Auxílio de Redes de Petri Coloridas na Implantação de Projetos Scrum

Daniel Biasoli, Lisandra Manzoni Fontoura

Departamento de Eletrônica e Computação – Universidade Federal de Santa Maria (UFSM)
Prédio Anexo B / CT, 3º andar – Sala 378 – 97.105-900 – Santa Maria – RS – Brasil
daniel@biasoli.com, lisandramf@gmail.com

Abstract. *The successful implementation of Scrum software project depends on achieving defined goals in each sprint. As a Scrum team is different, if necessary, the process can be tailored for a particular team. An activity diagram is elaborated at the beginning of each sprint for each item to be developed from the product backlog, making the team more confident and committed to the work being developed. Based on this diagram, we propose the elaboration of a colored Petri net for each item, so that it is possible to simulate results, anticipate problems develop, manage risk and even reallocate resources.*

Resumo. *O sucesso da implantação de Projetos Scrum é potencializado pelo sucesso no alcance das metas definidas para cada Sprint. Como uma equipe Scrum é diferente da outra, se necessário, é possível adaptar o processo e torná-lo mais adequado para uma equipe em específico. A elaboração inicial de um diagrama de atividades, para cada item a ser desenvolvido do Product Backlog, torna o time mais confiante e compromissado com o trabalho a ser desenvolvido. Com base neste diagrama, propõe-se a elaboração de uma rede de petri colorida para cada item, a fim de que seja possível simular resultados, antecipar problemas de desenvolvimento, gerenciar riscos e, até mesmo, realocar recursos.*

1. Introdução

Trabalhar com pessoas, romper padrões tradicionais no senso comum e promover comunicação honesta são experiências complicadas em um ambiente de desenvolvimento de software. Cada equipe é única e capaz de provocar resultados inesperados, interessantes e, muitas vezes, surpreendentes. Bons resultados podem ser conferidos em equipes que adotam a metodologia Scrum. No entanto, o sucesso da metodologia passa pelo sucesso da definição de cada item a ser colocado em produção.

Um item de um *product backlog* pode ser qualquer coisa, desde um pedido escrito a respeito de algo que interesse para o produto até algo escrito em alto nível – de forma que tenha uma profundidade detalhada suficientemente para que o time de desenvolvimento seja capaz de realizar o pedido descrito e o proprietário do produto seja capaz de priorizar o item [SHWABER, 2009].

Portanto, é importante que o time Scrum saiba e entenda como lidar com itens do *product backlog*. O entendimento da definição de um item é a base para o sucesso das metas de *sprints* quando se adota a metodologia Scrum.

Baseando-se nas experiências dos autores deste artigo no ramo de desenvolvimento de software e nos trabalhos desenvolvidos com Scrum, sabe-se que times com pouca experiência são propensos ao fracasso. Visando a minimizar o índice de insucesso na utilização da metodologia, desenvolveu-se um estudo que encontrou subsídios em uma pesquisa que utiliza redes de Petri coloridas (RdPC) para auxiliar o comportamento de times de desenvolvimento de software em formação.

A fim de tornar o time confiante e compromissado com os itens e, consequentemente, com as tarefas a serem desenvolvidas, modela-se o processo de produção de cada item do *product backlog* e, posteriormente, encaixa-se cada modelo em uma rede de Petri colorida. Após, simulam-se as redes implementadas, por intermédio de um software (CPNTools), de tal forma que seja possível avaliar as variações desses processos. Estas variações quando benéficas para o projeto, poderão ser adotadas pelo time nos pontos de inspeção da metodologia, os quais serão explicados na seção 2.1 deste artigo.

Com o auxílio da formalização usando redes de Petri, as equipes podem simular resultados, antecipar impedimentos ou problemas de desenvolvimento, gerenciar riscos e, até mesmo, realocar recursos, tornando o trabalho de times em formação mais produtivo e eficiente.

Assim, o artigo foi dividido em seções: a seção 1 descreve a metodologia Scrum e seus pontos de inspeção. A seção 2 trata do assunto "Modelagem e Simulação" como embasamento teórico para a formalização do uso de redes de Petri no auxílio à implantação de projetos Scrum. A seção 3 faz referências a trabalhos desenvolvidos no meio acadêmico com redes de Petri e redes de Petri coloridas (RdPCs). A seção 4 propõe um estudo de caso aplicado em uma empresa de desenvolvimento de software e explanam-se os resultados. A seção 5 apresenta as considerações finais do estudo realizado.

