5-2015

Automated Support for Scrum Projects Sprint Planning

Hanna Tátila Sousa
Inst. Federal do Espírito Santo, Serra, hanna.tati@gmail.com

Thanner Soares Silva
Inst. Federal do Espírito Santo, Serra, thannersoares@gmail.com

Paulo Sérgio Santos Junior
Inst. Federal do Espírito Santo, Serra, paulo.junior@ifes.edu.br

Rodrigo Fernandes Calhau
Inst. Federal do Espírito Santo, Serra, calhau@ifes.edu.br

Mateus Barcellos Costa
Inst. Federal do Espírito Santo, Serra, mbarcosta@ifes.edu.br

Apoio Automatizado ao Planejamento de Sprints em Projetos Scrum

Alternative Title: Automated Support for Scrum Projects Sprint Planning

Hanna Tátila de Sousa Thanner S. Silva
Paulo S. Santos Jr. Rodrigo F. Calhau Mateus Barcellos Costa

Lab. E. Soft. C. Eletrônico - SEEC
Inst. Federal do Espírito Santo
Serra - ES - Brasil
{hanna.tati,thannersoares}@gmail.com;
{paulo.junior,calhau,mbarcosta}@ifes.edu.br

RESUMO

Neste artigo é discutida uma metodologia para apoio automatizado ao planejamento de *Sprints* em projetos baseados em *Scrum*. A metodologia permite determinar o conjunto ótimo de tarefas de um *Sprint*, bem como a ordem de realização destas, considerando-se a informação *a priori* correspondente ao estado atual do projeto, à dificuldade e à importância das tarefas envolvidas, e relações entre tarefas. A solução proposta baseia-se na representação do espaço de decisão do projeto, definido por meio desta informação, como modelos de workflow. Para avaliar a abordagem, um protótipo foi desenvolvido e está sendo testado no contexto de um projeto de desenvolvimento real. Os resultados sugerem que o uso da abordagem provê um maior determinismo para o processo de planejamento de *Sprints*, maior facilidade para visualização e avaliação dos possíveis cenários do projeto e diminuição no tempo para determinação dos *Sprints*.

Palavras-Chave

Otimização da Ordem de Processos, Re-sequenciamento, Scrum, Gerenciamento de Workflow

ABSTRACT

In this paper an approach to support Sprint Planning in Scrum projects is discussed. This approach allows the automatic determination of the optimal set of tasks for a given sprint, as well as its execution flow, considering the *a priori* information concerning the current state of the project, tasks's difficulty and importance, and the relations between tasks. The proposed solution is based on the representation of the project decision space, defined by this information, as workflow models. To evaluate the approach, a prototype has been developed and is being assessed in the context of the Cornea Notification and Collecting Information System (Sincap) project. The results suggest that the use of the approach provides greater determinism for the sprint planning process, easies the viewing and evaluation of project scenarios and decreases planning time.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow management*; K.6.1 [**Management of computing and information systems**]: Project and people management—*System Analysis and Design*

General Terms

Management

Keywords

Process Order Optimization, Resequencing, Scrum, Workflow Management

1. INTRODUÇÃO

O Desenvolvimento de Software tem se tornado cada dia mais complexo. Os ambientes organizacionais em seu entorno têm condicionado esta atividade a uma dinâmica severa na qual aspectos como agilidade, equipes auto-organizáveis e gerenciamento permanente de contingências têm se tornado fatores críticos de sucesso. Para se incorporar estes fatores, um número ainda crescente de organizações desenvolvedoras de software tem feito a opção por Métodos Ágeis [5].

Um dos principais métodos de gerenciamento ágil de projetos de software é o *Scrum* [9]. O gerenciamento com *Scrum* implica na utilização de um framework de procedimentos e artefatos que conduzam aos objetivos da auto-organização das equipes de desenvolvimento e da agilidade na produção de código executável. Sendo assim, projetos que utilizam *Scrum* abandonam práticas documentais e de planejamento centralizado, em favor de um conjunto de princípios que garantam a integração entre planejamento e implementação. Em decorrência dessa opção de gerenciamento, surgem desafios para se garantir a qualidade da tomada de decisão

que, como regra geral, depende, fundamentalmente, da experiência e do conhecimento das equipes envolvidas.

Uma alternativa para se tratar estes desafios é a adoção, junto ao framework *Scrum*, de elementos provenientes de modelos típicos de desenvolvimento de software, por exemplo, de elementos que contemplem áreas chave do modelo CMMI [6]. Outra alternativa está no aprimoramento dos procedimentos adotados no ciclo de desenvolvimento padrão do *Scrum*. Considerando-se essa alternativa, foi desenvolvida, neste trabalho, uma metodologia de apoio à tomada de decisão que permite definir, de forma automática, o subconjunto de atividades a ser realizado no próximo ciclo do projeto, bem como a ordem em que estas atividades deverão ser realizadas (*Sprints*). A metodologia proposta permite gerar o espaço de decisão referente ao estado atual de um projeto por meio de um modelo de workflow que descreve relações de fluxo entre as atividades, por exemplo, relações de dependência de execução, e também relações quantitativas entre essas atividades. Por fim, o método permite determinar *sprints* por meio da avaliação desse modelo e busca das sequências de atividades ótimas.

Um protótipo para a avaliação da abordagem foi desenvolvido e está sendo correntemente avaliado no contexto do Projeto Sincap - Sistema de Informação de Notificação e Captação de Córnea. Partes desse projeto são usadas como exemplo ilustrativo. O restante do artigo está organizado da seguinte maneira: na Seção 2, aspectos do planejamento de *sprints* são discutidos. Na Seção 3, os passos da metodologia são apresentados. A Seção 4 apresenta os principais mecanismos utilizados na construção da metodologia. Na Seção 5, um exemplo de aplicação é apresentado. Na Seção 6, o artigo é concluído e trabalhos futuros são apresentados.

