiente de DW como sendo:

- O DW deve tornar fácil o acesso às informações da organização.
- O DW deve apresentar as informações da organização de forma consistente.
- O DW deve ser adaptativo e resistente a mudanças.
- O DW deve ser um bastião de segurança que protege os recursos de informações.
- O DW deve servir como fundação para uma melhoria da tomada de decisões
- A comunidade de negócio precisa aceitar o DW para que ele tenha sucesso.

Os principais componentes de um ambiente de DW são descritos na Figura 11. Nesta figura Kimball [Kimball & Ross, 2002] separa os componentes de um ambiente de DW em Sistemas Operacionais Fontes; Data Staging Area (DSA); Area de apresentação de dados, e ferramentas de acesso aos dados.

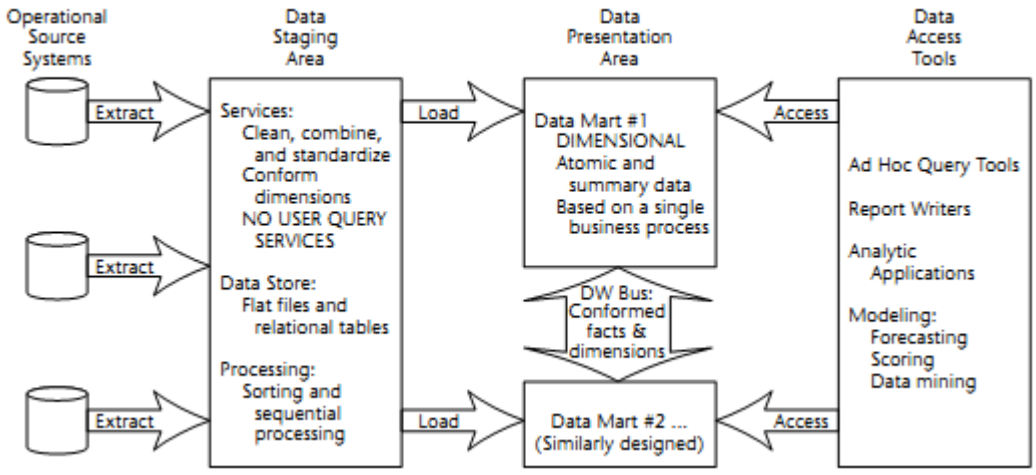

Figura 11 - Componentes DW

2.4.1 Sistemas operacionais fontes

São os sistemas operacionais que contém as fontes de dados utilizados pela organização. Estes dados em geral são armazenados em estruturas de dados relacionais como bancos de dados tradicionais ou ferramentas específicas como planilhas e arquivos de texto. Esses sistemas em geral são considerados como externos ao ambiente de DW por poderem estar atrelados a sistemas legados operacionais de armazenamento de dados servindo apenas como a fonte para extração dos dados que depois serão mais adequadamente manejados de forma a facilitar o acesso e consulta as informações necessárias à organização.

2.4.2 *Data staging area*

O DSA do ambiente de DW é tanto uma área de armazenamento de dados quanto um conjunto de processos usualmente referenciados como ETC (Extração – Transformação – Carga) [Kimball & Ross, 2002]. O DSA engloba todas as etapas desde a extração dos dados de suas fontes brutas até a preparação dos dados para a camada de apresentação. O primeiro passo do processo é pegar os dados de suas fontes e começar a passá-los para o ambiente de DW. Esse processo envolve ler e compreender os dados nas suas fontes e começar a sua transação para o ambiente de DSA para maiores manipulações. Uma vez que os dados estão no DSA diversas operações de limpeza devem ser realizadas (resolver erros de digitação em entradas, erros de domínio, dados faltantes) de modo que os dados restantes sejam consistentes e passíveis de demonstração na camada de apresentação.

2.4.3 Área de apresentação dos dados

Nesta área os dados são organizados, armazenados e tornados disponíveis para consultas diretas através de usuários, criadores de relatórios, e outras aplicações analíticas [Kimball & Ross, 2002]. Essa camada é tudo que a comunidade de negócios vê e interage através de ferramentas de acesso a dados. Esta organização é realizada através do conceito da modelagem dimensional, que é considerada a forma mais viável de entregar dados aos usuários de DW [Kimball & Ross, 2002].

A modelagem dimensional é bastante diferente da abordagem tradicional de modelagem de banco de dados usualmente conhecida como terceira forma normal ou modelos de entidade-relacionamento. A abordagem do modelo entidade relacionamento tem um foco na redução de redundância de dados. Essa abordagem é de grande utilidade em atividades de adição ou atualização de transições, pois os dados só precisam ser tocados em um único local na base de dados. Contudo, essa abordagem é demasiadamente complexa para consultas de DW, onde as consultas, devido a sua complexidade, sobrecarregam os sistemas da base de dados resultando em um desempenho desastroso [Kimball & Ross, 2002]. A modelagem dimensional endereça o problema da complexidade de esquemas de consulta utilizando as mesmas informações que um sistema normalizado, mas agrupando os dados em pacotes onde os objetivos são entendimento do usuário; desempenho de consultas e resistência a mudanças.

Recursos OLAP (*On Line Analytical Process*) são utilizados para a realização de consultas em bases de dados dimensionais tornando a apresentação dos resultados, textuais ou numéricos de maneira mais clara, considerando as diferentes perspectivas de análise e níveis de sumarização. Esses recursos possibilitam que o usuário possa utilizar os dados através de tabelas pivotantes possibilitando uma apresentação tabular e realização de operações OLAP de *drill-up* e *drill down*. Operações que quando implementadas mesclam atributos hierárquicos ou não de todas as dimensões disponíveis.

2.4.4 Ferramentas de acesso aos dados

Essa última camada de um ambiente de DW é responsável a prover ao usuário diferentes formas de realização de consulta e exibição de resultados da maneira mais

prática possível. Em geral, nessa camada são apresentados gráficos baseados em consultas realizados na base de dados dimensional criada na camada de apresentação de modo a facilitar a compreensão dessas informações e ajudar no processo de tomada de decisão.

2.4.5 Modelagem dimensional

Nesta seção serão explicados os principais componentes da modelagem dimensional, suas funções e organização.

- Tabelas Fato – As tabelas fatos são as principais tabelas em uma modelagem dimensional. Nestas tabelas as métricas de desempenho numérico do negócio devem ser armazenadas. O termo “Fato” refere-se a medida de negócio e está relacionada a uma intersecção de diferentes dimensões onde a lista de dimensões define a granularidade da tabela fato e nos diz qual é o escopo da métrica. A Figura 13 descreve um exemplo de uma tabela fato.

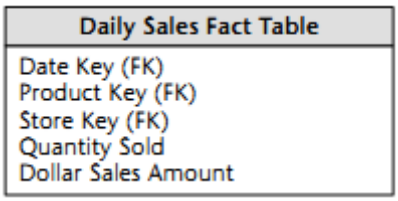

Figura 12 - Tabela Fato Exemplo [Kimball & Ross, 2002]

- Tabelas Dimensões – As tabelas de dimensões contêm as descrições textuais do negócio. Usualmente as mesmas possuam diversas colunas e atributos que são utilizados como as restrições de consultas, agrupamentos e requisições de relatórios. Os atributos das tabelas de dimensões têm um papel fundamental em um ambiente de DW uma vez que são responsáveis por fazer o DW utilizável e compreensível. A figura 14 demonstra um exemplo de uma tabela dimensão.

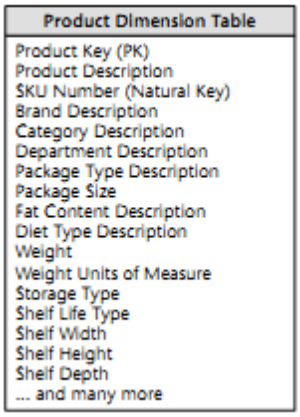

Figura 13 - Tabela Dimensão Exemplo [Kimball & Ross, 2002]

- Modelo dimensional analítico - Os modelos dimensionais analíticos estão relacionados à idéia da representação dos dados em formas de cubos, onde as células tem valores das métricas e as arestas são as dimensões dos dados. Os principais tipos de modelos dimensionais analíticos são o modelo estrela, caracterizado por conter uma única tabela fato relacionada a várias dimensões; O modelo floco de neve, que admite que as tabelas dimensões possuam outros relacionamentos; E, o Modelo Constelação de fatos onde podem existir mais de um fato conectados cada um as suas dimensões podendo compartilhar algumas delas. A Figura 15 demonstra como seria um modelo dimensional estrela.

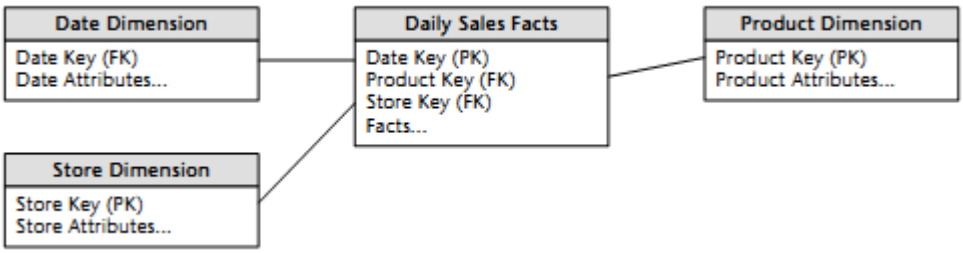

Figura 14 - Exemplo Modelo Estrela [Kimball & Ross, 2002]

2.5 TRABALHOS RELACIONADOS

2.5.1 Métricas e métodos ágeis

Métricas de software em métodos ágeis vêm sendo estudadas recentemente [Kunz, Dumke & Schmietendorf, 2007] e sua importância é cada vez mais reconhecida. A utilização de métricas possibilita a análise concisa da situação atual do projeto, a tomada

de decisão baseada em atributos concretos e a criação de um repositório de métricas para comparações com projetos futuros.

A qualidade do produto é um princípio considerado essencial ao Scrum e, devido a este fator, métricas de qualidade do produto começam a se tornar necessárias. Métricas de código para métodos ágeis foram estudadas [Sato, Goldman, & Kon, 2007], [Ambu et al., 2006] e consideradas como um bom indicador da qualidade do código. Já Kunz, Dumke, & Zenker [Kunz, Dumke & Zenker, 2008] propuseram uma ferramenta para a coleta de métricas de produto integrada à IDE de desenvolvimento Eclipse. Alshayeb & Li [Alshayeb & Li, 2005] propôs a métrica *System design instability* e Knoernschild [Knoernschild, 2006] propôs métricas de qualidade e *design* de código. Quanto a métricas de processo, Jeffries propôs a *running tested features* [Jeffries, 2004], Jakobsen [Jakobsen & Sutherland, 2009] propôs o acompanhamento do tempo de resposta a uma *build* falha e Mahnic [Mahnic & Zabkar, 2008] propôs o número de impedimentos levantados durante as reuniões diárias.

Já no âmbito das métricas de projeto, Fuqua [Fuqua, 2003] realizou a avaliação do esforço através de pontos por função em projetos XP e considerou a técnica inadequada aos princípios ágeis de desenvolvimento. Por definição, o Scrum mede o esforço através da quantidade de horas de esforço restantes às tarefas e itens de *backlog* [Schwaber & Beedle, 2001]. Mike Cohn propôs o planejamento de esforço através de pontos de histórias de usuário e velocidade, e o monitoramento através de gráficos de *Burndown* [Cohn, 2005], práticas estas, que vêm se tornando comuns ao Scrum e a outros métodos ágeis. Outro aspecto bastante explorado no contexto de métricas de monitoração de projeto foram as métricas de valor agregado, que foram abordadas em [Sulaiman, Barton, & Blackburn, 2006], [Cabri & Griffiths, 2006], [Erdogmus, 2010] e [Mahnic & Zabkar, 2008].

Além das pesquisas em métricas de software, também existem pesquisas nas métricas de caráter econômico, como o *Real Option Thinking* apresentado por Racheva [Racheva & Daneva, 2008], o *Earned Business Value* [Rawsthorne, 2006], em complementação as técnicas *Return on Investment* e *Internal Rate of Return*, comumente utilizadas em métodos ágeis [Cohn, 2005]. Estas métricas não são métricas de software e sim econômicas, contudo foram listadas aqui os trabalhos relacionados pois pelas alterações na comunicação com o cliente e entrega de produtos pesquisas foram realizadas para melhor medir estes aspectos do projeto, Contudo, para este trabalho, não foram consideradas as métricas econômicas como pertencentes ao escopo, limitando-se o escopo apenas às métricas de software.