Por não ser objetivo deste trabalho e por limitações de espaço, não serão explicados o funcionamento das redes de Petri e das redes de Petri Coloridas. Informações adicionais podem ser obtidas em PETERSON (1977), e MURATA (1989), CHIOLA (1997), GEHLOT *et al* (1985) e FRANCÊS (2003)

2. A Metodologia Scrum

Segundo SHWABER (2009), Scrum não se trata de um processo ou uma técnica para o desenvolvimento de produtos. Ao invés disso, é um *framework* no do qual pode-se empregar diversos processos e técnicas. Seu papel é fazer transparecer a eficácia relativa das suas práticas de desenvolvimento para que estas possam ser melhoradas, enquanto provê um *framework* a partir do qual produtos podem ser desenvolvidos.

Uma descrição detalhada a metodologia Scrum pode ser encontrada em: SHWABER (2009) e SHWABER (1995).

2.1. Inspeção de *Sprints*

Segundo SHWABER (2009), uma *sprint* é um evento regular com duração fixa (*time-box*). Vários elementos do Scrum possuem duração fixa, como a própria *sprint*, a reunião diária, a revisão da *sprint* e a retrospectiva da *sprint*, sendo estas, também, pontos chaves de inspeção e adaptação em Scrum.

A reunião diária situa o time quanto à direção da meta de uma *sprint*, além de alcançar adaptações que melhorem a produção no próximo dia de trabalho. As reuniões de revisão de *sprints* e o planejamento de *sprints* são empregados a fim de que o progresso em direção à meta da versão para entrega seja revisado, bem como sejam realizadas as devidas adaptações para a próxima *sprint*. A retrospectiva de uma *sprint* convém para revisar a *sprint* imediatamente anterior e definir que adequações deverão tornar o próximo *time-box* mais bem-sucedido.

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.

Em relação aos pontos de inspeção da metodologia Scrum, SHWABER (2009) afirma que os diversos aspectos do processo devem ser inspecionados com uma frequência suficiente para que variações inaceitáveis no processo possam ser detectadas. A frequência da inspeção deve levar em consideração que qualquer processo é modificado pelo próprio ato da inspeção. O problema acontece quando a frequência de inspeção necessária excede a tolerância do processo à inspeção. Outros fatores são a habilidade e a aplicação das pessoas em inspecionar os resultados do trabalho.

Aqui, identificou-se nas práticas de modelagem e simulação pautadas pelo embasamento formal das redes de Petri coloridas um possível mecanismo de auxílio à tomada de decisão nos pontos de inspeção da metodologia Scrum.

3. Modelagem x Simulação

De acordo com BRUSTOLINI *et al* (2007), as ferramentas de simulação permitem representar um sistema real, respeitando todas as regras e condições reais, prevendo o comportamento dos processos e comparando as mais diversas possibilidades sobre diferentes cenários propostos, tendo por conotação parâmetros técnicos e/ou econômicos.

A fim de que a simulação seja empregada na gestão de projetos, é imprescindível que a modelagem do sistema de produção a ser gerenciado seja realizada em uma etapa anterior. Assim, pela modelagem constroem-se modelos que representem o sistema real e que possibilitem experimentos, enquanto a simulação analisa e valida o modelo em questão [MELLO, 2007].

Segundo PRADO (1999), para se obterem resultados confiáveis, duas etapas devem ser seguidas: a construção do modelo da situação atual e a inclusão de alterações no modelo da situação atual para refletir a situação futura desejada.

De acordo com TAVIERA (1997), o objeto de estudo deste trabalho classifica-se como um modelo dinâmico, pois pode ser alterado no decorrer do seu tempo de vida. O

modelo deve seguir uma lógica para representar o processo, reconhecendo as variáveis de entrada e suas variações estatísticas ao longo do tempo. Assim, por meio da simulação, serão observados os resultados obtidos e, por conseguinte, comparados com os resultados reais do processo, conforme sugerem FOGARTY *et al* (1991).

4. Redes de Petri

Rede de Petri é uma técnica de descrição formal que faz uso de uma modelagem matemática e gráfica desenvolvida por Carl Adam Petri, com o intuito de representar sistemas concorrentes, controle, conflitos de sincronização e compartilhamento.

Mais informações a respeito de dados históricos de redes de Petri podem ser encontrados em MARRANGHELLO (2005), PETERSON (1977) e MURATA (1989).

