Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Aline Bicalho Santos

Um método para gerenciamento de custos em projetos de *software* desenvolvidos com metodologia Scrum.

São Paulo
2011

Aline Bicalho Santos

Um método para gerenciamento de custos em projetos de *software* desenvolvidos com metodologia Scrum.

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Software

Data da aprovação ____/_____/_____

_____
Prof. Dr. José Eduardo Zindel Deboni (Orientador)
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Membros da Banca Examinadora:

Prof. Dr. Dr. José Eduardo Zindel Deboni (Orientador)
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Prof. Dr. José Sidnei Colombo Martini (Membro)
USP – Universidade de São Paulo

Prof. Dr. Mario Yoshikazu Miyake (Membro)
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Aline Bicalho Santos

Um método para gerenciamento de custos em projetos de *software* desenvolvidos com metodologia Scrum.

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo – IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Software.

Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. José Eduardo Zindel Deboni

São Paulo
Dezembro/2011

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo – IPT

S237m **Santos, Aline Bicalho**
Um método para gerenciamento de custos em projetos de software desenvolvidos com metodologia Scrum. / Aline Bicalho Santos. São Paulo, 2011.
70 p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software

Orientador: Prof. Dr. José Deboni

1. Controle de custo 2. Projeto de software 3. Scrum 4. Valor econômico agregado 5. Tese I. Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Coordenadoria de Ensino Tecnológico II. Título

12-09 CDU 004.413(043)

RESUMO

A metodologia de desenvolvimento ágil *Scrum* não tem uma diretriz clara de gestão de custos. Esta pesquisa propõe um método para estimar o custo dos requisitos, mesmo não estando eles todos especificados, a fim de obter uma meta orçamentária do projeto. Além disso, o trabalho propõe uma adaptação da técnica de gestão de valor agregado (EVA) para projetos *Scrum* contribuindo para fazer a gestão do orçamento. A técnica de valor agregado permite a elaboração antecipada de planos de ação com o intuito de manter o orçamento original do projeto. O trabalho destaca a importância em se priorizar os requisitos, como forma de indicar a ordem de implementação das funcionalidades. Deve-se implementar primeiramente os requisitos que tragam maior benefício ao usuário e valor ao negócio da organização. A manutenção da lista de requisitos deve ser constante ao longo de todo projeto para validar sua aderência ao orçamento vigente. Para consolidar os custos estimados e realizados, propõem-se a utilização da ferramenta gráfica de *burndown*. A técnica proposta foi aplicada em um projeto de implementação de um sistema de *Supply Chain*. O projeto teve a duração de um ano, com uma equipe de 10 pessoas. Os resultados da utilização da técnica puderam ser captados através da entrega de valor a partir das primeiras funcionalidades implementadas e do controle de custos que evitou qualquer desvio no orçamento inicial durante todo o projeto.

Palavras-chave: Metodologia ágil *Scrum*, estimativa de custos, controle de custos, EVA, *burndown chart*.

ABSTRACT

A cost management method to software development projects using Scrum methodology.

Scrum agile methodology does not have a straight cost management driver. As in the beginning of the project, the requirements are not all specified, Scrum methodology is compared as a time and material contract, without budged restrictions. In this research, it is proposed a cost estimate method to capture budged baseline. The earned value analysis (EVA) technique was adapted to Scrum projects to help cost management. This technique helps to elaborate action plans to track and keep the project on the budget. This study also reveals the importance of prioritized requirements. The priority indicates the implementation requirements order. First requirements implemented are the one that brings more value to the business and user. The requirement list must be updated during the whole project. The burndown chart is adopted to visually demonstrate the relation between estimate cost and current cost. The main contribution of this research is to propose a method to estimate and control costs in Scrum projects, influencing in the adoption of the methodology in the companies. The proposal was applied in a project to implement a Commercial and Supply Chain System. One year project and 10 people involved in the development. The results were captured by the users that started using the functionalities as soon as they were available and by the cost control avoiding any budged deviation during the whole project.

Key-Words: Scrum agile methodology, cost estimate, cost control, EVA, burndown chart.

Lista de Figuras

Lista de Tabelas

Sumário

1. INTRODUÇÃO

1.1. Motivação

O gerenciamento de custos implica planejamento, estimativa, orçamentação e controle de custo (PMBOK, 2004). A orçamentação é uma compilação dos custos inicialmente estimados (MULCAHY, 2005). A estimativa é uma das entradas do processo de orçamentação e, portanto, devido a esta dependência, para este trabalho, estes dois processos serão simplificados em apenas um processo que visa estimar os custos. Estimar e controlar custos de um projeto de desenvolvimento de *software* são atividades complexas pela dificuldade em se definir com precisão os requisitos do software, em se visualizar o produto final antes que ele seja entregue, em se obter a aceitação do produto por parte de um cliente (SAKAMOTO, 2006).

Projetos de *software* com metodologia de desenvolvimento baseada em planejamento influenciam positivamente as atividades do gerenciamento de custos, pois estabelecem um plano de trabalho detalhado e, consequentemente, uma previsibilidade maior dos recursos necessários para que em determinado espaço de tempo se obtenha o produto desejado (MULCAHY, 2005).

No entanto, com o advento das metodologias de desenvolvimento de *software* ágeis, não baseadas em planejamento, apoiadas pelo Manifesto Ágil em 2001(BECK et al., 2001) observa-se um afastamento do planejamento antecipado, para um planejamento *ad hoc*. Este manifesto não propôs eliminar valores e técnicas de desenvolvimento utilizados até então, mas focou sua atenção, na colaboração e nas respostas rápidas às mudanças (BECK et al., 2001), formalizando os princípios para desenvolvimento ágil de *software*.

As práticas de gestão de desenvolvimento de *software* mais flexíveis, adaptáveis e produtivas são encontradas na proposta da metodologia ágil *Scrum*. No *Scrum*, o desenvolvimento é organizado em iterações denominadas *sprints, que,* usualmente, não excedem o período de um mês. A decisão do planejamento de cada *sprint* é tomada ao final do *sprint* anterior, que pode acomodar mudanças nos requisitos, dificultando um planejamento de prazos mais longos. Por outro lado, ao final de cada *sprint*, as funcionalidades previstas já podem ser avaliadas e

aprovadas pelos clientes. Contribui-se assim para diluir a aceitação do sistema ao longo do projeto, ao contrário de apenas uma validação ao final do desenvolvimento do sistema. As práticas de *Scrum* começaram a ser usadas por Jeff Sutherland, Ken Schwaber e Mike Beedle no início da década de 90 (SCHWABER; BEEDLE, 2001) com o foco na maximização do retorno sobre o investimento, possibilitando respostas rápidas às mudanças ocorridas no ambiente de negócio (HIGHSMITH, 2004). As mudanças de requisitos, a partir daí, passaram a ser tratadas durante a construção do software, e não mais apenas ao final do produto entregue.

Se por um lado, flexibilidade e adaptabilidade foram questões importantes para propor uma nova metodologia de desenvolvimento de *software*, a importância no gerenciamento de custos deste novo ambiente torna-se igualmente importante para fortalecer a implantação desta metodologia nas organizações e assegurar o retorno ao investimento. Adotar metodologias ágeis traz os benefícios da flexibilidade, melhor aceitação das mudanças, satisfação dos clientes e progresso gradual, no entanto, tais benefícios podem não ser justificados se o custo desta adoção exceder a expectativa orçamentária do projeto e não contribuir para a entrega do produto final. (BLACK et al., 2009).

Schwaber e Beedle (2001) afirmam que "*Scrum* é um sistema que se auto-organiza". De fato, pelas várias interações inerentes à metodologia, as pessoas estão mais próximas, comunicam-se mais e ajudam-se mutuamente culminando em auto-organização. No entanto, não exercem grande regulação do custo financeiro do desenvolvimento, pois se voltam mais à implementação gradativa das funcionalidades que se comprometem a desenvolver. As funcionalidades são definidas e especificadas somente no início de cada iteração. Não há um plano detalhado das atividades a serem realizadas como em projetos cuja metodologia se baseia em planejamento. A falta de visibilidade do detalhamento das funcionalidades no início do projeto influencia na forma como estimar os custos do projeto de *software*.

A implantação de uma nova forma de desenvolvimento de *software*, no caso, *Scrum*, transforma a cultura da empresa. Minimizar os efeitos desta implantação preservando as mesmas técnicas de estimativa e controle de custos utilizadas pode ser arriscado e comprometer o sucesso das técnicas. Novas metodologias de desenvolvimento de *software* exigem novas formas de gestão de custos.

1.2. Objetivo

Este trabalho propõe um novo processo para gestão dos custos de projetos de *software* desenvolvidos com a metodologia Scrum. O processo de gestão, aqui considerado, inclui as atividades de estimativa e controle do custo do *software*

Objetivos secundários desta pesquisa incluem o incentivo à adoção da metodologia Scrum, e o aumento do sucesso nos projetos que se utilizam dela. Não é escopo desta pesquisa abordar as técnicas de formação do preço de um projeto de *software*, atividade que pode estar mais ligada a oportunidades de mercado, marketing e outros objetivos comerciais. O estudo concentra-se, exclusivamente, no custo.

Através do controle de custos de um projeto, é possível analisar o custo já realizado e o custo a realizar. A visibilidade do custo a realizar é fundamental para tomada de ação e direcionamento do plano de desenvolvimento futuro a fim de não permitir que a soma dos custos realizados e a realizar exceda o orçamento inicialmente acordado para o projeto.

A contribuição deste estudo é potencializar a utilização do método ágil *Scrum* em projetos de desenvolvimento de *software* através do controle rígido dos custos do projeto. Não há a intenção em se demonstrar que a utilização do *Scrum* é um meio para se reduzir os custos dos projetos. A adoção de metodologias ágeis não significa desenvolver de maneira mais rápida e, portanto, mais barata, mas sim ter uma resposta rápida às necessidades de mudanças de requisitos (BLACK et al., 2009).

Este trabalho contribui para que tanto a estimativa quanto o controle dos custos sejam tratados de forma conjunta para garantir um processo mais assertivo na análise entre orçado e realizado. A proximidade entre cliente e fornecedor e a natureza de colaboração proposta pelos métodos ágeis influenciam positivamente na tomada rápida de decisão, evitando desvios entre o custo orçado e realizado.

O objetivo de toda empresa é gerar lucro para seus acionistas (ASSAF NETO, 2002) e através do controle dos custos é possível avaliar se o projeto de *software* não está desviando deste objetivo. A importância dada à disciplina de gestão dos

custos na gestão de projetos é devido à sua relação direta com a capacidade de contribuir para que a organização atinja um balanço positivo. Desta forma, outra contribuição deste trabalho é adaptar as a técnica do valor agregado (PMBOK, 2004), comumente utilizada para estimativa e controle de custos na engenharia de software, para projetos ágeis.

1.3. Método de trabalho

Inicialmente, apresenta-se a metodologia ágil *Scrum*.

Em seguida, realiza-se uma compilação das técnicas de estimativa de custos de software para definir os processos de estimativa de custos mais adequados aos projetos cuja metodologia não é voltada ao planejamento.

Posteriormente, é proposto um **método de estimativa e controle de custos** de *software* adaptado à metodologia de desenvolvimento ágil *Scrum*. As técnicas de estimativa de custo e controle de custos serão ajustadas para garantir o alinhamento dos processos de estimativa com os de controle de custos. Desta forma, garante-se que as variáveis a serem controladas são as mesmas variáveis estimadas.

Para validar a proposta, realiza-se a **aplicação do método**. O projeto considerado para aplicação do método é um projeto real desenvolvido para atender a necessidade de um cliente da empresa. A empresa é uma empresa de tecnologia brasileira especializada em desenvolvimento de sistemas. Este projeto será desenvolvido por uma equipe com experiência na metodologia de desenvolvimento *Scrum*. A aplicação do método proposto será realizada de forma paralela às técnicas utilizadas pela empresa para estimativa e controle de custo de software.

Por fim, a **análise dos resultados** visa uma comparação com os valores estimados e controlados ao longo do projeto pela técnica tradicional e os valores captados pela técnica proposta baseada na informação de gestão de custo. O objetivo desta atividade é verificar se a aplicação das técnicas de estimativa e controle dos custos propostas neste trabalho altera as decisões ao longo dos *sprints* e contribui para a adoção do *Scrum* no mercado provendo a empresa de dados estimados e controlados de acordo com a realidade do projeto.

1.4. Organização do trabalho

Além deste capítulo introdutório, o trabalho possui outros capítulos organizados da seguinte forma:

No Capítulo 2, Revisão Bibliográfica, são descritos o detalhamento da metodologia de desenvolvimento ágil *Scrum* e as técnicas de estimativa e controle de custos em projetos de *software*.

No Capítulo 3, Análise das Técnicas de Estimativa de Custo e Análise de Valor Agregado, são analisadas as técnicas que podem ser aplicadas em projetos de desenvolvimento ágil *Scrum*. Desta análise, são selecionadas as técnicas que farão parte do método a ser proposto.

No Capítulo 4, Método de Estimativa e Controle de Custos, é construído um método de estimativa e controle de custos para projetos *Scrum*. É proposto um método que priorize a implementação do *software* pela análise das funcionalidades prioritárias ao negócio.

No Capítulo 5, Aplicação do Método, aplica-se o método em um projeto de desenvolvimento de sistema utilizando metodologia *Scrum*. Por fim, é feita uma análise dos resultados obtidos com a aplicação do método.

Finalmente, no Capítulo 6, Conclusão, é feito um resumo do trabalho, apresentando os resultados alcançados e são sugeridas propostas para trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo são abordados assuntos de duas áreas de conhecimento: a metodologia *Scrum* e o gerenciamento de custos. No tópico Metodologia de Desenvolvimento *Scrum,* busca-se na literatura as características de utilização do método ágil *Scrum* em projetos de *software*. No tópico Gerenciamento de custos, a atenção é voltada às práticas de gerenciamento de custos com ênfase na estimativa e no controle dos custos.

2.1. Metodologia de Desenvolvimento Scrum

O *Scrum* surgiu acreditando na melhoria da comunicação entre a equipe e redução do prazo de entrega de código, em função das validações freqüentes ao longo do desenvolvimento do sistema. O *Scrum* estabelece o que pode ser produzido tendo como base a capacidade produtiva disponível da equipe de desenvolvimento (SCHWABER; BEEDLE, 2001).

Descrevem-se aqui as suas características gerais.

2.1.1. Visão geral

O *Scrum* (SCHWABER; BEEDLE, 2001) é um método para desenvolvimento de *software* iterativo e incremental. O produto final é obtido por meio de iterações sucessivas, cada uma com incremento nos requisitos.

A figura 1 ilustra o processo *Scrum*. Nesta figura, encontram-se alguns elementos deste processo.

- *Sprint* é o termo utilizado para nomear cada iteração do *Scrum*.
- *Product Backlog* é a lista de requisitos que o *software* deve contemplar. *Sprint Backlog* é uma pequena parte do *product backlog* que contém itens suficientes para serem desenvolvimentos ao longo de um *sprint*.
- *Deliverable* é parte do *software* que já pode ser validada após o desenvolvimento do *sprint*.

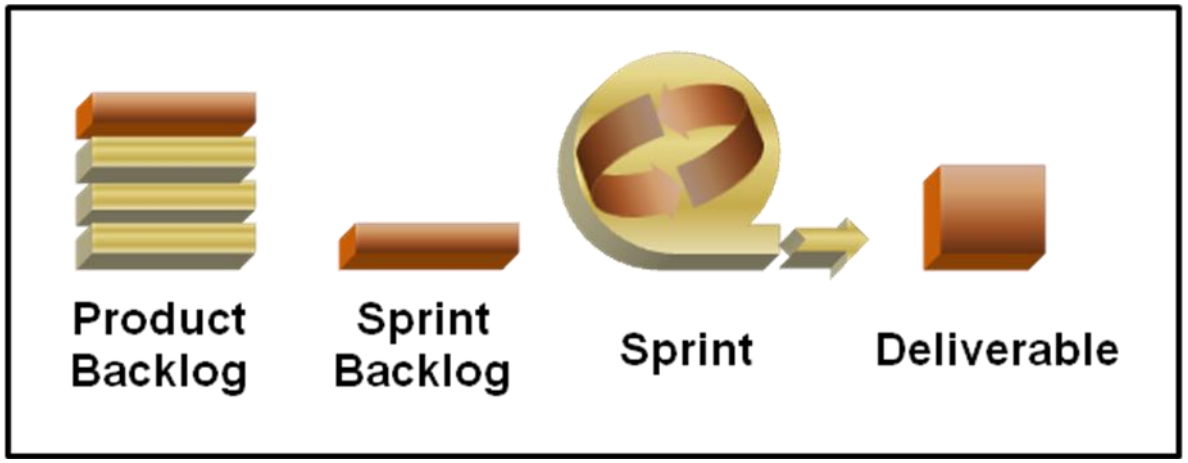

Figura 1 Ciclo de desenvolvimento Scrum
Fonte: Elaborado pelo autor

No início do projeto, os requisitos não estão, necessariamente, todos especificados. Podendo, até mesmo, não serem todos conhecidos. Assim, obter um plano detalhado de projeto neste momento torna-se tarefa difícil de ser realizada. A possibilidade de particionar o *product backlog* em *sprint backlog* permite concentrar a atenção nos requisitos que serão imediatamente desenvolvidos no próximo *sprint*. O *product backlog* não é uma lista de requisitos definitiva, completa, priorizada e nem tão pouco detalhada, mas é a lista que possibilita ter uma visão geral do que deve ser construído.