Desenvolvimento de programas de métricas para o Scrum já foram feitos por [Oualid & Lévesque, 2010], [Jakobsen & Sutherland, 2009] e [Mahnic & Zabkar, 2008]. Oualid e Lévesque [Oualid & Lévesque, 2010] realizaram uma *survey* com integrantes de times de desenvolvedores Scrum e levantaram métricas que os desenvolvedores consideravam importantes. Neste trabalho se aprofundaram na resolução do *Technical Debt,* custo relacionado à revisão de aspectos técnicos do sistema implementados de maneira não ideal - quanto mais complexa a solução, maior o custo de revisitação e, consequentemente, maiores os custos do projeto do time de desenvolvimento e ressaltaram as dificuldades de definir e implementar um programa de métricas apropriado em um ambiente de métodos ágeis que pode, por vezes, ser bastante hostil em relação a práticas não ágeis.

Tabela 3 - Métricas em Projetos Ágeis

Métricas	Processo	Produto	Projeto	Econômicas
Sato, Goldman, & Kon, 2007	-	X	-	-
Ambu, Concas, Marchesi, & Pinna, 2006	-	X	-	-
Kunz, Dumke, & Zenker, 2008	-	X	-	-
Alshayeb & Li, 2005	-	X	-	-
Knoernschild, 2006	-	X	-	-
Jeffries, 2004	X	-	-	-
Jakobsen & Sutherland, 2009	X	-	-	-
Mahnic & Zabkar, 2008	X	-	X	-
Fuqua, 2003	-	-	X	-
Sulaiman, Barton, & Blackburn, 2006	-	-	X	-
Cabri & Griffiths, 2006	-	-	X	-
Erdogmus, 2010	-	-	X	-
Racheva & Daneva, 2008	-	-	-	X
Rawsthorne, 2006	-	-	-	X
Spies & Ruiz, 2012	X	-	X	-

Já [Jakobsen & Sutherland, 2009] focaram na melhoria do processo de desenvolvimento através da sugestão de métricas para mensurar características do

processo, melhorando o comportamento organizacional e atingindo um maior nível de desempenho na execução do arcabouço Scrum na empresa Systematic. O trabalho de [Mahnic & Zabkar, 2008], por sua vez, foca na apresentação de um plano de métricas com o objetivo de monitoração de métricas de processo, produto e projeto. Baseado neste plano de métricas, eles ainda vão além e apresentam um modelo de um repositório de dados para o armazenamento e consulta destas métricas, o que é considerada uma boa prática ao se trabalhar com métricas de desenvolvimento [CMMI, 2010].
A Tabela 3 separa os trabalhos de acordo com a classificação das métricas sobre as quais focam. Sendo estas divididas em métricas de processo, produto, projeto e econômicas.

De acordo com a tabela, o trabalho mais similar a este, considerando métricas em Scrum, é a proposta de Mahnic e Zabkar [Mahnic & Zabkar, 2008], que também apresenta um plano de métricas para a monitoração de projetos que utilizam Scrum. Contudo, nos diferenciamos em alguns aspectos, como na monitoração através de valor agregado de Sulaiman [Sulaiman, Barton, & Blackburn, 2006], a não utilização de métricas de produto e a utilização das métricas de processo [Jakobsen & Sutherland, 2009]. A necessidade da realização destas alterações será mais bem explicada na seção 3 onde descrevemos mais detalhadamente a escolha das métricas pertencentes ao plano.

2.5.2 Ambientes de DW para coleta de métricas

A utilização de ambientes de Data Warehouse para o armazenamento de métricas de software já foi investigado e utilizado em alguns trabalhos [Palza, Furhman, & Abran, 2003], [Casati, 2007], [Castellanos, Casati, Dayal, & Shan, 2005], [Becker, Ruiz, Novello, & Cunha, 2006] e [Silveira, Becker, & Ruiz, 2010]. Nesta seção iremos descrever cada uma dessas abordagens juntamente com as suas principais características de modo a visualizar em que diferem da abordagem proposta neste trabalho.

2.5.2.1 *A Multidemensional Measurement Repository in CMMI Context*

O MMR proposto por Palza [Palza, Furhman, & Abran, 2003] tem por objetivo gerenciar, através de um repositório central de métricas e um modelo multidimensional para avaliação, métricas de software dentro de um contexto de CMMi dentro da empresa canadense *Enterprise Performance Unit* da *Ericson Research*.

A proposta deles é a criação de um repositório genérico utilizado para a coleta,

armazenamento, análise e relatório das métricas de acordo com o modelo sugerido pelo CMMi. Este repositório permite a análise dos dados através de cubos OLAP com pesquisas multidimensionais para uma melhor interação com os dados. Essa proposta busca apoiar a maturidade das organizações com diferentes níveis de CMMi através do suporte aos seus programas de métricas. Nessa abordagem os dados são entregues aos usuários através de um portal Web, através de indicadores e relatórios pré definidos. Quanto à coleta de métricas, as mesmas são coletadas manualmente e inseridas no repositório através de uma interface web, o que pode ser considerado como um ponto negativo.

2.5.2.2 BPI e iBOM

O BPI é uma arquitetura geral criada para dar suporte a gestão de qualidade do processo de negócios [Casati, 2007]. Essa arquitetura é composta por um conjunto de ferramentas HP(Hewlett and Packard) dedicadas a tarefas de gerência de projeto e permite realizar atividades de:

- **Análise**: Análises de execuções de processos, tanto da perspectiva de negócios como a dos analistas de TI. Permite a visualização de relatórios através de funcionalidades possibilitando identificar comportamentos em certos processos.
- **Previsão**: Derivação de modelos de previsão para aplicação em processos e busca de comportamentos não esperados.
- **Monitoração**: Monitoração de processos em execução.

O BPI é criado em cima da estrutura de gerência de *workflow* HP *Process Manager* gerando e armazenando dados em logs. Esses dados são armazenados em uma estrutura de DW e visualizados através de um componente chamado de *cockpit*.

2.5.2.3 SPDW+: Uma arquitetura de *datawarehousing*

O SPDW+ (*Software Development Process Performance Data Warehouse Plus*) [Silveira, Becker, & Ruiz, 2010] consiste em uma abordagem para solucionar o problema da necessidade da captura de métricas de software de uma maneira freqüente, automatizada e de pouco envolvimento por parte da equipe desenvolvedora, integrada com um repositório central para uma diferente gama de análises. Ele é uma melhoria

para a proposta do SPDW [Becker k. , 2006], que foi criado para apoiar o Programa de Métricas HP Enterprise And Services Brasil, como um projeto de parceria entre o Programa de Pós-Graduação de Ciências da Computação da PUCRS [PPGCC-PUCRS] e a HP EAS Brasil durante o processo de certificação CMMI nível 3.

O ambiente proposto apresenta uma AOS (Arquitetura Orientada a Serviços) parcial, onde as métricas relacionadas com o PDS são extraídas automaticamente das suas fontes de dados originais [processo ETC (Extração, Transformação e Carga)], com baixa intrusão [pouco envolvimento dos colaboradores de cada projeto], considerando diversas características heterogêneas dos projetos e das ferramentas utilizadas por eles. Após a extração, os dados são mantidos em uma área de armazenamento temporária, onde são submetidos a rotinas de limpeza e transformação. Concluída essa etapa, os dados são consolidados em um repositório central, a partir do qual as métricas são disponibilizadas para análise, através de componentes de apresentação.

O diferencial do SPDW+ é a total utilização de AOS, aliado a WS e metadados possibilitando oferecer um processo de ETC de métricas automatizado que pode ser utilizado a partir de um programa de métricas, de um modelo de gerenciamento e de um modelo analítico do repositório de dados. As principais características da abordagem adotado no SPDW+ é o tratamento do problema da heterogeneidade relacionada ao contexto de desenvolvimento de software, sendo automatizada, permitindo a captura de métricas em diferentes freqüências e latências permitindo tanto a análise quanto a monitoração das métricas de software. As vantagens do ambiente SPDW+ englobam: [1] flexibilidade para atender requisitos mesmo com mudanças freqüentes comuns em ambientes de PDS (Processo de Desenvolvimento de Software); [2] suporte ao monitoramento, que implica na execução frequente e incremental de cargas; [3] automação do custoso processo de captura de métricas. A solução do processo de ETC do SPDW+ é baseada em *web services* (WSs) e AOS, os quais são considerados conceitos chaves para alcançar flexibilidade e adaptabilidade em um PDS.

A arquitetura do ambiente SPDW+ é composta de:

- Um componente de repositório – que é um *Data Warehouse*
- Um componente de apresentação – que contém as funcionalidades analíticas para acessar o Data Warehouse. Este componente não é descrito, pois estava fora do escopo da pesquisa e não foi abordado.
- Os componentes de ETC – Subdivididos em Integração de Aplicação e Integração

de Dados

De modo geral a arquitetura do SPDW+ é demonstrada na Figura 15 onde os semicírculos pretos representam os WS do processo de ETC. O componente de integração com a aplicação automaticamente obtém as métricas das estruturas de dados de diferentes ferramentas, e os passa para o repositório temporário do componente de integração de dados. A extração é realizada através de *wrappers*, baseados em meta-dados de projetos e rotinas de extração. Para que então, baseado em padrões organizacionais, o componente de integração de dados possa limpar e realizar as rotinas de transformação padronizando os dados de mensurações, e após isso, realizar a carga dos dados dentro do DW.

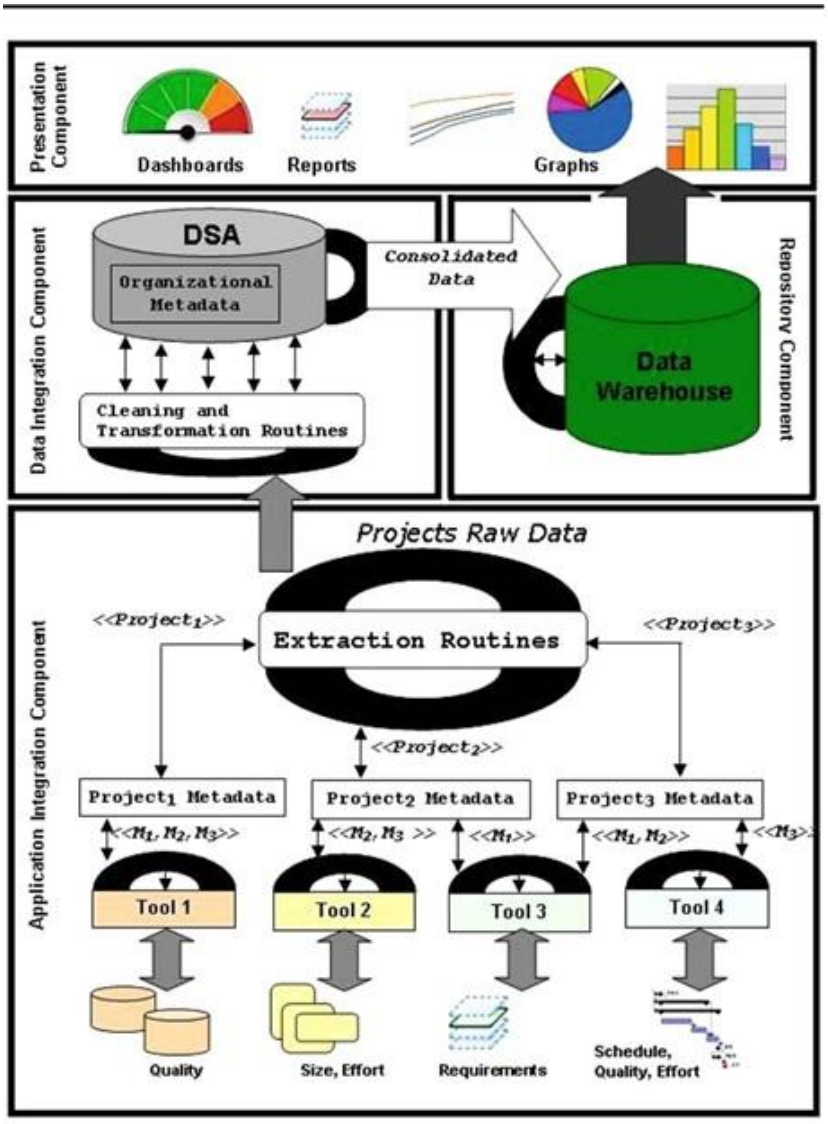

Figura 15 - Estrutura SDPW+ [Sil10]

Componente de integração de aplicação

O componente de integração de aplicação compreende os metadados do projeto, os *wrappers* e as rotinas de extração.