2. PLANEJAMENTO DE SPRINTS

Scrum é uma maneira ágil para organizar e gerenciar um projeto, geralmente de desenvolvimento de software. Segundo Schwaber [9], *Scrum* não é uma metodologia mas sim um framework para o gerenciamento, o qual pode e deve ser adaptado à realidade de cada situação. As informações sobre as atividades a serem desenvolvidas em um projeto *Scrum* são organizadas, decompostas e armazenadas como uma lista de funcionalidades (requisitos) e tarefas de apoio a serem realizadas. Esta lista é denominada de *Product Backlog* ou simplesmente *Backlog*[1]. Um *backlog* pode sofrer várias alterações durante o processo de execução do projeto, devido a novas questões que vão, aos poucos, sendo observadas.

A partir de um *backlog* inicial, o desenvolvimento do projeto ocorre em ciclos curtos de atividades denominados de *sprint*, que duram, normalmente, um período de tempo entre 2 e 4 semanas. Um *sprint* é definido em uma reunião de Planejamento com a presença de todos os envolvidos no projeto. O objetivo desta reunião é definir um subconjunto de funcionalidades do *backlog* que devem ser executadas no próximo ciclo. Cabe à equipe de desenvolvimento determinar quantos itens eles podem se comprometer a implementar. Os itens escolhidos são definidos pela equipe com base na importância de cada tarefa e na estimativa de tempo necessária para realizá-las, bem como na capacidade a ser alocada para o *sprint*. Essa estimativa é estipulada com base na percepção da equipe.

O *Scrum* conta com a habilidade da equipe para maximizar o uso de sua capacidade de trabalho, a fim de entregar no prazo as tarefas do *sprint* e responder rapidamente às necessidades emergentes. Por isso o *sprint* precisa ser muito bem definido, com alguma garantia de que tudo o que foi selecionado seja implementado no tempo previsto. Normalmente, espera-se que a equipe de desenvolvimento se auto-organize para determinar a melhor maneira de realizar o trabalho para atingir a meta estabelecida pelo *Product Owner*. Ou seja, a equipe é responsável por analisar o *backlog* e definir, baseando-se no autoconhecimento e experiência, qual poderia ser o melhor conjunto de funcionalidades a ser implementado em cada *sprint*. Dessa forma, esta decisão é crucial para garantir cumprimento dos objetivos.

A ideia do apoio automatizado ao planejamento de projetos não é recente e está registrada na literatura em inúmeros trabalhos, em particular nas áreas de Pesquisa Operacional e Apoio à Decisão. O apoio a projetos *Scrum* surge em iniciativas recentes, estimulados pelo fato de que projetos de maior escala ou que apresentem natureza particular como, por exemplo, projetos globais e descentralizados, vêm adotando esta metodologia com maior frequência [3, 11]. Um método em particular com o objetivo de melhorar a qualidade na execução de *sprints* é discutido por Biasoli e Fontoura [1]. Nestes modelos são utilizados redes de Petri coloridas para modelar os cenários que estão por trás de cada item de um *backlog* e assim, avaliar melhor a natureza das atividades que estes itens representam.

Um aspecto notado em diversos trabalhos no contexto do apoio ao planejamento de projetos, por exemplo em Maurer et Al [7], foi a ênfase ao comprometimento das atividades com os recursos disponíveis. De forma geral, o planejamento de projetos tem sido associado a modelos de workflow, visando o balanceamento entre o fluxo de trabalho e os recursos disponíveis. Diferentemente, neste trabalho não são considerados os recursos alocados às tarefas, supondo-se que esta relação entre tarefas e recursos é implícita. A partir desta consideração, o problema do re-sequenciamento de atividades (activities resequencing ou process order optimization) [8], levando-se em conta apenas as relações entre tarefas que impactam na realização das mesmas, foi considerado de forma particular e consiste no foco dessa pesquisa.

3. METODOLOGIA PARA OBTENÇÃO AUTOMÁTICA DE SPRINTS

A metodologia proposta neste trabalho prevê a determinação de todas as alternativas de planejamento possíveis dado um conjunto de restrições identificado no *backlog* do projeto, e a determinação, dentre estas alternativas, daquele mais eficiente, com relação aos parâmetros de avaliação considerados. Para a determinação dessas alternativas, o modelo de *backlog* foi modificado a fim de conter toda a informação *a priori* referente ao estado atual do projeto. Com base no modelo de *backlog* proposto é possível obter os diferentes modelos de fluxos de trabalho que podem ser seguidos no projeto a fim de atingir um estado a frente do estado atual. Por fim, estes fluxos podem ser inspecionados para se identificar melhores opções de *sprints*. Os conceitos de *Backlog* Estendido, Matriz de Restrições, Sistema de Transições e

[1]Dado que, para efeito no planejamento, uma funcionalidade ou uma tarefa tem o mesmo significado. Para simplificação do texto consideraremos doravante, um *backlog*, como sendo formado por uma lista de tarefas.

Grafo de Estados são discutidos a seguir.

3.1 Modelo de Backlog Estendido

Um *backlog* típico lista, para cada tarefa, os valores de dificuldade e importância. Ambos os valores são estimativas. A dificuldade é uma estimativa, em termos de esforço da equipe ou tempo, do quão difícil será a realização de uma tarefa. Estimativas podem se basear, por exemplo, na avaliação de dados históricos de desempenho da equipe. A importância, por sua vez, reflete a prioridade de uma funcionalidade dentro da estratégia do desenvolvimento ou a urgência da mesma considerando o ponto de vista do cliente. Tanto a dificuldade quanto a importância de uma funcionalidade podem variar no decorrer do projeto.