Especificar os requisitos o mais próximo do início do *sprint* permite à empresa decidir sobre o que efetivamente deve ser implementado. Qualquer necessidade de mudança ou alteração em regras previamente adotadas podem ser incorporadas no planejamento do *sprint* seguinte. A absorção de novas demandas pode ser analisada criteriosamente, pois implica em aumento de custo e tempo de desenvolvimento.

Mudanças ao longo do *sprint* devem ser evitadas, pois podem implicar parar o *sprint* e recomeçar o planejamento e desenvolvimento, desperdiçando tudo que já havia sido iniciado no *sprint* em questão.

Para Schwaber e Beedle (2001), Scrum é a “arte do possível” na qual o foco é o que pode ser feito no *sprint* atual uma vez que o *product backlog* é dinâmico. O *Scrum* requer grande colaboração entre a gerência, o cliente e a equipe de desenvolvimento para que os itens do *product backlog*, apesar de não estarem todos

definidos e especificados, sejam pelo menos conhecidos por todos para uma gestão de prazo e custo eficientes

O trabalho de gestão no processo *Scrum* deve ser pautado por equilibrar o orçamento disponível e o valor que o *software* pode aportar ao negócio do cliente. Com o modelo tradicional de desenvolvimento de *software* , os requisitos são planejáveis e as mudanças são tratadas à parte. No entanto, com o modelo de desenvolvimento ágil, há flexibilidade no conjunto de requisitos, porém a flexibilidade deve estar calcada em um planejamento adequado de custos. Aplicar o *Scrum* não é instituir o caos mas, sim, adaptar-se à capacidade da organização de se reinventar e priorizar o que adiciona valor ao seu negócio (SCHWABER; BEEDLE, 2001).

A adoção do *Scrum* muda a cultura da empresa. A mudança de cultura é sempre difícil pois altera valores, crenças, regras, papéis e práticas, porém necessária quando se acredita que são as pessoas que constroem bons produtos e não a estrutura (HIGHSMITH, 2004).

2.1.2. Equipe do *Scrum*

Na equipe de desenvolvimento *Scrum* destacam-se três papéis distintos: *Product Owner, Scrum Master* e *Scrum team.*

O *product owner* (*PO*) é o responsável pela definição do *product backlog*. Ele é quem conhece todos os requisitos e os prioriza em cada *sprint backlog*. Qualquer dúvida a respeito do *backlog* é respondida por ele.

O *Scrum master* (*SM*) é o líder que vai garantir que o processo *Scrum* esteja corretamente implementado. Ele é o elo entre a gestão e o time de desenvolvimento.

Scrum team é o termo usado para denominar a equipe de desenvolvimento. A equipe de desenvolvimento é responsável por entender os requisitos do *sprint backlog* e implementá-los em um *software*. A equipe deve possuir conhecimentos múltiplos e uma importante característica necessária a esta equipe é seu poder de auto-organização (SCHWABER; BEEDLE, 2001).

A equipe mantém-se alocada de maneira constante, sem mudanças na quantidade dos desenvolvedores e, desta forma, define-se a capacidade produtiva da equipe. A partir desta capacidade é que se limita o número de requisitos a serem

implementados. A diretriz não é decidir o que deve ser feito, mas decidir o que é possível ser feito diante da capacidade produtiva disponível.

O gerente de projeto atua ao lado do *product owner* na tentativa de priorizar e deixar o *product backlog* especificado e definido para ser implementado. A priorização ou a ordem de implementação das funcionalidades é uma questão fundamental para se atingir um dos objetivos da metodologia *Scrum*: entregas freqüentes de *software* testável.

2.1.3. Cerimônias

Para se desenvolver *software* utilizado a metodologia *Scrum*, devem ser realizadas reuniões. Estas reuniões são denominadas de cerimônias pelo *Scrum,* pois são reuniões pré-determinadas e com objetivos específicos. São elas: *sprint planning meeting*, *daily scrum meeting*, *sprint review meeting* e *sprint retrospective meeting*.

As *Sprint Planning Meetings* são reuniões de planejamento do *sprint*. Nelas o *product owner* (*PO*) é o responsável por explicar detalhadamente cada requisito do *sprint backlog*. À equipe de desenvolvimento, ficam reservadas as perguntas, questionamentos e dúvidas quanto à explicação do requisito. Após o entendimento do *sprint backlog*, a equipe de desenvolvimento reúne-se e verifica a possibilidade de se implementar todos os requisitos da forma como foram explicitados. Caso o esforço de implementação seja maior que a capacidade produtiva da equipe, o *PO* deverá priorizar alguns requisitos em detrimento de outros. Alternativamente, o *PO* poderá selecionar um novo requisito para ser implementado no *sprint* em questão.

As *Daily Scrum Meetings* são reuniões diárias que ocorrem com o intuito de verificar se o compromisso assumido no dia anterior foi realizado, de conhecer o que será realizado no dia e de identificar algum impedimento para a realização do plano assumido na *planning meeting*. Estas reuniões reforçam o princípio ágil de colaboração, pois todos tomam ciência do que está sendo feito por todos e contribuem para evitar a duplicidade de trabalho e até mesmo, a construção indevida de alguma parte do sistema.

As *Sprint Review Meetings* são reuniões de revisão e objetivam demonstrar o que foi desenvolvido. A presença do *PO* é importante para verificar se o que foi

especificado nas *sprint planning meetings* foi de fato implementado. Deve-se concentrar nos produtos passíveis de teste ou demonstração. Esta cerimônia também é conhecida como *demonstration meeting*. A demonstração e a validação do funcionamento dos requisitos implementados ao final de cada *sprint* são fundamentais para identificar possíveis novas melhorias que poderão ser executadas nos *sprints* futuros. Esta verificação antecipada permite que, ao final de todos os *sprints*, o produto entregue seja de fato o produto que atende, corretamente, às necessidades do *PO*.

As *Sprint Retrospective Meetings* são reuniões de retrospectiva que analisam o que se passou no *sprint* finalizado. Visam tirar as lições aprendidas e levantar pontos de melhoria do processo como um todo. Esta cerimônia incentiva a busca constante de melhoria, pois o objetivo é eliminar o que não foi bem sucedido e incentivar o que de melhor foi executado.

2.1.4. *Release sprint*

Ao final de cada *sprint*, um conjunto de funcionalidades pode ser testado. Normalmente, estes testes ocorrem em ambiente de teste. Aliado à filosofia ágil de entrega constante de funcionalidades, é importante que os requisitos entregues passem para o ambiente produtivo com o intuito de que seus benefícios já possam ser utilizados pela empresa.

A política para acessar o ambiente produtivo é geralmente normatizada nas empresas e exige um planejamento prévio. Tendo em vista, a necessidade de cumprir procedimentos para transporte ao ambiente de produção e a quantidade de funcionalidades aptas a serem implementadas, há um *sprint* específico para realizar este transporte para o ambiente de produção, denominado *release sprint*. Cada organização define a sua necessidade de encaminhar os desenvolvimentos para o ambiente de produção, não há número definido de *sprints* a serem empacotados dentro de um mesmo *release sprint*. No entanto, é importante manter a dinâmica de entregar funcionalidades para atender as necessidades dos usuários o quanto mais rápido for possível, para que o usuário perceba adição de valor ao seu negócio.

2.1.5. Ferramentas de gestão do *Scrum*

Uma das ferramentas bastante disseminada em projetos de *software* que utilizam metodologia ágil *Scrum* é o gráfico *burndown* do *sprint* (SLIGER, 2009). Este gráfico reflete o número total de horas disponíveis para finalizar um *sprint*. Este gráfico provê, de maneira rápida e visual, a situação real do *sprint*. Pode-se também fazer um gráfico de *burndown* de todo o projeto.

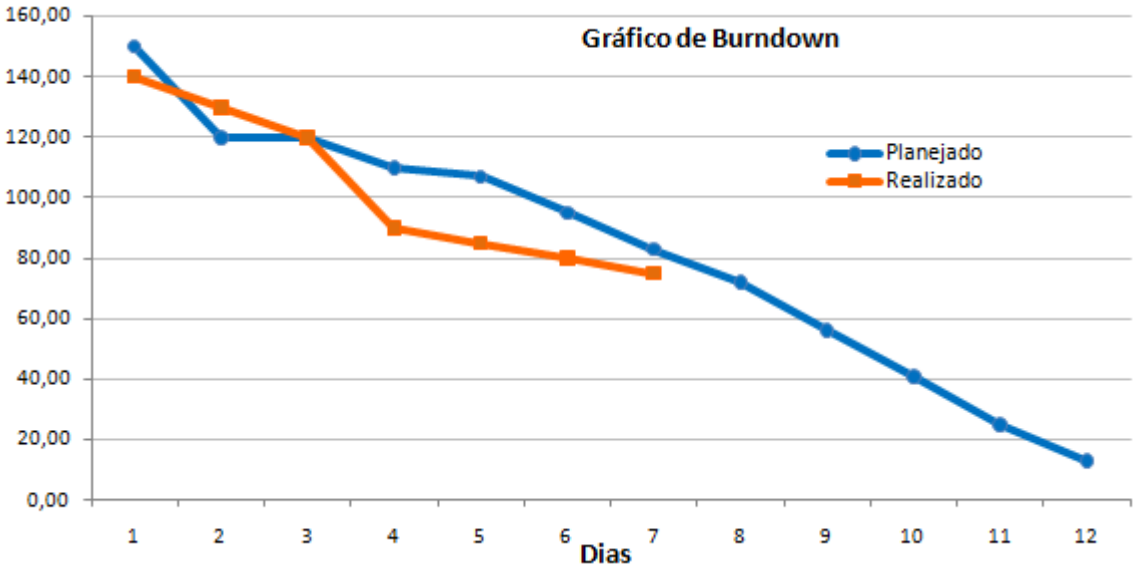

Figura 2 Gráfico Burndown
Fonte: Elaborado pelo autor

À medida que o projeto vai sendo implementado, este gráfico deve ser atualizado com a curva real. Caso a curva esteja abaixo da curva de referência significa que a equipe terá uma possibilidade de adiantar o fim do *sprint*. A curva real estar acima da curva de referência, por outro lado, significa que a equipe está trabalhando mais horas nas atividades do *sprint*. Nenhum dos dois casos é desejável, mas servem para balizar e encontrar a verdadeira capacidade de entrega da equipe.

Outro gráfico interessante acompanha a complexidade estimada dos itens acordada durante a *planning meeting* ao longo do *sprint*. Sliger e Broderick (2009) nomeiam este gráfico como “*burnup*” e enfatizam que este gráfico ajuda a visualizar a tendência e o desempenho da equipe.

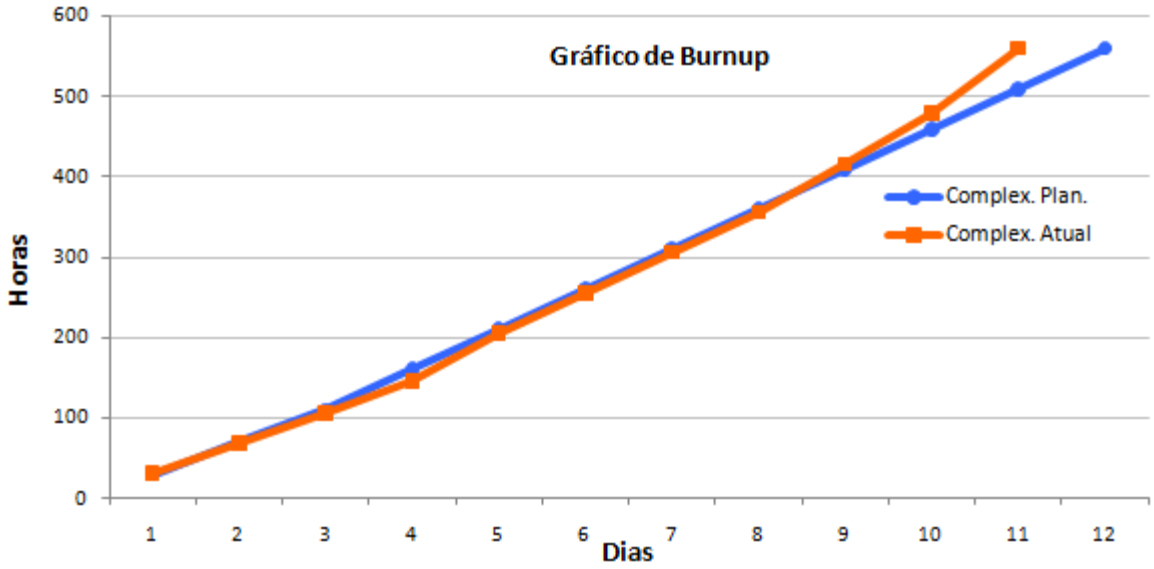

Figura 3 Gráfico Burnup
Fonte: Elaborado pelo autor

A atualização e o acompanhamento diário destes gráficos permitem tomadas de decisões e necessidades de negociação tão logo a curva de execução não esteja alinhada com a curva planejada, ou desvie mais do que um máximo aceitável.

2.1.6. Vantagens do *Scrum*

A vantagem do Scrum está em reunir um conjunto coerente de práticas que por meio de reuniões constantes visa reduzir o tempo de resposta a eventual problema ou impeditivo à execução do planejamento inicial dentro do *sprint*, bem como à execução do plano de projeto. A preocupação em ter ações rápidas diante de um desvio, a disseminação de boas práticas, bem como o alerta às condutas indevidas, são contribuições destas reuniões. No entanto, incorporá-las no dia a dia de um projeto passa por uma questão de cultura organizacional. As pessoas têm que apoiar e acreditar no benefício destas práticas.

Tendo em vista as reuniões constantes que a metodologia preconiza, a equipe age como uma equipe coesa; todos são envolvidos e participam das decisões e explicações das necessidades de negócio que devem ser implementadas. Este contato direto do time de desenvolvimento com o cliente acelera o entendimento das necessidades do negócio e influencia positivamente na construção do sistema.

O desenvolvimento do sistema em *sprints* fornece uma visualização das necessidades atendidas em um espaço de tempo curto e possibilita acomodar mudanças de escopo. Mudanças solicitadas em um *sprint* podem ser incorporadas no ciclo seguinte. Além disso, reduzir o tempo de implantação das funcionalidades por meio dos *release sprints* traz um retorno imediato à organização, uma vez que ela pode se beneficiar dos produtos do *sprint* à medida que eles vão sendo desenvolvidos.

Um pré-requisito para introduzir o *Scrum* é ter um alinhamento entre os níveis estratégicos, táticos e operacionais da organização (MOE, 2008). Todos, de desenvolvedores a usuários, gerentes a diretores de negócio, devem estar envolvidos e fazer parte da mudança. A participação do cliente, que é quem efetivamente conhece o negócio e vai usar o sistema, é fator decisivo no sucesso da implantação do *Scrum*. A vantagem de se ter um *product owner* , que responde pelo negócio de maneira rápida e participa das decisões com a equipe, proporciona um ambiente de desenvolvimento com maior previsibilidade dos requisitos. Essa maior confiança nos requisitos é fundamental para evitar problemas de qualidade de *software* causado, muitas vezes, pela falta de definição específica do que deve ser implementado.

Um aspecto que favorece a força competitiva do *Scrum* refere-se a requisições de mudanças. Metodologias de desenvolvimento tradicionais são limitadas quanto ao tratamento de requisições de mudanças após o início de um projeto (IONEL,2008). Já na metodologia *Scrum*, as requisições de mudança são absorvidas com uma das atividades do *sprint backlog* e priorizadas de acordo com a visão do *PO*. Há oportunidade para mudança tão logo ela seja identificada, e não apenas no final do processo, quando o sistema já está desenvolvido. Esta flexibilidade em gerenciar o escopo é benéfica para que o projeto acomode de fato as necessidades do negócio. No entanto, é fundamental fomentar a gestão do escopo de forma a priorizar de fato o que fortalece e contribui para o negócio dentro de um orçamento limitado. A gestão de escopo é um ponto explorado nesta pesquisa, de forma a contribuir positivamente na aplicação do *Scrum*.

Uma das características do *Scrum* é possuir equipes relativamente pequenas em *sprint*. A recomendação é que a equipe de desenvolvimento não ultrapasse 10 pessoas, e que tenha um conhecimento multifuncional (SCHWABER; BEEDLE,

2001). Em equipes pequenas, a comunicação e integração podem ser facilitadas pela proximidade entre todos os participantes, além de proporcionar várias possibilidades de aprendizado, tendo em vista a multidisciplinaridade da equipe (ASPRONI, 2006) (IONEL, 2008). O planejamento dos recursos é feito baseado em uma expectativa de esforço a ser desenvolvida, ou seja, uma capacidade produtiva que será utilizada para executar as atividades. Não são as atividades que direcionam o dimensionamento da equipe. Dessa forma, é novamente importante a gestão do escopo, evitando manter a capacidade ociosa da equipe.

Sprints são características positivas no *Scrum*. Eles são a menor unidade de planejamento possível durante o desenvolvimento do *software*. A entrega de funcionalidades a cada iteração (*sprint*) favorece o contato rápido com o *software*, cujas decisões de negócio foram relatadas pelo *PO* no *sprint planning meeting*. É recomendável que a duração do *sprint* não seja superior a 30 (trinta) dias para não quebrar a fluidez e dinamismo do processo de interação entre as pessoas envolvidas no projeto (SCHWABER; BEEDLE, 2001).

Após cada *release sprint*, os requisitos implementados já estão disponíveis para utilização pela área usuária. O atendimento rápido às necessidades do negócio é um fator que reforça o aumento da satisfação dos clientes com a adoção do *Scrum* (SCHWABER; BEEDLE, 2001) (IONEL, 2008).