Metadados de projeto

Os metadados de projeto são arquivos *XML* que contêm informações que determinam a localização dos dados de métricas para um determinado projeto. Estes metadados incluem parâmetros como o nome da métrica, a tabela fonte, tipo de dado, e o domínio. Cada projeto pode definir o conjunto de ferramentas utilizado e definir como a informação é estruturada nestas ferramentas. Os bancos de dados das ferramentas contêm o conjunto das métricas diretas que devem ser capturadas, e para cada uma dessas ferramentas, os metadados armazenam as suas informações de acesso (nome da ferramenta, URL, login e senha).

A adoção destes metadados facilita a extração das métricas e suas variações características [nome do atributo, tabela de origem, domínio]. E modificações nestes metadados permitem a alteração do processo de gerência do desenvolvimento de software sem a necessidade de modificação do componente de integração de aplicação, reduzindo desta maneira o esforço de codificação. Ainda, novos projetos não são restringidos pelo conjunto de ferramentas que utilizam ou como eles representam a informação.

Wrappers

Os *wrappers* são pacotes que encapsulam uma ou mais aplicações através de uma interface única. A abordagem utilizada no SPDW+ exige que exista um *wrapper* associado a cada ferramenta de suporte ao desenvolvimento que seja responsável pela captura de métricas em seus arquivos e bancos de dados subjacentes. Estes *wrappers* são baseados em WS e apresentam as seguintes vantagens: [1] Permitem a adição de novas ferramentas e projetos; [2] permitem o compartilhamento de serviços entre projetos que utilizam as mesmas ferramentas; [3] permitem a extração de dados de diferentes ferramentas, executadas em diferentes sistemas operacionais; e [4] oferecem liberdade na escolha da linguagem de programação utilizada para a codificação da ferramenta do serviço de extração de dados.

Rotinas de extração

As rotinas de extração são o cerne do componente de integração de aplicação, que é responsável pela coordenação das chamadas dos *wrappers*; o acesso as diferentes ferramentas; a sincronização da extração de métricas; consolidação das métricas em único pacote; e garantir a homogeneidade e consistência do processo de extração. Essas rotinas de extração são organizadas em duas diferentes camadas, a primeira sendo usada para a coordenação da comunicação dos *wrappers*, e a segunda para interagir diretamente com o *data staging area* (DSA). O DSA é um repositório temporário onde são realizadas as rotinas de limpeza e transformação, para que depois disso os dados já prontos possam ser movidos para o DW.

Componente de integração de dados

O componente de integração de dados da arquitetura do SPDW+ é responsável pela transformação e limpeza dos dados armazenados no DSA e também pelo processo de carga dentro do DW.

Limpeza e transformação

O DSA é composto por um conjunto de tabelas, onde os dados extraídos por bases de código são armazenados e pré-processados por rotinas de limpeza e transformação (ETC), considerando o modelo analítico do DW alvo. Nesta etapa as seguintes atividades de transformação são aplicadas:

- Solucionar domínios através dos *wrappers*: durante a extração das métricas, os conflitos de domínio são solucionados, e dados faltantes e formatos de valores são tratados, de acordo com um padrão pré-definido e ainda levando em conta as informações presentes nos metadados organizacionais.
- Combinar fontes de dados e verificar a integridade entre as chaves primárias.
- Criar chaves substitutas para cada dimensão de registro, de modo a evitar dependências em chaves definidas de legado.

Os metadados organizacionais definem um conjunto de regras para transformar dados brutos em dados com um padrão organizacional que possa ser comparável, de tal modo que uma visão unificada possa ser entregue ao DW. A automação provida pelas ETCs em conjunto com a flexibilidade proporcionada pelos metadados organizacionais tem um

papel significante em prover um processo de ETC livre de intrusão lidando com a heterogeneidade.

Carga de dados

O SPDW+ propõe um procedimento de carga incremental de modo a suportar o monitoramento e a manutenção do repositório em um estado atualizado e consistente na presença de cargas freqüentes. Com isso, é possível monitorar periodicamente os projetos e preservar o histórico de desempenho. Esta carga no SPDW+ deve ser realizada por intermédio de um WS após os dados estarem corretamente estruturados e consolidados no DSA, e como resultado os dados são inseridos no DW. Essa abordagem orientada a serviços apresenta as seguintes vantagens: [1] a possibilidade de modificar o repositório de dados sem comprometer todo o cenário; [2] usar o mesmo serviço de carga em ambientes similares; [3] adicionar novas métricas ao procedimento de carga pela simples adição de um novo serviço; e [4] uma carga livre de intrusão da chamada dos WS, sem nenhuma intervenção humana.

Componente repositório

O componente do repositório do SPDW+ trata a representação das métricas de um PDS de acordo com um modelo multidimensional e um conjunto de *guidelines* para a criação do ambiente de DW. O trabalho apresentado com o SPDW+ ainda propõe a utilização de tempo de validade para as tabelas de fato, para permitir a carga incremental no repositório.

Tabela 4 - Métricas SPDW+

Quality Areas	Derived Metrics	Direct Metrics
Schedule	**SV** – Schedule Variance (Original and Revised Baselines – **SVOB, SVRB**)	**ASD** – Actual Start Date **AED** – Actual End Date **OBSD** – Original Baseline Start Date **OBED** – Original Baseline End Date **RBSD** – Revised Baseline Start Date **RBED** – Revised Baseline End Date
Effort	**EV** – Effort Variance (Original and Revised Baselines – **EVOB, EVRB**) **PR** – Productivity	**AE** – Actual Effort **OBE** – Original Baseline Effort **RBE** – Revised Baseline Effort **AS** – Actual Size
Size	**SZV** – Size Variance (Original and Revised Baselines – **SZVOB, SZVRB**)	**OBS** – Original Baseline Size **RBS** – Revised Baseline Size **AS** – Actual Size
Cost	**CV** – Cost Variance **SV** – Schedule Variance **CQ** – Cost of Quality **CVE** – Cost Variance Earned **SVE** – Schedule Variance Earned **CPI** – Cost Performance Index **SPI** – Schedule Performance Index	**AC** – Actual Cost **OBC** – Original Baseline Cost **RBC** – Revised Baseline Cost **ACAR** - Actual Cost of Activity Revision **ACTP** – Actual Cost of Test Phase **ACAQ** – Actual Cost of Activity Quality **ACRWA** – Actual Cost of Rework Activity **%WC** – %Work Complete **SD** – Status Date
Requirements	**RV** – Requirements Volatility	**NACR** – No. of Approved Change Requests **NAR** – No. of Added Requirements **NDR** – No. of Deleted Requirements **NMR** – No. of Modified Requirements
Quality	**DRE** – Defect Removal Efficiency **DDD** – Delivered Defect Density **IDD** – Internal Defect Density **RE** – Review Efficiency **CS** – Customer Satisfaction	**NDI** – No. of Defects Internally Found **NED** – No. of Defects Found by Clients **AS** – Actual Size **CSI** – Customer Satisfaction Index

Quanto às métricas usualmente é comum a criação de um programa de métricas, e para qualquer programa de métricas é necessário que exista a definição das métricas utilizadas pela organização e como as mesmas serão combinadas para obter métricas mais complexas. O SPDW+ não propõe um conjunto de métricas estático, permitindo que diferentes métricas sejam adicionadas ou removidas de modo a melhor atender diferentes programas de métricas em projetos distintos atendendo as suas particularidades. No estudo de caso realizado na HP EAS Brasil utilizando o SPDW+ o conjunto de métricas utilizado na época era focado na utilização do método EVA (*Earned Value Analysis)* para a monitoração e controle do projeto, sendo utilizado o conjunto de métricas demonstrado na .

A estrutura de dados do SPDW+ ainda permite que diferentes estruturas de projetos sejam implementadas e modeladas de modo a atender diferentes modelos de PDS. No estudo de caso apresentado pelo SPDW+ a estrutura de dados do projeto é demonstrada como na Figura 17 esta figura mostra o diagrama UML (*Unified Modeling Language*) da estrutura de projetos existente no estudo de caso, e ela representa a possibilidade da criação de diversas *releases*, cada uma delas podendo ser subdividida em diversas fases ou iterações. Uma fase pode ou não pertencer a uma iteração. Fases contêm atividades relacionadas a seu tipo [e.g. trabalho, retrabalho, revisão]. Já os defeitos são mensurados de acordo com as fases e com sua severidade associada [e.g. High, Low].

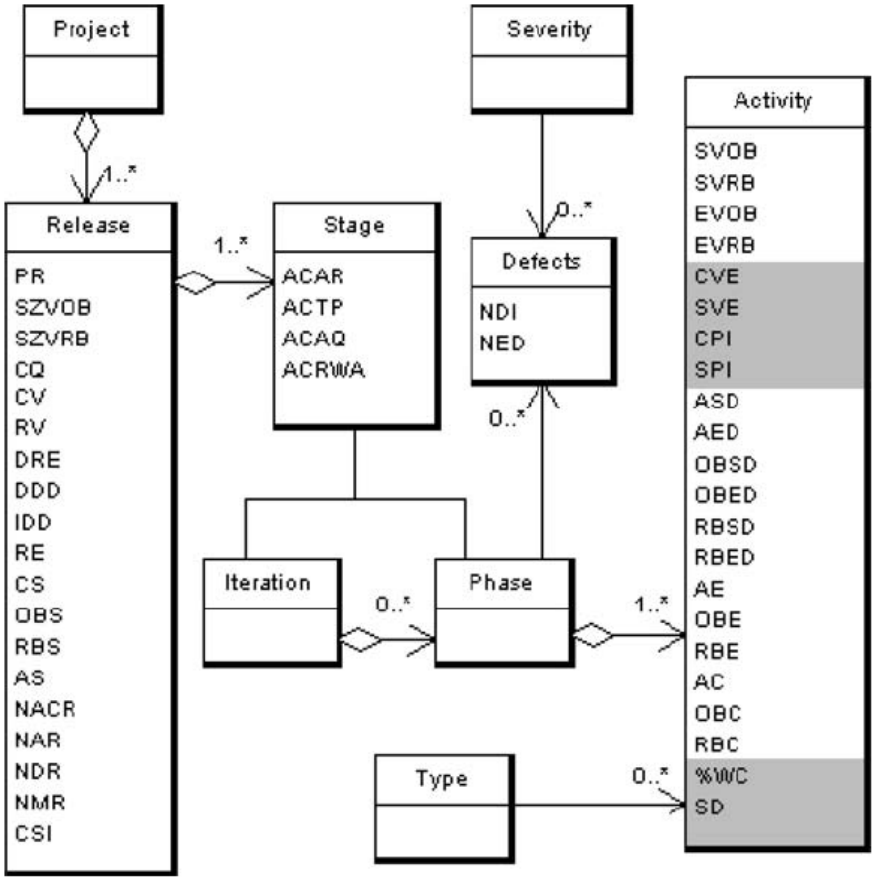

Figura 16 - Estrutura do Projeto do SPDW+

O modelo analítico é representado como uma constelação de fatos relacionados ao conjunto de métricas especificadas. Esse modelo analítico provê diferentes perspectivas de análise baseados em tabelas de fatos. O SPDW+ ainda é flexível podendo acomodar diferentes métricas desde que as mesmas sejam representadas na estrutura do projeto, e tenham a sua granularidade e informações definidas nas tabelas fato e suas dimensões. Desta forma, é parte da atividade de *design* mapear as métricas na estrutura do projeto e refletir isto no modelo multidimensional de acordo com a granularidade desejada.

Para obter um suporte satisfatório ao processo de monitoração da mensuração das métricas de software, é necessário que o processo de captura de métricas seja baixa latência e alta freqüência. Além disso, as métricas devem estar disponíveis em um repositório central de modo agregado, consistente e atualizado. Para a resolução desse problema a abordagem utilizada no SPDW+ consiste na utilização de um tempo de validade para os fatos, baseado no modelo TRM (*Temporal Relational Model*) [Navathe & Ahmed, 1993].

2.5.3 Considerações sobre os trabalhos relacionados

Considerando os trabalhos relacionados, existem alguns que precisamos destacar como os mais fundamentais a esta pesquisa. O SPDW+ [Silveira, Becker, & Ruiz, 2010] foi utilizado como uma das principais influências, ele apresentava um ambiente de DW para monitoramento de projetos com o processo de ETC automatizado, baseado neste trabalho buscamos evoluir esta proposta de modo que fosse possível trabalhar com métodos ágeis de desenvolvimento, dado o crescente ganho de popularidade deste tipo de projeto.

