METODOLOGIA HÍBRIDA APLICADA AO GERENCIAMENTO DE PROJETOS DE SOFTWARE

Vilmar Santos Nepomuceno
Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE) – Garanhuns
R. Padre Agobar Valença s/n, Severiano Moraes Filho. 55297-400 - Garanhuns – PE - Brasil.
E-mail: vilmar.nepomuceno@garanhuns.ifpe.edu.br

Marcele Elisa Fontana
Universidade Federal de Pernambuco-UFPE
Centro Acadêmico do Agreste – CAA. Rodovia BR-104 km 59 - Nova Caruaru. 55002-970 – Caruaru, PE – Brasil.
E-mail: marcelelisa@gmail.com

RESUMO

No ambiente de projetos de software há uma crescente necessidade em se estudar métodos que auxiliem e otimizem o seu gerenciamento, sem retirar a flexibilidade inerente das metodologias ágeis. Desta forma, este trabalho consiste na determinação de um cronograma e na atribuição preliminar de tarefas através da hibridização da metodologia *Scrum* com o método de programação linear de designação de tarefas. Esta proposta é baseada em um estudo de caso e mostrou-se eficiente na minimização dos tempos de concretização de uma *sprint*, bem como no auxílio ao gerenciamento da execução das tarefas sem restringir possíveis mudanças desejadas pela equipe.

PALAVARAS CHAVE. Gerenciamento de projetos de software, Metodologia Scrum, Programação linear.

ABSTRACT

In the environment of software projects there is a growing need to study methods that help and optimize its management without removing the inherent flexibility of agile methodologies. Therefore, this paper consists in determining a schedule and in the tasks preliminary allocation through the hybridization between the Scrum methodology and the linear programming method of assignment problem. This proposal is based on a case of study and it showed to be effective in time minimization to completion of a sprint, as well as it aid of the tasks execution management, without restricting possible changes desired by the team.

KEYWORDS. Software project management, Scrum methodology, Linear programming.

1. Introdução

O desenvolvimento de software depende significativamente do desempenho da equipe, assim como todo o processo que envolve a interação humana. Para Dybå (2000) o desenvolvimento de software está enraizado no paradigma racionalista, que promove um plano de abordagem orientado à linha de produtos para seu desenvolvimento, usando um padronizado, controlável e previsível processo de engenharia de software.

As metodologias tradicionais são concebidas para responder à imprevisibilidade dos ambientes externos e do desenvolvimento no início de um ciclo de melhoria, o que pode atrasar a visibilidade de problemas no projeto. Ao mesmo tempo em que as metodologias ágeis destinam-se a serem mais flexíveis, elas fornecem mecanismos de controle para o planejamento de um lançamento de produto e, em seguida, a gestão de variáveis com o avanço do projeto. Isso permite que as organizações alterem o projeto e as entregas em qualquer ponto no tempo, entregando a versão mais adequada (Schwaber, 1995). Segundo Nerur (2007 *apud* Moe *et al.*, 2010) esta visão tradicional mecanicista do mundo é desafiada pela perspectiva ágil que confere primazia à singularidade, à ambiguidade, à complexidade e à mudança, ao contrário de previsão, variabilidade e controle. O objetivo da otimização está sendo substituído por aqueles de flexibilidade e capacidade de resposta.

Em outras palavras, o foco, antes em processos e ferramentas, documentação abrangente, negociação de contratos e existência de um plano de curso, foi direcionado para a valorização de indivíduos e interações, software em funcionamento, colaboração com o cliente e resposta rápida à mudanças, sem contudo esquecer dos demais itens presentes na visão tradicional (Silva *et al*, 2012).

Neste contexto, observou-se uma empresa do ramo de desenvolvimento de projetos de software, que executa suas atividades com a adoção de equipes autogestoras de execução das atividades. Atualmente, a atribuição de tarefas, entre os desenvolvedores de cada equipe de trabalho, é feita pela metodologia *Scrum.* O *Scrum* é uma metodologia extremamente ágil e flexível, que tem por objetivo definir um processo de desenvolvimento iterativo e incremental que pode ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa, proporcionando um excelente entrosamento entre as equipes de desenvolvimento (Bissi, 2007).

Embora existam vários trabalhos relatando o uso do *Scrum*, verifica-se que os mesmos não levam em consideração as variações de tempo de execução das tarefas ou itens de *backlog* pelos desenvolvedores, o que pode levar a atrasos na finalização das etapas do projeto. Contudo, verifica-se na prática que este tempo, em geral, é tanto menor quanto maior for o nível de senioridade do desenvolvedor. Por este motivo, o objetivo deste trabalho é propor uma metodologia híbrida para gerenciamento de projetos de software. Nesta, as tarefas devem ser preliminarmente atribuídas a cada nível de senioridade pelo método de programação linear para designações de tarefas, de tal forma a minimizar o tempo total de execução das etapas do projeto. A gestão do projeto é, então, realizada pelo *Scrum.* A ideia central deste artigo se baseia na conciliação entre os benefícios da otimização da visão tradicional com os benefícios de equipes autogestoras das metodologias ágeis.

Além desta introdução o artigo apresenta outras cinco Secções. Na segunda Secção são apresentados alguns conceitos a cerca do gerenciamento de projetos, mais especificamente do método usado pela empresa objeto de estudo. Em seguida é feita uma explanação de como a atribuição de tarefas ocorre atualmente na empresa, de modo a ampliar a compreensão das melhorias propostas. Na quarta Secção são descritas todas as etapas da metodologia proposta. Algumas análises e discussões da metodologia são feitas na quinta secção. E por fim, realizam-se as considerações finais.