Conforme AALST (1998) e YU *et al* (2009), as redes de Petri podem ser utilizadas como um modo de comunicação visual, auxiliando na modelagem de sistemas, de forma similar a fluxogramas, diagramas de blocos e de redes. Além disso, são utilizadas em redes de simulação de atividades dinâmicas e em simulação de sistemas.

Segundo PETERSON (1977), redes de Petri são um poderoso método para descrever e analisar o fluxo de informações e controle de sistemas, principalmente aqueles que apresentam comunicação síncrona, assíncrona, atividades concorrentes e paralelas.

De acordo com KHADKA *et al* (2007), redes de Petri suportam simulações de sistemas com simultaneidade em diferentes fases da sua execução.

Diversos trabalhos têm valorizado a estrutura formal das redes de Petri para representar e potencializar a modelagem e a análise de sistemas reais. PÁDUA *et al* (2003), OLIVEIRA (2008) e GEHLOT *et al* (1985) são boas referências para iniciar um estudo nesse sentido.

Atualmente existem diversas extensões de redes de Petri. Dentre elas, podem-se destacar as coloridas (RdPC), as temporizadas e as estocásticas.

Este artigo aborda somente redes de Petri coloridas por estas serem capazes de modelar sistemas suficientemente complexos, devido à quantidade de recursos de que dispõem, possibilitando uma redução no tamanho dos modelos.

4.1. Redes de Petri Coloridas

Para YU *et al* (2009), redes de Petri são ferramentas de modelagem gráfica e matemática aplicáveis a vários sistemas, mas tendem a aumentar rapidamente de tamanho quando a complexidade dos sistemas estudados é ampliada. Dessa forma, redes de Petri muito grandes tornam-se inconvenientes para uso, tornando-se impraticável a utilização deste recurso para modelagem de sistemas complexos. A fim de solucionar problemas como este, YU *et al* (2009) destaca que é possível reduzir o tamanho do modelo com a utilização de redes de Petri coloridas (RdPC).

O trabalho de GEHLOT *et al* (1985) propõe redes de Petri coloridas aplicadas à simulação de serviços WEB para que se possa avaliar a simultaneidade, bem como recursos de comunicação e suas restrições, além da qualidade de serviços que possam

utilizar tanto comunicação síncrona quanto assíncrona, podendo, ainda, descrever o impacto de aspectos não-funcionais como, por exemplo, a granularidade de serviços. Verifica-se, no trabalho em questão, um grande potencial no que diz respeito à formalização de modelagem e utilizando-se de redes de Petri coloridas, pois a solução proposta pode ser adaptada.

KRISTENSEN *et al* (2004) produziu estudos de RdPC e teve bons resultados quanto à formalização e ao planejamento de requisitos em projetos de desenvolvimento de software.

RdPC demonstram ser uma técnica capaz de analisar situações complexas de maneira formal, pois, na literatura pesquisada, foram muito eficientes em relação a concorrência, paralelismo, sincronização, não-determinismo e exclusão mútua.

Mais informações sobre redes de Petri coloridas podem ser encontradas em CHIOLA (1997), GEHLOT *et al* (1985) e FRANCÊS (2003).

5. Simulação e Inspeção de *Sprints* Scrum

O sucesso da metodologia Scrum passa pelo sucesso da definição um item 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.

Um time Scrum deve entender como lidar com itens do *product backlog*. O entendimento da definição de um item de um *product backlog* é a base para o sucesso de qualquer meta de uma *sprint* em Scrum.

Na próxima seção será mostrado, de maneira sintética, como é possível desenvolver o ciclo de vida de um item de um *product backlog* utilizando como base um diagrama de atividades simplificado.

5.1. Ciclo de Vida de Um Item do *Product Backlog*

Para modelagem de processos, a equipe piloto deste trabalho adotou o diagrama de atividades, em detrimento à BPMN (*Business Process Modeling Notation*), pois possui mais habilidade com as ferramentas de UML.

Segundo FREITAS FILHO (2008), os diagramas de atividades são um dos cinco diagramas disponíveis na UML para a modelagem de aspectos dinâmicos de sistemas.

Um diagrama de atividades pode ser a base fundamental para a descrição do processo de manipulação de todo o item de um *product backlog*.

Mais especificamente no trabalho apresentado neste artigo, o diagrama de atividades serve como base para a modelagem de processos utilizando redes de Petri coloridas.