2.1.7. Dificuldades no uso do *Scrum*

Nas primeiras reuniões de *sprint planning*, o conhecimento da equipe é muito básico, pois não há uma fase específica de conhecimento do negócio ou até mesmo levantamento de requisitos. À medida que as iterações avançam, o conhecimento da equipe no negócio é maior e, portanto, o entendimento dos requisitos leva a uma melhor estimativa de esforço de implementação de cada *sprint* (SCHWABER; BEEDLE, 2001). Desta forma, a capacidade produtiva de uma célula *Scrum* é variável e tende a se estabilizar após um determinado período, quando o conhecimento do negócio tiver sido consolidado. A possibilidade de realizar um levantamento de requisitos ágil antes do início de desenvolvimento do primeiro *sprint*, para garantir uma previsibilidade constante de esforço do time, pode ser um caminho para abreviar este período de aprendizagem.

O não envolvimento do responsável pelas definições de negócio pode atrasar a definição do *sprint backlog* e, até mesmo, impactar o andamento do *sprint*, caso as decisões impeçam a continuidade do desenvolvimento. Se faltarem ao *PO* conhecimento do negócio e poder de decisão, qualquer tentativa de estimativa de projeto pode ser inviabilizada já que a previsão será sempre baseada em fatos incompletos ou imprecisos. A disponibilidade do responsável pelo negócio é fator preponderante para o sucesso da aplicação do processo de desenvolvimento ágil *Scrum* (IONEL, 2008).

Como as iterações são muitas e constantes, encontrar uma equipe coesa e com conhecimento múltiplo é sempre tarefa difícil nas organizações. Profissionais competentes que tenham habilidade de comunicação são fundamentais para suportar um processo de desenvolvimento ágil (HIGHSMITH, 2004).

A falta de conhecimento especializado entre os membros da equipe pode ser um potencial problema. O tempo de resposta e produtividade de cada um são individualizados. *Web designers*, por exemplo, são mais produtivos realizando atividades de *design* do que de *testers*. Sendo assim, a produtividade da célula normalmente é medida pela capacidade de entrega em situação normal, não em situações nas quais todos os papéis estão exercendo atividades nas quais são menos produtivos. A alocação dos recursos em determinadas atividades dentro de um *sprint* influencia bastante na estimativa e controle de esforço de uma célula *Scrum* (IONEL, 2008).

A gestão do escopo é fundamental para garantir a melhor produtividade de cada recurso dentro dos *sprints.* Como a equipe é multidisciplinar, se em determinado sprint for concentrado apenas esforço de prototipação de tela de sistemas, provavelmente os desenvolvedores especialistas em determinada linguagem de programação não serão tão efetivos e rápidos nas atividades quanto um *web designer*, proporcionando desta forma uma baixa produtividade da célula de desenvolvimento e, consequentemente, um desperdício financeiro.

A falta de uma diretriz clara de gestão de custos no *Scrum* é um das dificuldades para adoção desta metodologia. A flexibilidade para acomodação de mudanças de requisitos pode ser associada ao não controle do orçamento destinado ao projeto.

2.2. Gerenciamento de custos

O gerenciamento de custos é uma disciplina relevante na condução de projetos de qualquer natureza. O gerenciamento de custos envolve o processo de estimativa e controle de custos (PMBOK, 2004). Em termos de custos, os produtos de software contêm uma complexidade específica que outros produtos não têm. Booch (1994) afirma que a complexidade do *software* é inerente ao processo e justificada através de quatro pontos: a complexidade do domínio do problema, a dificuldade em gerir o processo de desenvolvimento, a flexibilidade esperada e os problemas que caracterizam o comportamento dos sistemas.

Assaf Neto (2002) sintetiza que “a principal característica de diferenciação das empresas é a forma como as decisões econômicas são tomadas”. Uma das medidas que norteia estas decisões é a minimização de seus custos. Os custos influenciam diretamente o desempenho econômico da empresa levando a perdas quando mal gerenciados.

Nas empresas, a maioria das metas está ligada ao desempenho financeiro; o gerenciamento de custos permite potencializar ganhos, refletindo positivamente no lucro líquido da organização (FALCONI, 2009).

Prahalad e Hamel (1990) enfatizam que a competitividade de uma empresa deriva de uma capacidade de formar, a custos menores, as competências essenciais que a diferenciam da concorrência. Desta forma, monitorar e gerir os custos permitem tomadas de decisão e planos de ação que geram vantagens competitivas.

As reais fontes de vantagem competitiva encontram-se na capacidade de traçar ações para que as empresas consolidem seus negócios sem que a estimativa de custos seja excedida, e até mesmo podendo ser reduzida ou otimizada.

A estimativa dos custos associados a um projeto deve ser realizada a fim de estabelecer um orçamento para o projeto e compor um preço para o cliente. A formação do preço considera também aspectos sociais, políticos, econômicos e estratégicos da organização.

Existe uma diferença entre preço e custo. O preço é o resultado da soma entre os custos do projeto e o lucro que se pretende praticar pela venda de um produto ou serviço (SOMMERVILLE, 2003). Já Nóbrega (2010) enfatiza que “preço é

o mercado que determina e custo é você quem determina" e a diferença entre preço e custo é valor. Se o valor trouxer o benefício esperado pelo negócio, o projeto é justificável. Este trabalho não pretende abordar a questão preço de um projeto de *software* ficando única e exclusivamente focado na abordagem de custo do projeto de *software*. Para evitar quaisquer questionamentos a respeito da palavra preço, nesta dissertação, o preço de um *software* será considerado igual ao custo deste *software*. Desta forma, caso haja alguma referência sobre a palavra preço fechado, uma terminologia conhecida no mercado de *software* como uma forma de contratação de um projeto de *software*, a atenção deve sempre estar voltada à meta de orçamento de um determinado cliente, à quantidade monetária que ele quer investir no desenvolvimento do *software*. Projetos com esta clara visão da meta de custos a ser garantida precisam de uma estimativa e controle de custos bastante apurados a fim de garantir maior confiabilidade dos clientes nos contratos de projetos de desenvolvimento de *software*.

Sommerville (SOMMERVILLE, 2003) cita três fatores envolvidos no cálculo total de custos de um projeto de desenvolvimento de *software*:

- custos de infraestrutura (*hardware*, *software*)
- custos variáveis (despesas com viagem, treinamento da equipe de desenvolvimento)
- custos fixos (custo dos recursos envolvidos)

O gerenciamento do custo é uma diretriz de qualquer empresa, pois contribui diretamente no resultado operacional. Projetos com preço fechado deixam clara a meta orçamentária, pois o orçamento é igual ao custo estimado no início do projeto. Porém, um eventual desvio na estimativa inicial pode ser prejudicial à empresa que não consegue repassar seus custos. Projetos contratados com a abordagem *time and material* também possuem orçamentos limitados e permitem acomodar a flexibilidade de mudanças de requisitos e estratégias sem a necessidade de renegociação do contrato (FRANKLIN, 2008). Contratos *time and material* são contratos cobrados de acordo com unidades de trabalho e o valor a ser faturado é a quantidade de unidades necessárias para se concluir o trabalho. No entanto, contratos *time and material* são difíceis de serem vendidos por causa da incerteza do cliente quanto ao fim do projeto e da possibilidade de desvio de orçamento (ECKFELDT, 2005).

Encontrar o equilíbrio entre a meta orçamentária dos contratos preço fechado e a flexibilidade para adaptação dos contratos *time material* é um desafio que será perseguido nesta pesquisa.

2.2.1. Estimativa de custo de *software*

A estimativa dos custos de um projeto de *software* deve se basear nas funcionalidades do *software*. Cada funcionalidade exige um esforço específico para ser implementada, testada, integrada e homologada. Em geral, há um documento de requisitos de sistemas elaborado que resume o que deve ser desenvolvido. A partir desta visão geral dos requisitos, equipes especialistas são alocadas para estimativa de esforço e consequentemente dos custos do projeto (SOMMERVILLE, 2003).

"Estimar é prever à luz da incerteza e conhecimento incompleto" (BOEHM, 2000). Ferramentas e técnicas podem auxiliar na atividade de estimativa de custos. Ter uma base histórica de estimativas de projetos semelhantes e aplicá-la no projeto atual através de uma analogia, é uma das técnicas de estimativa

A estimativa *botton*-up prevê a divisão de uma funcionalidade em atividades menores e a estimativa é realizada para cada atividade individualmente, sendo a soma do esforço de cada atividade, a estimativa de custos daquela funcionalidade. A estimativa paramétrica é uma técnica que utiliza um modelo matemático considerando dados históricos e parâmetros característicos do projeto, como por exemplo tamanho do *software* (SOMMERVILLE, 2003) (PMBOK, 2004). A estimativa de três pontos baseia-se em estimar três cenários: mais provável, otimista e pessimista e a média aritmética das três estimativas passa a ser a própria estimativa (PMBOK, 2004).

Um dos primeiros métodos a estimar o desenvolvimento de *software* com alguma precisão e métrica é conhecido como Pontos de Função (TAVARES, 2002). Os pontos de função não dependem do tipo de linguagem de programação em que o sistema será desenvolvido e são mais apropriados aos sistemas de processamento de dados com grande volume de entradas e saídas. O número de ponto de função é estimado considerando as entradas e as saídas externas, as interações com os usuários, as interfaces externas e os arquivos utilizados pelo sistema. Cada uma

dessas características é avaliada em termos de complexidade e recebe um valor ponderado. O somatório do número de elementos de determinado tipo multiplicado pelo seu peso fornece a medida da contagem de ponto de função não ajustada (*unadjusted function-point acount - UFC*). A contagem final de pontos de função é finalmente calculada através do produto entre a contagem de ponto de função não ajustada e o fator de complexidade do projeto. Esta complexidade está baseada em algumas características como, por exemplo, quantidade de reuso, grau de processamento distribuído e desempenho (SOMMERVILLE, 2003) . Portanto, este grau de complexidade pode variar de acordo com a opinião e conhecimento do profissional que está realizando a estimativa, permitindo certo grau de subjetividade à estimativa. Por esta razão, há diferentes opiniões sobre o valor de pontos de função (FUREY e KITCHENHAM, 1997).

A exatidão das estimativas de custos que utilizam um modelo matemático depende da assertividade das informações fornecidas sobre o sistema. O modelo algorítmico COCOMO (*constructive cost model* – modelo de custo construtivo) foi lançado, em 1981, por Boehm, tendo como base um sistema desenvolvido utilizando um processo em cascata e tendo como premissa um desenvolvimento sem aproveitamento de qualquer linha de código possível. No entanto, devido à evolução das metodologias de desenvolvimento de *software*, através de processos incrementais e introdução da reengenharia de *software* primando pelo reaproveitamento de código, o modelo foi aperfeiçoado para considerar tais mudanças através do COCOMO 2 (SOMMERVILLE, 2003). Dentre os atributos que devem ser estimados para o cálculo do algoritmo do esforço, podem ser citados o tamanho do *software*, o reuso requerido, a confiabilidade e complexidade do produto, a dificuldade da plataforma, a capacitação pessoal, prazo e recursos de suporte. As estimativas de custo são derivadas da estimativa de esforço e de atributos referentes ao produto, ao hardware, ao pessoal e ao projeto. As dificuldades para aplicação do modelo referem-se a estimar o verdadeiro tamanho do *software*, em fase inicial do desenvolvimento de *software*, e à subjetividade quanto à avaliação dos atributos de processo, produto e desenvolvimento.

Boehm (2000) afirma que os modelos e técnicas de estimativa são componentes essenciais da engenharia de *software* e, como o processo de

desenvolvimento sempre evolui, as técnicas de estimativa de custos devem necessariamente mudar.

A partir do manifesto ágil (BECK, 2001), a ênfase em *software* funcionando desencorajou o planejamento detalhado de todas as funcionalidades do sistema no início do desenvolvimento e ressaltou como valor a entrega parcial das funcionalidades priorizadas pelo *product owner* (SULAIMAN, 2006). O princípio ágil permite mudanças freqüentes nos requisitos, prejudicando bastante o processo de estimativa de custo do projeto (SHAHIR, 2008).

User story é o termo utilizado nos métodos ágeis para descrever funcionalidades desejadas do sistema a ser implementado. As *user stories* são bons exemplos da natureza mais dinâmica do desenvolvimento ágil, pois contemplam descrições simples das orientações sobre o que deve ser desenvolvido para cada funcionalidade (REIDEMANN, 2009) (REES, 2002). A partir das *user stories*, a equipe de desenvolvimento ágil estima o esforço de implementação baseado em experiência, dados históricos, referências e, ao final, a estória recebe uma pontuação denominada *story points*. As estimativas de esforço em desenvolvimento de *software* ágil serão medidas por *story points*.

As *user stories* contêm definições bastante objetivas de negócio. Estas estórias são divididas em tarefas de desenvolvimento e, desta forma, a equipe inicia a atividade de estimativa de esforço. A equipe também se baseia em analogia e, desta forma, a menor estória possível é constantemente considerada para efeitos de pontuação das estórias seguintes. Há sempre uma comparação entre o esforço da estória que está sendo pontuada e a estória de referência. Todas as estórias são pontuadas por todos da equipe, de maneira individual, e a pontuação final é obtida quando todos chegam a um consenso sobre a mesma complexidade de pontuação.

A medida inicial capturada pelas *story points* é a base para o cálculo da estimativa inicial de cada funcionalidade listada no *product backlog* e, consequentemente, cálculo da estimativa inicial do projeto.

O método de estimativa sugerido por McConnell (2006) consiste em definir um tamanho relativo de “valor de negócio” e “custo de desenvolvimento” para os itens a serem desenvolvidos. Os tamanhos relativos assemelham-se à numeração de camisas P, M, G (*t-shirt size*). Desta forma, cada funcionalidade tem duas classificações P, M, G, uma referente ao valor de negócio e outra ao custo. O

relacionamento entre estas medidas proporcionam o resultado do valor de negócio líquido. Este modelo apóia, principalmente, os *stakeholders* não técnicos a estabelecer uma prioridade dos requisitos.

Benefício/valor ao negócio	**Custo do desenvolvimento**		
	G	M	P
G	0	2	3
M	-2	0	1
P	-3	-1	0

Tabela 1 Pesos das classificações de benefício e custo
Fonte: McConnell, 2006

Observa-se que no modelo proposto por McConnell (2006) o peso da funcionalidade que tem um custo grande, mas um benefício médio é -2. A funcionalidade não é muito relevante, pois seu peso é baixo. Na tabela de Pesos das classificação, o fator negativo não significa que o custo de implementação seja negativo.

Funcionalidade	Benefício	Custo	Classificação
A	G	M	2
B	P	M	-1
C	P	G	-3

Tabela 2 Classificação das funcionalidades com base no benefício e custo
Fonte: McConnell, 2006

Mc Connell (2006) recomenda que cada membro da equipe realize a estimativa individualmente, mas chegue a um consenso em conjunto. O foco para a realização desta classificação deve ser inicialmente a estimativa de esforço, e a partir desta estimativa passa-se a computar o esforço, prazo e custo.

Em todas as técnicas de estimativa de *software* citadas o fator base histórica está presente. Quanto maior a experiência do desenvolvedor em projetos diversos, maior a sua sensibilidade e assertividade na proposta da estimativa. Desta forma, ela pode variar bastante de empresa para empresa e, dentro da mesma empresa, de grupo de desenvolvedores para grupo de desenvolvedores. Resumidamente, para se estimar os custos, é preciso inicialmente conhecer as demandas e os requisitos que serão especificados e desenvolvidos. Depois com base em experiência, dados

históricos e uma determinada técnica de estimativa, inicia-se o processo de estimativa de esforços que culminará na estimativa de custos. Uma estimativa pode conter alguns desvios e erros. Normalmente, se não há margem de negociação e revisão das estimativas, o custo da incerteza é considerado para acomodar eventuais desvios. No entanto, neste trabalho o custo da incerteza não será considerado, pois ele apenas eleva o patamar de custos de um software, ou seja, o controle seria realizado em cima deste novo patamar e isto não contribuiu na otimização da relação estimado e orçado.

A estimativa do custo de desenvolvimento de *software*, no início do projeto, independente do modelo de desenvolvimento adotado, deve ser feita para calibrar expectativa de meta orçamentária a ser investida pelo cliente. Estimativas realizadas após o início do projeto passam a ser mais detalhadas a partir da melhor compreensão das funcionalidades e conhecimento do negócio. Ambas afirmativas são verdadeiras para sistemas desenvolvidos utilizando modelo ágil. No entanto, em equipe ágil, a equipe deve obrigatoriamente dividir o processo de estimativa entre os envolvidos; todos devem ter conhecimento das estimativas de todos (DERBIER, 2003).

2.2.2. Introdução ao controle do custo de *software*

Uma importante parte do controle de custos envolve medir o que foi realizado, verificar possíveis variações com o planejado e decidir se um plano de ação deve ser praticado para evitar desvios quanto ao orçamento definido inicialmente (MULCAHY, 2005).

Para se controlar os custos, é importante conhecer perfeitamente a sua natureza e fonte geradora, pois a partir deste controle dos custos é possível propor ações corretivas e preventivas de forma a evitar uma nova ocorrência (FALCONI, 2009).

O controle dos custos pode influenciar o escopo. Para se ajustar ao orçamento, pode-se reduzir as funcionalidades a serem desenvolvidas ou demonstrar a necessidade de aumento do investimento em um projeto de *software*. O controle dos custos é importante para que a organização não seja surpreendida por resultados inesperados.