2. Gerenciamento de Projetos de software: abordagem ágil

Sistemas de informação exercem um importante papel no mundo moderno e seu rápido desenvolvimento e correto funcionamento são uma grande vantagem de negócio (Schoepping & Vilain, 2011). Assim, é preciso gerenciar todas as atividades de um projeto para que seja possível atingir ao objetivo final, ou conclusão do mesmo. Durante as duas últimas décadas, metodologias

de gestão vêm sendo cada vez mais empregadas, principalmente em função de fatores como surgimento de negócios na área de tecnologia da informação (TI) com soluções para problemas específicos, competitividade empresarial e necessidade de processos organizacionais que reflitam a real necessidade em um dado momento (Carneiro *et al.*, 2012).

Neste ponto, o gerenciamento de projetos de software pelo desenvolvimento ágil de projetos é diferente da visão tradicional e se apresenta como uma importante alternativa para o desenvolvimento de sistemas de informação (Slattery *et al.*, 2008; Schoepping & Vilain, 2011). Os métodos ágeis têm despertado um grande interesse na comunidade de desenvolvimento de sistemas. Acredita-se que, devido à esta demanda, uma considerável quantidade de métodos, apresentando características ágeis, têm surgido nos últimos anos. Podem-se citar alguns destes, como: *Extreme Programming* (XP), *Scrum*, *Feature Driven Development* (FDD) e *Adaptive Software Development* (ASD) (Fagundes *et al.*, 2008).

Dentre estes, neste artigo, será enfatizado a metodologia *Scrum*, uma vez que já está sendo utilizada pela empresa objeto de estudo. Assim, podem-se destacar as seguintes diferenças entre a abordagem tradicional e a abordagem ágil pelo *scrum* (Slattery *et al.*, 2008):

- ✓ Foco no planejamento e programação usando iterações em oposição à abordagem tradicional de planejamento e programação de um projeto inteiro;
- ✓ Uso de equipes menores do que as equipes tradicionais, onde os membros da equipe são selecionados com base nas habilidades necessárias para o projeto;
- ✓ Inclusão dos clientes durante todo o projeto e reunião em uma base diária por um curto período de tempo, em contraste com reuniões semanais que são realizados em um projeto tradicional;
- ✓ Criação de menos documentação, mas mais útil em comparação com projetos tradicionais, onde grandes quantidades de documentação podem ser criadas.

Na visão ágil há ênfase na equipe de colaboradores. Em uma equipe de software, os membros são solidariamente responsáveis pelo produto final. Assim, os objetivos do projeto, requisitos de sistema, planos e riscos do projeto, as responsabilidades individuais e o *status* do projeto deve ser visível e compreendido por todas as partes envolvidas (Moe *et al.*, 2010). Neste contexto, uma equipe (time) é "um pequeno número de pessoas com habilidades complementares que estão comprometidas com um propósito comum, conjunto de metas de desempenho e abordagem para a qual eles próprios são responsáveis" (Katzenbach, 1993).

2.1. Metodologia Scrum

Jeff Sutherland, Ken Schwaber e Mike Beedle criaram, em 1993, a metodologia de desenvolvimento de produtos chamada *Scrum*, que foi formalizada dois anos após (Carvalho, 2009; Rose & Mello, 2010). *Scrum* é um método de gestão de projetos orientado pelo desenvolvimento ágil. A autogestão é uma característica essencial no *Scrum*. Esta metodologia representa uma abordagem radicalmente nova para o planejamento e gerenciamento de projetos de software, porque ela traz a autoridade da tomada de decisão para o nível operacional de problemas e incertezas (Moe *et al.*, 2010). As fases de desenvolvimento *Scrum* podem ser divididas basicamente em três (Bissi, 2007), são elas:

- ✓ **Planejamento:** Definição de uma nova funcionalidade requerida pelo sistema baseado no conhecimento do sistema como um todo;
- ✓ **Desenvolvimento:** Desenvolvimento dessa nova funcionalidade respeitando o tempo previsto, requisitos exigidos e qualidade. Esses itens definem o fim do ciclo de desenvolvimento;
- ✓ **Encerramento:** Preparação para a entrega do produto persistindo as atividades de teste, documentação do usuário, treinamento e marketing.

Segundo Schwaber (1995) a tecnologia orientada a objeto fornece a base para a metodologia *Scrum*. Objetos, ou as características do produto, oferecem um ambiente discreto e gerenciável para o uso desta metodologia. Código processual, que tenha muitas interfaces e

entrelaçados, torna-se, no entanto, inadequado. Ainda segundo Schwaber (1995) o *Scrum* pode ser seletivamente aplicado a sistemas processuais, com interfaces limpas e orientação de dados forte. Para entender a metodologia é importante conhecer alguns dos termos utilizados (Bissi, 2007; Mallmann, 2011; Rose & Mello, 2010; Schwaber, 1995; Silva *et al*, 2012), como segue.

A Reunião de Planejamento da *Release* acontece no início do projeto, servindo para que todos os envolvidos no *Scrum* estejam juntos e possam definir as diretrizes para o projeto. A *Sprint* é um conjunto de atividades de desenvolvimento ao longo de um período pré-definido, geralmente uma a quatro semanas, onde o projeto (ou apenas algumas funcionalidades) é desenvolvido. O intervalo é baseado na complexidade do produto, avaliação de risco e grau de supervisão desejado. *Backlog* do produto (*Product Backlog*) é uma lista de todas as funcionalidades a serem desenvolvidas durante o projeto completo, sendo bem definido e detalhado no início do trabalho, deve ser listado e ordenado por prioridade de execução.