Além da importância e dificuldade, outras informações podem ser consideradas no planejamento. Dentre estas estão as relações de dependência e de benefício relativo. Relações de dependência indicam restrições na realização de uma tarefa. Relações de benefício relativo indicam o impacto ou efeito da realização de uma tarefa sobre a dificuldade de realização de outra tarefa. Tais informações têm sido utilizadas em projetos *Scrum*, não necessariamente de forma sistemática. No contexto do Projeto Sincap discutido na Seção 5, essas relações são obtidas em uma etapa chamada mapeamento de *User Stories*. Para a metodologia proposta, elas devem ser registradas no *backlog*, derivando uma estrutura que denominamos de *Backlog* Estendido. A estrutura completa do *backlog* estendido possui os seguintes itens de informação:

1. Identificador de Tarefa: Compreende nome da tarefa e deve ser único. Em adição a convenção adotada para o identificador de uma tarefa permite também identificar diferentes alternativas de uma mesma tarefa.

2. Dependência: Expressa uma restrição de execução de uma tarefa causada por outras tarefas presentes no *backlog*. Para uma tarefa t e um subconjunto do *backlog* t, r, s, alguns possíveis valores para a dependência de t podem ser observados a seguir:

   (a) N/A: A tarefa t não depende da execução de nenhuma outra tarefa;

   (b) r: A tarefa t só pode ser realizada se a tarefa r já se encontra executada;

   (c) r,s: A tarefa t só pode ser realizada se as tarefas r e s já se encontram executadas;

   (d) r;s: A tarefa t só pode ser realizada se as tarefas r ou s já se encontram executadas.

3. Importância: É uma medida que expressa o quão necessário é a realização da tarefa em questão. A importância é expressa em valores inteiros positivos.

4. Estimativa: É uma ordem de grandeza, expressa em valores inteiros positivos, que expressa o esforço estimado para a execução de uma tarefa, ou seja, o quão difícil é a realização de uma tarefa.

5. Status: Indica o estado da tarefa atual conforme registrado no *backlog*. O estado de um tarefa é considerado binário. Assim, uma tarefa possuirá o status "S" quando já realizada e o status "N" caso contrário.

6. Benefício Relativo: Valor que indica o impacto ou efeito da realização de uma tarefa sobre a dificuldade de realização das demais tarefas do *backlog*. Para cada par ordenado de tarefas do *backlog* é estabelecido um valor percentual de redução da estimativa de outra tarefa.

A Figura 1 apresenta um exemplo do modelo de *backlog* estendido. Nesse exemplo, podem ser observadas 6 tarefas (A, B, C, $D1$, $D2$, E) bem como suas respectivas relações de dependência, importância, estimativa, status e benefício relativo (representado pelas colunas "A", "B", "C", "D1", "D2" e "E").

Tarefa	Dependência	Importância	Estimativa	Status	A	B	C	D1	D2	E
A	N/A	20	21	S	0	0,2	0	0	0	0
B	N/A	40	24	S	0,4	0	0	0	0	0
C	A,B	50	45	N	0	0	0	0	0	0
D1	C	10	56	N	0	0	0	0	0	0
D2	C	30	71	N	0	0	0	0	0	0
E	D1,D2	25	12	N	0	0	0	0,1	0,2	0

Figura 1: Exemplo de *Backlog* Estendido.

3.2 Modelo de Workflow Equivalente

Os valores atuais de um *backlog* estendido refletem o estado de um projeto e, com o andamento do projeto, as mudanças de estado do mesmo são registradas por meio de alterações neste *backlog*. A evolução de um projeto pode seguir diferentes percursos, com relação aos seus estados alcançáveis, os quais podem ser determinados em função da informação *a priori* disponível no *backlog*. O agrupamento desses diferentes percursos, em um único modelo, leva a um modelo de workflow que pode ser usado para representar o espaço de decisão do projeto. A Figura 2 ilustra o modelo de Workflow derivado do *backlog* da Figura 1 como um grafo de estados e transições. Cada estado do grafo é resultante dos estados individuais de cada tarefa. O estado inicial é o estado em que nenhuma tarefa foi realizada. Novos estados são inseridos de acordo com as regras de dependência presentes no *backlog*. As transições representam as mudanças de estado possíveis a partir de um estado origem.

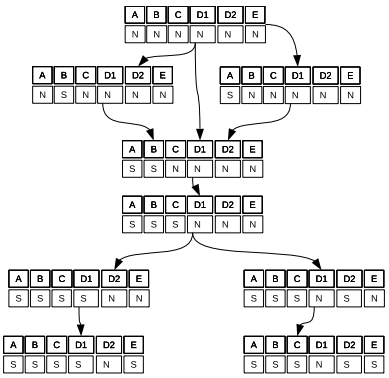

Figura 2: Modelo de Workflow derivado da Tabela da Figura 1.

3.3 Representação do modelo de Workflow como um Sistema de Transições

A Figura 2 apresenta uma representação de um modelo de Workflow como um sistema de transições. Genericamente,

um sistema de transições é uma estrutura contendo um conjunto de estados e um conjunto de transições entre estes estados. Esta estrutura pode conter ainda elementos adicionais, tais como estados iniciais, estados finais, relações causais para as transições e durações associadas às transições [10]. Em princípio, o número de estados de um sistema de transições pode ser infinito. Contudo, para a maioria das aplicações práticas este número é finito.

Um sistema de transições de um modelo de Workflow pode ser definido por um conjunto de tarefas, estados e transições. Uma tarefa é uma atividade atômica que quando realizada promove uma transição do sistema para um novo estado. Um estado representa um determinado momento do sistema modelado, representado por valores de variáveis de estado. A partir da representação de um modelo de workflow por meio de um sistema de transições, pode-se inferir o seu comportamento. Para atingir cada estado do sistema, um determinado percurso de tarefas necessita ser realizado. O estado atingido irá, portanto, refletir os resultados obtidos por este percurso. Sendo assim, um sistema de transições possibilita demonstrar não só a estrutura de um workflow, mas também o seu comportamento.