A análise do valor agregado *EVA (Earned Value Analysis)* é comumente utilizada para controlar os custos em projetos de desenvolvimento de software gerenciados de acordo com as práticas do PMBOK (PMBOK, 2004). Por meio deste modelo, tem-se a visibilidade dos custos realizados ao longo do projeto, do desvio de custos referentes à estimativa inicial realizada e da projeção de custo final. Com estes dados, decisões podem ser tomadas para melhorar a eficiência do projeto

2.2.3. Análise do valor agregado

O uso da técnica *EVA* integra custo, prazo e escopo e pode ser usado para prever desempenho futuro do projeto (MULCAHY, 2005). A metodologia de gerenciamento do valor agregado (*Earned Value Management – EVM*) é baseada em um planejamento inicial bem definido, com uma estrutura bem detalhada das atividades, associadas a um custo e prazo (SULAIMAN, 2006) (SAKAMOTO, 2006).

A análise do valor agregado fornece meios para prever variações de prazo e custos através de três medidas: *PV – planned value*, *EV – earned value* e *AC – actual cost*.

O *BAC – budget at completion* – é o orçamento aprovado no início do projeto. A meta orçamentária que o projeto deve cumprir.

O *PPC – planned percentual concluded* – é o percentual de trabalho planejado acumulado.

O *PV – planned value* – representa o custo acumulado orçado do projeto até um determinado momento. (*PV = BAC * PPC*)

O *APC – actual percentual concluded* – é o percentual de trabalho realizado acumulado.

O *EV – earned value* – é a parcela do orçamento comprometida até um determinado momento. É calculada através do custo orçado e do percentual do trabalho realizado. (*EV = BAC * APC*)

O *AC – actual cost* – é o custo acumulado real do projeto.

O cálculo da variação de custo e prazo demonstra o desvio ocorrido no projeto e, respectivamente, são apresentados a seguir:

- *CV – cost variance* – é o desvio de custo do projeto
  - *CV = EV - AC*

- *SV – schedule variance – é* o desvio de prazo do projeto
  - *SV = EV – PV*

Além dos desvios, alguns índices são também calculados para apoiar a análise do projeto:

- *SPI – Schedule performance index* – é o índice de performance do projeto. Sendo este índice menor que 1, o projeto pode ser considerado como atrasado.
  - *SPI = EV / PV*
- *CPI – cost performance índex* – é o índice de performance de custo. Sendo este índice menor que 1, o projeto está gastando mais do que o orçado.
  - *CPI = EV / AC*

A utilização destas métricas permite uma análise do projeto em um determinado momento. São métricas obtidas de acontecimentos passados. A técnica de análise de valor agregado também fornece ferramentas para projetar resultados futuros dos projetos. A partir dos resultados elabora-se um plano de ação para aplicar uma decisão corretiva (PMBOK, 2004) (ALLEMAN, 2003).

- *ETC – estimated TO complete* – é o custo estimado que ainda será realizado até o fim do projeto, levando em consideração o histórico de desempenho do projeto.
  - *ETC = BAC – EV*
  - *EV = BAC * APC*
- *EAC – estimated AT complete* – é o custo estimado para terminar o projeto.
  - *EAC = ETC + AC*

Sakamoto (2006) conclui que “a utilização da técnica de análise de valor agregado encontra algumas barreiras devido à alta volatilidade dos cenários de desenvolvimento... e instabilidade do escopo”. Attarzadeh (2009) cita em seu artigo que a aplicação da técnica de análise de valor em projetos tradicionais é uma boa técnica utilizada para monitorar custos e tomar ações corretivas ao longo do seu desenvolvimento. No entanto, ele ressalta que é esperado que a mudança de escopo seja mínima.

A partir das duas citações, projetos de *software* que utilizam metodologia de desenvolvimento ágil dificilmente poderiam se utilizar dos benefícios da técnica de *análise* de valor agregado. Pois o escopo em projetos ágeis oscila bastante, além de não serem conhecidos em detalhes no início do projeto. Porém, adaptar a aplicação da técnica aos métodos ágeis pode ser um ganho no controle de custos de projetos ágeis.

2.2.4. Análise de valor agregado ágil

Para Schwaber e Beedle (2001), todo projeto de desenvolvimento de software é limitado por quatro variáveis: tempo, custo, qualidade e escopo. O custo considera mão de obra, o ambiente de desenvolvimento e outros recursos que desempenham atividades no projeto para eliminar os impedimentos. Um projeto de *software* deve considerar não apenas o custo unitário do *sprint*. O custo a ser controlado é o custo de todos os *sprints* do projeto. Controlar estes *sprints*, de forma que, ao final do projeto, as necessidades de negócio especificadas tenham sido atendidas ao custo planejado, é a grande missão do controle de custos.

Controlar custo planejado com custo realizado é medir apenas o esforço dispensado durante um período de tempo. O aspecto crítico da análise de valor agregado está centrado na idéia de entrega de valor (*EV* – parcela do orçamento consumida até o momento) com o tempo ou custo investido (*AC* – *actual cost*) no desenvolvimento do *software* (ALLEMAN, 2003). Se a entrega de valor é igual ou superior ao custo atual, o produto entregue está na meta ou acima da meta orçamentária, respectivamente.

Sulaiman (2006) propõe uma adaptação do gerenciamento por valor agregado para aplicação em projetos de *software* ágeis, especialmente utilizando metodologia *Scrum*. Esta proposta é chamada de gerenciamento ágil por valor agregado ágil – *Agile EVM*.

Para aplicação da técnica, o escopo deve ser inicialmente conhecido. Em projetos ágeis, o escopo é listado no *product backlog*. No entanto, ele é melhor explicitado no início de cada *release*, quando ao final do release entrega-se *software* funcionando. O escopo de cada atividade é finamente detalhado no início de cada *sprint*. Qualquer mudança solicitada é analisada no início de cada *sprint* e o plano do

release é então revisto para analisar se a alteração pode ser suportada pelo orçamento definido para aquele *release*. Devido ao melhor detalhamento dos requisitos no início do *release* é interessante aplicar a técnica para controlar o trabalho em cada *release.*

Foram definidos alguns parâmetros para acompanhamento:

Parâmetros do *Release*	
BAC	meta orçamentária do sprint, o controle será realizado pelo custo total do release
L	tamanho de cada *sprint.*
PS	número de *sprints* planejados no *release*
SD	a data de início do *release*
PRP	total de pontos (tamanho do esforço) planejados para o *release*
Parâmetros do *Sprint*	
N	número do *sprint* (classificação começa com 01)
PC	quantidade de esforço ou pontos entregues do product backlog no sprint
PA	quantidade de pontos adicionados ou retirados do release backlog durante o sprint

Tabela 3 Parâmetros de Acompanhamento
Fonte: SULAIMAN, 2006

Na tabela 4, há uma comparação entre os termos do gerenciamento de valor agregado (*EVM)* e do gerenciamento de valor agregado ágil (Agile*EVM)*.

Termo	***EVM***	***AgileEVM***
Prazo	Duração total do projeto	Total dos sprints (*PS)* multiplicado pelo tamanho do sprint (*L).*
Orçamento (*BAC)*	Orçamento total do projeto	Orçamento do somatório de todos os *releases*.
Percentual planejado (*PPC)*	% completo esperado em determinado momento do projeto	Número do sprint atual (*n)* dividido pelo número total de sprints planejados (*PS)*
Percentual concluído (*APC)*	Custo do trabalho realizado dividido pelo custo total do projeto	Número de *story points* entregues dividido pelo total de *story points* planejados.

Tabela 4 Comparação entre termos *EVM*
Fonte: SULAIMAN, 2006

Detalhes sobre todas as equações válidas para o *AgileEVM* obtidas pela adequação das fórmulas do *EVM* Tradicional podem ser encontradas no artigo de SULAIMAN (2006).

2.2.5. Conclusão

A adoção do modelo *t-shirt size* proposta por McConnel (2006) agrega a idéia de valor de negócio e custo ao mesmo tempo. Estas duas variáveis são importantes mas não devem ser classificadas ao mesmo tempo. Além disto, para a classificação de custo, McConnel propõe que inicialmente seja realizada a análise de esforço da complexidade de implementação e somente a partir deste esforço deriva-se o custo. As avaliações devem ser feitas por pessoas distintas, área de negócio e equipe de TI. Funcionalidades de *software* podem variar bastante quanto ao grau de complexidade e esforço de implementação. Considerar apenas três medidas para classificá-las (P, M, G) pode limitar e distorcer a análise.

Sulaiman (2006) propõe uma adaptação ao modelo de gerenciamento do valor agregado, *EVM*, usando conceitos definidos no *Scrum* para controle dos custos de cada iteração, individualmente, de um projeto de desenvolvimento de *software*. Ela não ressaltou a aplicação das técnicas de *EVM* para todo o projeto contemplando todas as iterações. No entanto, para efeitos de gestão orçamentária do projeto é fundamental ampliar o foco de visão para todo o projeto e não somente para as *release sprints* do projeto.

A ativação de valor no pensamento do dia-a-dia de qualquer projeto deve ser realizada a ponto de sugerir que os itens de *backlog* a serem desenvolvidos estejam de fato alinhados à estratégia de valor, que gera maiores benefícios de negócio para o cliente.

O alinhamento entre a gestão de custo e os benefícios de valor para o negócio deve ser buscado a fim de garantir que o projeto não exceda o orçamento mas também garanta que as funcionalidades implementadas, de fato, são úteis e trazem benefícios para á área usuária.

3. ANÁLISE DAS TÉCNICAS DE ESTIMATIVA DE CUSTO E VALOR AGREGADO

Em projetos de desenvolvimento ágil de *software*, os requisitos são especificados ao longo do projeto. No início do desenvolvimento, há uma noção dos itens de *backlog* mas não o completo detalhamento dos mesmos. Desta forma, toda tentativa de estimativa de esforço total nos projetos ágeis ser uma previsão com grande incerteza.

Apesar das estimativas conterem um grau de incerteza, elas são imprescindíveis para guiar o esforço de prazo e custo. O orçamento de uma área para determinado projeto é finito e acomodar as expectativas (estimativas) com o orçamento determinado (custo) é fundamental para não desviar do objetivo estratégico da empresa.

A sugestão de uma técnica de estimativa do *backlog* será proposta de maneira a adequar o processo de estimativa de requisitos de *software* ao conceito ágil de desenvolvimento *Scrum*, evitando desperdício de esforço e maximizando o valor que o processo pode aportar ao ciclo de desenvolvimento.

3.1. Adaptação das técnicas de estimativa e controle de custos

A proposta deste trabalho é consolidar um método para adequar os processos de estimativa e controle de custos a fim de formatar um processo de gestão de custo ágil.

A figura 4 apresenta o mapeamento da utilização de um novo método para realizar a estimativa de custos, *T-shirt size expanded,* e adaptação de uma técnica para realizar o controle de custos, *Earned Value Management*, de forma a contribuir à formatação de um processo de gestão de custos ágeis.

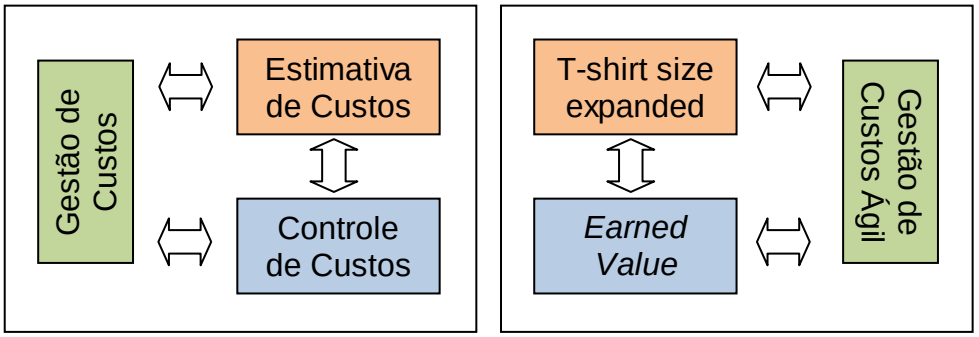

Figura 4 Mapeamento das técnicas utilizadas e suas interações
Fonte: elaborado pelo autor

3.2. Estimativa de complexidade

A identificação dos requisitos de *software* não é uma tarefa trivial. As dificuldades atreladas à identificação destes requisitos referem-se a questões de conhecimento da efetiva necessidade a ser atendida do próprio solicitante até as constantes mudanças de negócio motivadas interna e externamente à empresa.

Em metodologias orientadas a planejamento, as mudanças de requisito são normalmente negociadas à parte do orçamento inicial do projeto através de uma gestão de mudanças. Já em projetos ágeis, a incorporação de mudanças acontece ao longo do projeto e são absorvidas em iterações futuras.

Independente da metodologia de desenvolvimento a ser utilizada, um dos passos iniciais de um projeto de *software* é o levantamento dos requisitos básicos para formulação de uma proposta a fim de estimar prazo e custo. No *Scrum*, seguindo um dos princípios *lean*, a especificação detalhada das funcionalidades ocorre ao longo do projeto, desta forma, a estimativa inicial deve-se basear nas informações disponíveis no início do projeto. Estas informações podem ser alteradas ao longo do desenvolvimento e, portanto, a estimativa deve ser ajustada durante todo o projeto. O controle dos custos do projeto é essencial para viabilizar que as alterações de necessidades da área de negócio ao longo do projeto sejam permitidas, mas não ultrapassem o limite do orçamento.

3.2.1. Product backlog

Os requisitos básicos levantados no início do projeto devem ser listados para formar o *product backlog* do projeto. A partir do *product backlog*, o projeto é estimado.

As estimativas são realizadas através de uma noção da solução técnica necessária para suprir determinada funcionalidade. Ou seja, entende-se a necessidade e detalha-se uma solução.

Esta atividade de estimativa é demorada, pois exige um detalhamento para cada item do *backlog.* Revisitar todo o *backlog* ao longo do projeto para verificar possíveis melhorias nas estimativas iniciais, consome tempo de análise. Para evitar que o tempo da equipe seja consumido revisando estimativas de funcionalidades que ainda serão detalhadas no seu devido momento, no início dos *sprints*, o método *t-shirt size expanded* será aplicado.

3.2.2. O modelo *T-shirt size expanded*

McConnell (2006) afirma que antes de se classificar o custo, deve-se focar na estimativa de esforço. Desta forma, deve-se utilizar o modelo P, M, G apenas para estimativa de esforço. A estimativa de custo será realizada como derivação da estimativa de esforço. No vestuário, a numeração de camisas segue um padrão P, M, G. Uma das idéias deste modelo é padronizar o tamanho das funcionalidades do sistema em pequenas, médias e grandes.

A utilização de apenas três tamanhos, P, M, G, sugerida por McConnell (2006), limita a classificação. Desta forma, uma maneira de sanar esta limitação é expandir a numeração PP (itens que exigem muito pouco esforço), P (item de pequena complexidade), MP (pequena média complexidade), M (média complexidade), MG (média grande complexidade), G (grande complexidade) e GG (muito grande complexidade).

Desta forma, os itens do *product backlog* são listados e para cada item estima-se uma solução técnica e, posteriormente, cada item é classificado de acordo com as métricas *t-shirt size expanded*, associado à complexidade desta solução.

Funcionalidade	Solução estimada	*T-shirt size expanded*
Requisito 1	Criar um novo campo	M
Requisito 2	Criar uma tabela	G
Requisito 3	Alterar o layout	PP

Tabela 5 Passo 1 da aplicação do método
Fonte: elaborado pelo autor

Mesmo que o requisito não esteja detalhado, é importante estimar uma possível solução para atender à necessidade de negócio. A solução final será detalhada somente no momento em que ela for desenvolvida. A diferença entre a estimativa e a solução real será acompanhada no processo de controle de custos.

Ao final da classificação de todos os itens do backlog, é importante iniciar uma estimativa de esforço. Seleciona-se pelo menos um item de cada numeração e inicia-se o processo de estimativa refinada. Na tabela 6, foi exemplificada apenas 3 numerações PP, M e G.

Diante da solução técnica selecionada para determinado item, um analista faz o desdobramento tarefa a tarefa e estima detalhadamente cada uma delas para apreender o esforço. Com base em projetos análogos, as tarefas são estimadas em horas e somando-se o tempo de cada tarefa, tem-se, no final, o esforço para desenvolver determinada funcionalidade.

Funcionalidade	Solução estimada	*T-shirt size expanded*	Solução detalhada	Esforço (h)
Requisito 1	Criar um novo campo	M	Criar na tabela um novo campo Validar se o valor do campo é múltiplo	5
Requisito 2	Criar uma tabela	G	Criar tabela no banco Criar 5 campos Manter tabela	13
Requisito 3	Alterar fonte da tela	PP	Alterar fonte da tela no arquivo	1

Tabela 6 Passo 2 da aplicação do método
Fonte: elaborado pelo autor

Para validar o esforço de determinada numeração, é importante compará-la com outros itens do *backlog* de mesma numeração, bem como de numeração superior e inferior. O esforço calculado para desenvolver uma funcionalidade M analisada deve ser suficiente para desenvolver outra funcionalidade classificada como M. Caso não seja suficiente, há duas possibilidades a serem investigadas: ou a funcionalidade deve ser reclassificada com outro *t-shirt size* ou o esforço para implementar a funcionalidade detalhada deve ser modificado. Análises entre funcionalidades com classificação diferente, M e G, por exemplo também podem acontecer de forma a certificar que o esforço para determinada numeração está coerente com o esforço das demais numerações.

A partir da determinação e validação de esforço de cada numeração, todo o *product backlog* já classificado em *t-shirt size* recebe seu respectivo cálculo de esforço e, desta forma, tem-se uma medida estimada do tamanho do sistema a ser desenvolvido.

Há um esforço para executar este modelo *t-shirt size expanded* em todo o *backlog*. O modelo não é preciso, pois a análise detalhada é realizada para apenas alguns itens durante o início do projeto. Porém, realizar uma análise precisa em cima de itens ainda não precisos não gera valor para o projeto. Poupar o esforço de estimativa, mas mesmo assim estimar o tamanho do sistema é o objetivo deste método.