Cada elemento desta lista é conhecido como *estória* e deve possuir uma breve descrição das funcionalidades requeridas pelo *Product Owner* (é o responsável por priorizar o *backlog* do produto) para um dado produto. A equipe pode dividir uma *estória* em tarefas, também chamadas de itens de *backlog*, de preferência pequenas, para que, com a conclusão de todas as tarefas, a própria *estória* esteja concluída.

Na metodologia *Scrum* existem reuniões de acompanhamento diárias (*Daily Scrum*), em que as dificuldades encontradas e os fatores de impedimento são identificados e resolvidos. Além destas, após a elaboração do *backlog* do produto há a necessidade de se realizar uma reunião conhecida como Reunião de Planejamento do *Sprint*, em que se selecionam quais são as atividades que farão parte da *estória.* Também ocorrem reuniões para a revisão da *Sprint* e retrospectiva da *Sprint*.

A Revisão da *Sprint* acontece antes da reunião de planejamento da próxima *Sprint*. Nela, a equipe demonstrará ao *Product Owner* a evolução de suas atividades e, consequentemente, a funcionalidade que o *Product Owner* havia combinado na reunião de planejamento. Nesse momento, são também discutidas ideias que ajudarão na definição da próxima *Sprint*, com base nas lições aprendidas nesse período de desenvolvimento. Após a revisão do *Sprint*, os envolvidos deverão realizar a Retrospectiva. Nesta reunião o *Scrum Master* (Mestre do *Scrum* - tem como objetivo principal defender o *Scrum*, garantindo que a metodologia esteja funcionando de forma adequada e de acordo com as regras) deve fazer com que a equipe reflita sobre todas as lições que foram aprendidas durante a *Sprint*. A ideia principal desta reunião é gerar uma melhora contínua através destas lições prendidas.

O planejamento e a retrospectiva de uma *Sprint* também servem como espaços temporais para inspeção e alteração, quando necessária, na estrutura do processo que está sendo seguido, pois a cada iteração é possível que se façam alterações que melhorem a produtividade do time. O sucesso da metodologia *Scrum* passa pelo sucesso da definição dos itens de um *product backlog*, que não é uma tarefa trivial para equipes de desenvolvimento de software muito novas ou acostumadas com os métodos tradicionais de desenvolvimento. Uma equipe *Scrum* deve entender como lidar com itens do *product backlog*. O entendimento da definição deste é a base para o sucesso de qualquer meta de uma *Sprint* em *Scrum* (Biasoli e Fontoura, 2011).

Em todo gerenciamento de projetos é importante se fazer uma estimativa do tempo necessário para a realização de cada atividade, para que então, seja possível estimar o tempo total para finalizar o projeto em si. Como em seu desenvolvimento há o envolvimento de um grupo de pessoas, deve-se fazer uma estimativa do tempo que represente este time. Várias técnicas podem ser usadas com este intuito e pode-se destacar seis das técnicas mais comuns, que são: *Delphi*, *Wideband Delphi*, *Planning Poker*, *Unstructured groups*, *Statistical groups* e *Decision markets* (Moløkken-Østvold *et al.*, 2008).

Na empresa objeto de estudo usa-se a técnica *planning poker* (planejamento por *poker*). Esta técnica se constituiu no princípio de que várias cabeças pensam melhor que uma, garantindo que todos os desenvolvedores participem igualmente no processo de estimação, independentemente do seu nível de senioridade na empresa. A participação de pessoas com diferentes níveis ajuda na redução do otimismo excessivo do julgamento de especialistas para

identificar problemas que mais afetam a implementação e aliviar os problemas de ancoragem e personalidades fortes (Mahnic & Hovelja, 2012).

Com o uso de um baralho de cartas de *poker,* as instruções para jogar o *planning poker* são as seguintes etapas: (1) a estimativa da tarefa é apresentada, (2) a tarefa é discutida no grupo, e (3) cada participante escolhe uma carta representando a sua estimativa para aquela tarefa. (4) Uma vez que todos os participantes tenham escolhido uma carta, as cartas são viradas (com o número a mostra) simultaneamente. (5) Se todas as cartas mostram o mesmo número, esta será a estimativa, (6), caso contrário, o grupo discute centrando-se nos valores mais distantes, (7) os passos 3-6 são repetidos até que as estimativas convirjam, ou seja, quando a equipe chega a um consenso sobre aquela estimativa (Børte *et al.*, 2012).

Contudo, Moe *et al.* (2012) mostraram que a introdução do *Scrum* exige mudanças em todos os níveis da organização: de como os recursos são alocados, como a organização apoia as equipes e como atrasos estão alinhados com as decisões estratégicas, mudanças de poder na tomada de decisões e implementação de um processo de compartilhamento na tomada de decisão ao nível operacional.

2.2. Programação linear: Método de designação de tarefas

O Modelo da Designação é um caso particular do Modelo de Transporte da Programação Linear que procura representar situações onde se faz necessário alocar os recursos (indivisíveis) disponíveis para atender de maneira exclusiva às atividades de interesse, de modo que alguma medida de efetividade (custo total ou tempo de execução) do sistema modelado seja otimizada. Neste tipo de problema, deve ser alocado um recurso para cada atividade e toda atividade deve receber apenas um recurso. Situações em que o número de recursos é diferente do número de atividades requerem a aplicação de um artifício simples, com a inclusão de recursos ou atividades fictícias conforme necessário (Marins, 2011).