O sucesso da criação dos diagramas de atividades por parte de um time Scrum potencializa sua confiança, facilita o mapeamento dos processos com a utilização de redes de Petri coloridas e dá suporte à tomada de decisões do Gerente de Projetos, na figura do *ScrumMaster*. Como uma equipe Scrum é diferente da outra, pode-se adaptar o processo e torná-lo adequado para um time em específico nas retrospectivas, se necessário.

A elaboração inicial do diagrama de atividades de um item de um *product backlog* por parte do time Scrum o torna confiante e compromissado com o item a ser desenvolvido.

Quando um time de Scrum tem o diagrama de atividades elaborado para cada item do seu *product backlog*, pode-se escolher uma ferramenta de apoio para monitorar, inspecionar e validar o processo de desenvolvimento.

5.2. O Estudo de Caso

Para que o trabalho possa ser viabilizado, adotou-se um estudo de caso elaborado em uma empresa de desenvolvimento de software.

Uma equipe que adota Scrum como *framework* para seus projetos, composta por um *ScrumMaster* (SCM), um *Product Owner* (PO), um *designer*, um analista de sistemas e dois programadores, foi incumbida de desenvolver um software para reservas de mesas em eventos, por associados de clubes sociais.

A versão WEB deste software deve ser iniciada nas próximas *sprints*. Deseja-se integrar o sistema de reservas com o site do clube. Cada *sprint* tem uma semana, ou seja, mais ou menos cinco dias úteis de duração.

Por limitações de espaço, exemplifica-se o processo apenas com um item do *product backlog*.

O próximo item a ser priorizado e que deve entrar na próxima *sprint* é o "Pagamento da Reserva de Mesas". É importante lembrar que este item só funcionará corretamente na companhia de outros itens que se supõe estarem finalizados.

O processo de desenvolvimento de umitem do *backlog* pode ser visualizado no diagrama de atividades mostrado na Figura 1 e elaborado pelo time:

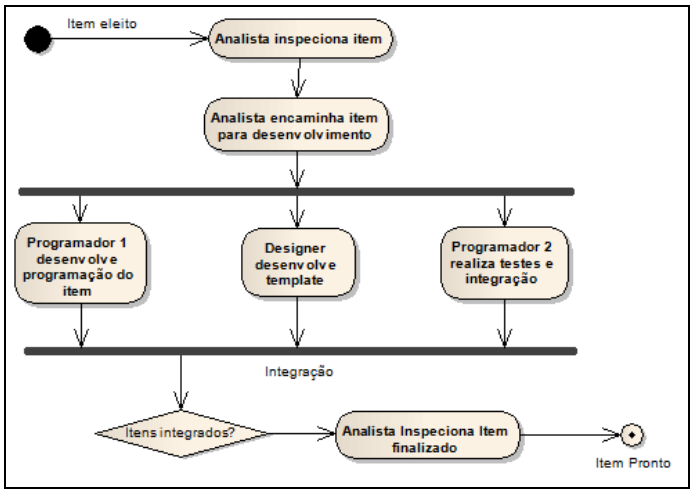

Figura 1. Diagrama de Atividades com o processo de desenvolvimento de um Item

Esse processo é de fácil compreensão. Devido a isso, o recurso será utilizado como base para a modelagem do processo em uma rede de Petri colorida para o item discutido.

5.3. O Desenvolvimento do Modelo

Para esta etapa do desenvolvimento, fez-se uso do da ferramenta CPN Tools versão 2.2, que serve para editar, simular e analisar redes de Petri coloridas. Esse software é o resultado do projeto de pesquisa CPN2000, da universidade de Aarhus, patrocinada pelo Danish National Centre for IT Research (CIT), George Mason University, Hewlett-Packard, Nokia e Microsoft. Ela pode ser obtida através do endereço HTTP://www.daimi.au.dk/CPnets/CPN2000/.

Na rede de Petri colorida, desenvolvida para o processo de desenvolvimento do item, representada pela Figura 2, os lugares que representam estados de desenvolvimento do item foram chamados de “Scrum”. Para se diferenciar um tipo de trabalho de outro, alguns lugares recebem valores referentes para cada tipo de colaborador (“Analista”, “Designer”, “Programador1”, “Programador2”). Esses tipos de colaboradores receberam marcações para que seja possível avaliar seu período de atuação durante o processo de desenvolvimento do item. Em redes de Petri coloridas, o tempo de marcação é exposto ao lado do símbolo @.