Durante o processo de desenvolvimento, quando o conhecimento for mais profundo do código e processos de negócio, a numeração estimada de cada item deve ser revista, o que afeta diretamente o esforço estimado inicialmente. Esta alteração entre o estimado e o realizado é foco do controle de custos.

Resumidamente, os passos para aplicação do método são:

1) Listar as funcionalidades
2) Estimar solução técnica
3) Classificar cada funcionalidade usando *t-shirt size expanded*
4) Detalhar algumas funcionalidades de cada medida do *t-shirt size*
5) Estimar o esforço das funcionalidades detalhadas
6) Replicar o esforço nas demais funcionalidades de acordo com sua numeração

A tabela a seguir resume os passos citados para a aplicação do método.

Funcionalidade 1	Solução 2	*T-shirt size* 3	Detalhamento 4	Esforço 5
Requisito 1	Solução	P	Detalhamento	1
Requisito 2	Solução	M	Detalhamento	3
Requisito 3	Solução	MP	Detalhamento	2
Requisito 4	Solução	GG	Detalhamento	13
Requisito 5	Solução	M		3 6
Requisito 6	Solução	GG		13

Tabela 7 Aplicação do Modelo *T-shirt size expanded*
Fonte: elaborado pelo autor

Observa-se que através do passo 6, o Requisito 5 recebeu um esforço semelhante ao esforço do Requisito 2, pois ambos foram classificados como M.

3.2.3. Plano de execução

Com a estimativa inicial do sistema, o processo de desenvolvimento do sistema deve ser desenhado de forma a considerar o número de iterações (*sprints*) e plano de implementação. Este é um plano que deve ser revisto ao longo de todo projeto para garantir sua aderência, mostrar a realidade do projeto tendo como base o que foi planejado e o realizado.

O planejamento da equipe deve levar em consideração o tamanho do sistema inicialmente estimado e o prazo a ser entregue. Os *sprints* podem ter de 2 a 4 semanas, dependendo da natureza do sistema, do envolvimento do *product owner*, da criticidade da aplicação. Além dos *sprints* de desenvolvimento, o plano deve conter os planos de entrada em produção (*release sprint)*. Um dos princípios das metodologias ágeis de desenvolvimento é entregar o *software* com as novas funcionalidades tão logo elas estejam prontas para que a área de negócio já passe a colher os benefícios da utilização.

Figura 5 Plano de Projeto
Fonte: elaborado pelo autor

Durante o *release sprint*, toda a equipe dedica-se às atividades de implantação da solução: compilação do código, correção de pequenos defeitos, verificação e validação de ambientes.

Os custos são atualizados de acordo com este plano de projeto e, a partir, deste momento, são estabelecidas as estimativas iniciais de todo o projeto.

3.3. O modelo Earned Value

Um dos princípios ágeis é implementar as funcionalidades tão logo elas sejam homologadas. Desta forma, os usuários se beneficiam da solução que atende seus requisitos de negócio. As atividades de implantação são realizadas durante os *release sprints*.

A data de fim (Fim) de projeto é calculada pela relação da data de início do *sprint* (Início) somada aos números (n) de *sprints* subseqüentes incluindo neste número os *release sprints*, considerando a duração dos *sprints* (L), temos:

(1) Fim = Início + (L x n)

Sulaiman (2006) expandiu a equação acima, associando-a com custos do projeto.

(2) Fim = início + L*(n.EAC / AC)

Onde, *AC* é o custo realizado até o momento presente. O *AC* é o custo referente aos custos de pessoas, como também viagens, aquisição de material e estes deverão ser computados ao final de cada sprint.

EAC é o custo estimado para finalizar o projeto. O *EAC* é calculado a partir da relação do percentual do projeto concluído e o custo gasto (AC). Através desta relação, estima-se que para concluir 100% do projeto, o custo deverá ser o equivalente ao *EAC*.

O percentual concluído real (*APC*) será calculado a partir da conta entre o número de *story points* entregues ao final de cada *sprint* até o momento e o número

total de *story points* do projeto. Para efeitos de simplificação, vamos considerar *APC* igual a PC (percentual concluído).

$$(3)\ PC = \frac{\text{story points entregues}}{\text{Story points total}}$$

$$(4)\ EAC = \frac{AC}{PC}$$

O custo total do projeto é considerado o *BAC* (*budged at complete)* que é calculado a partir da estimativa inicial de esforço. De acordo com o plano de projeto, o fluxo de pagamento é acordado ao final de cada *sprint* ou ao final de cada mês. O somatório do fluxo de pagamento inicialmente planejado resulta no valor do *BAC.*

BAC	AC	PC	EAC	ETC
Fluxo pagto 1	AC 1	PC 1	EAC 1	EAC1 – AC1
Fluxo pagto 2	AC 2	PC 2	EAC 2	EAC2 – AC2
Fluxo pagto 3	AC 3	PC 3	EAC 3	EAC3 – AC3
Fluxo pagto n	AC n	PC n	EAC n	EACn – ACn

Tabela 8 Consolidação de dados de custo
Fonte: elaborado pelo autor

A diferença entre o *BAC* e o fluxo de caixa acumulado é o orçamento disponível para finalizar o projeto. No entanto, o *ETC* indica qual será a necessidade de orçamento para finalizar o projeto tendo em vista o que já foi realizado.

(5) ETC = EAC – AC ou

$$(6)\ ETC = \frac{AC * (1-PC)}{PC}$$

3.4. Gestão ágil de custos

Diante do plano de execução do projeto, é importante realizar uma priorização das funcionalidades a serem implementadas. Como as estimativas foram realizadas baseadas em algumas premissas, há a possibilidade das estimativas serem reajustadas ao longo do projeto, a partir de um maior conhecimento do código e até mesmo do negócio. O reajuste das estimativas pode impactar severamente o custo

do projeto e desta forma, é importante garantir que a ordem de desenvolvimento dos requisitos seja focada na real necessidade de implementação destes requisitos para atender a área usuária.

De acordo com o conceito *lean*, as necessidades de negócio devem ser atendidas evitando desperdício de tempo e custo. O pensamento de geração de valor deve abranger a decisão da ordem de prioridade dos itens de *backlog* a serem desenvolvidos. As funcionalidades mais importantes ao negócio devem ser priorizadas a ponto de poder entregá-las mais rapidamente à área de negócio, a fim de que a mesma já passe a usufruir dos benefícios da funcionalidade em produção.

Cabe ao *product owner* listar todas as funcionalidades e pontuar o que é mais importante e menos importante, qual o benefício para o negócio de cada funcionalidade. Não é foco deste trabalho, discorrer sobre o como fazer esta priorização ou montar um modelo de classificação. A curva a seguir sintetiza esta idéia de classificação e mostra uma ordem das funcionalidades mais importantes ao negócio para as menos importantes.

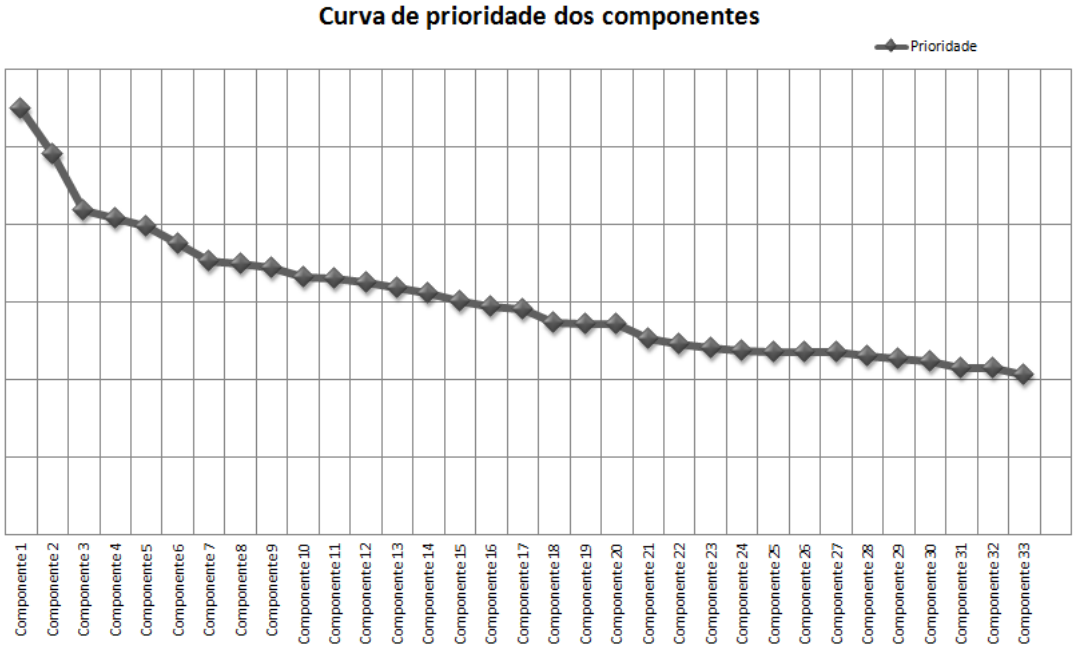

Figura 6 Gráfico de prioridade funcionalidades
Fonte: elaborado pelo autor

À medida que o projeto for sendo detalhado, pode acontecer que um componente classificado com baixa importância ao negócio, seja reclassificado com alta importância, e isto, apenas, sugere que o componente a ser detalhado

primeiramente será o de mais alta importância. Esta análise deve desconsiderar os componentes já implementados, pois as funcionalidades destes componentes já foram entregues.

Uma vez que estas funcionalidades estão classificadas pelo importância que elas geram ao negócio, o objetivo é gerir o custo baseado no valor gerado ao negócio através da implantação das funcionalidades.

Ao longo do projeto, caso uma nova funcionalidade seja inserida no projeto, ela deve ser classificada com relação ao seu potencial de gerar valor ao negócio da mesma forma que as demais funcionalidades. É importante que ela seja classificada comparando as funcionalidades ainda não implementadas, para guiar efetivamente quais destas funcionalidades devem ser consideradas para o *sprint* seguinte.

3.4.1. Curva de prioridade com estimativa de custo

Diante da curva de priorização, e à luz das estimativas de custo de cada funcionalidade obtido através do modelo *T-shirt size expanded*, um novo gráfico pode ser gerado consolidando as curvas de priorização e de custo de cada componente ou funcionalidade .

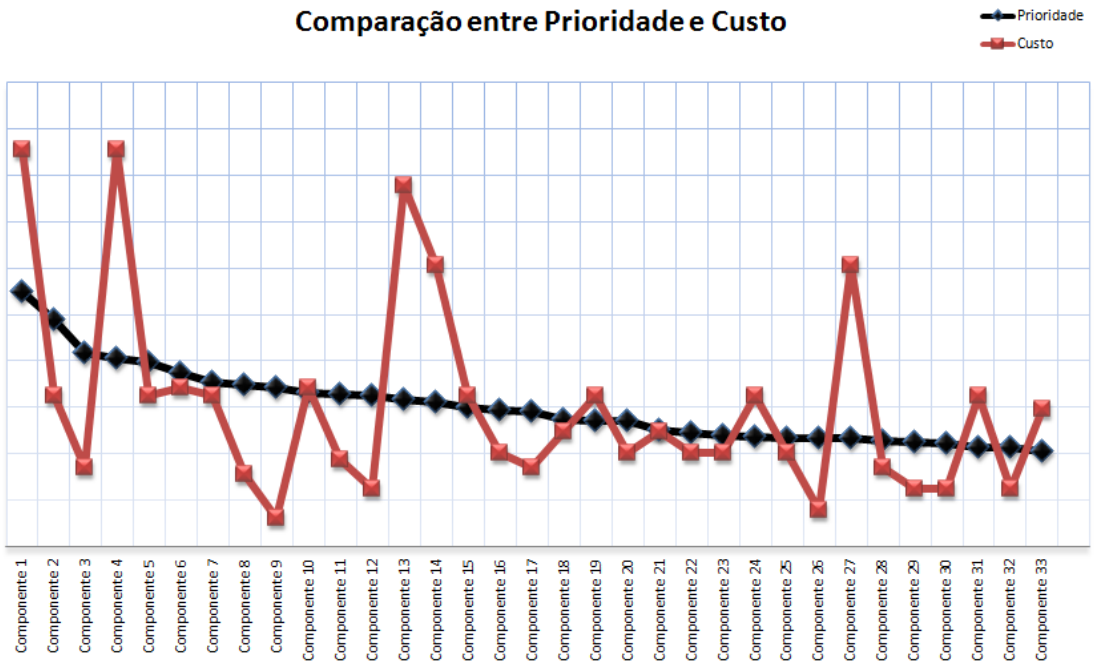

Figura 7 Gráfico de Comparação Prioridade e Custo
Fonte: elaborado pelo autor

No gráfico de comparação de prioridade e custo, quando a curva de custo fica muito acima da curva de priorização, suscita um plano de ação de revisar a definição de negócio a respeito do benefício gerado pela implementação de uma funcionalidade e o custo de sua implantação.

Esta técnica mostra de forma visual a relação entre a prioridade / benefício que a implementação das funcionalidades aportará ao negócio e custo das funcionalidades. Uma das ações imediatas ao analisar a curva, é tentar validar se a a importância do componente e a estimativa de custos são de fato as que o gráfico aponta. Como plano de ação, deve-se tentar buscar alternativas de simplificação do requisito e adoção de uma solução técnica menos rebuscada. Não necessariamente, toda funcionalidade que tiver estimativa de custo maior que a estimativa de prioridade e importância para o negócio deve ser eliminada, mas no mínimo uma reflexão mais apurada é necessária a fim de validar as reais necessidades de implementá-la.

Outro benefício deste gráfico é trazer a idéia, muito constante na metodologia de desenvolvimento *Scrum*, de validação contínua do *product backlog*. À medida que o projeto se desenvolve, as estimativas de custos e a percepção de importância que as funcionalidades podem agregar ao negócio são refinadas, devido ao conhecimento e detalhamento maiores dos requisitos. Este melhor conhecimento de custo e importância (prioridade) reflete diretamente no gráfico.

3.4.2. Burndown de Orçamento

No gráfico de comparação de prioridade e custo, não se tem uma visão clara do controle do custo ao longo do projeto. Um gráfico comumente utilizado para controle das funcionalidades implementadas é o gráfico *burndown* apresentado no capítulo 2 no tópico ferramentas de controle do processo *Scrum*. A ampliação de uso deste gráfico para o controle de orçamento será aplicado neste trabalho a fim de que a gestão do custo realizado e do custo projetado possa ser visualmente mostrada.

Para obter o gráfico de *burndown* de orçamento, a tabela 6 “Consolidação de dados de custo” é formatada com os valores acumulados de custo real e percentual concluído. Desta forma, o custo real do mês dois é custo dos dois primeiros meses

do projeto e o percentual concluído no mês dois, é o PC entregue até o momento em relação ao comprometido em todo o projeto.

Além disso, é importante que a coluna *BAC* seja demonstrada de forma decrescente. O BAC é reduzido mês a mês do valor referente ao fluxo de caixa negociado inicialmente.

BAC	AC	PC	ETC
B1 = BAC - Fluxo pagto 1	AC1	PC 1	ETC 1
B2 = B1 - Fluxo pagto 2	AC1 + AC2	PC1 + PC 2	ETC 2
B3 = B2 - Fluxo pagto 3	AC1 + AC2 + AC3	PC1 + PC2 + PC 3	ETC 3
Bn = Bn-1 - Fluxo pagto n	AC1 + ... + ACn	PC1 + ... + PC n	ETC n

Tabela 9 Consolidação de dados de custo
Fonte: elaborado pelo autor

A partir do custo inicialmente planejado (*BAC* e fluxo de caixa) e do *ETC* calculado após cada *sprint* são montadas as curvas do gráfico abaixo.

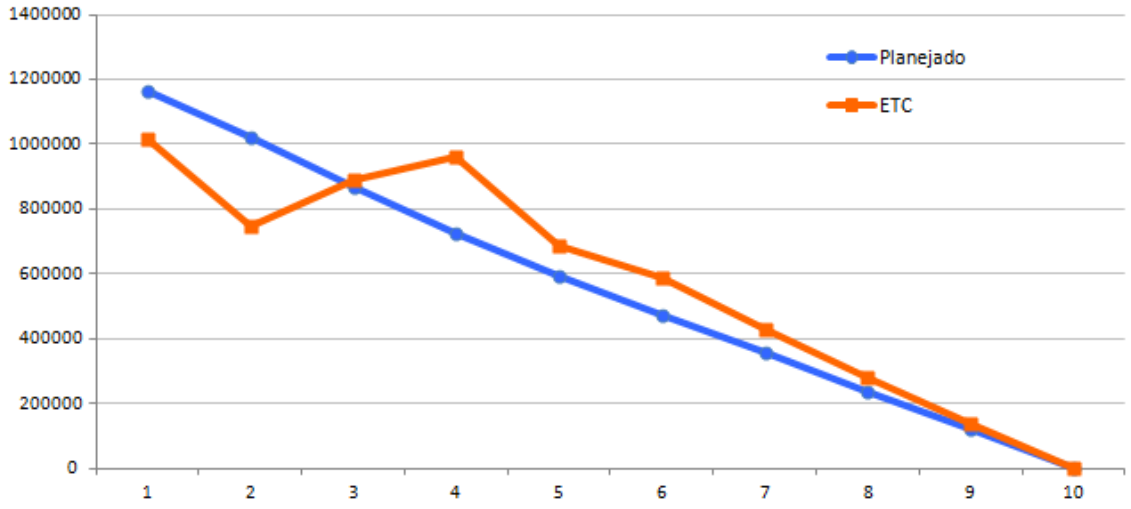

Figura 8 Gráfico de Burndown de Orçamento
Fonte: elaborado pelo autor

Caso a curva *ETC* esteja acima da curva do planejado, deve-se tomar uma ação corretiva de modo a eliminar este impacto negativo no projeto através de discussão de priorização das funcionalidades de negócio, simplificação dos requisitos solicitados e até mesmo uma nova negociação para incremento do orçamento destinado ao projeto.