Assim, para o problema proposto, a formulação do problema será: dado i itens de *backlog*, $i=\{1, 2, ..., n\}$, e j número de desenvolvedores, $j=\{1, 2, ..., m\}$; sendo t_{ij} a duração (em horas) de cada item de *backlog* por desenvolvedor (os desenvolvedores com o mesmo nível de senioridade possuem a mesma duração para cada atividade); x_{ij} representa a designação do desenvolvedor j para determinada tarefa i (itens de *backlog*), sendo que se $x_{ij}=1$ é designado o item de *backlog* 'i' ao desenvolvedor 'j' e 0 caso contrário. Assim, o modelo para designação das tarefas se dá conforme equação 1.

$$Min\ Z = \sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} * x_{ij}$$

s.a.

$\sum_{j=1}^{m} x_{ij} = 1$, para $i = \{1, 2, ..., n\}$ (1)

$\sum_{i=1}^{n} x_{ij} = 1$, para $j = \{1, 2, ..., m\}$

$x_{ij} = 0$ ou 1

Para a sua resolução é comum utilizar o método Húngaro. Após montar uma matriz com todos os dados, os passos básicos são (Moreira, 2009):

1. Subtrair de todos os números de cada linha o menor deles;
2. Subtrair de todos os números de cada coluna o menor deles;
3. Traçar o número mínimo de retas horizontais / verticais de forma a cobrir todos os zeros gerados;
4. Adicionar o menor número não coberto aos números que estão nas intersecções entre as retas; ao mesmo tempo esse menor número é subtraído de todos os demais números não cobertos pelas retas.
5. A solução será aquela em que seja possível atribuir pelo menos um recurso a cada atividade onde aparecem zeros. Voltar para o passo 3 caso não tenha encontrado uma solução viável.

3. Estudo de caso

A empresa objeto de estudo, localizada no porto digital no município de Recife/PE, tem como ramo de negócio o desenvolvimento de projetos de software conforme especificações do cliente. A empresa é projetizada, ou seja, não possui um produto único em desenvolvimento, mas sim uma gama de projetos específicos para cada cliente. Neste sentido ela precisa de uma metodologia de apoio a gestão no desenvolvimento de seus softwares que permita o envolvimento do cliente durante o processo.

Atualmente, a empresa segue, no seu processo de desenvolvimento de software, a metodologia de gerenciamento ágil, o *Scrum*. Neste, cada projeto utiliza a metodologia de acordo com as suas necessidades, nível de interação com cliente, nível de maturidade da equipe, tamanho do escopo e tempo para entrega do projeto. Após a definição do *backlog* da *sprint* é realizada a definição dos itens de *backlog* (IB). Os itens são apresentados aos desenvolvedores, onde cada um selecionará aquele que irá realizar. A duração dos itens de *backlog* é variada e baseada em dias, dependendo dessa duração os itens podem ser quebrados em atividades diárias para um melhor acompanhamento do progresso do desenvolvimento.

A equipe destes projetos é constituída por desenvolvedores multifuncionais em diversos níveis de senioridade (Estagiário, *Trainee*, Junior, Pleno, Sênior). No entanto a alocação dos itens de *backlog* não leva em consideração essa informação, já que, uma vez disponível no quadro, o item pode ser escolhido por qualquer desenvolvedor da equipe, o que pode acarretar no atraso da entrega deste item, dependendo de sua complexidade.

Verifica-se na prática que o tempo de execução de uma tarefa, em geral, é tanto menor quanto maior for o nível de senioridade do desenvolvedor. Além disso, caso os itens não sejam bem alocados dentre os desenvolvedores, ou seja, se os desenvolvedores escolherem itens que estão em um nível de complexidade acima do seu nível de senioridade, isso pode acarretar em um atraso na entrega da *sprint*. Como exemplo, suponha uma equipe composta por 04 desenvolvedores, tal que $j=\{1, 2, 3, 4\}$, e sua senioridade distribuída da seguinte maneira: 01 *Treinee*; 02 Juniores; 01 Pleno. A tabela 1 apresenta os tempos médios esperados de execução de itens de *backlog* baseados no histórico de um projeto realizado pela empresa estudada. Os tempos foram obtidos através do *Planning Poker*.

Tabela 1. Tempos esperados de execução dos itens de *backlog* (IB)

Itens	Tempo (dias)	Itens	Tempo (dias)	Itens	Tempo (dias)
IB_1	4	IB_4	5	IB_7	4
IB_2	6	IB_5	7	IB_8	3
IB_3	3	IB_6	2	IB_9	5

Durante o decorrer da *sprint* foi utilizada, para a designação das tarefas, a forma padrão do *Scrum*, em que os envolvidos escolhem um dos itens que estiverem disponíveis no quadro para ser executado e que ao término dessa execução outro IB deve ser escolhido do quadro. A figura 1 mostra o tempo real de execução dos itens de *backlog* pelos desenvolvedores e a sequência de IB's que foi escolhida por cada desenvolvedor. De maneira intuitiva, os desenvolvedores tenderam a "puxar" as atividades na ordem apresentada.