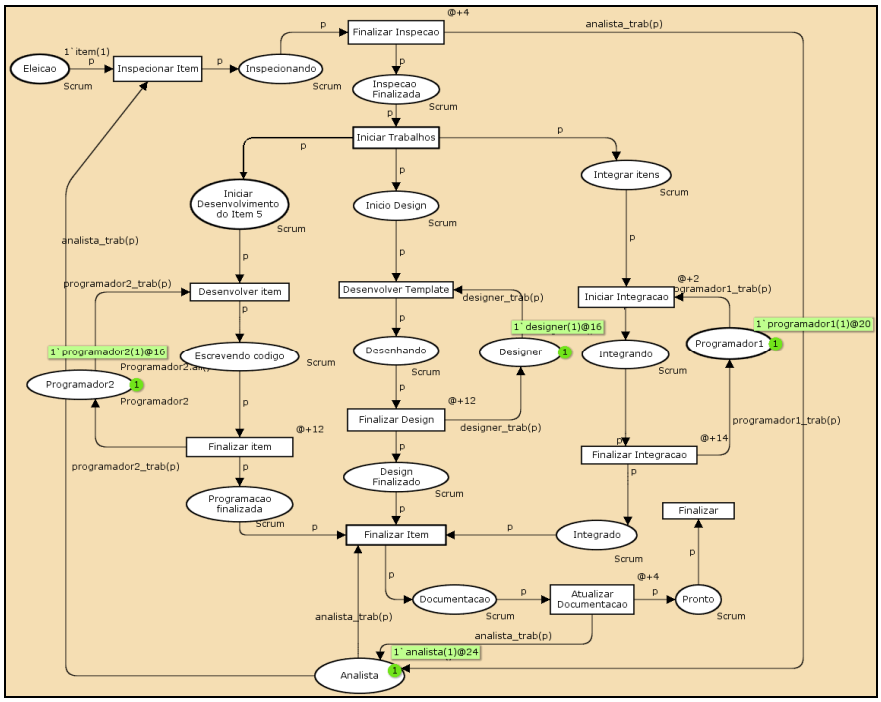

Figura 2. Rede de Petri Colorida Para o Item

No modelo, cada tarefa do processo é representada por transições. Assume-se que duas transições ou mais podem disparar simultaneamente, conforme se pode observar nas transições “Desenvolver item”, “Desenvolver *template*” e “Iniciar integração”.

Um contador foi desenvolvido para delinear o tempo de término de cada atividade de um colaborador por marcações temporais das transições do modelo. Todos os colaboradores envolvidos em atividades (transições) que representem custo temporal tiveram suas marcações alteradas para indicar o término de sua última atividade.

Nesse contexto, um dos lugares é representado pelo membro do time ocioso, esperando trabalho, e o outro representa o membro do time em atividade. Aqui duas transições foram necessárias. Uma faz com que o programador inicie sua atividade de integração e outra que ele volte a estar ocioso. Assim, modela-se o colaborador de tal forma que não fique eternamente desenvolvendo uma única tarefa. Nesse modelo, também é possível inserir mais de um programador para realizar uma atividade. No entanto, para o estudo de caso, modelou-se o processo com apenas um.

No modelo, uma transição é dita habilitada para ocorrer quando: (i) for possível associar valores às variáveis das expressões dos arcos conectados à transição, e (ii) sua guarda é avaliada como verdadeira. A função "programador1_trab" mapeia cada tarefa a um programador, removendo um *token* (ou marcação) do lugar "programador1". Quando a atividade é finalizada (Finalizar Integração), um *token* é colocado no lugar Programador1, novamente e o horário de término da atividade é marcado.

6. Resultados

As tarefas foram modeladas de acordo com a quantidade de horas a serem trabalhadas em cada uma delas e, nesse caso, sugeridas pelo "time".

O modelo inicia sua simulação sendo inspecionado pelo analista de sistemas, após o item ser eleito para desenvolvimento.

O tempo inicia sua contagem em zero (0). O analista de sistemas, que estava ocioso, inicia a inspeção do item. Estima-se, no modelo, um trabalho de quatro (4) horas.