Como as ações para correção de desvio são bastante influenciadas pelo negócio, é interessante ter uma visão clara deste desvio ao longo das entregas das funcionalidades. Desta forma, atrelar a visão de orçamento à curva de prioridade das funcionalidades permite de maneira visual um melhor manejo e rearranjo do *product backlog* através de inserção ou exclusão de funcionalidades nos *sprints* seguintes. À medida que a curva do custo planejado for se aproximando do valor absoluto zero, o orçamento do projeto já foi todo consumido. Todas as funcionalidades da curva de prioridade que estão abaixo deste ponto de interseção (Planejado = zero) não podem sem implementadas por falta de orçamento. Ou poderão ser implementadas se o orçamento do projeto for incrementado.

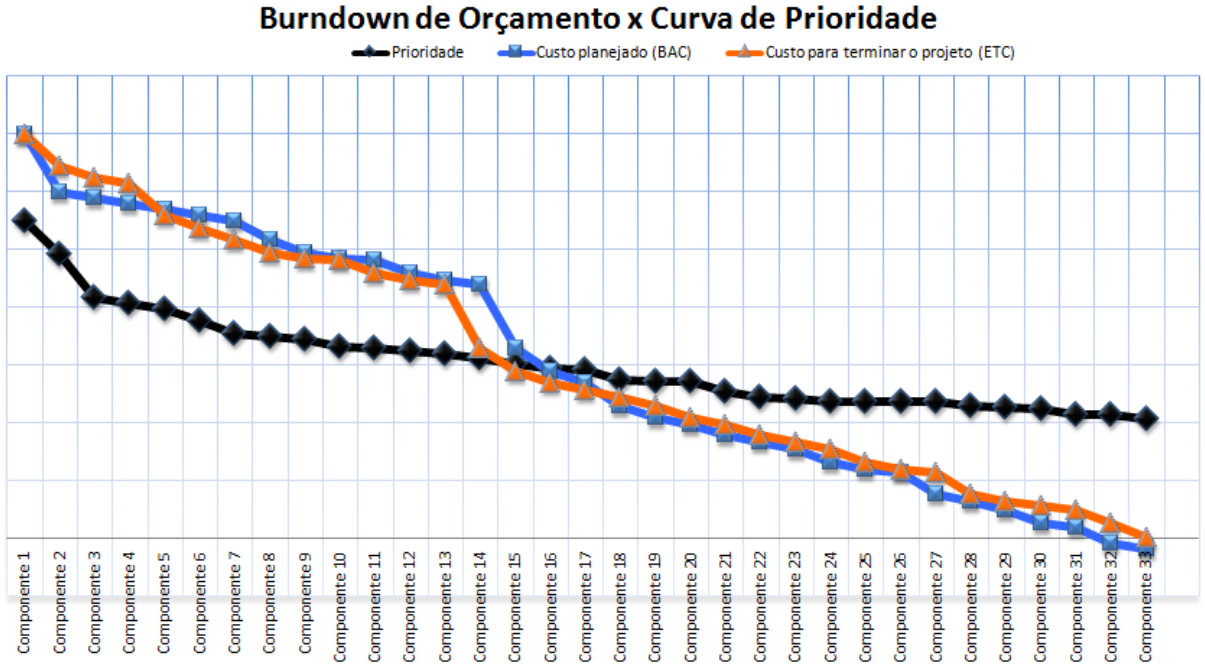

Figura 9 Gráfico de Burndown de Orçamento e Curva de Valor
Fonte: elaborado pelo autor

O gráfico de orçamento e curva de prioridade apresenta um cenário, que pode sofrer alterações ao longo do projeto. Desta forma, é importante ter cautela nas tomadas de decisão e, principalmente, no plano de comunicação destas informações para não criar expectativas a ponto de o negócio inserir novas funcionalidades no *product backlog* ou decepcionar-se diante de uma possibilidade de não implantação daquilo que havia sido planejado inicialmente.

4. EXEMPLO DE APLICAÇÃO

O método proposto no capítulo anterior é aplicado em um projeto de *software* que utiliza a metodologia de desenvolvimento ágil *Scrum*. O objetivo desta aplicação é validar a viabilidade do método. O projeto selecionado é considerado um grande projeto com mais de um ano de duração.

4.1. Histórico do Projeto

O projeto selecionado para aplicação do método é um projeto de implantação de um sistema comercial e *supply chain*. Este sistema foi desenvolvido pela matriz da empresa localizada no Japão. A implantação e a otimização do sistema para atender os requisitos regionais ficam sob responsabilidade de cada país.

A implantação do sistema no Brasil está alinhada à implantação do sistema nos Estados Unidos, desta forma, o projeto é considerado um projeto Américas. Além do Brasil e Estados Unidos, a implantação do sistema ocorrerá também no Canadá, México e Argentina.

Uma das diretrizes estratégicas da empresa é padronizar processos de negócio, desta forma, alguns requisitos regionais, quando possível, têm sua prioridade reduzida. No entanto, outros requisitos apesar de serem regionais devem ser implementados, caso contrário, o sistema não suportará as operações atuais de vendas, o que vai contra a outra diretriz estratégia da empresa de não interromper o fluxo de vendas atual.

O sistema comercial e *supply chain* visa atender os processos de previsão da demanda, vendas, expedição, estoque, compra entre unidades, e faturamento.

Reuniões com as pessoas de negócio foram realizadas a fim de se mostrar as atuais funcionalidades do sistema. Uma análise entre o que o sistema oferece e a realidade e necessidade do negócio fez surgir uma lista de requisitos a serem implantados.

Estes requisitos tornaram-se o *product backlog* inicial do projeto.

4.2. Priorização dos requisitos

Diante da grande quantidade de requisitos levantados durante as reuniões com as áreas de negócio, os itens precisaram ser agrupados para serem gerenciados.

O agrupamento dos itens seguiu inicialmente a linha dos módulos que o sistema trata: previsão, vendas, expedição, estoque, compra entre unidades e faturamento. No entanto, como estas áreas são complexas e englobam muitos processos, adotou-se a diretriz de realizar uma quebra nos módulos e agrupar separadamente os requisitos relacionados a um mesmo componente do processo.

Este processo inicial contou com a participação da área usuária e da área de TI. Os 300 requisitos foram classificados em 110 componentes. Esta primeira classificação levou 10 horas para ser finalizada. Posteriormente, o grupo foi dividido (Brasil e Estados Unidos) e individualmente cada grupo validou a classificação inicial. Ao final, chegou-se a um consenso e os requisitos foram classificados ao todo em 65 componentes.

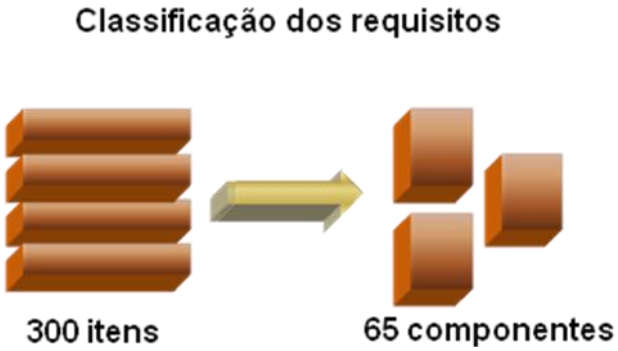

Figura 10 Relação dos requisitos e componentes
Fonte: elaborado pelo autor

4.2.1. Priorização dos componentes

Os 65 componentes precisavam ser priorizados para guiar as discussões de forma a discutir primeiramente os componentes cuja prioridade fosse maior. A decisão de prioridade envolveu a questão de valor para o negócio, medida através do benefício percebido pelo usuário diante da implantação do componente.

A estratégia da empresa foi o norteador para o início das decisões. Além do da estratégia, foi questionada a competência necessária para se alcançá-la. E por último, como o componente de negócio contribuía para cada competência.

Entre as competências, foi selecionada a que mais contribuía para atingir a estratégia e a que menos contribuía. Desta forma, todas as competências receberam um peso, quanto mais alto o peso maior a influência desta competência para se atingir a estratégia.

O mesmo grupo que realizou o agrupamento dos requisitos iniciais em 65 componentes executou o trabalho de escolher quais competências a organização precisava atingir para alcançar o objetivo estratégico. Este grupo também é quem estabeleceu o peso relativo de cada competência.

Posteriormente, todos os componentes foram listados e cada um pontuado em relação a cada competência. Se um componente influenciava mais uma determinada competência, o peso do componente nesta competência era superior ao peso em outra competência, na qual este componente tinha pouca influência.

A priorização dos componentes foi estabelecida através da soma dos produtos entre o peso do componente em determinada competência versus o peso da própria competência em relação à estratégia.

O componente com o maior valor de priorização era o componente que mais contribuía com a estratégia da empresa e, portanto, a sua implementação quanto mais adiantada fosse, maior valor traria para a empresa.

	Estratégia								
Competência	Competência 1	Competência 2	Competência 3	Competência 4	Competência 5	Competência 6	Competência 7	Competência 8	Priorização
Peso Competência	34	2	1	5	21	3	13	8	
Componente 1	8	5	1	13	13	1	21	21	**1065**
Componente 2	1	1	1	8	21	1	13	5	**730**
Componente 3	13	8	1	1	13	21	8	13	**1008**
Componente 4	1	1	1	21	21	1	1	13	**703**
Componente 5	13	1	1	8	8	1	1	3	**693**
Componente 6	1	1	1	13	21	1	1	13	**663**
Componente 7	13	3	8	13	8	5	21	13	**1081**
Componente 8	21	1	1	1	21	1	5	8	**1295**
Componente 9	1	1	1	21	21	2	21	5	**902**
Componente 10	13	1	1	8	2	5	5	13	**711**
Componente 11	21	1	1	1	8	1	5	8	**1022**
Componente 12	3	1	1	13	21	1	21	3	**911**
Componente 13	1	1	1	13	21	1	21	3	**843**
Componente 14	5	1	1	13	21	1	1	8	**759**
Componente 15	8	1	1	21	13	13	1	5	**745**
Componente 16	5	1	1	8	21	1	5	1	**730**
Componente 17	8	1	1	5	21	1	3	8	**847**
Componente 18	8	1	2	1	21	3	1	5	**784**
Componente 19	2	1	1	1	21	1	1	13	**637**
Componente 20	21	1	1	13	21	1	21	3	**1523**

Tabela 10 Priorização das funcionalidades
Fonte: elaborado pelo autor

4.2.2. Curva de valor

A fim de facilitar a visualização dos componentes do projeto e associando a idéia da ferramenta de controle do *Scrum*, *burndown chart,* a curva de valor foi gerada com os requisitos que devem ser entregues ao longo do projeto com base na priorização da tabela mostrada anteriormente. Na tabela anterior, o componente 20 foi o componente de maior prioridade e portanto é mostrado na curva como o primeiro componente da curva de priorização por valor.

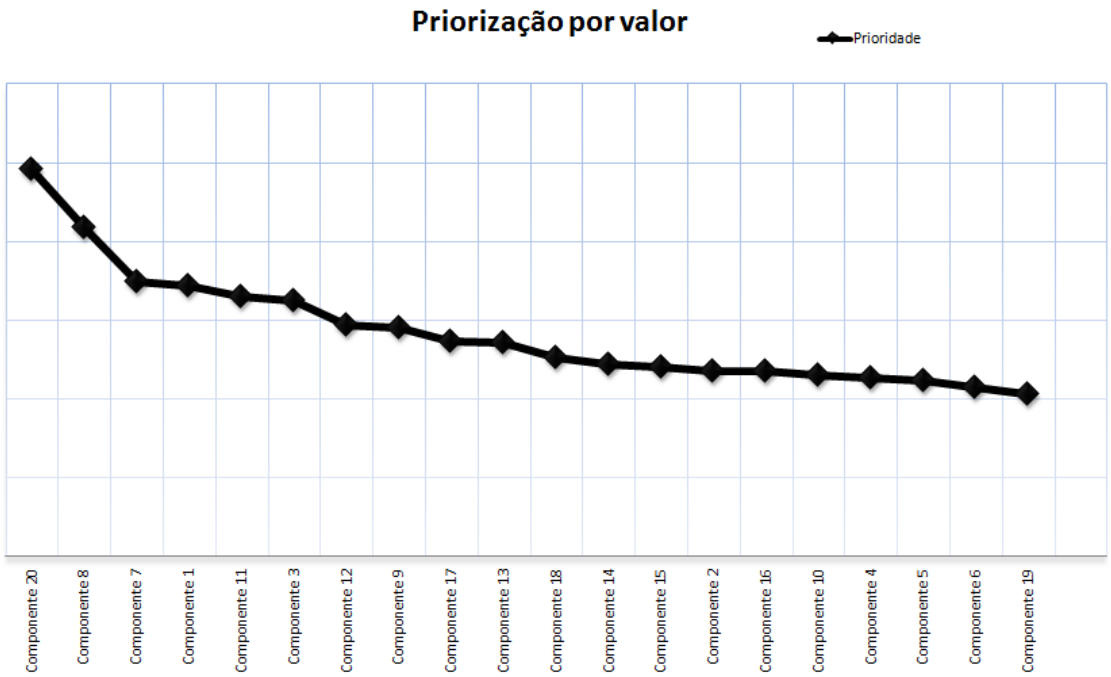

Figura 11 Priorização dos componentes
Fonte: elaborado pelo autor

O componente de maior classificação de valor deve ser mais prioritário na implementação. Os requisitos listados neste componente são capazes de trazer mais benefícios à estratégia da empresa se comparado a componentes menos prioritários.

Para a realização deste trabalho: seleção das competências, ponderação entre as competências e ponderação de todos os componentes nas devidas competências, foram consumidos um dia útil de 8 pessoas, totalizando 64 horas.

4.3. Estimativa de esforço

A realização da atividade de ponderação dos componentes às competências não impactou o *product backlog*. Nenhum item foi retirado ou incluído por consequência de baixa prioridade. Eles foram apenas classificados em seus devidos componentes. Através desta lista priorizada, a atividade de estimativa de esforço foi iniciada a fim de se estabelecer uma estimativa de custos para o projeto.

Para cada item, foi descrita uma solução técnica viável, não foram inseridos detalhes de implementação uma vez que os itens não estavam detalhados. A proposta da solução técnica para cada item de negócio foi realizada por duas pessoas técnicas com grande experiência em desenvolvimento de *software*. Cada um fez o detalhamento de um item, exceto em casos específicos, quando uma pessoa não havia entendido o requisito de negócio e, neste caso, a solução técnica considerada foi feita pelos dois.

Apenas o trabalho de especificar os 300 itens do product backlog consumiu 2 dias úteis de cada pessoa, totalizando 32 horas.

Baseados na solução técnica descrita, estas mesmas pessoas classificaram todos os itens através da numeração *T-shirt size expanded*.

Nome do Componente	Priorização de valor	Item / Requisito	Solução estimada	T-Shirt Size Expanded
Componente A	273	Item 1	Solução estimada 1	P (pequeno)
Componente A	273	Item 2	Solução estimada 2	G (grande complexidade)
Componente B	221	Item 3	Solução estimada 3	GG (muito grande complexidade)
Componente B	221	Item 4	Solução estimada 4	MP (pequena média complexidade)
Componente B	221	Item 5	Solução estimada 5	M (média complexidade)
Componente n	n	Item n	Solução estimada n	MG (média grande complexidade)

Tabela 11 Aplicação do *T-shirt size*
Fonte: elaborado pelo autor

O trabalho de classificação através da numeração *T-shirt size* foi baseado em muita analogia. Itens classificados como “M” eram comparados com outros itens “M”. E até mesmo com itens classificados com outros tamanhos, para diminuir a incerteza.

O trabalho também foi individualizado na maioria dos itens, salvo alguns itens que forçaram a discussão entre as duas pessoas que estavam executando a atividade para sanar dúvidas e possíveis inconsistências.

O trabalho de classificação de todos os itens em *T-Shirt Size* demorou na média 4 horas.

A medida *T-shirt size* apenas forneceu uma idéia de complexidade, traduzi-la em esforço foi um passo adicional do processo de estimativa. Este novo passo baseou-se no detalhamento da solução técnica de alguns itens. A intenção era alcançar uma correspondência entre complexidade (*T-shir size*) e esforço (horas).

Solução estimada	T-Shirt Size Expanded	Solução detalhada	Esforço (h)
Solução estimada 1	P (pequeno)	Solução detalhada 1	
Solução estimada 2	G (grande complexidade)	Solução detalhada 2	
Solução estimada 3	GG (muito grande complexidade)	Solução detalhada 3	
Solução estimada 4	MP (pequena média complexidade)	Solução detalhada 4	
Solução estimada 5	M (média complexidade)	Solução detalhada 5	
Solução estimada n	MG (média grande complexidade)	Solução detalhada n	

Tabela 12 Estimativa inicial de esforço
Fonte: elaborado pelo autor

Diferentemente da solução estimada que foi feita por cada desenvolvedor individualmente, a solução detalhada dos itens selecionados foi elaborada pelas duas pessoas juntas. A estimativa de esforço foi validada em conjunto, a fim de evitar qualquer tendência devido à experiência diferenciada destas duas pessoas.