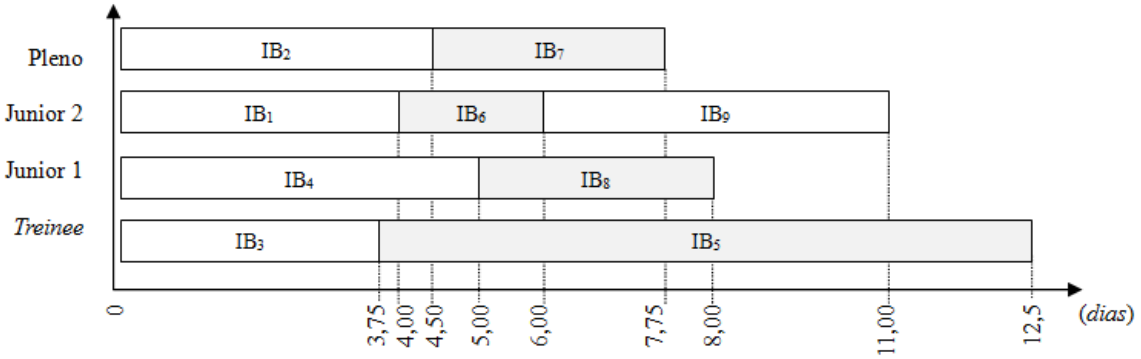

Figura 1. Tempo real de execução dos itens de *backlog*

Como pode ser visto na figura 1, houve um atraso na entrega do IB_5 por parte do *treinee*, uma vez que ele levou mais tempo para concretizar o item de *backlog* do que o tempo estimado. Este fato levou ao atraso na entrega da *Sprint*. Além disso, enquanto os demais desenvolvedores aguardam o *treinee* concluir sua tarefa, eles ficam subutilizados após a conclusão dos itens escolhidos pelos mesmos.

Em alguns casos os integrantes ociosos podem colaborar com aqueles que ainda não finalizaram suas tarefas, mas essa colaboração é subjetiva, pois pode depender do estágio atual de desenvolvimento do IB, se já está próximo ao término e o desenvolvedor está bloqueado por um problema específico, e se não existem atividades paralelas a *sprint*, como a investigação de uma nova tecnologia para ser usada futuramente no projeto. Assim, a equipe constatou que a forma como os IB's foram escolhidos pelos desenvolvedores, sem levar em consideração a senioridade e o tempo médio esperado de execução, não foi eficiente.

O problema da alocação das tarefas sem considerar o nível de senioridade, muitas vezes, pode ser resolvido por um simples acompanhamento por parte do gerente do projeto. No entanto, em grandes empresas, como é o caso deste estudo, os gerentes são responsáveis por várias equipes, dificultando o seu acompanhamento sobre todo o processo. Portanto, o uso de uma ferramenta de apoio à decisão pode, além de auxiliar no acompanhamento do andamento das atividades pelo gerente, ajudar os desenvolvedores na escolha das tarefas que melhor se adéquam a sua senioridade.

O objetivo desta metodologia proposta não é impedir a flexibilidade da alteração das designações das atividades. Mas, caso ocorra alguma mudança, o gerente obterá informações com as quais seja possível avaliar o impacto destas mudanças sobre o prazo de entrega, podendo adotar medidas que minimizem ou, mesmo, eliminem este atraso.

4. Aplicação da metodologia proposta

A partir do problema ressaltando anteriormente, propõe-se o desenvolvimento de uma metodologia híbrida de apoio na decisão quanto a designação de tarefas e visualização do andamento do projeto. A figura 2 apresenta uma ilustração dos passos da proposta.

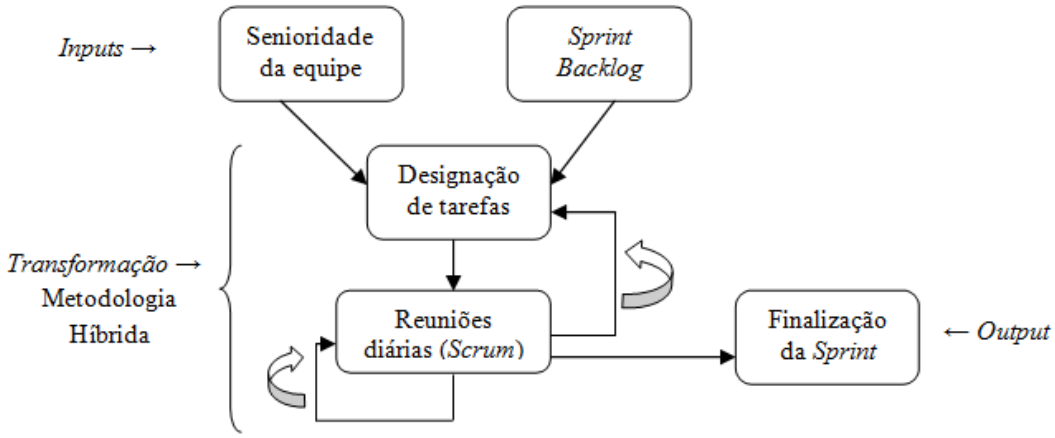

Figura 2. Fluxograma da proposta

A metodologia proposta recebe como *input* os IB's escolhidos para a *sprint*, associados ao tempo médio de execução esperado para cada um deles, assim como a senioridade dos integrantes da equipe, essas informações serão utilizados na etapa de designação de tarefas. Os tempos esperados de execução são estimados pelo "*planning poker*".