No início da inspeção, o *token* (ou marcação) do analista é decrementado e, ao término do seu trabalho, é acrescentada uma marcação de tempo, bem como seu *token* é incrementado. Assim, o analista volta a estar disponível para um próximo trabalho, como sugere a Figura 3:

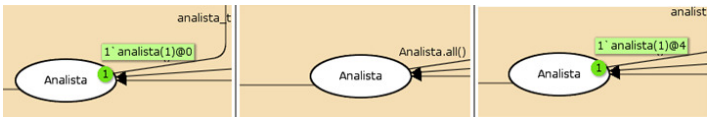

Figura 3. Rede de Petri Colorida Para o Item

De acordo com a simulação, o analista estará disponível durante vinte (20) horas, podendo ser encaixado em outras tarefas (como, por exemplo, iniciar a análise do próximo item do *selected product backlog*). Vinte (20) horas representam dois (2) dias e meio de trabalho em uma empresa que adote um expediente de trabalho de oito (8) horas diárias. A simulação deixa claro, também, que é possível fazer um cálculo de horas trabalhadas vezes o valor da hora trabalhada por colaborador, embora não tenha sido construído no modelo.

Continuando a simulação, iniciam-se os trabalhos de desenvolvimento. O próximo passo (Figura 4) mostra a transição do item distribuído em tarefas (transições: Desenvolver item, Desenvolver Template e Iniciar Integração).

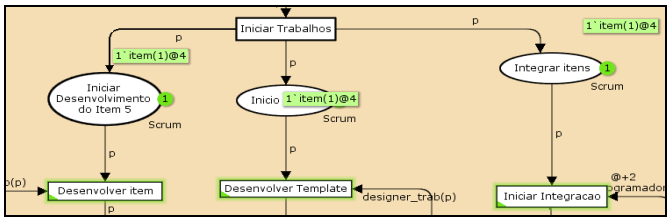

Figura 4. Rede de Petri Colorida Para o Item

Na Figura 5, percebem-se os estados de cada um dos colaboradores antes, durante e após o término de seus trabalhos.

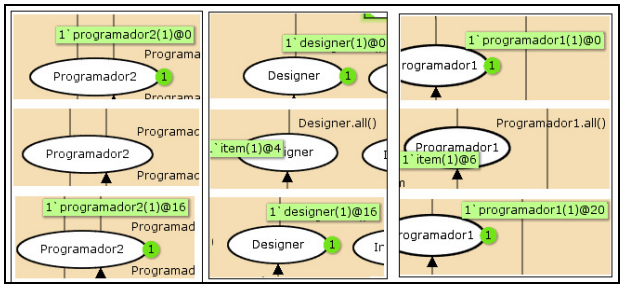

Figura 5. Rede de Petri Colorida Para o Item

Analisando a Figura 5, nota-se que as atividades realizadas em paralelo são encerradas quando o programador 1 termina sua atividade. O programador 2 e o *designer* encontram-se ociosos por mais quatro horas (num total de oito desde o início do projeto), podendo serem engajados no desenvolvimento de outras funcionalidades pendentes ou, no caso do desenvolvimento Scrum, no suporte a melhorias de itens programados em *sprints* anteriores que necessitem manutenção por parte do time.

Observa-se, também, que se algum profissional falhar ou não completar todo o processo que lhe é conferido, o trabalho do time não é concluído, uma vez que as atividades são executadas paralelamente. Um trabalho depende do outro, principalmente porque temos aqui processos interdependentes. Aqui se pode observar a importância da simulação do processo e o entendimento do modelo por parte dos profissionais envolvidos nesse projeto: os membros de um time Scrum estão ligados para se alcançar e compartilhar um objetivo comum, criando vínculos diante dos seus desafios cotidianos de trabalho. Todos os envolvidos devem remar na mesma direção para poderem avançar em seus projetos e possibilidades, compartilhando tanto seus acertos como erros e tendo em mente que uma boa relação estimula a cooperação.

Finalmente, pouco antes do término da simulação, o Analista entra em ação novamente. Dessa vez há uma inspeção do item já desenvolvido, que poderá durar quatro (4) horas, segundo estimativa do time. Em geral, essa atividade envolve toda a equipe.

Ao terminar a execução da simulação do modelo no programa CPN Tools, a estimativa é de que a equipe utilize vinte e quatro horas para finalizar o item, oito horas a mais do que fora previsto em uma estimativa analisada friamente pelo time, em um primeiro momento, como é tradicionalmente feito em projetos que utilizam metodologia Scrum.

A simulação do modelo não diminuiu o tempo de execução de um item de um *product backlog*. No entanto, traz à tona, com alguma eficiência, espaços significativos no que diz respeito ao intervalo temporal entre tarefas a serem desenvolvidas. Nesse ponto é importante que o time antecipe as correções necessárias tão logo seja possível.