Uma vez com o foco determinado em métricas de acompanhamento e monitoramento do projeto o trabalho do Mahnic [Mahnic & Zabkar, 2008] foi uma das influências uma vez que ele propõe um repositório de métricas para projetos em Scrum. Ele apresentou um plano de métricas e realizou a modelagem de um repositório para o seu armazenamento e ainda sugere a criação de um ambiente de DW para melhor utilização destas métricas.

Outro trabalho bastante importante foi a proposta de uma abordagem de avaliação de valor agregado em projetos ágeis [Sulaiman, Barton, & Blackburn, 2006]. Este trabalho apresenta uma abordagem do método EVM utilizando as características de gerência de projetos ágeis, métricas desta abordagem foram adicionadas ao plano de métricas deste trabalho de modo a ser possível obter o monitoramento do projeto através de mais essa ferramenta.

3. PLANO DE MÉTRICAS PARA SCRUM

Para o desenvolvimento deste plano de métricas buscamos na bibliografia básica do arcabouço Scrum as métricas já utilizadas. Também buscamos seus princípios e valores para compreender as necessidades em tal tipo de projeto. Como passo seguinte, passamos a procurar em diferentes trabalhos científicos propostas de métricas para este tipo de projeto, selecionando métricas compatíveis com o nosso foco em monitoramento e acompanhamento de projeto [essencialmente métricas de projeto] e compatíveis com o arcabouço Scrum e heurísticas de métricas ágeis [Hartmann & Dymond, 2006]. O Acompanhamento e monitoramento através da utilização de métricas também podem ser aplicados a outros aspectos do projeto, como por exemplo, a qualidade de produto monitorando informações como coesão, tamanho de código, complexidade entre outros.

Como ponto inicial a procura por métricas de projeto foi realizada nos principais livros que descrevem o arcabouço Scrum [Cohn, 2005],[Cohn, 2004], [Schwaber, 2004], [Schwaber & Beedle, 2001]. Neste ponto foram levantadas as métricas já presentes e utilizadas por padrão no arcabouço de desenvolvimento Scrum.Não existe uma declaração explícita das métricas coletadas, uma vez que o Scrum não coloca como mandatória a utilização de um programa de métricas para o acompanhamento do projeto. Desta forma, estas métricas foram levantadas através da observação do método de gerência do trabalho, principais artefatos, e processo de desenvolvimento do método, levantando assim métricas que seriam implícitas às realizações destas etapas.

Como segundo passo, foram realizadas pesquisas em trabalhos científicos buscando métricas propostas para o acompanhamento e monitoramento deste tipo de projeto e as mesmas também fora sendo adicionadas ao plano.

Também foram avaliadas as principais métricas de monitoramento de projetos tradicionais [Kan, 2002], [Laird & Brennan, 2007]. Elãs foram avaliadas quanto aos princípios ágeis de desenvolvimento [Manifesto for Agile Software Development, 2001] e as heurísticas de definição de métricas ágeis adequadas [Hartmann & Dymond, 2006] adicionando métricas já consideradas como tradicionais buscando não ferir os princípios ágeis e adicionando um pouco mais de controle. Desta forma, agrupamos um conjunto de métricas que consideramos adequadas para o objetivo

de monitoramento e acompanhamento de projetos Scrum, as quais são apresentadas nas tabelas 2, 3 e 4. A primeira demonstrando o conjunto de métricas primitivas juntamente com seu ponto de coleta, e as duas seguintes contendo as métricas derivadas de projeto e de processo respectivamente, e suas fórmulas de obtenção.

As métricas propostas neste plano têm seu ponto de coleta nas reuniões padrões já estabelecidas no Scrum [Schwaber & Beedle, 2001]. Por padrão, a métrica definida no Scrum para a realização do monitoramento do projeto é a quantidade de trabalho restante [Schwaber & Beedle, 2001]. Para a definição do plano de métricas, nos baseamos na descrição do arcabouço Scrum e também nas técnicas de estimativas de pontos de histórias e de velocidade, apresentadas por Cohn [Cohn, 2005]. A inclusão das métricas de histórias de usuário e de velocidade ocorreu devido às mesmas já serem práticas comuns ao Scrum utilizado no mercado. Para o monitoramento e análise de tempo e custo, utilizamos a abordagem AgileEVM proposta por Sulaiman [Sulaiman, Barton, & Blackburn, 2006] devido à análise de valor agregado ser uma técnica de grande poder informativo, tanto que, muitas vezes, acaba sendo uma exigência contratual, ou até mesmo a sua utilização é uma política organizacional. O AgileEVM apresenta a técnica de análise de valor agregado de maneira simples, sem ferir os princípios ou prejudicar a agilidade de projetos ágeis. Desta forma, as métricas necessárias para a utilização desta técnica foram adicionadas ao plano de métricas proposto de modo a viabilizar um monitoramento mais eficaz do projeto de desenvolvimento.

Tabela 5 - Plano de Métricas Scrum

Áreas de Qualidade	Métricas Derivadas	Métricas Diretas	Ponto Coleta
Esforço	V – Velocidade TRT – Trabalho restante para todas as atividades da Sprint PRC – Pontos de release completados até o momento	TR – Trabalho restante em cada tarefa do backlog de Sprint [Schwaber & Beedle, 2001]	Reuniões diárias
		TRA – Trabalho restante de atividades adicionado [Schwaber & Beedle, 2001]	Reuniões diárias
		TRR – Trabalho restante de atividades removido [Schwaber & Beedle, 2001]	Reuniões diárias
		PHPR – Pontos de história de usuário planejados para release	Planejamento de Release
		PHC – Pontos de história completados na sprint [Cohn, 2005]	Planejamento de Sprint
		PHA – Pontos de história adicionados [Cohn, 2005]	Planejamento de Sprint
		PHR – Pontos de história removidos [Cohn, 2005]	Planejamento de Sprint
Processo	MTRI – Média de tempo para resolução de impedimentos	NI – Nro impedimentos por Sprint/Release [Mahnic & Zabkar, 2008]	Reuniões diárias
		DAI – Data de abertura do impedimento [Mahnic & Zabkar, 2008]	Reuniões diárias
		DFI – Data de fechamento do impedimento [Mahnic & Zabkar, 2008]	Reuniões diárias
		NBS – Nro de Builds realizadas com sucesso [Jakobsen & Sutherland, 2009]	Reuniões diárias
Tempo	DER – Data da entrega da Release	TS – Tamanho da Sprint [Sulaiman, Barton, & Blackburn, 2006]	Planejamento de *Release*
		NP– Nro de Sprints planejadas [Sulaiman, Barton, & Blackburn, 2006]	Planejamento de *Release*
		DIR – Data de início da Release [Sulaiman, Barton, & Blackburn, 2006]	Planejamento de *Release*
		NS – Nro da Sprint atual [Sulaiman, Barton, & Blackburn, 2006]	OS
Time		NIT – Nro. de integrantes do time [Mahnic & Zabkar, 2008]	Planejamento de *Release*
		DT – Percentual de dedicação do time ao projeto [Mahnic & Zabkar, 2008]	Planejamento de *Release*

Tabela 6 - Plano de Métricas Scrum (continuação)

Custo	TCR – Percentual de trabalho completado real TCP – Percentual de trabalho completado planejado VPA – Variação de prazo agregada VCA – Variação de custo agregada VA – Valor agregado VP – Valor Planejado PC – Percentual planejado completado IDC – Indice de desempenho de custo IDP – Índice de desempenho prazo	CPR – Custo planejado para a Release [Sulaiman, Barton, & Blackburn, 2006]	Planejamento de *Release*
		CR – Custo real da Release [Sulaiman, Barton, & Blackburn, 2006]	Planejamento de *Release*
		CS – Custo real da Sprint anterior [Sulaiman, Barton, & Blackburn, 2006]	Planejamento de *Sprint*
Qualidade	MRDI – Média de tempo para resolução de defeitos internos MRDE – Média de tempo para resolução de defeitos externos ERD – Eficiência de remoção de defeitos	NDI – Nro defeitos internos [Mahnic & Zabkar, 2008]	Reuniões diárias
		NDE – Nro defeitos externos [Mahnic & Zabkar, 2008]	Reuniões diárias
		DADI – Data de abertura do defeito [Mahnic & Zabkar, 2008]interno	Reuniões diárias
		DFDI – Data de fechamento do Defeito interno [Mahnic & Zabkar, 2008]	Reuniões diárias
		DADE – Data de abertura do defeito externo [Mahnic & Zabkar, 2008]	Reuniões diárias
		DFDE – Data de fechamento do defeito externo [Mahnic & Zabkar, 2008]	Reuniões diárias

As métricas são divididas em diretas ou derivadas. As primeiras sendo atributos coletados em diferentes pontos do processo: planejamento de release, planejamento de Sprint e reuniões diárias. As derivadas, por sua vez, são calculadas a partir das métricas diretas com o intuito de obter um diferente nível de informação. As métricas estão divididas em diferentes áreas de qualidade:

- Tempo – Área fundamental para o monitoramento do projeto através da análise de valor agregado. Em métodos ágeis ao invés de considerar datas de início e fim de atividades do cronograma, o cálculo é realizado através da avaliação da quantidade de *Sprints* planejadas e a duração fixada para cada *Sprint*.

- Esforço – Nesta área o programa de métricas é dividido em dois diferentes níveis. No primeiro o esforço é monitorado através da avaliação do trabalho restante nas tarefas da *Sprint*, realizado através da mensuração da quantidade de itens de *Backlog* de *Sprint* e a atualização diária do trabalho restante para cada um dos itens. Já em nível de Release o esforço é monitorado a partir da noção de pontos de histórias de usuário planejados para a release e a velocidade do time de desenvolvimento.
- Custo – Métricas de custo são essenciais à monitoração de valor agregado do projeto. Esta monitoração será realizada em nível de *Release* e envolve toda a análise das métricas de valor agregado, variação de custo, cronograma e os índices de desempenho clássicos a esta técnica.
- Processo – Métricas relacionadas ao monitoramento das práticas processuais. Estas métricas têm por objetivo avaliar se o processo de desenvolvimento adotado está sendo devidamente utilizado.
- Qualidade – Indicadores da qualidade são coletados por motivos de registro e referência histórica. Estes atributos refletem a qualidade do produto que, apesar de importantes, não desempenham um papel fundamental no monitoramento e controle do andamento do projeto.
- Time – Métricas indicadoras do time de desenvolvimento. Usualmente times de desenvolvimento Scrum são pequenos [7-10] e se busca alcançar dedicação exclusiva ao projeto por parte dos integrantes [Cohn, 2005] .
  Estas métricas têm por objetivo avaliar estas características do arcabouço Scrum, porém não desempenham um papel no acompanhamento do projeto.

Na Tabela 7 são descritas as métricas derivadas com foco no monitoramento do projeto.

Tabela 7 - Métricas Derivadas [Projeto]

Métricas Derivadas	Objetivos	Equação
Percentual de trabalho completado real [TCR]	Medir o percentual do trabalho completado até o momento baseado nas histórias planejadas e completadas.	TCR = PHC/PHPR
Percentual de trabalho completado planejado [TCP]	Medir o progresso estimado para o projeto até o momento	TCP = NS/NP
Valor Agregado [VA]	Medir o valor agregado do projeto até o momento.	VA = TCR*CPR
Valor Planejado [VP]	Medir o valor que o projeto tinha planejado entregar até o momento.	VP = TCP*CPR
Variação de prazo agregada [VPA]	Fornecer a relação entre a estimativa de prazo original e o percentual atual de trabalho completado até a data de status.	VPA=VA-VP
Variação de custo agregada [VCA]	Fornecer a relação entre o custo original planejado e o percentual de trabalho completado até a data de status.	VCA = VA - CR
Índice de desempenho de custo [IDC]	Fornecer a relação entre os custos consumidos pelo projeto e o valor agregado.	IDC = VA/CR
Índice de desempenho de prazo [IDP]	Fornecer a taxa de conversão do valor estimado em valor agregado	IDP = VA/VP
Trabalho restante para todas as atividades da Sprint [TRT]	Mede o total de trabalho restante para a Sprint atual	$TRT = \sum_{k=1}^{n} TR_k$ Onde $n > 0$
Velocidade [V]	Mede a capacidade de implementação de histórias do time, utilizada para a realização dos cálculos de data de entrega.	$V = \frac{1}{n}\sum_{k=1}^{n} V_k$ Onde $n > 0$
Pontos de Release completados [PRC]	Mede o total de pontos de histórias restantes para a Release atual	$PRC = \sum_{k=1}^{n} PHC_k$ Onde $n > 0$
Média de tempo para a resolução de impedimentos [MTRI]	Tem o objetivo de avaliar o tempo médio que se leva para resolver os impedimentos relatados pelo time de desenvolvimento	$MTRI = \frac{\sum(DAI - DFI)k}{NI}$

Já na Tabela 8 estão as métricas derivadas de processo e qualidade do produto. Para cada uma delas, estão descritas as métricas primitivas das quais estas métricas se originam, seu propósito e a equação correspondente.