Deve-se ressaltar que na prática do "*planning poker"*, comumente espera-se que os desenvolvedores sejam capazes de estimar um tempo de execução de apenas uma senioridade. Em geral, isto é feito para um Júnior, pois este se encontra em um nível de senioridade intermediário aos demais. Porém, como já ressaltado, esta estimativa padrão dos tempos, nivelada pela "média" das senioridades, pode levar à subutilização de desenvolvedores de alta senioridade e atrasos por parte daqueles de menor senioridade.

Por isso, a atribuição preliminar das tarefas proposta utiliza o nível de senioridade dos desenvolvedores com o intuito de minimizar o tempo total de execução da *sprint*. Desta forma, ao invés do desenvolvedor "puxar" uma tarefa pra ele, a metodologia indicará quais atividades são indicadas ao seu nível de senioridade.

Para aplicar o método de designação é preciso conhecer os tempos de execução esperados a cada nível de senioridade e não o tempo médio esperado, como é feito, normalmente. Na prática da empresa estudo observaram-se, empiricamente, os seguintes tempos padrões para uma mesma atividade: 1,25 *x* dias executada por um *Treinee*; 1,5*x* por um estagiário; *x* dias por um Junior; 0,75*x* por um Pleno; e 0,5*x* por um Sênior. Essas velocidades podem variar de acordo com o projeto e as especialidades dos integrantes da equipe. Assim, considerando que no exemplo apresentado anteriormente os tempos apresentados eram de um Junior, a tabela 2 mostra os tempos dos demais desenvolvedores.

Tabela 2. Tempos médios esperados de execução baseados na senioridade dos desenvolvedores

Desenvolvedores	**Tempo das atividades (em dias)**								
	IB_1	IB_2	IB_3	IB_4	IB_5	IB_6	IB_7	IB_8	IB_9
Treinee	5,00	7,50	3,75	6,25	8,75	2,50	5,00	3,75	6,25
Junior 1	4,00	6,00	3,00	5,00	7,00	2,00	4,00	3,00	5,00
Junior 2	4,00	6,00	3,00	5,00	7,00	2,00	4,00	3,00	5,00
Pleno	3,00	4,50	2,25	3,75	5,25	1,50	3,00	2,25	3,75

No cronograma de atividades deverá aparecer a designação dos itens de *backlog* por senioridade e sua duração. Como pode ser visualizado na tabela 2, há um número maior de atividades do que desenvolvedores. Isto é normal em gerenciamento de projetos de software. Assim, uma vez que os primeiro itens foram alocados, aqueles remanescentes devem ser alocados na sequência temporal, ou seja, nos espaços disponíveis no cronograma, reaplicando o método de designação. Portanto, várias "rodadas" do método são realizadas até que todas as atividades sejam alocadas. Os desenvolvedores poderão escolher os itens de acordo com sua senioridade, caso haja mais de um desenvolvedor com a mesma senioridade. Após cada desenvolvedor ter concluído seu primeiro item, deverá escolher outro (da segunda "rodada") e, assim, por diante.

Portanto, aplicando-se o método de designação, a partir dos dados na tabela 2, tem-se o seguinte resultado, na sequência de execução: (1) *Treinee* = IB_6+ IB_1= 7,5 dias; (2) Junior 1 = IB_3 + IB_4= 8 dias; (3) Junior 2 = IB_8+ IB_9= 8 dias; Pleno = IB_7+ IB_2 + IB_5= 12,75 dias. Note que a duração total da *Sprint* foi de *z* = 12,75 dias, enquanto que na forma intuitiva de atribuição dos itens levou-se *z* = 12,5 dias. Porém, uma vantagem foi a alocação de maior responsabilidade aos profissionais de maior senioridade, o que possibilita um melhor gerenciamento das atividades.

Assim, percebeu-se a necessidade de que o número de itens de *backlog* seja um múltiplo do número de desenvolvedores. Para tanto, basta agregar ou dividir as tarefas definidas preliminarmente. Para melhorar a compreensão, imagine a agregação dos itens de *backlog* mais simples, ou seja, os itens IB_6 + IB_8 (poderia ser o IB_3). A tabela 3 apresenta o resultado das duas rodadas do método de designação. Os itens em cinza são as atividades designadas a cada desenvolvedor.

Tabela 3. Aplicação do método de designação

1° Rodada: como o Pleno possui os melhores tempos em todas as atividades, optou-se por designar a ele os itens de *backlog* com os maiores tempos. Como consequência tem-se a atribuição das maiores complexidades ao maior nível de senioridade.

Desenvolvedores	IB_1	IB_2	IB_3	IB_4	IB_5	IB_6+ IB_8	IB_7	IB_9
Treinee	0,25	1,25	0,00	0,75	1,75	0,75	0,25	0,75
Junior 1	0,00	0,50	0,00	0,25	0,75	0,25	0,00	0,25
Junior 2	0,00	0,50	0,00	0,25	0,75	0,25	0,00	0,25
Pleno	0,00	0,00	0,25	0,00	0,00	0,00	0,00	0,00

Continuação da Tabela 3 - 2° Rodada				
Desenvolvedores	IB_2	IB_4	IB_6+ IB_8	IB_9
Treinee	0,25	0,00	0,00	0,00
Junior 1	0,00	0,00	0,00	0,00
Junior 2	0,00	0,00	0,00	0,00
Pleno	0,00	0,25	0,25	0,25

A partir dos dados da tabela 3 chega-se a uma nova proposta de cronograma totalizando z = 10 dias, como mostra a figura 3.