Os itens foram selecionados dentro de cada classificação *T-shirt size*. Desta forma, garantiu-se que cada classificação *t-shirt size* fosse avaliada detalhamente quanto ao esforço de implementação.

Para evitar que uma determinada complexidade pudesse ter valores bastante discrepantes de esforço, itens diferentes com a mesma classificação *T-shirt size* também foram analisados. O intuito era obter um esforço padrão para cada medida do *T-shirt size*. Caso um item de complexidade M tivesse o esforço de 5 horas e outro item também de complexidade M tivesse o esforço de 8 horas, os dois itens eram revistos, podendo ao final ser ajustada a solução detalhada de um item, ou até mesmo a reclassificação de complexidade do item. Ao final, todos os itens de classificação M deveriam ter o mesmo esforço, no caso deste projeto, 8 horas.

Esta atividade de dimensionamento de esforço e discussão entre as duas pessoas que estavam executando as atividades durou 16 horas.

Baseado no esforço de implementação de cada item, o custo de implementação foi derivado a partir do custo do perfil da equipe que executaria o projeto e desta forma, montou-se a curva de valor e custo.

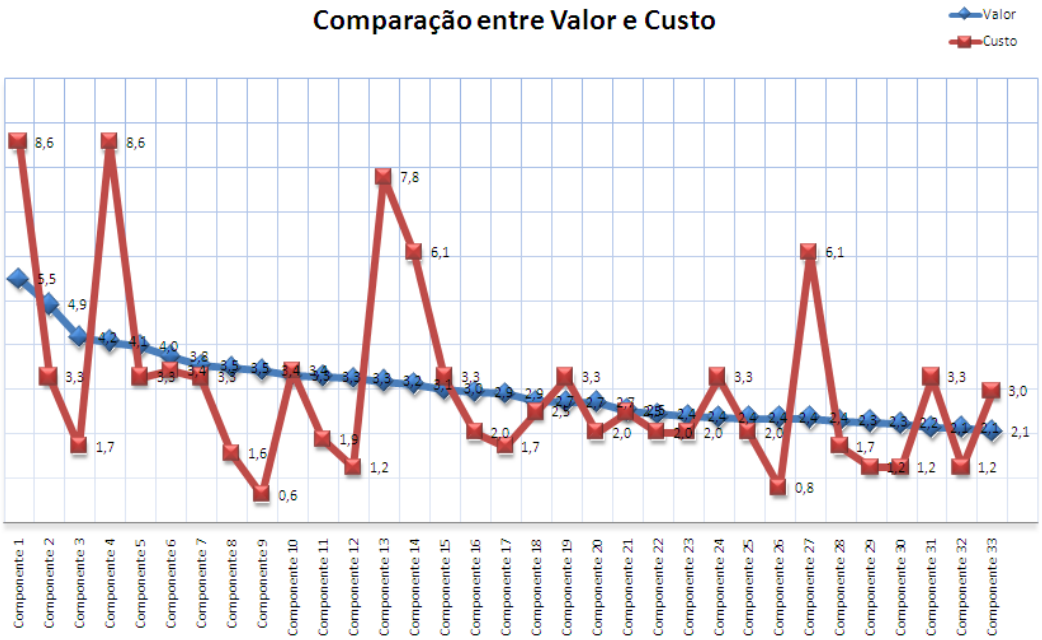

Figura 12 Comparativo entre valor gerado para o negócio e custo dos componentes
Fonte: elaborado pelo autor

As escalas de valor e custo não são relacionadas. No caso acima, o componente que aporta maior benefício ao negócio é um dos mais caros do backlog. O componente 27, apesar de ter menor prioridade em termos de valor, benefício de negócio, também é bastante custoso se comparado com os demais componentes.

A discrepância entre custo e valor ao negócio pode levar a questionamentos para validar se o componente agrega o devido valor para o projeto e traz o benefício a que se propõe. Também pode ser questionado se há uma maneira menos custosa de implementá-lo.

Tentar relacionar o custo do desenvolvimento com o benefício que uma funcionalidade traz e estabelecer um limite para que a implementação só aconteça respeitado este limite pode trazer complicações para o negócio. Há o risco de funcionalidades essenciais por questões regulatórias, apesar de não contribuírem fortemente para a estratégia da empresa, serem eliminadas do *backlog* pelo seu custo de implementação. Não é intenção deste gráfico discorrer sobre um possível retorno sobre o investimento de cada funcionalidade. A classificação de valor ao negócio considerada neste trabalho está atrelada única e exclusivamente à importância para o negócio e não ao retorno financeiro que determinada funcionalidade pode trazer ao negócio. Este benefício é traduzido pela priorização

dos componentes em relação ao peso nas competências que contribui para atingir a estratégia corporativa.

A idéia da curva de custo foi mostrar a distribuição de custo de todo o projeto ao longo dos componentes a serem implementados. Havendo uma discrepância entre os valores percentuais de valor (priorização dos componentes) e custo (estimativa de custo do componente) recomenda-se uma validação da priorização e da estimativa. Não há erro na aplicação do método caso existam, de fato, estas discrepâncias.

4.4. Plano de Projeto

A partir da estimativa de esforço (horas) realizada, o plano de projeto foi desenhado.

A estimativa de esforço foi baseada no tempo de construção da solução técnica detalhada, no entanto, em projetos de *software* é imprescindível também considerar o tempo de testes, correção de defeitos e o tempo para que a equipe cumpra as cerimônias do *Scrum* (*daily, planning, retrospective, demo meeting*).

Sendo assim, o somatório de esforço estimado do *product backlog* foi a base para o total de esforço real para implementação do projeto.

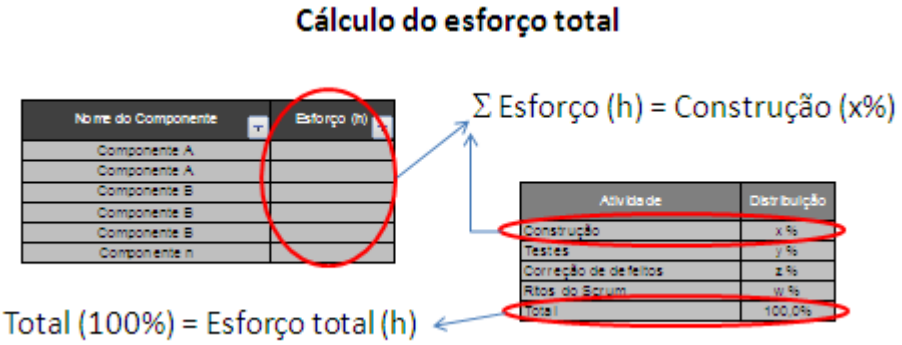

Tabela 13 Esforço total da equipe
Fonte: elaborado pelo autor

O projeto considerado para a aplicação do método discorrido neste trabalho contou com uma equipe de 7 desenvolvedores, um *scrum master*, um arquiteto e um gerente de projeto. Adotou-se no projeto, *sprints* com duração de 15 dias úteis.

Cálculo número de sprints

Dias úteis Sprint	A
Desenvolvedores	B
Não desenvolvedores	C

Horas úteis por dia = n° desenvolvedores (B) x 8

ΣEsforço (h) / horas úteis por dia = Dias de implementação

Número sprints = Dias de implementação / dias úteis sprint (A)

Tabela 14 Cálculo do número de *sprints*
Fonte: elaborado pelo autor

Diante do número de *sprints* e da estratégia de implantação em produção (*release sprint*), o plano de projeto foi finalizado.

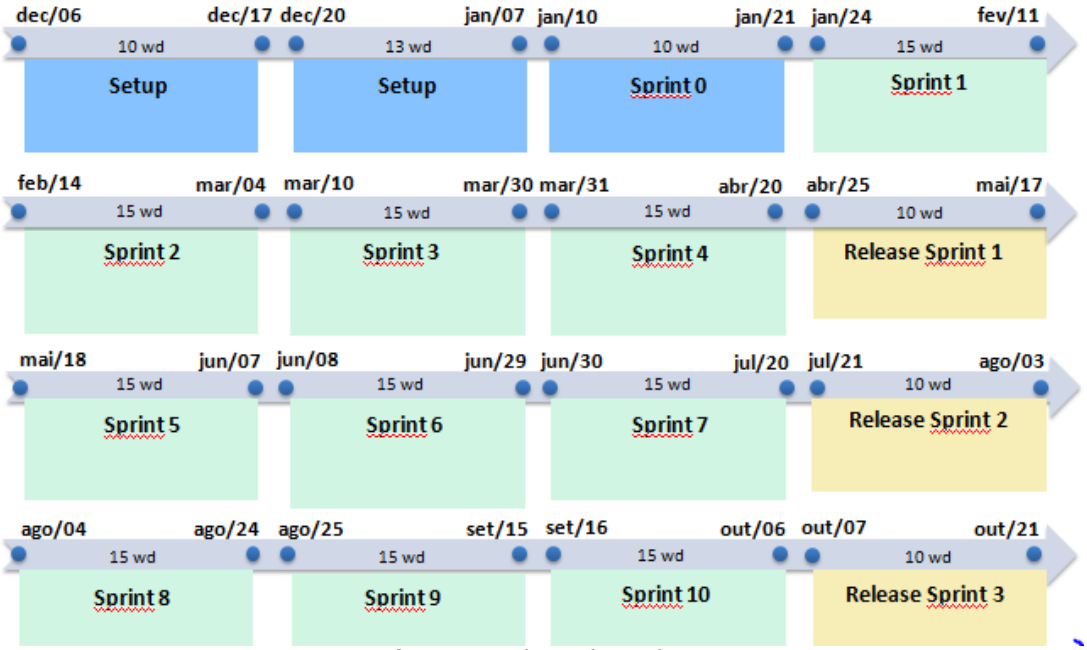

Figura 13 Plano de projeto
Fonte: elaborado pelo autor

O plano de projeto para o projeto em questão contou com 10 *sprints* e 3 *release sprints*. A partir do plano de projeto e da equipe alocada, o custo total do projeto pode ser calculado.

Para elaborar o plano de projeto, base para o orçamento inicial, e também adaptar as técnicas de controle do valor agregado para o controle do custo ao longo do projeto foi necessária uma semana de trabalho, 40 horas.

4.5. Gestão de custos

A adoção da técnica de gerenciamento por valor agregada (*EVA*) baseou-se no orçamento inicial (*BAC*), nos custos já ocorridos (*Actual Cost - AC*) e no percentual já concluído (*PC*). A partir do percentual concluído do projeto e do custo realizado (*AC*) fez-se uma projeção de custo necessário para finalizar o projeto.

Com o plano de projeto concluído, o custo total do projeto foi consolidado a partir da duração e números dos *sprints* e a alocação da equipe ao longo dos sprints.

Para este trabalho, consideramos que o orçamento disponível foi o custo total para a implementação de todo o projeto. E sobre este orçamento inicial, foi realizada a gestão de custos. Para a aplicação *EVA* este orçamento foi o *BAC*.

O orçamento foi baseado no custo mês a mês do projeto. A fim de visualizar o planejamento de custo ao longo do projeto, foi feito um fluxo de caixa inicial do projeto. Este previsão de fluxo de caixa foi a base para análise e gestão dos custos ao longo do projeto.

Custo planejado mês a mês

Mês 1	Mês 2	Mês 3	Mês 4	Mês 5	Mês 6	Mês 7	Mês 8	Mês 9	Mês 10
Custo 1	Custo 2	Custo 3	Custo 4	Custo 5	Custo 6	Custo 7	Custo 8	Custo 9	Custo 10

ΣCusto = Orçamento = BAC

Tabela 15 Planejamento de custo
Fonte: elaborado pelo autor

Para o acompanhamento do custo atual (*AC*) mês a mês, apurou-se o custo de alocação de todos os recursos naquele determinado mês. E o percentual concluído foi simulado de duas maneiras através do número de *sprints* e através da somatória das complexidades das *user stories* entregues mês a mês. Inicialmente, a aplicação do método seguiu o percentual concluído baseado nos *sprints*.

Percentual concluído baseado em *sprints*

	Mês 1	Mês 2	Mês 3	Mês 4	Mês 5	Mês 6	Mês 7	Mês 8	Mês 9	Mês 10
Custo	AC 1	AC 2	AC 3	AC 4	AC 5	AC 6	AC 7	AC 8	AC 9	AC 10
Sprint	1	2	3	4	5	6	7	8	9	10
PC	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0

Tabela 16 Cálculo do percentual concluído por *sprints*
Fonte: elaborado pelo autor

No entanto, associado à idéia de entregar funcionalidades, foi também simulado o percentual concluído através da complexidade das estórias entregues ao longo dos *sprints*, desta forma, um eventual desvio no custo poderia ser melhor explicado se, por exemplo, naquele mesmo período um número maior de complexidade fosse entregue.

Percentual concluído baseado em complexidade (*T-shirt size*)

	Mês 1	Mês 2	Mês 3	Mês 4	Mês 5	Mês 6	Mês 7	Mês 8	Mês 9	Mês 10
Custo	AC 1	AC 2	AC 3	AC 4	AC 5	AC 6	AC 7	AC 8	AC 9	AC 10
Sprint	1	2	3	4	5	6	7	8	9	10
T-shirt size	TS 1	TS 2	TS 3	TS 4	TS 5	TS 6	TS 7	TS 8	TS 9	TS 10
PC	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7	PC 8	PC 9	PC 10

Onde, PCn = TSn / ΣTS

Tabela 17 Cálculo do percentual concluído por *T-shirt size*
Fonte: elaborado pelo autor

Baseado, na complexidade das funcionalidades entregues e no custo realizado até o momento, a projeção de custo necessário para finalizar as demais funcionalidades foi obtida através da razão entre o custo realizado (*AC*) e o percentual concluído (*PC*).

Projeção de custo ao longo do projeto

	Mês 1	Mês 2	Mês 3	Mês 4	Mês 5	Mês 6	Mês 7	Mês 8	Mês 9	Mês 10
Custo	AC 1	AC 2	AC 3	AC 4	AC 5	AC 6	AC 7	AC 8	AC 9	AC 10
PC	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7	PC 8	PC 9	PC 10
ETC	ETC 1	ETC 2	ETC 3	ETC 4	ETC 5	ETC 6	ETC 7	ETC 8	ETC 9	ETC 10

Onde, ETC = [AC * (1-PC)] / PC

Tabela 18 Projeção de custos mensal
Fonte: elaborado pelo autor

A projeção de custo para finalizar o projeto (*ETC*) e o custo realizado não devem ultrapassar o *BAC*. Caso o *ETC* eleve a expectativa de orçamento e supere o patamar planejado do orçamento (*BAC*) decisões devem ser planejadas na tentativa de retomar o orçamento disponível ao seu patamar original.

4.6. *Burndown* de orçamento

A ferramenta do gráfico de *burndown* facilita o controle visual das atividades ao longo dos *sprints.* Permite tomadas de decisão rápidas que visam contornar possíveis desvios entre o realizado e o planejado.

Aplicar o mesmo gráfico para acompanhar o desempenho orçamentário do projeto foi uma solução que contribui para elaboração de um plano de ação para eliminar perdas ou potencializar ganhos.

Para aplicar o gráfico *burndown* para controle de orçamento, foram consideradas a curva de orçamento planejado e orçamento real.

A curva de orçamento planejado foi construída a partir do BAC e do fluxo de caixa planejado ao longo dos meses. No eixo vertical, tem-se a variável orçamento e no eixo horizontal, a variável mês. Para obter o efeito do gráfico *burndown*, o fluxo de caixa considerado foi acumulado mês a mês.

Curva Orçamento Planejado

Mês 0	Mês 1	Mês 2	Mês 3	...	Mês 10
BAC	Fluxo 1	Fluxo 2	Fluxo 3	...	Fluxo 10
BAC	BAC - soma fluxo (1)	BAC - soma fluxo (1 e 2)	BAC - soma fluxo (1 a 3)	...	BAC - soma fluxo (1 a 10)

Tabela 19 Fluxo de caixa
Fonte: elaborado pelo autor

A curva de orçamento realizada também se iniciou a partir do BAC, mas foi atualizada mês a mês com base no cálculo da projeção de custo necessária para finalizar o projeto (*ETC*). Os números expostos são fictícios, mas contribuem para um entendimento da aplicação do método.

O projeto foi estimado em aproximadamente R$1.300.000,00 e o seu fluxo de caixa estimado de acordo com a coluna “fluxo planejado”.

		Burned down			Burndown Chart			
Mês	**Fluxo planejado**	**PC**	**AC**	**AC Acumulado**	**Planejado**	**EAC**	**Diferença**	**ETC**
0	**0**	**0**	**0**	**0**	**1.286.265,05**	**1.286.265,05**	**0,00**	**1.286.265,05**
1	122.582,00	0,11	122.582,00	122.582,00	1.163.683,05	1.140.297,64	-145.967,40	1.017.715,65
2	140.044,76	0,26	140.044,76	262.626,76	1.023.638,29	1.010.102,92	-276.162,13	747.476,16
3	156.200,08	0,32	156.200,08	418.826,84	867.438,21	1.308.833,87	22.568,82	890.007,03
4	140.476,18	0,37	140.476,18	559.303,02	726.962,03	1.521.912,97	235.647,92	962.609,95
5	133.868,38	0,50	128.373,34	687.676,35	593.093,65	1.375.352,70	89.087,66	687.676,35
6	118.618,73	0,58	118.618,73	806.295,08	474.474,92	1.396.181,96	109.916,91	589.886,88
7	118.618,73	0,68	118.618,73	924.913,81	355.856,19	1.355.185,07	68.920,03	430.271,26
8	118.618,73	0,79	118.618,73	1.043.532,54	237.237,46	1.325.120,69	38.855,64	281.588,15
9	118.618,73	0,89	118.618,73	1.162.151,27	118.618,73	1.302.130,28	15.865,23	139.979,00
10	118.618,73	1,00	118.618,73	1.280.770,00	0,00	1.280.770,00	-5.495,04	0,00

Tabela 20 Fluxo de caixa estimado
Fonte: elaborado pelo autor

O gráfico foi desenhado usando as variáveis Planejado (que foi baseada na estimativa inicial de custos) e *ETC* (que foi captada através da gestão dos custos).