3.3.1 Matriz de Transições

Uma Matriz de Transições é uma forma simples e poderosa de se representar um sistema de transições. Considerando-se um modelo de tarefas com estado binário (realizada ou não realizada), uma matriz de transições é uma matriz $[2^n \times 2^n]$ tal que n corresponde ao número de tarefas do modelo. Cada linha e coluna da matriz de transições corresponde a um estado. Um estado é representado por uma cadeia binária de tamanho n, em que cada bit representa o estado de cada tarefa individualmente. Sendo assim, para um modelo composto pelo conjunto de tarefas $t_1, t_2, ..., t_n$, um estado e_x com $t_i = 0$ significa que a tarefa i não foi realizada no percurso até o estado e_x. Por sua vez, $t_i = 1$ indica que a tarefa i foi realizada no percurso até o estado e_x. A matriz será composta de 0's e 1's, sendo que posições que possuam o valor 0 corresponderão a ausência de uma transição do estado referente a linha para o estado referente a coluna. Uma posição contendo o valor 1, por sua vez, corresponderá a uma transição entre os estados em questão.

A Figura 3 apresenta um exemplo de matriz de transições para um modelo com duas tarefas. Este modelo permite duas transições: $00 \rightarrow 10$ e $10 \rightarrow 11$.

	00	01	10	11
00	0	0	1	0
01	0	0	0	0
10	0	0	0	1
11	0	0	0	0

Figura 3: Matriz de Transições.

Uma limitação para Sistemas de Transições é relativa ao crescimento exponencial do número de estados [10]. Porém, conforme discutido previamente em [2], na modelagem de workflow, quando esta é exercida de forma pragmática, o número de estados válidos e de transições entre esses estados tende a reduzir significativamente. Por meio desta representação, o espaço de decisão de modelagem é obtido de maneira precisa e soluções específicas podem ser obtidas por métodos de busca neste espaço. Tomando como exemplo a Figura 2, supondo que se deseja um *sprint* que execute as tarefas A, B e C, temos dois percursos diferentes que podem ser tomados. Como estes dois percursos possuem transições com custos diferentes, uma solução para o problema está em encontrar o menor caminho entre o estado inicial do sistema e o estado onde as três tarefas encontram-se realizadas.

3.3.2 Representação de valores de Importância, Dificuldade e Benefício Relativo

Para se obter uma representação completa do *backlog* estendido como um modelo de Workflow, é necessário agregar ao sistema de transição, um modelo de representação dos parâmetros de interesse do problema em questão. Neste caso, esses valores correspondem à Importância, Dificuldade e Benefício Relativo das tarefas do *backlog*. O modelo utilizado foi denominado de matriz de custo. Uma matriz de custo é uma matriz $[2^n \times 2^n]$ tal que n corresponde ao número de tarefas do modelo e as células $[i, j]$ indicam o custo relativo à realização da tarefa que provoca a transição entre os estados i e j. Uma matriz de custos será então capaz de representar o custo de realização das transições. Ou seja, ela indica o custo de execução de uma tarefa levando em conta o estado a partir do qual ela será executada. Um exemplo de matriz de custos pode ser observado na Figura 4.

	00	01	10	11
00	0	7.5	5.5	–
01	–	0	–	2.6
10	–	–	0	0.9
11	–	–	–	0

Figura 4: Matriz de Custos.

No exemplo da Figura 4, temos que o custo de realização da tarefa 1 dado que nenhuma tarefa se encontra realizada é 5,5. Por sua vez, o custo de realização da mesma tarefa levando em consideração que a tarefa 2 se encontra realizada é 2.6. O custo de uma transição pode ser determinado de diferentes formas, dependendo do problema em questão. Para a determinação de *sprints*, este custo, conforme será visto na seção 4, é função dos valores de Importância e Estimativa. Pode-se notar ainda que, por meio dessa representação, o custo de execução de uma tarefa não é fixo, podendo variar em função, por exemplo, do histórico de atividades que precedem a mesma. Assim, o modelo permite também incorporar no valor do custo o benefício que cada tarefa, previamente realizada, exerce na realização da tarefa responsável pela transição. O modelo para se determinar o custo em função desse benefício relativo é visto também na Seção 4.

Um aspecto adicional a ser considerado na representação do sistema de transições é o estado atual das tarefas conforme o *backlog*. Inicialmente um *backlog* pode possuir um conjunto de tarefas já concluídas. Dessa forma, um estado do sistema de transições, em que todas as tarefas já se encontram concluídas, vai ser marcado como inativo. Esta marcação permite a inclusão desse estado em percursos sem contabilizar os custos de alcançá-lo.

3.4 Determinação de Sprints

Para a determinação de *sprints*, foram consideradas duas abordagens utilizadas em projetos *Scrum* e que vêm sendo aplicadas no Projeto Sincap. A Abordagem de Estimativa baseia-se no esforço da equipe que será alocado para o *sprint*.

Com base nesse esforço a abordagem determina uma sequência de tarefas com estimativa total equivalente e que maximize o fator importância. A outra abordagem, denominada Abordagem de Subconjunto, permite à equipe do projeto indicar um subconjunto de tarefas que deve ser concluída no *sprint*. As sugestões de *sprint* nesse caso são sequências de tarefas de melhor relação entre os fatores importância e estimativa. Estas sequências incluem o subconjunto indicado, bem como as tarefas das quais este subconjunto depende para ser realizado.

3.4.1 Abordagem de Estimativa

A Abordagem de Estimativa busca encontrar a menor relação de estimativa e importância, tendo como fator de limitação um esforço previamente estabelecido representado como uma unidade denominada tempo limiar. Essa variável é fornecida pelo usuário no momento em que consulta o sistema para o planejamento do *sprint*. Para um dado tempo limiar, é estabelecido um limite inferior que deverá ser considerado. Esse limite, também fornecido pelo usuário, se trata de uma porcentagem a ser considerada abaixo do tempo limiar. A estratégia adotada busca encontrar todos os percursos cuja soma das estimativas de cada transição do fluxo não ultrapasse o tempo limiar dado como entrada. Isso é possível a partir da análise da Matriz de Transições e de uma Matriz de Estimativas correspondente ao *backlog*. Lembrando que no cálculo do custo dos percursos são considerados apenas os nós ativos do grafo. Ou seja, a transição associada a uma tarefa com status "S" no *backlog*, não terá seu custo computado no percurso.