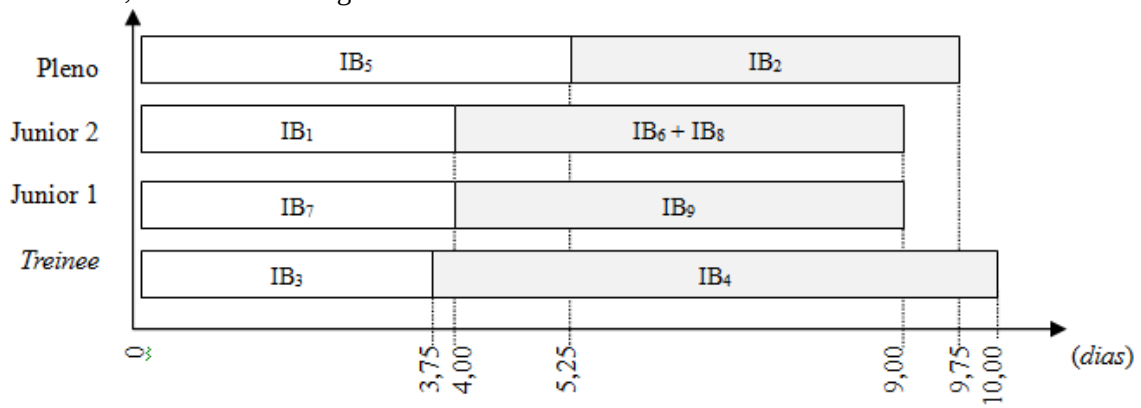

Figura 3. Cronograma de atribuição dos itens de *backlog*

Durante a etapa de reuniões diárias de acompanhamento, a equipe pode decidir por não seguir o cronograma proposto pela designação, ou seja, pode haver troca de IB's entre os desenvolvedores, ou que ainda existam atrasos na entrega de algum IB. A metodologia prevê que isso possa ocorrer por ser uma característica inerente ao *Scrum* e ao desenvolvimento de projetos, e coloca para o gerente de projeto a possibilidade de executar novamente o processo de designação, desta vez, sem as atividades já concluídas e com a alteração das atribuições realizadas pelos desenvolvedores. Essa nova rodada poderá servir, inclusive, para que o gerente do projeto visualize os impactos dessa mudança no cronograma.

5. Análises e discussões

A metodologia proposta não impede as possíveis alterações nas atribuições das tarefas, uma vez que não se objetiva eliminar a flexibilidade do processo de desenvolvimento colocado pelo *scrum*, mas sim contribuir para o bom andamento do projeto, ajudando o gerente no acompanhamento da execução dos IB's.

Esta metodologia propicia um cenário no qual os desenvolvedores de maior senioridade "puxam" as atividades de maior complexidade para sua responsabilidade. Algumas equipes já realizam esse procedimento, porém a metodologia tem o intuito de tornar essa prática comum, já que se percebe nos projetos realizados na empresa estudo, que essa prática acelera o processo de desenvolvimento do projeto.

Um tempo padrão pode ser estimado pelo uso da metodologia "*planning poker*", como já ocorre. Contudo, a avaliação, do tempo esperado de um Júnior (nível intermediário), feita pelo próprio Júnior ou por um desenvolvedor de maior senioridade pode ser intrínseco, mas muito difícil para aqueles de menor senioridade. Por isso, acredita-se que o melhor é usar como base a menor senioridade da equipe e todos os envolvidos devem estimar o tempo de concretização das tarefas por este. Depois, basta aplicar as razões estimadas anteriormente (estagiário*: Treinee*: Júnio: Pleno: Sênior = 1,50: 1,25: 1,00: 0,75: 0,50). Portanto, dado o tempo de um estagiário igual a 6 dias, o tempo de um Júnior será: x=(6*1,00)/1,50 = 4 dias.

Durante a execução do método, a cada rodada da designação, os IB's de maior complexidade são atribuídos aos integrantes de maior senioridade (Junior, Pleno e Sênior), enquanto que as atividades de menor complexidade são entregues aos desenvolvedores de menor senioridade (Estagiário e *Treinee*).

Para o melhor funcionamento do método aplicado necessita-se que as atividades executadas durante a *sprint* sejam múltiplas do número de desenvolvedores da equipe. Caso não sejam, a agregação de atividades menores, ou a quebra de atividades maiores (mais complexas) é uma maneira de tornar a aplicabilidade do método eficaz. Isto não retrata uma limitação da metodologia, uma vez que esta agregação ou quebra é facilmente realizada pelas equipes.

Um ponto a ser considerado na aplicação do método proposto é a ajuda proporcionada aos gestores em cobrar mais da equipe, uma vez que, conhecido o desempenho esperado de cada um e os tempos médios esperados é possível visualizar os atrasos e verificar quais dificuldades podem não ter sido estimadas ou se houve algum desempenho irregular por parte dos integrantes da equipe.

Os tempos esperados nas execuções dos IB's nem sempre ocorrem. Assim, o método deve ser atualizado a cada concretização de itens, possibilitando alterações nas designações. Além disso, durante o processo de desenvolvimento é possível que a equipe deseje alterar as atribuições feitas pelo método por motivos alheios ao seu desenvolvimento. Quando isto ocorrer, as designações devem ser atualizadas e um novo cronograma gerado.