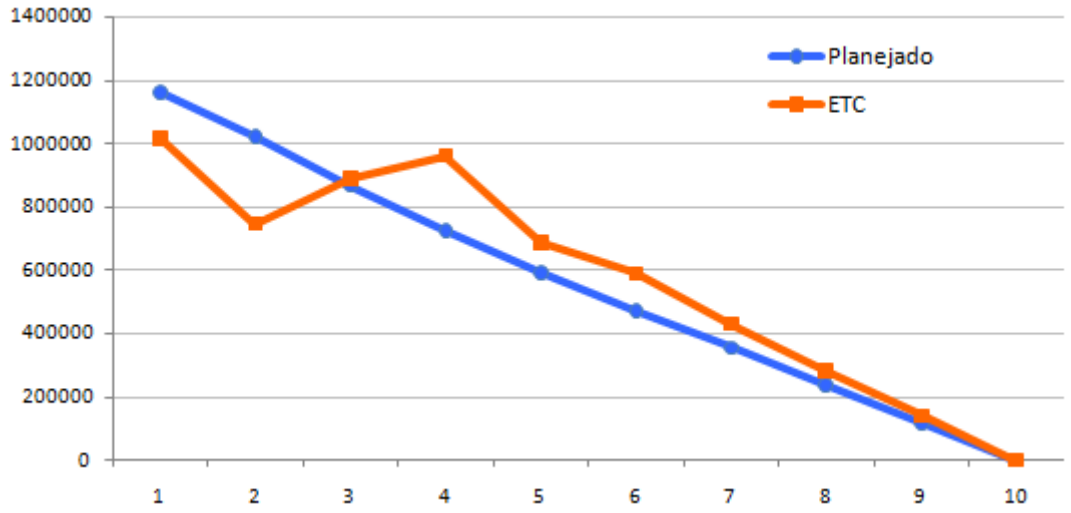

Figura 14 Burndown de orçamento do projeto
Fonte: elaborado pelo autor

Quando a curva *ETC* está abaixo da curva Planejado, o projeto está executando com maior rapidez o que havia sido projetado. Neste caso, há uma possibilidade do projeto ser entregue antes do prazo ou até mesmo novas funcionalidades serem adicionadas ao *product backlog*. No entanto, a análise deve ser criteriosa, pois tem-se que estudar o provável comportamento futuro para que a comunicação possa ser assertiva. Quando a curva *ETC* está acima da curva Planejado, há uma probabilidade de que nem todas as funcionalidades listadas no *product backlog* sejam implementadas, ou de repente, pode-se buscar uma simplificação na solução técnica das funcionalidades a fim de implementá-las mas de maneira menos complexa.

Ao longo do projeto com a variação do *ETC* a cada período, a diferença acumulada entre o que o projeto poderá custar e o planejado pode ser acompanhada através do *EAC* (*EAC* = *ETC* + *AC*) e do *BAC*. A diferença entre estas duas variáveis consolida o quanto o realizado está diferente do estimado.

4.7. Análise da aplicação

Foram utilizadas 126 horas para classificar as funcionalidades, organizá-las em componentes, estimar uma solução de alto nível para cada uma delas, classificá-las no modelo *T-shirt size* e finalmente elaborar para algumas funcionalidades uma solução detalhada a fim de realizar o processo de estimativa de custo.

Para elaborar o plano de projeto, mais 40 horas foram necessárias, totalizando um investimento de aproximadamente um mês (126 / 40 horas) para se iniciar o desenvolvimento do sistema.

Os benefícios da aplicação desta fase inicial de projeto, fase de preparação, podem ser justificados pela organização do *product backlog* de acordo com a priorização do valor e uma maior garantia de controle de custos, evitando desvios ao longo do projeto. No entanto, esta fase de preparação deve estar contemplada também no plano de projeto, para considerar o custo desta fase.

No primeiro mês de adoção da ferramenta, apenas o percentual concluído e custo real do primeiro mês eram conhecidos, pois era o início do projeto e a diferença entre o *EAC* e o *BAC* causou um primeiro ponto de reflexão. A estimativa inicial foi questionada, pois poderia ter sido super dimensionada, já que no primeiro mês o projeto mostrava um possível ganho de mais de 10%. Também foi questionado o percentual concluído, se não houve algum erro no cálculo.

Para evitar, tomadas de ação que pudessem ser influenciadas pela falta de informação dos meses seguintes, o *AC* dos meses futuros foi idêntico aos valores planejados e o *PC* de cada mês foi simulado de maneira que ao final o percentual concluído fosse 100%.

A partir da adoção desta medida, a diferença considerada para efeitos de análise foi a diferença acumulada, considerando todos os meses do projeto. E desta forma, o desempenho de um mês isolado não foi a referência para decisão de uma possível mudança de estratégia no projeto.

Uma limitação encontrada na adoção das técnicas de valor agregado, quando percentual concluído foi baseado em *sprints*, foi referente ao tamanho dos *sprints*. Os *sprints* de desenvolvimento são normalmente de mesmo tamanho e os *release sprints* podem ter tamanhos diferentes. No projeto em questão, os *sprints* tinham duração de 15 dias úteis e os *release sprints* de 10 dias úteis. A adoção de 15 dias úteis para os *release sprints* se fez necessária para que a técnica fosse aplicada, mas neste caso, também por necessidade do projeto, devido a um problema na condução da homologação. Como a homologação não foi contínua, o *release sprint* ficou maior a fim de que os usuários pudessem também fazer uma força tarefa para execução da homologação durante *os release sprints*.

5. CONCLUSÃO

No capítulo 2, Gerenciamento de Custo, citou-se que um dos objetivos desta pesquisa era contribuir para o equilíbrio entre a manutenção do orçamento de contratos preço fechado e a flexibilidade para aceitar alterações de requisitos em contratos *time and material*. Nos projetos *Scrum*, o apelo a esta flexibilidade de requisitos é característica forte da metodologia. Por isso, muitas vezes há uma rejeição à aplicação do *Scrum* por associá-lo a contratos com características *time and material*. Desta forma, a aplicação do modelo proposto neste trabalho favoreceu a adoção da metodologia *Scrum* uma vez que permitiu o controle de custos sem limitar as alterações de escopo ao longo do projeto.

Como a solução detalhada foi realizada apenas para algumas funcionalidades e o conhecimento dos requisitos de negócio era superficial no início do projeto, era imprescindível que houvesse uma manutenção contínua da curva de comparação entre valor e custo (figura 12). Tanto questionando a classificação de valor de um determinado componente, quanto questionando a solução inicialmente proposta e medida através do *T-shirt size*. A manutenção do product backlog foi incorporada como uma das atividades ao longo do *sprint*. O intuito desta manutenção foi analisar a necessidade de se incluir ou remover requisitos e a necessidade de reavaliar a classificação original de complexidade.

A ampliação do método *T-shirt size* facilitou a melhor adequação do esforço necessário para implementação de cada funcionalidade. A revisão da classificação inicial foi realizada periodicamente após cada *sprint*. A assertividade entre a classificação inicial e a classificação final, após cada *sprint*, foi bem alta. O investimento de tempo e esforço despendidos no início do projeto para classificação de todas as funcionalidades foi um diferencial para primeiramente orçar o custo inicial e depois controlá-lo, à medida que revisões periódicas eram realizadas.

A adoção da técnica de valor agregado teve resistência por parte das pessoas do projeto, pois consideraram, em um primeiro momento, que captar estas métricas seria um procedimento adicional. No entanto, a adaptação da técnica para projetos que rodam *Scrum* reduziu este desconforto inicial, pois a aplicação da técnica não onerou o processo, uma vez que os dados de custos e percentual concluído já eram

disponibilizados e apenas foram consolidados de forma a apoiar as métricas de *EAC* e *ETC*. A aplicação do método exigiu certo investimento inicial, fase de preparação.

Um ponto alto da aplicação do método foi a credibilidade que o mesmo causou no cliente. Através da forma visual do gráfico de *burndown* de orçamento, o cliente acompanhava de maneira clara a projeção de custos do projeto. As reuniões mensais para acompanhamento de projeto, no que tange à gestão de custos, resumiam-se apenas a apresentar o gráfico e a diferença acumulada entre o *EAC* e o *BAC*. Inclusive, através da análise da curva, identificou-se que os componentes entregues nos primeiros dois meses foram componentes de complexidade técnica baixa, e portanto, a taxa de percentual concluída foi superior, causando desta forma uma diferença negativa que poderia levar a uma redução de orçamento para o projeto ou a possibilidade de inserir novos requisitos. Como era de conhecimento de todos, funcionalidades de complexidade técnica maior ainda estavam para ser implementadas e isto poderia consumir o ganho de orçamento no início do projeto. A transparência a respeito do controle do orçamento foi salientada nas duas pesquisas de satisfação realizada com o cliente.

A consolidação dos dados, cálculo e demonstração dos números foi criteriosa, pois para se obter o gráfico de *burndown* as variáveis foram tratadas, por exemplo, acumulando os valores de custos e percentual concluído.

Por fim, este trabalho contribui para que sejam implementadas as funcionalidades que efetivamente o sistema exige. Em um trabalho realizado por Johnson et al. (1999) foi identificado que 20% das funcionalidades entregues no produto final são utilizadas sempre ou com frequência, 16% são utilizadas algumas vezes, e a grande maioria, 64%, raramente ou nunca, é utilizada. Desta forma, a ordenação das funcionalidades pela importância ao negócio no *Scrum* levou o time de projeto a uma conscientização da capacidade produtiva e priorização das funcionalidades mais relevantes. Esta assertividade na ordem de implementação das funcionalidades também foi percebida através do retorno dos usuários. Os usuários elogiaram bastante as novas funcionalidades e fizeram grande uso das mesmas logo após sua implementação.

A ferramenta de *burndown* de custo foi muito bem assimilada pela área de negócio, ou seja, ela saiu do âmbito somente do comitê de gestão do projeto. Através desta ferramenta, a própria área usuária começou a questionar a

classificação de valor inicialmente feita no projeto. À medida que eles acompanhavam a evolução dos custos e o que ainda estava disponível para implementação, tendo como diretriz não ultrapassar o orçamento inicial, eles próprios propuseram uma nova análise na curva de classificação das funcionalidades por valor. Vários componentes foram simplificados, alguns foram eliminados e outros foram priorizados diante do objetivo claro de manter o projeto no orçamento inicial.

5.1. Contribuições

Dentre as principais contribuições deste trabalho, destacam-se:

- A investigação das técnicas de estimativa e controle de custo conhecidas no mercado, para estender sua utilização em projetos cuja metodologia de desenvolvimento é *Scrum*.
- A análise entre o benefício gerado pela implementação de uma funcionalidade à área de negócio (valor) e o custo de sua implementação levando a um questionamento de simplificação da solução.
- A adaptação de técnicas utilizadas em projetos de desenvolvimento de *software* para projetos *Scrum*.
- A expansão da utilização da ferramenta de *burndown* para o âmbito da gestão.
- A incorporação do controle de custos para tomada de decisão de negócio por parte da área usuária.

5.2. Trabalhos futuros

Como trabalhos futuros, derivados deste, pode-se propor a aplicação do método proposto em outros projetos, com outros culturas e experiências, para confirmar os benefícios da técnica. A extensão do uso do *burndown* para incorporar o valor agregado e não só o orçamento do projeto também pode ser explorada. Há oportunidades de pesquisa na priorização dos requisitos de projeto assim como na estimativa do valor de negócio para o projeto.

A técnica demonstrada neste trabalho pode ser automatizada, de modo a integrar em um sistema de software as diversas planilhas de controle utilizadas reduzindo a complexidade e resistência na adoção do método. Outra possível fonte de pesquisa é a seleção adequada do tamanho de *sprints* de desenvolvimento e *release.* O tamanho do *release sprint* não tem relação com o tamanho dos *sprints* de desenvolvimento e, portanto, adaptar a técnica de valor agregado para considerar estes *sprints* de forma isolada pode ser proposta de um novo trabalho.

REFERÊNCIAS

ALLEMAN, G. B.; HENDERSON M. ***Making Agile Development Work in a Government Contracting Environment***. IEEE Agile 2003. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

ASPRONI, G. ***An Introduction to Scrum***. *Software* Developer´s Journal, junho, 2006.

ASSAF NETO, A. **Estrutura e Análise de Balanços: Um Enfoque Econômico-Financeiro**. Editora Atlas, ed. 7°, 2002, 44p.

ATTARZADEH, I.; HOCK O. S. ***A New Forecasting Model to Improve Earned Value Index to Achieve an Accurate Project Time and Cost Estimation***. IEEE Agile 2009. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

BECK, K. et al.. ***Manifesto for agile software development***. 2001. Disponível em: www.agilemanifesto.org. Acesso em: 01.06.2009.

BLACK, S. et al. ***Formal versus agile: survival of the fittest?*** IEEE Agile 2009. IEEE Magazine, Volume 42, Number 9, p. 37-45.

BOEHM, B. W.; FAIRLEY R. E. ***Software Estimation Perspectives***. IEEE Agile 2005. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

BOOCH, G. ***Object-Oriented Analysis and Design With Applications***. Addison Wesley, 1994, 608p.

DENNIS, P. **FAZENDO ACONTECER A COISA CERTA**. Lean Institute Brasil, 2007, 246p.

DERBIER, G. ***Agile Development in the Old Economy***. IEEE Agile 2003. IEEE Development Conference, 4-8 Aug . 2008, p. 413-416.

EDWARDS, M. D. ***Overhauling a failed project using out of the box Scrum***. IEEE Agile 2008. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

ECKFELDT B.; MADDEN R. ***Selling Agile: Target-Cost Contracts***. IEEE Agile 2005. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

FALCONI, V. **O VERDADEIRO PODER**. INDG Tecnologia e Serviços, 2009, 158p.

FRANKLIN, T. ***Adventures in Agile Contracting: Evolving from Time and Materials to Fixed Price, Fixed Scope Contracts***. IEEE Agile 2008. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

HIGHSMITH, J. ***Agile Project Management: Creating Innovative Products***. Addison Wesley, 2004, 312p.

Institute of Electrical and Electronics Engineers. ***SWEBOK a Guide to the Software Engineering Body of Knowledge***. IEEE, 2004.

IONEL, N. ***Critical Analysis of the Scrum project management methodology***. The Academy of Economic Studies Bucharest, Management Faculty, 2008.

JOHNSON, J. et al. ***CHAOS: A Recipe for Success***. Published Report, The Standish Group, 1999. 12p.

MCCONNELL, S. ***Software Estimation: The Black Art Demystified***. Microsoft Press, 2006, 308p.

MOE, N. B.; AURUM A. ***Understanding Decision-Making in Agile Software Development: A Case-study***. *Software* Engineering and Advanced Applications, 2008. SEAA 08. 34th Euromicro Conference, 03-05 Ago. 2008, p. 216-223.

MULCAHY, R. ***PMP EXAM PREP***. RMC Publications, ed. 5°, 2005, 445p.

NOBREGA, C.; LIMA, A.R. **INNOVATRIX Inovação para Não Gênios**. Nova Fronteira, 2010, 163p.

PRAHALAD, C.K.; HAMEL, G. **A Competência Essencial da Corporação**. 1990. Editado no livro Estratégia: A Busca da Vantagem Competitiva, organizado por MONTGOMERY, C.A.; PORTER, M.E. Editora Campus, ed. 7°,1998, 293-316,

Project Management Institute Inc. ***PMBOK a Guide to the Project Management Body of Knowledge***. PMI, 2004.

REES, M. J. ***A Feasible User Story Tool for Agile Software Development?***. IEEE Agile 2002. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

RIEDEMANN, C.; FREITAG, R. ***Modeling Usage: Techniques and Tools***. IEEE Agile 2009. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

SAKAMOTO, H. T. **Guia de Utilização da Análise de Valor Agregado (EVMS) e do Valor Econômico no Controle de Projetos de *Software***. 2006. 119f. Dissertação (Mestrado em Engenharia de Computação) – Instituto de Pesquisas Tecnológicas do Estado de São Paulo, São Paulo, 2006.

SCHWABER, K.; BEEDLE, M. ***Agile Software Development with Scrum***. Prentice Hall, 2001, 158p.

SHAHIR, H.Y. et al. ***Improvement Strategies for Agile Processes: a SWOT Analysis Approach***. IEEE Agile 2008. IEEE Conference, 4-8 Aug . 2008, p. 413-416.

SLIGER, M.; BRODERICH, S. ***The Software Project Manager's Bridge to Agility***. Addison Wesley, 2008, 384p.

SOMMERVILLE, I. **Engenharia de *Software***. Addison Wesley, 2003, 579p.

SULAIMAN, T et al. ***Earned Value Management in Scrum Projects***. IEEE Agile 2006. IEEE Conference, 13-18 Jul. 2006, p. 10-16.

TAVARES, H. C.; CARVALHO, A. E., CASTRO, J. F. **Medição de Pontos por Função a Partir da Especificação de Requisitos**. Universidade Federal de Pernambuco, 2002.

TENGSHE, A.; NOBLE, S. ***Establishing the Agile PMO: Managing variability across projects and portfolios***. IEEE Agile 2007. IEEE Conference, 13-17 Ago. 2007, p.188-193.