Tabela 8 - Métricas Derivadas (Processo e Produto)

Métricas Derivadas	Objetivos	Equação
Média de tempo para a resolução de Defeitos Externos [MRDE]	Tem o objetivo de avaliar o tempo médio que se leva para resolver os defeitos internos relatados pelo time de desenvolvimento	$MRDE = \frac{\sum(DFDE - DADE)k}{NDE}$
Média de tempo para a resolução de Defeitos Internos [MRDI]	Tem o objetivo de avaliar o tempo médio que se leva para resolver os defeitos externos relatados pelo time de desenvolvimento	$MRDI = \frac{\sum(DFDI - DADI)k}{NDI}$
Eficiência para a remoção de defeitos [ERD]	Tem o objetivo de avaliar a capacidade do time de desenvolvimento de encontrar os defeitos antes de o produto chegar ao cliente	$ERD = \frac{\mathrm{NDI}}{\mathrm{NDE} + \mathrm{NDI}}$
Data de entrega da Release [DER]	Dado o progresso do time até o momento tem o objetivo de proporcionar uma possível data de entrega da release baseado no progresso do time até o momento	$DER = DIR + TS \cdot \left(\frac{NS}{TCR}\right)$

O monitoramento do projeto ocorre em dois diferentes estágios, o monitoramento da *Sprint* e o monitoramento da *release*. No decorrer da *Sprint* o esforço despendido nas tarefas é monitorado através da quantidade de trabalho restante. Este valor já é atualizado diariamente, por padrão, nos *Daily Scrum*, e visualizado através de gráficos de *Burndown*. Já em nível de planejamento de Release é aplicada a técnica *AgileEVM*, avaliando não mais a granularidade de esforço relacionado às atividades, e sim a quantidade de histórias de usuários completadas, número de *Sprints* realizadas e a velocidade média de implementação do time. Baseado nestes dois níveis de monitoramento, a análise de progresso pode ser tanto em nível de *Sprint,* através de gráficos de *burndown* de atividades, quanto em nível de release através de gráficos de *burndown* de histórias em conjunto com os índices de desempenho de custo e cronograma calculados através do valor agregado.

3.1 CONSIDERAÇÕES SOBRE O PLANO DE MÉTRICAS

Considerando os trabalhos relacionados e o plano de métricas proposto, podemos identificar as principais diferenças existentes entre as diferentes abordagens. Os dois trabalhos relacionados que mais se assemelham a este são o SPDW+[Silveira, Becker, & Ruiz, 2010] e o repositório de métricas proposto por

Mahnic [Mahnic & Zabkar, 2008].

3.1.1 Em relação ao SPDW+

O SPDW+ propôs um programa de métricas para projetos tradicionais de modo que fosse possível realizar o acompanhamento e monitoramento dos projetos através da coleta e análise de métricas, obtidas com pouca interferência do time através de um processo de ETC automatizado. Contudo, este programa de métricas proposto não compreende a realidade de métodos ágeis. As principais diferenças entre o programa de métricas de SPDW+ e o plano de métricas proposto neste trabalho são:

- Monitorar trabalho restante de atividades – Em métodos ágeis o esforço é controlado através do trabalho restante das atividades (e histórias). No SPDW+ as atividades eram monitoradas de acordo com estimativas de duração de horas as quais eram atualizadas de acordo com o esforço gasto diariamente nas mesmas. A abordagem de trabalho restante é padrão a métodos ágeis e facilitam a análise e monitoramento através de ferramentas como os gráficos de *burndown*.
- Monitoramento de Histórias de usuário – Em métodos ágeis o monitoramento e acompanhamento vão acontecendo em dois diferentes níveis de abstração. O primeiro acompanha o desenvolvimento da *sprint* corrente e o segundo o andamento do projeto como um todo. Para este acompanhamento em dois níveis o plano de métricas proposto monitora além do trabalho restante em atividades o trabalho restante em pontos de histórias, uma unidade de medida padrão em métodos ágeis. Desta forma este plano de métricas difere do SDPW+ ao monitorar na dimensão de pontos de história o que não acontece no SPDW+.
- Tamanho de software – As métricas de monitoramento de tamanho de software estão presentes no programa de métricas do SPDW+ apesar de as mesmas não serem utilizadas para o monitoramento devido as diferenças de técnicas utilizadas para esta mensuração. No plano de métricas deste trabalho as métricas de tamanho não foram adicionadas ao plano por tratarem-se de métricas de produto diferindo do foco do monitoramento e acompanhamento de métricas de projeto não tendo relação direta ao monitoramento de projetos Scrum de acordo com o encontrado na literatura.

- Volatilidade de requisitos – O SPDW+ monitora também a quantidade de modificações aos requisitos do software, prática comum em métodos tradicionais onde mudanças nos requisitos eram evitadas para manter a integridade do planejamento. No plano de métricas proposto não colocamos métricas de volatilidade de requisitos (poderiam ser as histórias), em virtude dos princípios ágeis de abraçar as mudanças e não incentivar práticas que mantenham uma as histórias de forma estática.
- Qualidade – O SPDW+ monitora a qualidade através dos defeitos e densidade de defeitos. No plano de métricas proposto apenas são monitorados o número de defeitos e não a densidade, devido que para a densidade seria necessário possuir uma métrica de tamanho de código. No plano de métricas proposto no âmbito de qualidade outra métrica monitorada é a quantidade de impedimentos, onde o monitoramento dos impedimentos é uma das funções básicas do *Scrum Master*.
- Valor Agregado – A análise de valor agregado é uma técnica tradicional bastante reconhecida para o monitoramento dos projetos, no SPDW+ eles utilizam esta técnica para a realização do monitoramento dos projetos. No plano de métricas proposto utilizamos uma adaptação desta técnica para projetos ágeis de desenvolvimento chamada *AgileEVM* [Sulaiman, Barton, & Blackburn, 2006] que utiliza as métricas do Scrum para a realização deste tipo de monitoramento.

3.1.2 Em relação à Mahnic [Mahnič & Vrana, 2007]

Quando comparando este plano de métricas ao de Mahnic e Zabkar [Mahnic & Zabkar, 2008], ressaltamos as seguintes diferenças. Métricas de tamanho não foram utilizadas em virtude que as mesmas não serem definidas, por padrão, no *Scrum*. Usualmente, métodos ágeis utilizam a noção de pontos de história como a medida de tamanho e esforço que, apesar de compartilhar características com técnicas como Pontos de função e pontos de caso de uso, é uma métrica de tamanho relativo e não pode ser utilizada para a comparação com outros projetos. Pontos de Função e de casos de uso possuem procedimentos que se replicados

tendem a obter o mesmo valor possuindo as mesmas entradas, já em pontos de história a valoração é feita pela mensuração do tamanho de forma subjetiva sem a presença de um procedimento padrão o que pode levar em diferentes formas de medição quando lidando com diferentes pessoas.

Uma métrica física que poderia ser utilizada é a contagem de linhas de código. Entretanto, esta não é uma prática recomendada [Kan, 2002] em virtude da grande variação entre plataformas e por esta não levar em consideração a complexidade da funcionalidade sendo desenvolvida. Desta forma, optamos por não utilizar o tamanho físico do código como uma métrica a ser analisada por padrão..

As métricas qualitativas baseadas em *surveys*, apesar de úteis para mensurar atributos como a satisfação do time e cliente, não têm foco na monitoração e controle do projeto. Além disso, elas apresentam um nível de envolvimento do time com as atividades de formulação, aplicação e avaliação da *survey*. Este formato de métrica não é adequado para a utilização no ambiente SPDW+ baseado em um processo de ETC automatizado.

Mahnic e Zabkar [Mahnic & Zabkar, 2008] também propõem realizar o monitoramento do projeto através da análise de valor agregado. Contudo, optamos pela utilização da técnica proposta por Sulaiman [Sulaiman, Barton, & Blackburn, 2006] em virtude de a mesma medir um nível acima [Hartmann & Dymond, 2006], monitorando o projeto em nível de *release*, e utilizando técnicas bastante comuns no cenário ágil de desenvolvimento, como a velocidade e as histórias de usuários. Além disso, a abordagem de Mahnic e Zabkar para avaliação do valor agregado implica na mensuração do esforço realizado em cada uma das tarefas, o que é uma prática considerada inadequada para o Scrum, como descrito por Schwaber [Schwaber & Beedle, 2001].

> "*Relatórios de trabalho restante atualizam o número estimado de horas necessárias para completar uma tarefa. Isso não deve ser confundido com relatórios de tempo, que não são parte do Scrum. Não existem mecanismos no Scrum para monitorar a quantidade de tempo que um time trabalha. Times são mensurados através do cumprimento de objetivos, e não através da quantidade de horas que eles levam até cumprir o objetivo. Scrum é orientado a resultados, não à processos*."

4. MODELAGEM DIMENSIONAL

Nesta seção descrevemos a modelagem dimensional do ambiente de Data Warehouse para o plano de métricas descrito na Seção 3. Esta modelagem dimensional segue uma orientação de uma constelação de fatos de modo a englobar diferentes aspectos de mensuração relacionados às suas respectivas dimensões de avaliação. A escolha da abordagem da constelação de fatos foi necessária devido que as métricas numéricas a serem monitoradas tem diferentes naturezas e relações com as dimensões. Desta forma, através de um modelo dimensional de constelação de fatos buscamos realizar a captura das métricas [fatos] em três diferentes granularidades de acordo com as perspectivas de gerência de projetos Scrum.

4.1 TABELAS FATOS

Para a realização da modelagem dimensional desse ambiente de DW, foram consideradas três diferentes granularidades: A granularidade de atividades (atualização diária); A granularidade de *Release* (Atualização Semanal); e A granularidade de defeitos/Impedimentos (atualização conforme ocorrências). Baseados nessas diferentes granularidades foram modeladas três diferentes tabelas fatos para endereçar as necessidades do plano de métricas e do monitoramento dos projetos Scrum. As três tabelas fatos criadas foram: FatoAtividade; FatoHistórias; FatoImpedDef. Cada uma dessas fatos será descrita mais detalhadamente a seguir.

Desta forma com essas três tabelas fatos temos duas perspectivas de monitoramento dos projetos Scrum. A primeira de monitoramento da sprint e do trabalho nas atividades compositoras das histórias dando uma visão de uma pequena janela de tempo do projeto de modo que as mesmas possam ser vistas e gerenciadas em uma forma mais aprofundada e a segunda uma visão geral de mais alto nível para avaliar o progresso do projeto de maneira geral. E a terceira já não avaliando mais o trabalho nas diferentes perspectivas do projeto, mas sim os aspectos de qualidade através do monitoramento de defeitos e impedimentos que vão sendo controlados durante o decorrer de todo o projeto.

4.1.1 Tabela fato de atividades

Essa tabela fato é a responsável pela coleta das métricas primitivas da granularidade das atividades realizadas diariamente durante a ocorrência da *Sprint*. Nessa tabela fato a métrica principal é a quantidade de trabalho restante (TR) das atividades das histórias de usuário sendo desenvolvida na *Sprint*. A escolha dessa métrica foi realizada em decorrência de o lançamento de TR nas atividades é uma ocorrência diária durante as reuniões diárias tradicionais ao Scrum, baseado na atualização do TR nessas reuniões é que o progresso da *Sprint* vai sendo monitorado, principalmente, através dos gráficos de *Burndown* de *Sprint*. A Figura 17 demonstra a modelagem dimensional desse Fato, juntamente com as suas respectivas dimensões de avaliação. Essa granularidade representa o monitoramente e acompanhamento do projeto durante o decorrer de uma *sprint*, representando o trabalho diário nas atividades compositoras das histórias envolvidas na *sprint* atual, essa é granularidade representa as iterações frequentes de métodos ágeis de desenvolvimento e reforça os princípios de pequenas iterações, feedback constante, e práticas como as reuniões diárias. Essa tabela fato irá representar o projeto nessa perspectiva de monitoramento do projeto dando uma visão de uma pequena fração do projeto que vai sendo desenvolvida em cada *sprint*.