6. Considerações finais

Neste artigo apresentou-se uma nova metodologia híbrida para gerenciamento de projetos de software. A proposta se baseia na combinação do método de programação linear de designação de tarefas com o método de gerenciamento de tarefas o *Scrum*. A proposta auxilia o gerenciamento e otimiza os tempos de concretização da *sprint* sem, contudo, retirar a flexibilidade inerente das metodologias ágeis.

Conclui-se que, a metodologia proposta, pode, além de atribuir as tarefas de modo a minimizar o tempo total de execução do *sprint*, auxiliar os gerentes na avaliação dos desenvolvedores, uma vez que será possível verificar se estes estão cumprindo os prazos dentro das metas estabelecidas para o seu nível de senioridade, além de não interferir nas práticas utilizadas pelo *scrum*.

Alguns autores vêm discutindo a possibilidade de que algumas tarefas quando combinadas propiciarem um ganho (em tempo) maior do que quando realizadas por indivíduos diferentes. No caso apresentado, alguns itens de *backlog* tem semelhanças em sua execução que podem ser otimizadas no caso do mesmo desenvolvedor as realizar. Assim, como futuro trabalho estuda-se meios de realizar a atribuição preliminar com base nas melhores combinações. Para tanto, é preciso realizar um estudo a cerca dos ganhos nos tempos com cada combinação. Além disso, como futuro trabalho, deixa-se a necessidade do desenvolvimento de um software que realize o procedimento proposto e auxilie na visualização do cronograma.

Referências

Biasoli, D. e Fontoura, L. M. (2011), Auxílio de redes de Petri coloridas na implantação de projetos Scrum, *VII Simpósio Brasileiro de Sistemas de Informação*, Salvador, BA, 178 – 189.

Bissi, W. (2007), Scrum – Metodologia de desenvolvimento ágil, *Campo Dig.*, Campo Mourão, 2 (1), 3-6.

Børte, K., Ludvigsen, S. R., Mørch, A. I. (2012), The role of social interaction in software effort estimation: Unpacking the ''magic step'' between reasoning and decision-making, *Information and Software Technology*, 54, 985–996.

Carneiro, L. S., Queiroz, M. A., Barros, R. M., Brancher, J. D. (2012), Implementação da ISO 9001 com Scrum um estudo de caso, *VIII Simpósio Brasileiro de Sistemas de Informação*, São Paulo, SP, 222 – 230.

Carvalho, B. V. (2009), Aplicação do método ágil Scrum no desenvolvimento de produtos de softwares em uma empresa de base tecnológica, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Itajubá/MG. (Dissertação)

Dybå, T. (2000), Improvisation in small software organizations, *IEEE Software*, 17 (5), 82–87.

Fagundes, P. B., Deters, J. I., Santos, S. S. (2008), Comparação entre os processos dos métodos ágeis: XP, SCRUM, FDD e ASD em relação ao desenvolvimento iterativo incremental, *E-Tech: Atualidades Tecnológicas para Competitividade Industrial*, 1 (1), 37-46.

Katzenbach, J.R. e Smith, D.K. (1993), The discipline of teams, *Harvard Business Review*, 71 (2), 111–120.
Mahnic, V. e Hovelja, T. (2012), On using planning poker for estimating user stories, *The Journal of Systems and Software*, 85, 2086– 2095.
Mallmann, P. R. (2011), Um modelo abstrato de gerência de software para metodologias ágeis, Universidade do Vale do Rio dos Sinos - UNISINOS, Dep. Computação Aplicada. São Leopoldo. (Dissertação)
Marins, F. A. S. *Introdução à Pesquisa Operacional*, São Paulo, Cultura Acadêmica: Universidade Estadual Paulista, Pró-Reitoria de Graduação, 2011.
Moe, N. B., Aurum, A., Dybå, T. (2012), Challenges of shared decision-making: A multiple case study of agile software development, *Information and Software Technology*, 54, 853–865.
Moe, N. B., Dingsøyr, T., Dybå, T. (2010), A teamwork model for understanding an agile team: A case study of a Scrum project, *Information and Software Technology*, 52, 480–491.
Moløkken-Østvold, K., Haugen, N. C., Benestad, H. C. (2008), Using planning poker for combining expert estimates in software projects, *The Journal of Systems and Software*, 81, 2106–2117.
Moreira, D. A. *Administração da produção e operações*, 2.ª edição revista e ampliada, São Paulo, Pioneira, 2009
Rose, T. P. R. e Mello, C. H. P. (2010), Proposta de sistemática para gestão de projetos baseada na metodologia ágil scrum, **XXX Encontro Nacional de Engenharia de Produção,** São Carlos, SP, Brasil.
Schoepping, G. e Vilain, P. (2011), Analisando a Agilidade em Processos Ágeis, *VII Simpósio Brasileiro de Sistemas de Informação*, Salvador, BA, 274 – 285.
Schwaber, K. (1995), Scrum Development Process, *Workshop on Business Object Design OOPSLA'95.*
Silva, A. L. M., Cavalheiro, L. T. A., Roman, N. T., Chaim, M, L. (2012), Implementação de metodologia de desenvolvimento ágil em projetos com time alocado e não alocado, *VIII Simpósio Brasileiro de Sistemas de Informação*, São Paulo, SP, 327 – 336.
Slattery, S., Rooney, M, Treacy, F., Staunton, C., McHugh, O. (2008), A study of XP & Scrum: A Project Management Perspective, *Proceedings of the 3rd International Business Informatics Challenge and Conference*, Dublin City University, September 25th,.