À medida que os possíveis percursos vão sendo determinados, se a estimativa total do mesmo estiver entre o tempo limiar e a margem válida, ele é armazenado como percurso candidato. Caso não haja nenhum percurso que alcance o tempo limiar, uma nova estratégia é formulada. Nesta estratégia todos os percursos de estimativa total sub limiar encontrados são determinados. A maior estimativa total encontrada é agora definida como tempo limiar. Com base nesse novo tempo, os percursos que se encontram dentro da margem do novo tempo limiar são considerados como percursos candidatos.

Caso mais de um percurso candidato seja determinado, é aplicada uma nova avaliação, agora em relação a uma métrica resultante da combinação entre a importância e a estimativa de cada transição. Esse peso já se encontra definido na Matriz de Custos. O percurso que possuir menor peso total será recomendado como caminho ótimo.

3.4.2 Abordagem de Subconjunto

A Abordagem de Subconjunto possui como entrada um subconjunto de tarefas mandatório para um dado *sprint*. A estratégia adotada determina os possíveis subconjuntos de tarefas que podem ser realizados e que incluem o subconjunto pré-definido, determina os percursos mínimos para cada um desses subconjuntos e, por fim, elege dentre estes percursos a solução com melhor benefício em termos de importância e estimativa.

Nesta abordagem, temos a opção de considerar dois tipos de comportamento:

1. Se o projeto analisado possuir um *backlog* dinâmico, que sofra alterações constantes, pode-se então optar por uma política de eficiência de *sprints* que preze por uma visão de curto prazo. Neste caso, a estratégia adotada consiste em determinar como *sprints* candidatos as sequências mínimas que contenham o subconjunto de tarefas de entrada. Os *sprints* indicados serão então os percursos de custo mínimo correspondentes a uma destas sequências. Percebe-se então que, por meio desta estratégia, será obtida forma ótima de realizar as tarefas do subconjunto considerando apenas a situação presente do projeto no momento do planejamento do *sprint*, sem considerar a consequência futura da realização da sequência de tarefas com relação ao desempenho global do projeto.

2. Se uma organização possui um *backlog* com maior estabilidade, pode-se adotar uma política de eficiência a longo prazo. Neste caso, a estratégia determinará o percurso contendo os subconjuntos, mas incluirá nesse percurso os caminhos que conduzirão até o fim do projeto ou até um estado futuro que indique um marco importante para o projeto. Encontrados estes percursos completos, uma análise será feita para verificar quais possuem o subconjunto pré-definido e destes, qual possui a melhor eficiência - menor relação estimativa × importância. Dessa forma, a estratégia permite a seleção de um *sprint* que busca a eficiência no escopo global do projeto.

4. ASPECTOS DE CONSTRUÇÃO DA METODOLOGIA

Nessa seção, a metodologia proposta é discutida em detalhes. O *backlog* estendido apresentado na Figura 1 é utilizado como exemplo ilustrativo.

4.1 Obtenção do Sistema de Transições

Conforme discutido na Seção 3, a aplicação da metodologia requer, primeiramente, descrever o espaço de decisão do planejamento de *sprints* por meio de um modelo de Workflow, que, no nosso caso, será descrito por meio de um Sistema de Transições. O sistema de transições para um *backlog* é obtido por meio da análise e combinação das dependências definidas no *backlog*.

Para serem processadas, as dependências definidas no *backlog* são representadas na forma de uma matriz de restrições. Uma matriz de restrições é uma matriz $n \times n$, onde n corresponde ao número de tarefas considerado. Os valores assumidos pelas posições i, j da matriz são 0, indicando que a tarefa i não depende da tarefa j para ser realizada, 1 indicando que a tarefa i depende da tarefa j para ser realizada e 2 indicando que a tarefa i não participa do contexto representado pela matriz. Com base na definição, um exemplo de matriz de restrições que contenha 3 tarefas (T_1, T_2 e T_3), onde a tarefa T_1 depende da execução da tarefa T_2 e a tarefa T_3 não participa do fluxo, pode vir a ser observado na Figura 5.

$$\begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \end{array} \begin{array}{c} \begin{array}{ccc} T_1 & T_2 & T_3 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 2 & 2 & 2 \end{bmatrix} \end{array}$$

Figura 5: Matriz de Restrições.

As relações de dependência de um *backlog* criam restrições relativas a um determinado contexto de realização de

tarefas. Isto é, dependendo do contexto, uma dependência vai gerar uma restrição de execução, ao passo que podem existir outros contextos em que a dependência não vai gerar a restrição ou poderá gerar restrições distintas. Por exemplo, se uma tarefa (C) depende de duas tarefas alternativas (A ou B). Neste caso, C *depende* de A é uma restrição válida num contexto. Já em outro contexto a restrição válida é C *depende* de B. Pode-se dizer então que o contexto global de um *backlog* é formado pela combinação de diversos contextos. Em adição, observa-se que uma única matriz de restrições é muitas vezes incapaz de representar todo este contexto global. Isso se dá pelo fato de que, existindo somente uma matriz de restrições, uma tarefa ao ser excluída da matriz de restrições significaria que a mesma jamais deveria vir a aparecer no contexto em questão. Além disso, incluir as restrições de todos os contextos em uma única matriz irá criar inconsistências. A solução encontrada foi utilizar várias matrizes de restrições que buscam representar todos os possíveis contextos de realização das tarefas de um *backlog*.

O processo de descoberta das matrizes de restrições de um *backlog* ocorre por meio da geração de uma árvore de matrizes de restrições denominada Árvore de Restrições [2]. A árvore de restrições busca representar todas as tarefas, dependências e exclusividades existentes em um *backlog* estendido. A árvore de restrições da Figura 6 corresponde ao *backlog* estendido da Figura 1.

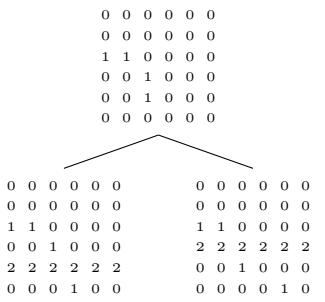

Figura 6: Árvore de Restrições.