Com base na simulação, pode-se propor a realocação de colaboradores ociosos, a fim de que possam executar outras tarefas que não sejam exatamente as desempenhadas na implementação do item.

A simulação torna o time mais confiante quanto à tomada de decisões por parte do *ScrumMaster*. Consegue-se, após a realização da simulação do desenvolvimento do item, visualizar com mais precisão a execução das tarefas e precisar melhor os seus tempos de desenvolvimento.

Também é possível utilizar a flexibilidade que a ferramenta CPN Tools proporciona, adicionando mais colaboradores para o desenvolvimento de cada tarefa. Ainda pode-se criar uma espécie de trava no desenvolvimento do item, caso algum impedimento ocorra, como faltas ao trabalho, por exemplo. A linguagem CPN-ML, que trabalha as redes de Petri coloridas, proporciona, ainda, a programação de fenômenos aleatórios, deixando a simulação do processo ainda mais real.

É possível fazer com que a equipe veja e entenda a divisão dos itens em mais ou menos tarefas por meio da simulação. No caso do item simulado, o Programador 1 fica ocioso, esperando o término das outras atividades de seus colegas para integrar seus trabalhos. Por isso, estimou-se um prazo maior para sua conclusão. Além disso, considerando a atividade de análise do analista de sistemas, teríamos um prazo de desenvolvimento do item de pelo menos 16 horas acrescendo-se quatro horas de inspeção.

7. Conclusões e Trabalhos Futuros

Lidar com itens de um *product backlog* é um dos passos mais importantes na transição de equipes de desenvolvimento tradicionais para o Scrum. Compreender o fluxo de trabalho de um item de um *product backlog*, na opinião dos autores deste trabalho, é o melhor ponto de partida para a discussão do ciclo de vida de um item.

A criação de um diagrama de atividades ajuda substancialmente a entender a posição de um item de um *product backlog* em relação ao *framework* Scrum, auxiliando o sucesso de uma *sprint*. Com os processos a serem executados em vigor, definidos e aceitos pelo time Scrum, os detalhes específicos de um item são de menor importância. Um ciclo de vida definido ajuda a equipe a compreender que, antes de começar a realizar um item de um *product backlog*, este deve ter sido apreciado por todos na equipe Scrum, para que todos tenham a oportunidade de esclarecer a intenção deste item.

No contexto deste trabalho, as redes de Petri coloridas serviram como uma base de sustentação para uma análise paralela do time Scrum sem muita experiência na metodologia. Após a observação de um fluxograma de trabalho a ser desenvolvido pelo time, em torno de um item do *product backlog*, foi aplicada uma modelagem utilizando-se redes de Petri coloridas, a qual retorna valores temporais simulados e mais condizentes com a realidade.

Elaborar um modelo para cada item do *product backlog* pode tornar-se um processo desgastante, como se pode observar na pesquisa realizada. No entanto, quando o time possui dados históricos suficientemente bons para que possam ser equiparados a itens em desenvolvimento, não se faz necessário que todo o processo seja modelado novamente. Com o tempo, a equipe adquire experiência e o processo de simulação torna-se menos frequente.

A modelagem utilizando redes de Petri coloridas comprovou que essas redes podem ser bastante eficientes para representar um sistema de *Workflow*, permitindo que erros sejam detectados ainda em fase de projeto, evitando que falhas ocorram durante a execução.

Diante do exposto, obtiveram-se representações das atividades e suas sincronizações, dos recursos determinados para as atividades, das regras do processo e principalmente do fluxo de trabalho. Dessa maneira, é possível trocar informações padronizadas entre os participantes do projeto, além de analisar o processo antes de sua execução, tornando o processo de desenvolvimento mais compreensível e eficiente.

Como sugestão para trabalhos futuros, entende-se que se possa explorar o uso de outros tipos de ferramentas, como o ARENA, software da empresa Paragon. Outra possibilidade seria trabalhar com redes de Petri no auxílio a gerência de riscos.

8. Referências

AALST, V. D. (1998) “The Application of Petri Nets to Workflow Management”. The Journal of Circuits, Systems and Computers, 8(1):21–66.

BOOCH, G., RUMBAUGH, James, JACOBSON, I. (2000) “UML, Guia do Usuário”, 14. ed. Rio de Janeiro: Elsevier.