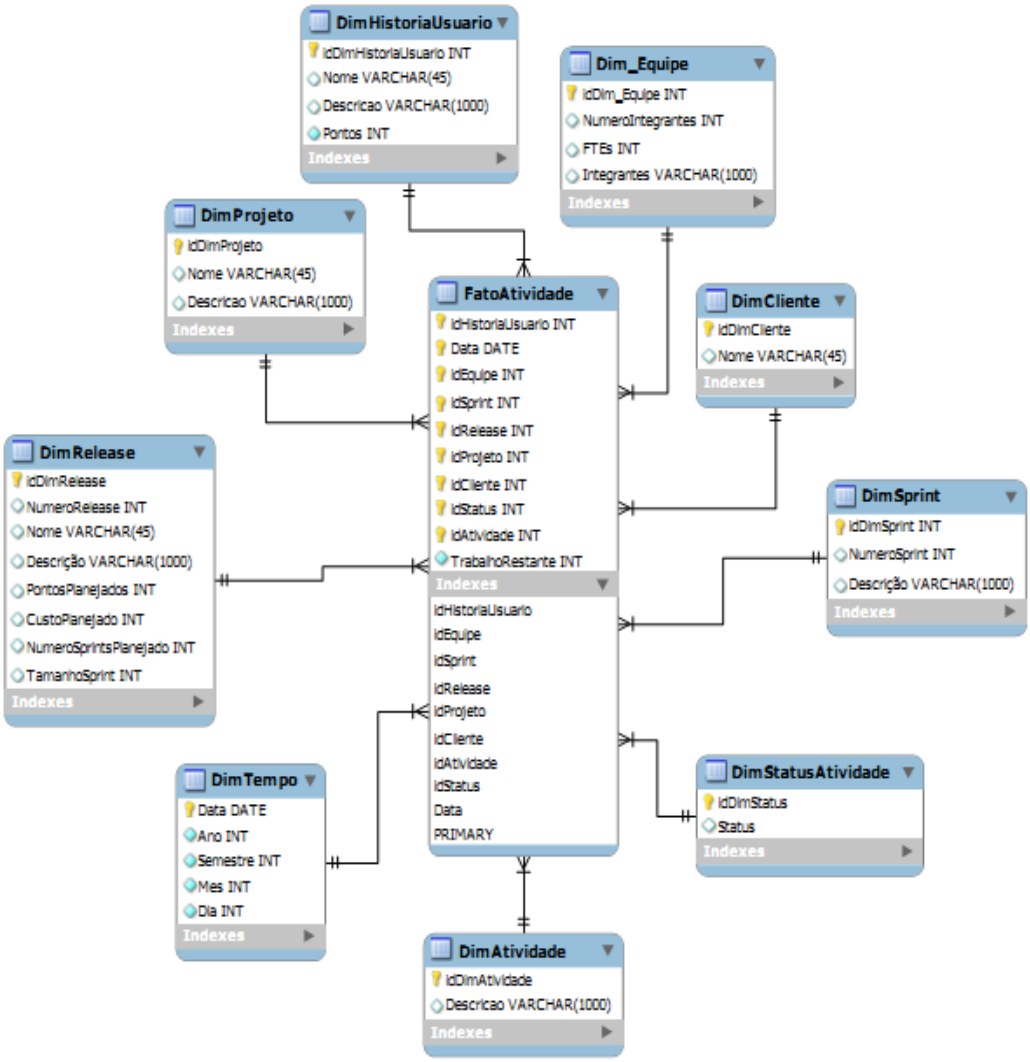

Figura 17 - Fato Atividades

4.1.2 Tabela fato histórias

Já esta segunda tabela fato endereça uma segunda granularidade de monitoramento dos projetos ágeis, que é a das histórias de usuários. Nessa granularidade estão as histórias de usuário que estão sendo desenvolvidas para o release do projeto. Essas métricas são armazenadas no ambiente de DW durante as reuniões de Fim de *Sprint*, onde é atualizada a quantidade de pontos de histórias completadas durante aquela *Sprint* de modo a ter uma noção geral do andamento do projeto em relação a Release. O monitoramento da release é realizado através da atualização de um gráfico de *Burndown* de release onde é possível avaliar a

quantidade de histórias adicionadas e removidas avaliando assim o progresso do time em relação ao projeto. Esta tabela pode ser vista na Figura19 abaixo.

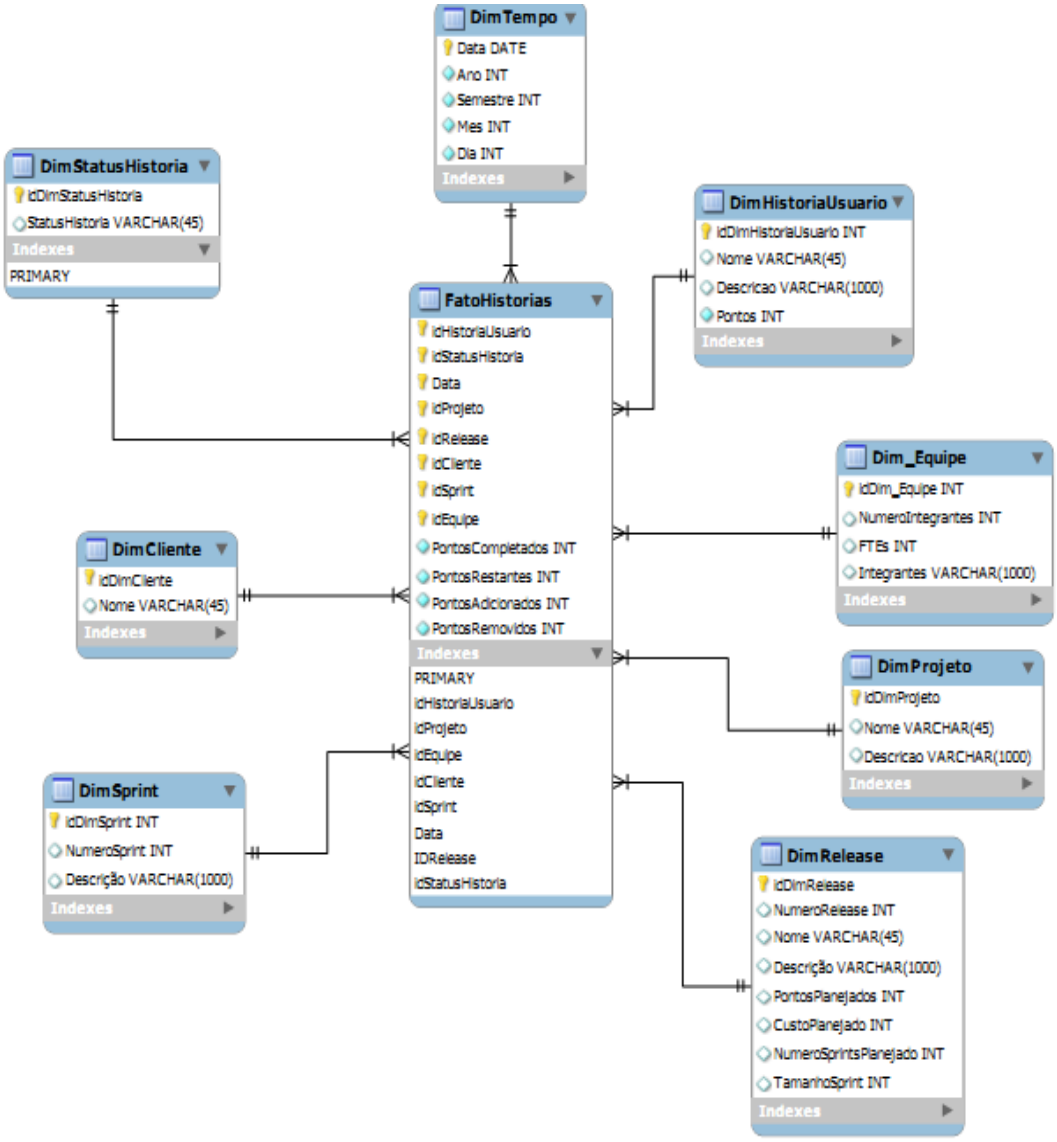

Figura 18 - Fato Histórias

Nessa granularidade é possível observar o progresso do projeto de uma forma mais geral como um todo. As histórias por serem descrições mais genéricas das funcionalidades a serem desenvolvidas e a sua pontuação em pontos de história vão sendo atualizadas ao final de cada *sprint* de modo a existir uma visualização semanal do *status* geral do projeto de desenvolvimento. O planejamento de *releases* é realizado acompanhando o ritmo de desenvolvimento de histórias e o progresso sendo realizado em cada uma das *sprints*.

4.1.3 Tabela Fato impedimentos / defeitos

A terceira tabela fato já tem por objetivo monitorar não as métricas de projeto como as outras duas tabelas fatos anteriores e sim os aspectos de produto e processo do projeto Scrum. Nesta tabela fato, atualizada a cada ocorrência de defeito ou impedimento, estarão os defeitos e impedimentos encontrados durante o decorrer do projeto. O fato nesta tabela é a quantidade de defeitos/impedimentos, os dois estão agrupados como uma mesma métrica e poderão ser diferenciadas através da manipulação da dimensão “Categoria”. Desta forma os defeitos e impedimentos, características relacionadas tanto ao processo de desenvolvimento poderão ser monitoradas e observadas sob diferentes perspectivas em virtude da natureza da modelagem dimensional adotada provendo informações e consultas de maneira bastante simplificada sobre estas informações. A imagem da modelagem da tabela fato de impedimentos bem como suas relações com as diversas dimensões podem ser visualizadas na Figura 20.

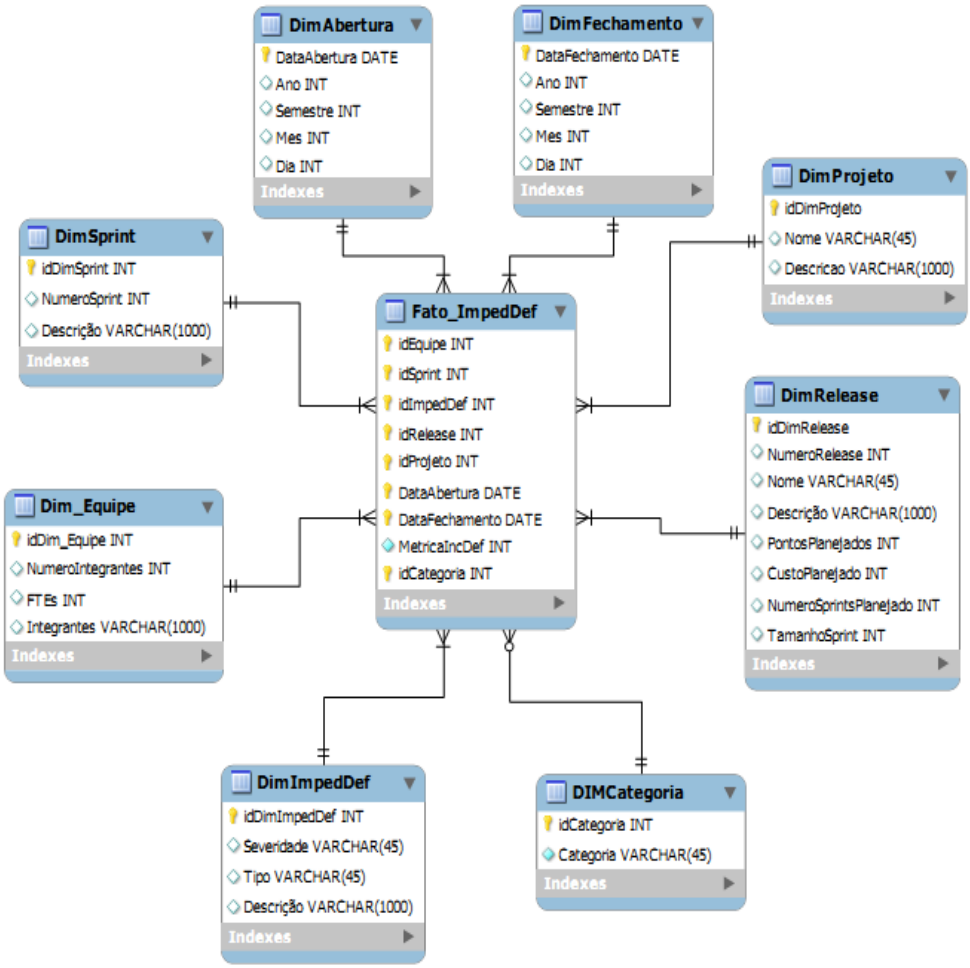

Figura 19 - Fato Impedimento/Defeitos

4.2 Dimensões

As tabelas anteriores (fatos) representam as métricas que são coletadas e monitoradas durante o desenvolvimento do projeto. Já as dimensões representam as diferentes perspectivas de visualização destas métricas. Buscando desta forma, prover à gerência destes projetos uma visualização rápida e sob diferentes filtros sobre as principais métricas coletadas em seus projetos. Nesta seção serão descritas as 14 dimensões criadas e as perspectivas sobre as quais elas irão permitir visualizar os dados.