4.1.1 Obtenção da Matriz de Transições baseado na Árvore de Restrições

Finalmente, a matriz de transições equivalente ao *backlog* pode ser obtida a partir da árvore de restrições. Este processo consiste em percorrer cada nó folha da árvore de restrições e gerar, para todos os estados, as transições permitidas pelas matrizes de restrições correspondentes a estes nós. A matriz de transições da Figura 7 corresponde à árvore de matrizes de restrições da Figura 6.

	010 000	100 000	110 000	111 000	111 010	111 011	111 100	111 101
000000	1	1	0	0	0	0	0	0
010000	0	0	1	0	0	0	0	0
100000	0	0	1	0	0	0	0	0
110000	0	0	0	1	0	0	0	0
111000	0	0	0	0	1	0	1	0
111010	0	0	0	0	0	1	0	0
111100	0	0	0	0	0	0	0	1

Figura 7: Matriz de Transições.

4.2 Obtenção da Matriz de Custos

A matriz de custos permite definir o custo da execução das tarefas levando em consideração o estado em que se encontra. Para a geração da matriz de custos são considerados o grau de importância, de estimativa e os benefícios relativos das tarefas presentes no *backlog* estendido.

Suponha uma tarefa t_x, que possua importância I_{t_x}, num domínio contendo n tarefas $(t_1..t_n)$, cuja estimativa da tarefa t_x seja Ef_{t_x} e o benefício variável de cada tarefa seja, respectivamente, $(B_{t_1}..B_{t_n})$. Sendo assim, o custo de execução da tarefa t_x será:

$$(1)\quad C_{t_x} = (Ef - (Ef \cdot \sum_{k=1}^{n}(B_{t_k} \cdot f(t_k)))) \div I_{t_x}$$

Na equação 1 $f(t_k)$ é 0 se a tarefa t_k já foi executada e 1 caso contrário. Por exemplo, suponha que na Figura 1 somente a tarefa A já tenha sido executada (100000) e se deseja executar a tarefa B (110000). Nesse caso, o custo para a transição 100000 → 110000 será dado por:

$C_{100000\times110000} = (24 - (24 \cdot (1 \cdot 0.4 + 0 \cdot 0.0 + 0 \cdot 0.0 + 0 \cdot 0.0 + 0 \cdot 0.0 + 0 \cdot 0.0))) \div 40 = 0.36$

Com a obtenção das matrizes de transições e de custos pode-se representar todos os elementos, de maneira sucinta, como um grafo de estados. A Figura 8 representa o grafo de estados do *backlog* estendido da Figura 1.

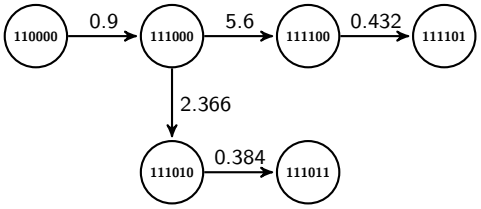

Figura 8: Grafo de Estados correspondente ao backlog da Figura 1.

4.3 Algoritmo de Planejamento de Sprints

Conforme discutidas na Seção 3, o planejamento de *sprints* poderá seguir as abordagens de Estimativa e de Subconjunto. Nesta seção, são discutidos os algoritmos que implementam estas abordagens tendo o espaço de decisão dos projetos representado como um sistema de transições.

4.3.1 Abordagem de Estimativa

A Abordagem de Estimativa requer encontrar, dentre todos os percursos com estimativa total dentro de uma margem, aquele com melhor relação entre importância e estimativa. O Algoritmo 1 apresenta uma solução para determinar este percurso, por meio de uma adaptação do algoritmo de busca em profundidade em grafos. No algoritmo, a busca por percursos válidos é iniciada a partir de um estado inicial do sistema de transições. O algoritmo percorre recursivamente, a partir deste estado, todos os percursos válidos que atendam a restrição definida para a estimativa. Para efeito de simplificação, no algoritmo foi considerado apenas a métrica de esforço como parâmetro de avaliação dos percursos. Na implementação do protótipo, entretanto, uma outra métrica, obtida pela combinação da estimativa e importância, também é considerada. Tendo os percursos que

atendem a restrição de estimativa, é avaliado o custo total desses percursos, que pode ser encontrado a partir da análise da Matriz de Custos. O percurso com menor custo total é retornado, sendo gerado como uma sequência de estados do sistema de transições. O *sprint* é então determinado pelas tarefas que determinam esta sequência.

Algoritmo 1: Planejamento por Estimativa

```
Funcao determinaPercursos(S : SistemaDeTransicao, ei :
Estado);
início
    para cada estado e ∈ S faça
        cor[e] = BRANCO;
        predecessor[e] = nil ;
    fim
    Visitas ← 1;
    menorPercurso ← ∅;
    percurso ← ∅;
    menorCusto ← ∞;
    visita(ei) ;
fim
Funcao visita(e : Estado);
início
    cor[e] ← CINZA;
    rota ← rota ∪ e;
    para todo (estado v adjacente a e faça
        se
        ((cor[v] = BRANCO) e (custo + dificuldade(u, v) <=
        estimativa)) então
            Visitas ← Visitas + 1;
            custo ← custo + dificuldade(u, v);
            predecessor[v] ← e;
            visita(v);
        fim
    fim
    se ((custo <= esforco) e (custo >=
    estimativa ∗ (1 − margemCusto))) então
        menorPercurso ← percurso;
    fim
    cor[u] ← BRANCO;
    percurso[visitas] ← 0;
    visitas ← visitas − 1 ;
    custo ← custo − dificuldade(predecessor[u], u);
fim
```

4.3.2 Abordagem de Subconjunto

Para implementação da Abordagem de Subconjunto, o algoritmo projetado determina os menores caminhos para os estados que, quando alcançados, atendem o subconjunto de entrada. Caso a estratégia seja de eficiência local, consideram-se os estados mínimos que contenham o subconjunto. Caso a estratégia seja de eficiência global, busca-se os estados adiante, que contenham o subconjunto adicionado das demais tarefas necessárias para se alcançar o ponto do projeto desejado. Finalmente elege-se dentre estes caminhos, aquele que representa o percurso de melhor relação entre estimativa e importância. O Algoritmo 2 ilustra esta implementação, de maneira simplificada, considerando apenas uma métrica de custo. O menor percurso até os estados válidos para o subconjunto pode ser obtido por meio de um algoritmo de caminhos mínimos. Na implementação atual, foi utilizado o algoritmo de Dijkstra [4], onde a fila de prioridade foi implementada numa lista linear.