BRUSTOLINI, J. R; SILVA, E. M (2007) “Simulação do Processo de Congelamento em uma Unidade Produtora de Aves”. XXVII Encontro Nacional de Engenharia de Produção: Anais.

CHIOLA, G. & *et al*. (1997) “A Symbolic Reachability Graph For Coloured Petri Nets”. Dipartimento di Informatica, Università di Torino, Italy. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.6066.

FOGARTY, D. W.; BLACKSTONE Jr.; J. H.; HOFFMANN, T. R. (1991) “Production & Inventory Management”. 2 ed. Cincinnati: South-Western Publishing Co.

FRANCÊS, Carlos Renato Lisboa. (2003) “Introdução às Redes de Petri”. Laboratório de Computação Aplicada, Universidade Federal do Pará.

GEHLOT, V.; EDUPUGANTI, K. (1985) “Use of Colored Petri Nets to Model, Analyze, and Evaluate Service Composition and Orchestration”. Proceedings of the 42nd Hawaii International Conference on System Sciences, 2009 - Center of Excellence in Enterprise Technology, Department of Computing Sciences, Villanova University, Villanova, PA, USA.

KHADKA, B., MIKOLAJCZAK, B. (2007) “Incorporating Object-Orientedness in Transformations from Live Sequence Charts to Colored Petri Nets”. Fifth International Conference on Information Technology: New Generations, Computer

and Information Science Department - University of Massachusetts Dartmouth, Dartmouth, MA.

KRISTENSEN, L. M.; JORGENSEN, J. B.; JENSEN, K. (2004) "Application of Coloured Petri Nets in System Development". Department of Computer Science, University of Aarhus, Denmark, ACPN 2003, LNCS 3098, pp. 626–685, 2004. Springer-Verlag Berlin Heidelberg.

MARRANGHELLO, N. (2005) "Redes de petri: Conceitos e aplicações", DCCE/IBILCE/UNESP. Disponível em: http://www.dcce.ibilce.unesp.br/~norian/cursos/mds/ApostilaRdP-CA.pdf.

MELLO, B. A. (2007 "Modelagem e Simulação de Sistemas". Universidade Regional Integrada do Alto Uruguai e das Missões.

MURATA, T. (1989) "Petri nets: Properties, analysis and applications". Proceedings of the IEEE. Disponível em: http://ieeexplore.ieee.org.

OLIVEIRA, C. A. de L. (2008) "Uma Abordagem para Melhoria de Workflow Baseada em Redes de Petri Estocásticas Generalizadas". Dissertação de Mestrado - Universidade de Pernambuco, Programa de Pós-Graduação em Engenharia da Computação.

PÁDUA, S. I. D.; SILVA, A. R. Y.; INAMASU, R. Y.; PORTO, A. J. V. (2003) "Aplicações e Potencial das Redes de Petri na Engenharia de Produção". In: Simpósio de Engenharia de Produção, 10, Bauru, UNESP, 2003. Anais. Avaliable from WWW: URL: http://www.bauru.unesp.br/acontece/simpep.html, maio.

PÁDUA, S. I. D.; INAMASU, R. Y. (2008) "Mapeamento do Modelo de Processos de Negócio do EKD em Redes de Petri". Produção, v. 18, n. 2, p. 260-274.

PETERSON, J. L.(1977) "Petri nets". Department of Computer Sciences, The University of Texas.

PRADO, D. (1999). "Teoria das Filas e da Simulação". DG: Belo Horizonte/MG.

SCHWABER, K. (1995) "The scrum development process". In: OOPSLA'95 Workshop on Business Object Design and Implementation, ACM Press, Austin, Texas, USA.

SCHWABER, K. (2009) "Scrum Guide. ScrumAllienace". Disponível em www.scrumalliance.org/resource_download/598, outubro.

TAVIERA, R. (1997) "Uma metodologia para aperfeiçoamento da mudança para um sistema de produção just-in-time em uma indústria metalúrgica, usando simulação discreta e técnicas de projeto de experimentos de Taguchi". Dissertação (Mestrado em Engenharia de produção e Sistemas). PPGEPS - UFSC, Florianópolis.

YU, Y.; LI, T.; LIU, K.; DAI, F.; ZHAO, N. (2009) "OR-transition Colored Petri Net and its Application in Modeling Software System". Second International Workshop on Knowledge Discovery and Data Mining - School of Software, Yunnan University, Kunming, China.