- **Dimensão história de usuário**

A Dimensão história de Usuário mostra a perspectiva das histórias criadas para a determinada *sprint*. Através dessa dimensão será possível avaliar a quantidade de TR e a quantidade de defeitos/impedimentos para cada uma dessas histórias.

- **Dimensão equipe**

Dimensão responsável por exibir a perspectiva do TR, Defeitos/impedimentos, e Pontos Restantes para cada uma das equipes presentes nos dados armazenados no ambiente de DW

- **Dimensão cliente**

Dimensão responsável por exibir a perspectiva do TR, Defeitos/impedimentos, e Pontos Restantes para cada um dos clientes presentes nos dados armazenados no ambiente de DW

- **Dimensão *sprint***

Dimensão responsável por exibir a perspectiva do TR, Defeitos/impedimentos, e Pontos Restantes para cada uma das *Sprints* presentes nos dados armazenados no ambiente de DW

- **Dimensão *status* atividade**

Essa Dimensão tem por objetivo agrupar as diferentes atividades de acordo com o seu *status*. Essa realização é possível graças à possibilidade da criação de *slowly changing dimensions* [Kimball & Ross, 2002] onde então o *status* de uma atividade pode ser alterado de aberta para completada, onde as atividades completadas estarão com um TR igual a zero.

- **Dimensão atividade**

Nomes e descrições das atividades cadastradas no ambiente de DW.

- **Dimensão tempo**

Dimensão responsável para monitorar o trabalho restante em diferentes tempos do projeto, provendo bases de comparação em diferentes semanas. De modo a avaliar o decorrer do TR nas atividades em diferentes momentos do projeto,

mantendo desta forma esta informação histórica.

- **Dimensão *release***

Dimensão responsável por armazenar todos os atributos da *release* sobre a qual se está trabalhando, não só isso como o monitoramento de horas restantes das atividades das *Sprints* do projeto.

- **Dimensão projeto**

Informações sobre o projeto que está sendo desenvolvido, bem como os dados de quantidade de defeitos/impedimentos em cada projeto cadastrado.

- **Dimensão *status* historia**

Dimensão responsável por armazenar as informações necessárias ao *status* de história, podendo identificar desta forma quando uma história foi finalizada.

- **Dimensão abertura**

Dimensão responsável pelo monitoramento do tempo de abertura de incidentes e impedimentos.

- **Dimensão fechamento**

Dimensão responsável pelo monitoramento do tempo de fechamento de incidentes e impedimentos.

- **Dimensão categoria**

Dimensão responsável pela categorização da métrica de impedimentos/defeitos em impedimentos e defeitos.

4.3 Escolha do projeto exemplo X validação

Devido a restrições de tempo e de contato com empresas com uma realidade de coleta de métricas em Scrum, não foi possível utilizar métricas de um projeto de desenvolvimento real. Desta forma, o ambiente de DW criado foi testado a partir de métricas coletadas a partir de um projeto exemplo existente em um livro. Na escolha deste projeto fictício buscamos encontrar um exemplo claro e clássico de um projeto utilizador de Scrum. O Projeto escolhido para a representação do funcionamento do cubo é o projeto presente no livro do Cohn [Cohn, 2005] retratando um projeto de

uma empresa fictícia de nome “Bomb Shelter Studios” que está desenvolvendo um jogo de tabuleiro em formato digital e passa a utilizar o Scrum pela primeira vez em seus projetos.

Desta forma, foram abstraídos os dados para este projeto exemplo e, os mesmos, foram passados para planilhas de Excel que seriam as ferramentas utilizadas para o registro das métricas coletadas nas reuniões do Scrum. Foram então criadas as planilhas, e as mesmas foram preenchidas de acordo com as descrições do projeto no livro.

Estas planilhas do Excel então passaram por um processo de ETC através da ferramenta de *Integration Services* da Microsoft sobre as quais o ambiente de *Data Staging Area* foi criado, para após esse momento a ferramenta de *Analisys services* realizar a criação do cubo *OLAP* destas métricas.

Aqui abaixo será realizada uma pequena descrição do projeto, bem como estas informações foram passadas para as diferentes planilhas e como isso foi gerado o cubo.

A Figura 20 é um exemplo de uma das planilhas criadas baseado no projeto exemplo utilizado. Esta planilha cobre a parte da fato das atividades da primeira reunião diária da primeira *sprint* realizada neste projeto de desenvolvimento.

	A	B	C	D	E	F	G	H	I	J
1	ID Histórià	ID Atividad	Cliente	Projeto	Release	Equipe	Sprint	StatusAtiv	Estimativa	Data Atualização
2	4	1	1	1	1	1	1	1	16	03/07/2012
3	4	2	1	1	1	1	1	1	10	03/07/2012
4	4	3	1	1	1	1	1	1	12	03/07/2012
5	4	4	1	1	1	1	1	1	12	03/07/2012
6	4	5	1	1	1	1	1	1	8	03/07/2012
7	4	6	1	1	1	1	1	1	4	03/07/2012
8	4	7	1	1	1	1	1	1	4	03/07/2012
9	4	8	1	1	1	1	1	1	12	03/07/2012
10	4	9	1	1	1	1	1	1	4	03/07/2012
11	4	10	1	1	1	1	1	1	2	03/07/2012
12	9	11	1	1	1	1	1	1	4	03/07/2012
13	9	12	1	1	1	1	1	1	2	03/07/2012
14	9	13	1	1	1	1	1	1	4	03/07/2012
15	10	14	1	1	1	1	1	1	6	03/07/2012
16	10	15	1	1	1	1	1	1	6	03/07/2012
17	10	16	1	1	1	1	1	1	8	03/07/2012
18	5	17	1	1	1	1	1	1	4	03/07/2012
19	5	18	1	1	1	1	1	1	6	03/07/2012
20	5	19	1	1	1	1	1	1	12	03/07/2012
21	5	20	1	1	1	1	1	1	6	03/07/2012
22	5	21	1	1	1	1	1	1	4	03/07/2012
23	5	22	1	1	1	1	1	1	8	03/07/2012
24	5	23	1	1	1	1	1	1	4	03/07/2012
25	5	24	1	1	1	1	1	1	2	03/07/2012
26	5	25	1	1	1	1	1	1	4	03/07/2012
27	5	26	1	1	1	1	1	1	4	03/07/2012
28	5	27	1	1	1	1	1	1	4	03/07/2012
29	5	28	1	1	1	1	1	1	16	03/07/2012
30	5	29	1	1	1	1	1	1	6	03/07/2012

Figura 20 - CSV Fato Atividades

Nesta planilha CSV foram colocadas todas as atividades planejadas para as quatro histórias de usuários planejadas para aquela *Sprint*. Cada atividade deve estar relacionada a uma dessas histórias. Dentro deste arquivo CSV utilizado para popular a Fato de Atividades ainda é necessário ter as informações das dimensões às quais a Tabela Fato está relacionada, de modo que possibilite uma visualização das métricas sobre estas diferentes perspectivas.

Ou seja, neste caso de projeto exemplo as reuniões diárias ainda seguem o procedimento de atualização diário desta planilha com TR a cada uma dessas atividades. Então para cada dia da *sprint*, ao final de cada reunião diária, o arquivo CSV deve ser atualizado e passado ao processo de ETC para a geração do cubo OLAP das métricas do projeto.

O mesmo funcionamento também ocorre para a população das outras tabelas fatos e também das dimensões de perspectivas. Cada vez que forem realizadas alterações nos arquivos CSV eles são passados pelo processo de ETC de modo que o cubo seja processado contendo essas novas informações. A Figura 21 mostra o arquivo CSV da dimensão de release, que deve ser atualizado a cada nova *release* planejada para o projeto.

	A	B	C	D	E	F	G	H	I
1	ID Release	Numero Release	Nome Release	Data Inicio	Descrição Release	Estimativa (Pontos)	Estimativa (Sprints)	Custo Planejado	Tamanho Sprint(dias)
2	1	1	Release do Havannah	01/11/2012	Entrega do Jogo Havannah como um todo	146	9	50000	14
3									
4									
5									

Figura 21 - CSV Dimensão Release

Com todas essas dimensões tendo os seus respectivos dados assinalados nos momentos descritos, os arquivos são gerados e o processo de ETC faz a transferência destes dados para o DSA. Estes dados das dimensões pouco provavelmente serão alterados durante o decorrer do projeto, sendo o maior esforço de atualização focado na atualização das métricas presentes nas tabelas fatos criadas.

Após o processo de ETC finalizado através da ferramenta *Integration Services*, a ferramenta *Analisys Services* realiza a montagem do cubo.

4.4 APLICAÇÃO E MONTAGEM DO CUBO

Uma vez com os dados devidamente colocados no DAS através do processo de ETC. A modelagem e processamento do cubo permitem navegar por estes dados de uma maneira a avaliá-los por diferentes perspectivas. As imagens abaixo apresentam os pontos de perspectiva das métricas coletadas no primeiro dia da Sprint do Projeto exemplo avaliadas por três diferentes dimensões: a De Sprint [Figura 22], a de História [Figura 23] e a de Atividades Figura 24.

Descrição Historia	Trabalho Restante
As a Player, I can play against a weak engine that recognizes bridges	82
As a Player, I can play against a weak engine that recognizes rings	112
As a Player, I want the computer to recognize a winning shape	20
As a Player, I'd like to be able to use the system to play against another human on my computer	10

Figura 22 - Cubo (TR por História de Usuário)

Descricao Atividade	Trabalho Restante
additional product research	12
Automate test cases for blocking a human player maki...	2
Automate test cases for making a blocked ring	4
Automate tests	8
Automate tests for blocking another player from makin...	4
Automate tests for choosing between making a bridge ...	2
Automate tests for forming a bridge around obstacles	4
Automate tests for giving up on a bridge	2
Clicking on a hexagon adds a new piece of the right color	4
Code state management classes	16
Computer knows when a piece completes a winning pa...	6
Design tests	6
Draw empty board	2
Engine can choose appropriately between placing a pie...	16
Engine can find path from one corner to another (that ...	4
Engine can form a bridge around obstacles	12
Engine knows when to give up on a particular bridge	8
Engine tries to prevent a another player from forming ...	4
Have move engine pursue a ring even if the human pla...	8
Have move engine pursue an unblocked ring	12
Have the move engine try to block a human player wh...	12
Identify and automate tests for simple bridge design	6

Figura 23 – Cubo (TR por Atividade)

Descricao Sprint	Trabalho Restante
Implementação de algumas histórias	224

Figura 24 - Cubo (TR por *Sprint*)

Nestas três imagens é possível avaliar a informação da métrica de trabalho restante sobre as diferentes perspectivas (Dimensões) e essa avaliação ainda pode ser realizada no decorrer do tempo, avaliando as alterações nos valores para o monitoramento deste tipo de projeto.

Neste Cubo estão armazenadas todas as métricas primitivas necessárias para calcular as métricas presentes no plano de métricas proposto neste trabalho. As métricas derivadas, contudo não estão modeladas, pois as mesmas não necessariamente fazem parte do escopo da modelagem dimensional e de construção do cubo OLAP do projeto, muitas delas sendo calculadas e exibidas através de players de cubo OLAP.

Os dados provenientes do exemplo do livro não descreviam o projeto em sua totalidade, desta forma para melhor exemplificar o funcionamento do cubo,

incrementamos os dados do exemplo com valores de modo a demonstrar o decorrer do trabalho durante uma *sprint* para ser possível ver à redução do trabalho restante diário como demonstrado na Figura 25. Os valores decrescentes de acordo com os dias podem ser utilizados por uma ferramenta de análise OLAP gerando os gráficos de *burndown* comumente utilizados para o monitoramento de projetos em Scrum.

Date	Trabalho Restante
Tuesday, July 03 2012	224
Wednesday, July 04 2012	190
Thursday, July 05 2012	152
Friday, July 06 2012	112
Saturday, July 07 2012	66
Sunday, July 08 2012	0

Figura 25 - Trabalho Restante (Fim *Sprint*)

Desta forma teria sido completada a primeira *Sprint* deste projeto, além dos dados de atividades também foi realizada a atualização dos dados de histórias de usuário, subtraindo os pontos de história completados, desta forma podendo visualizar o andamento do projeto na granularidade de histórias de usuário como pode ser visualizado na Figura 26.