A implementação do gerador de sistema de transições para *backlogs* e os algoritmos de determinação de *sprints* foram implementados usando o software de Computação Matemática Scilab e linguagem C. O protótipo implementado pode ser usado em reuniões de *sprint*, desde que a metodologia seja adotada e o *backlog* estendido tenha sido previamente

Algoritmo 2: Planejamento por Subconjunto.

```
Funcao determinaSubConjunto(S :
SistemaDeTransicao, e_inicial : estado, C : Tarefas);
início
    menorCusto ← 0;
    melhorSequencia ← ∅;
    Dijktra(S, e_inicial);
    para cada estado e ∈ S faça
        se (e[t_i] = Realizada para cada t_i ∈ C) então
            e_candidatos ← e_candidatos ∪ e;
            se (custo[e] < menorCusto) então
                menorCusto ← custo[e];
                melhorSequencia ← menorcaminho(e)
            fim
        fim
    fim
fim
```

definido. O protótipo recebe uma planilha correspondente ao *backlog* estendido em formato .XLS e produz como saídas as sequências de tarefas recomendadas.

5. ESTUDO EXPERIMENTAL

A metodologia proposta foi avaliada no contexto do Projeto Sincap. O Projeto Sincap é uma iniciativa para o desenvolvimento de um sistema de informação que permitirá o CNCDO-ES (Centro de Notificação, Captação e Distribuição de Órgãos do Espírito Santo) controlar algumas atividades do processo de doação e captação de córnea e, também, auxiliará na gestão da informação. O sistema de informação em questão será mediador das comunicações entre o CIHDOTT (Comissão Intra-Hospitalar de Doação de Órgão, Tecido e Transplantes), o CNCDO e o Banco de Olhos. A Figura 9 apresenta a relação entre o Sincap com as demais entidades envolvidas no processo. O LEDS (Laboratório de Extensão em Desenvolvimento de Sistemas), localizado no Ifes campus Serra, é o responsável pelo desenvolvimento do projeto e onde o estudo foi realizado.

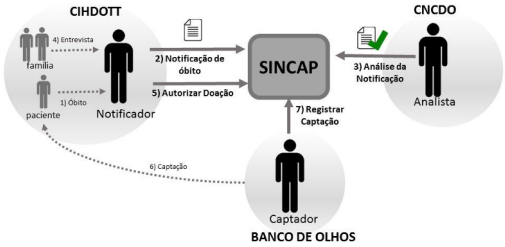

Figura 9: Arquitetura de Sistemas envolvidos no Projeto Sincap.

Para a discussão da metodologia, um subconjunto de tarefas do *backlog* do Projeto Sincap foi selecionado e é apresentado na Figura 10.

Um dos primeiros aspectos a ser avaliado na metodologia foi a eficiência dos métodos e estruturas utilizadas. Como pode ser observado, o *backlog* do Projeto Sincap possui duas tarefas exclusivas entre si, *L1 e L2*, resultando em uma árvore de restrições com dois nós folhas. O sistema de transições equivalente é representado por uma matriz $2^{17} \times 2^{17}$. Entretanto, o número de transições dessa matriz foi baixo, conforme previsto. Foram 8353 transições, o que levou a uma matriz de densidade $4,86 \cdot 10^{-7}$, ou seja, bastante

ID	Tarefa	Dependência	Importância	Estimativa	Status	A	B	C	D	E	F	G	H	I	J	K	L1	L2	M	N	O	P
A	Cadastro de óbito	N/A	100	20	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	Manipulação dos dados de óbito	A	97	48	S	0	0	0	0	0,05	0	0	0,05	0	0	0	0	0	0	0	0	0
C	Notificar interessados do óbito	B	91	20	N	0	0	0	0	0	0,15	0	0	0,15	0	0	0	0	0	0	0	0
D	Cadastrar entrevista	A	99	20	S	0,2	0	0	0	0,1	0	0,2	0	0	0	0	0	0	0	0	0	0
E	Manipular dados da entrevista	D	90	32	S	0	0,05	0	0,2	0	0	0	0,05	0	0	0	0	0	0	0	0	0
F	Notificar interessados da entrevista	E	86	20	N	0	0	0,15	0	0	0	0	0	0,05	0	0	0	0	0	0	0	0
G	Cadastrar captação	D	98	8	S	0,1	0	0	0,1	0	0	0	0,05	0	0	0	0	0	0	0	0	0
H	Manipulação dos dados de captação	G	85	24	S	0	0,05	0	0	0,05	0	0,15	0	0	0	0	0	0	0	0	0	0
I	Notificar interessados da captação	H	82	20	N	0	0	0,15	0	0	0,15	0	0	0	0	0	0	0	0	0	0	0
J	Arquivar processo de notificação	H	81	1	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	Listar processos arquivados	J	80	8	N	0	0	0	0	0	0	0	0	0	0,1	0	0	0	0	0	0	0
L1	Criar banco de dados A	N/A	69	56	N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L2	Criar banco de dados B	N/A	69	50	N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	Criar termo de doação	D	79	20	S	0,1	0,1	0	0,1	0,1	0	0,1	0,1	0	0	0	0	0	0	0	0,2	0
N	Gerar Relatórios	G	78	145	N	0,1	0,1	0	0,1	0,1	0	0,1	0,1	0	0	0	0	0	0	0,15	0	0
O	Configurar notificação	L1,L2	68	20	N	0	0	0	0	0	0	0	0	0	0	0	0,05	0,1	0	0	0	0
P	Fazer contato mobile	I	71	120	N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 10: Excerto do Backlog Estendido do Sincap.

baixa. Apesar da baixa densidade, o Scilab apresentou limitações para suportar matrizes de dimensões muito grandes, exigindo assim, o uso combinado de Scilab e linguagem C.