Date	Pontos Restantes
Tuesday, July 03 2012	146
Sunday, July 08 2012	128

Figura 26 - Historias Restantes (Fim *Sprint*)

Através deste monitoramento tanto das histórias quanto das atividades se tem a possibilidade de gerenciar o andamento dos projetos Scrum. As métricas derivadas presentes no plano podem ser calculadas manualmente através das métricas primitivas presentes no cubo, ou pode ser realizado um esforço para a criação de uma camada de apresentação que se realiza estes cálculos de maneira automatizada.

5. CONCLUSÃO

Desde a criação do manifesto ágil é visível o crescente interesse nos mesmos tanto dentro da indústria quanto em trabalhos acadêmicos [Torgeir Dingsøyr, 2012]. Isso é evidente quando avaliando a quantidade e qualidade de publicações nos últimos anos e no número de países envolvidos nessas pesquisas [Torgeir Dingsøyr, 2012]. A maioria das publicações científicas na área é baseada no método XP apesar do fato de o Scrum estar ganhando bastante tração na indústria de desenvolvimento de software [Torgeir Dingsøyr, 2012].

Dentro destas pesquisas acadêmicas realizadas, os temas de pesquisa variam desde as práticas e princípios utilizados por estes métodos, assim como a gestão destes projetos em comparação a métodos considerados como tradicionais. Este trabalho focalizou essencialmente no aspecto da utilização de métricas de desenvolvimento como uma ferramenta para gerenciamento de projetos utilizadores do arcabouço Scrum de desenvolvimento. Trabalhos nesta área já foram realizados e atentam quanto à importância da utilização de métricas para o controle de projetos ágeis [Fowler, 2009], [Jakobsen & Sutherland, 2009].

Dentro deste contexto de métricas e estimativas dentro do Scrum, buscamos maneiras de utilizar métricas para mensurar o desenvolvimento deste tipo de projeto, utilizando-as para o acompanhamento e monitoramento destes projetos sem ferir os princípios base declarados no manifesto ágil e baseado nas práticas particulares de gestão deste tipo específico de projeto. O plano de métricas proposto é baseado nas heurísticas de [Hartmann & Dymond, 2006] da utilização coerente de métricas em métodos ágeis de modo a buscar um nível de controle sobre o projeto sem ferir os princípios sobre os quais estes métodos operam.

Considerando o plano de métricas desenvolvido, a proposta de um ambiente de DW para o armazenamento e consulta sobre este conjunto de métricas. A utilização de um ambiente de DW para a coleta, armazenamento e consulta de métricas, funcionando como uma um repositório central das métricas coletadas também já tinha sido considerada como uma prática adequada para o monitoramento de projetos [Castellanos, Casati, Dayal, & Shan, 2005],[Palza, Furhman, & Abran, 2003] e [Silveira, Becker, & Ruiz, 2010]. Buscando desta forma,

utilizar desta ferramenta para reforçar os princípios ágeis de desenvolvimento como a utilização de ferramentas de automatização facilitando o acesso as informações relevantes ao monitoramento deste tipo de projeto.

As principais limitações desta pesquisa foram o fato de não termos conseguido métricas de um projeto real para tentar avaliar o desempenho, bem como limitações de tempo para devidamente realizar o acompanhamento e monitoramento de um projeto. Devido a estas limitações escolhemos apenas cobrir a parte de métricas de projeto, não dando ênfase às métricas de produto e processo e também avaliando apenas um projeto exemplo da literatura sem contar desta forma com um caso real de avaliação.

Desta forma, espera-se que através da utilização deste plano de métricas em conjunto com o ambiente de DW projetos utilizando o arcabouço Scrum consigam monitorar e acompanhar os seus projetos sem causar um aumento grande de esforço na prática de coleta de métricas, uma vez que as mesmas são coletadas nas reuniões já existentes deste arcabouço, e que essas informações possam ser utilizadas como referencial histórico de uma maneira simplificada e visualizada através de diferentes dimensões de análise.

Como perspectiva de trabalho futuro poderia ser realizada a continuação da evolução do SPDW+[Silveira, Becker, & Ruiz, 2010] devidamente criando o ambiente de DW de maneira completa compreendendo o processo de ETC automatizado e bem como a camada de exibição de dados e relatórios em um player OLAP de modo que este ambiente pudesse adequadamente ser utilizado para o monitoramento de métricas de projeto e útil para o processo de tomada de decisões baseado nestas métricas.

Bibliografia

[Amb06] Ambu, W., Concas, G., Marchesi, M., & Pinna, S. [2006]. “Extreme Programming and Agile Processes in Software Engineering”. Berlin, Springer Berlin / Heidelberg, 2006. 230p.

[Bec06a] Becker, k. “SPDW: a Software Development Process Performance Data Warehousing Environment”. In: SEW '06 Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop, 2006, pp. 107-118.

[Cab06] Cabri, A., & Griffiths, M. “Earned Value and Agile Reporting”. In: AGILE '06 Proceedings of the conference on AGILE, 2006, pp. 17-22.

[Cas07] Casati, F. “A Conversation on Web Services: what´s new, what´s true, what´s not. And what´s not”. In: Workshop on Knowledge Transformation for the Semantic for the Semantic Web at the European Conference on Artificial Intelligence. 2007, pp. 1-2.

[Cas05] Castellanos, M., Casati, F., Dayal, U., & Shan, M. “iBOM: A Platform for Intelligent Business Operation Management”. In: International Conference on Data Engineering [ICDE], 2005, pp. 1084-1095

[CMM06] CMMI. “CMMI for Development, Version 1.2. Software Engineering Institute, Carnegie Mellon University”. Capturado em: http://www.sei.cmu.edu/reports/10tr033.pdf Novembro 2013.

[Coc04] Cockburn, A. “Crystal Clear A Human-Powered Methodology for Small Teams”. Addison-Wesley Professional, 2004, 336p.

[Coh05a] Cohn, M. “Agile Estimating and Planning”. Prentice Hall PTR, 2005, 368p.

[Coh04b] Cohn, M. “User Stories Applied: For Agile Software Development”. Addison Wesley Longman Publishing Co., Inc. 2004, 304p.

[Erd10] Erdogmus, H. “Tracking progress through earned value”. IEEE Software vol 27 no. 5, September 2010, pp. 2-7.

[Fow09] Fowler, M. “*Flaccid Scrum”*. Martin Fowler. Capturado em: http://martinfowler.com/bliki/FlaccidScrum.html Novembro 2013

[Fuq03] Fuqua, A. M. “Using function points in XP – Considerations”. In: XP'03 Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering, 2003. pp.

340-342.

[Har06] Hartmann, D., & Dymond, R. “Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value”. In: AGILE '06 Proceedings of the conference on AGILE, 2006, pp. 126 – 134.

[Jak08a] Jakobsen, C. R., & Johnson, K. A. “Mature Agile with a Twist of CMMI”. In: AGILE '08 Proceedings of the Agile, 2008, pp. 212-217.

[Jak09b] Jakobsen, C. R., & Sutherland, J. “Scrum and CMMI – Going from Good to Great”. In: Agile Conference AGILE '09. 2009, pp. 333 – 337.

[Jef04] Jeffries, R. “A Metric Leading to Agility”. Capturado em: http://xprogramming.com/articles/jatrtsmetric/ Novembro 2013

[Kan02] Kan, S. H. “Metrics and Models in Software Quality Engineering”. Boston, MA, USA.: Addison-Wesley Longman Publishing Co., Inc. 2002, 2nd edition, 560p.

[Kim02] Kimball, R., & Ross, M. “The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling”. Wiley Computer Publishing. 2002, 464p.

[Kno06] Knoernschild, K. “Using Metrics To Help Drive Agile Software”. Capturado em: http://www.agilejournal.com/articles/columns/the-agile-developer/56-using-metrics-to-help-drive-agile-software Novembro 2013.

[Kun08a] Kunz, M., Dumke, R. R., & Zenker, N. “Software Metrics for Agile Software Development”. In: ASWEC '08 Proceedings of the 19th Australian Conference on Software Engineering. 2008, pp. 673-678.

[Kun09b] Kunz, M., Dumke, R., & Schmietendorf, A. “How to Measure Agile Software Development”. In: Software Process and Product Measurement . Springer-Verlag Berlin, Heidelberg 2008, pp. 95 - 101.

[Lai07] Laird, L. M., & Brennan, M. C. “Software Measurement and Estimation: A Practical Approach”. Washington, DC, USA: IEEE Computer Society. 2007, 280p.

[Lon11] Longstreet, D. “The Agile Method and Other Fairy Tales”. Capturado em: http://softwaremetrics.com/Agile/Agile%20Method%20and%20Other%20Fairy%20Tales.pdf Novembro 2013

[Mah07a] Mahnič, V., & Vrana, I. “Using stakeholder-driven process performance measurement for monitoring the performance of a Scrum-based software development process”. In: *Electrotechnical Review*, 2007, pp. 241-247.

[Mah08b] Mahnic, V., & Zabkar, N. “Measurement repository for Scrum-based software development process”. In: *CEA'08 Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications*, 2008, pp. 23-28.

[Man01] “*Manifesto for Agile Software Development”*. Capturado em: http://agilemanifesto.org/ Novembro 2013.

[Nav93] Navathe, S., & Ahmed, R. “Temporal extensions to the relational model and SQL”. In: R. Ahmed, *Temporal databases: Theory design and implementation,* Benjamin/Cummings, 1993, pp. 92-109.

[Oua10] Oualid, K., & Lévesque, G. “Designing and implementing a measurement program for Scrum teams: what do agile developers really need and want?” In: *C3S2E '10 Proceedings of the Third C* Conference on Computer Science and Software Engineering* pp, 2010, 101-107.

[Pal03] Palza, E., Furhman, C., & Abran, A. “Establishing a Generic and Multidimensional Measurement Repository in CMMI context”. In: *Annual IEEE/NASA Software Engineering Workshop*, 2003, pp. 12-20.

[Pre92] Pressman, R. S. “*Software Engineering [3rd Ed.]: A Practitioner's Approach”.* New York: McGraw-Hill, 1992, 880p.

[Rac08] Racheva, Z., & Daneva, M. “Using measurements to support real-option thinking in agile software development”. In: APOS '08 Proceedings of the international workshop on Scrutinizing agile practices or shoot-out at the agile corral, 2008, pp. 15-18.

[Raw06] Rawsthorne, D. “Calculating Earned Business Value For An Agile Project*”.* Capturado em: http://www.agilejournal.com/index2.php?option=com_content&do_pdf=1&id=54 Novembro 2013.

[Sat07] Sato, D., Goldman, A., & Kon, F. “Tracking the evolution of object-oriented quality metrics on agile projects”. In: XP'07 Proceedings of the

8th international conference on Agile processes in software engineering and extreme programming, 2007, pp. 84-92.

[Sch04a] Schwaber, K. “Agile Project Management With Scrum”. Redmond, Microsoft Press, 2004, 188p .

[Sch01b] Schwaber, K., & Beedle, M. “Agile Software Development with Scrum”. Upper Saddle River , Prentice Hall PTR, 2001, 158p.

[Sil10] Silveira, P. S., Becker, K., & Ruiz, D. D. “SPDW+: a seamless approach for capturing quality metrics in software development environments”. In: *Software Quality Journal*, 2010, pp. 227-268.

[Som11] Sommerville, I. “Software Engineering [9th edition]”. New York, Pearson Addison Wesle, 2011, 782p.

[Spi12] Spies, E., & Ruiz, D. D. “Proposta de um plano de métricas para o monitoramento de projetos Scrum”. In: Workshop Brasileiro de Metodos Ageis [WBMA], pp. 84-95.

[Sul06] Sulaiman, T., Barton, B., & Blackburn, T. “AgileEVM - Earned Value Management in Scrum Projects”. In: AGILE '06 Proceedings of the conference on AGILE, 2006, pp. 7 - 16.

[Tor12] Torgeir Dingsøyr, S. N. “A decade of agile methodologies: Towards explaining agile software development”. *Journal of Systems and Software. Vol 85, issue 6*, jun-2012, pp. 1213-1221.

[Wil11] Williams, L., Brown, G., Meltzer, A., & Nagappan, N. “Scrum + Engineering Practices: Experiences of three Microsoft Teams”. In: ESEM '11 Proceedings of the International Symposium on Empirical Software Engineering and Measurement, 2011, pp. 463-471.