Com relação à funcionalidade da metodologia, foram verificados diversos cenários. Para exemplificar a Abordagem de Subconjunto, foi considerada uma situação em que a equipe *Scrum* deseja saber qual a melhor escolha de *sprint* onde foi observado que precisa-se, obrigatoriamente, executar as tarefas C, P e O do *backlog* estendido do Projeto Sincap. Tendo sido escolhida a estratégia de eficiência "local", apenas um percurso foi escolhido como a melhor opção. Sendo, o *sprint* retornado, formado pela sequência das seguintes tarefas: $C \rightarrow I \rightarrow L2 \rightarrow O \rightarrow P$, com custo total(estimativa $\times$ importância) de aproximadamente 3.107. No caso da Abordagem de Subconjunto com a abordagem de visão "global" e a mesma entrada, o *sprint* retornado seria: $K \rightarrow C \rightarrow F \rightarrow I \rightarrow L2 \rightarrow O \rightarrow N \rightarrow P$. Observa-se que o *sprint* desse exemplo é consideravelmente maior que o da abordagem anterior, isso ocorre porque essa abordagem considera os benefícios a longo prazo.

Como exemplo da Abordagem de Estimativa, foi considerada uma situação onde a equipe *Scrum* deseja encontrar uma melhor opção de *sprint* que tenha como soma total de estimativa alcançando no máximo uma medida de dificuldade de 142, denominada tempo limiar. Sendo que, possam ser consideradas, também, sequências que obtenham dificuldade em 10 por cento abaixo desse valor, ou seja, os *sprints* válidos podem ter soma total de estimativa entre 127,8 e 142. O objetivo é descobrir qual a sequência de tarefas menos custosa a se realizar nessa margem de estimativa. Dado o tempo limiar como entrada, assim como a margem a ser considerada, o sistema identifica 649 fluxos que possuem soma de estimativa dentro da faixa procurada, são então percursos candidatos a melhor *sprint*. Dentre esses percursos candidatos, é analisado o que possui menor custo total(estimativa $\times$ importância). Feita essa análise, o sistema retorna, como melhor *sprint* para a equipe, a sequência de tarefas: $N \rightarrow L2 \rightarrow K \rightarrow C$.

Os resultados obtidos foram bem recebidos pela equipe do projeto, sendo estes compatíveis com as decisões da equipe. O planejamento serviu em algumas situações para uma melhor visualização do *backlog* e também para pequenos ajustes no planejamento. De forma geral, a metodologia foi bem aceita, sendo que o ponto crítico considerado foi o levantamento dos benefícios relativos pela equipe.

6. CONCLUSÃO

Neste trabalho foi apresentada uma metodologia para apoio automatizado ao planejamento de *sprints* em projetos *Scrum*. O foco da metodologia foi o re-sequenciamento de atividades [8], tendo em conta as relações lógico temporais e de custo entre as mesmas. Na metodologia, duas estratégias de planejamento foram consideradas: planejamento por subconjunto e por estimativa. Para análise do espaço de decisão do planejamento, foi utilizado um modelo de workflow e uma representação deste modelo por meio de sistemas de transições. Embora tenha sido testada apenas em caráter experimental, os resultados da utilização da metodologia indicaram uma concordância entre as sugestões de *sprints* feitas pelo protótipo com o que seria planejado pela equipe. Um dos aspectos da metodologia considerado como crítico é o modelo de representação dos benefícios relativos. Não foi considerado por exemplo, um modelo de benefícios relativos não acumulativos. Ainda sobre o modelo de benefícios, observou-se que a obtenção de valores de benefício relativo é, por si só, um grande desafio para projetos *Scrum* poderem usufruir melhor da metodologia proposta. Entretanto, acreditamos que este desafio pode ser apoiado, por exemplo, por meio de Mineração de Processos.

7. REFERÊNCIAS

[1] D. Biasoli and L. M. Fontoura. Auxílio de redes de petri coloridas na implantação de projetos scrum. In *Anais do VII Simpósio Brasileiro de Sistemas de Informação*, pages 498–502. SBC, 2011.

[2] M. B. Costa and T. S. Silva. Uma abordagem para recomendação no apoioa modelagem de processos de negócio. In *Anais do XXIX Simpósio Brasileiro de Banco de Dados*, pages 177–186. SBC, 2014.

[3] J. A. Crowder and S. Friess. Productivity tools for the modern team. In *Agile Project Management: Managing for Success*, pages 43–48. Springer, 2015.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[5] M. Laanti, O. Salo, and P. Abrahamsson. Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information and Software Technology*, 53(3):276–290, 2011.

[6] A. S. C. Marçal, B. C. C. de Freitas, F. S. F. Soares, M. E. S. Furtado, T. M. Maciel, and A. D. Belchior. Blending Scrum Practices and CMMI Project Management Process Areas. *Innovations in Systems and Software Engineering*, 4(1):17–29, 2008.

[7] F. Maurer, B. Dellen, F. Bendeck, S. Goldmann, H. Holz, B. Kötting, and M. Schaaf. Merging project planning and web-enabled dynamic workflow technologies. *IEEE Internet Computing*, 4(3):65–74, 2000.

[8] H. Reijersa and S. L. Mansarb. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega*, 33:283–306, 2005.

[9] K. Schwaber. *Agile project management with Scrum*. Microsoft Press, 2004.

[10] W. Van der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer, 2011.

[11] J. Vlietland and H. van Vliet. Towards a governance framework for chains of scrum teams. *Information and Software Technology*, 57:52–65, 2015.