UNIVERSIDADE FEDERAL DA GRANDE DOURADOS- UFGD

FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RAFAEL OLIVEIRA DE BIASSIO

UMA BUSCA SISTEMÁTICA POR PROCESSOS E METODOLOGIAS COM FOCO EM QUALIDADE DE SOFTWARE

Dourados/MS

2019

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DA GRANDE DOURADOS

FACET – Faculdade de Ciências Exatas e Tecnologia

RAFAEL OLIVEIRA DE BIASSIO

UMA BUSCA SISTEMÁTICA POR PROCESSOS E METODOLOGIAS COM FOCO EM QUALIDADE DE SOFTWARE

Orientadora: Evanise Araujo Caldas Ruiz
Área de Concentração: Metodologia e Técnicas de Computação

Dourados/MS

2019

Uma busca sistemática por processos e metodologias com foco em qualidade de software

Rafael Oliveira de Biassio, Evanise Araujo Caldas Ruiz

[1] Faculdade de Ciências Exatas e Tecnologia – FACET
Universidade Federal da Grande Dourados – UFGD

rafaelobiassio@gmail.com, evanisecaldas@ufgd.edu.br

Abstract. *This article presents an exploratory study about the development process agile Scrum and the MPS-BR improvement model. But also a systematic mapping of Scrum and MPS-BR, emphasizing its effective compatibility. Based on the study results, it can be concluded that Scrum is MPS-BR level G compliant, and their joint application is encouraged to achieve software maturity in small businesses.*

Keywords: Scrum; MPS-BR; agile software; software process improvement; software maturity;

Resumo. *Este artigo apresenta um estudo exploratório sobre o processo de desenvolvimento ágil Scrum e o modelo de melhoria MPS-BR. Mas também, um mapeamento sistemático sobre o Scrum juntamente com o MPS-BR, enfatizando sua efetiva compatibilidade. Com base nos resultados do estudo, é possível concluir que o Scrum é compatível com o nível G do MPS-BR, e sua aplicação em conjunto é incentivada para alcançar a maturidade de software em pequenas empresas.*

Palavras-chave: Scrum; MPS-BR; métodos ágeis; melhoria de software; maturidade de software;

1. Introdução

No início da engenharia de software o seu processo era sistemático e disciplinado, de forma que a visão global do sistema era concebida no início do projeto, na fase de levantamento de requisitos. Essa abordagem provou-se eficaz em grandes sistemas de software, com poucas mudanças durante o projeto [Pressman 2011]. Porém, quando trata-se de sistemas de pequeno e médio porte a resposta era diferente, esse modelo convencional aumentava o tempo total de construção do projeto, dificultava as mudanças e assim, refletia no valor final investido pelo cliente [Sommerville 2011].

Todos esses atrasos e erros de valores geraram insatisfações em torno da engenharia de software, com isso, no começo na década de 1990 diversos profissionais da área trabalhavam em uma nova abordagem, que se popularizou no ano de 2001 pelo "Manifesto do Desenvolvimento Ágil de Software": os métodos ágeis [Beck et al. 2001]. Essa metodologia contém como princípios: foco nos integrantes do projeto, simplicidade de implementação, documentação estritamente necessária e tratar as mudanças com pouca resistência.

Existem várias vertentes dos métodos ágeis, um dos principais é o Scrum, que foi criado por *Jeff Shuterland* no início dos anos 1990 contém princípios consistentes com o manifesto ágil, usados para orientar as atividades de desenvolvimento dentro de um processo [Pressman 2011]. Ademais, o Scrum possui práticas para simplificar o processo de desenvolvimento de software, seguindo seus cinco valores: foco, coragem, franqueza, compromisso e respeito. Dessa forma, maior visibilidade do projeto, assim como resultados que agregam valor final ao produto [Alliance 2012].

Contudo, estudos afirmam que cada contexto de desenvolvimento deve ser tratado individualmente, com adaptações do método para a equipe alcançar a maturidade de software [Soares et al. 2007]. Apesar das boas práticas defendidas pelo Scrum, apenas a metodologia não satisfaz a procura por maturidade, já que contém princípios e valores práticos e simples, ainda que com foco na qualidade final do produto - bem como satisfação do cliente - não há métodos para medir essa maturidade [Salgado et al. 2010]. Sendo assim, para metrificar essa maturidade, foram criados modelos de processo de melhoria de software, um dos principais modelos é o MPS-BR (Melhoria do Processo de Software Brasileiro) criado pela Softex, que além de aumentar a garantia de software de qualidade é compatível com práticas ágeis, e tem resultados satisfatórios juntos ao Scrum [Oliveira et al. 2007].

Com isso, o objetivo desse trabalho é desenvolver uma pesquisa investigativa e sistemática quanto ao uso do método ágil Scrum no desenvolvimento de software, priorizando o modelo de qualidade MPS-BR, e como esse programa modifica o projeto de software nas empresas. Mas também, identificar as adaptações no método Scrum para utilização do MPS-BR em conjunto no processo de desenvolvimento de software.

Desse modo, o trabalho é dividido nessas etapas: a seção 2 contém o desenvolvimento sobre Métodos Ágeis e Scrum. A seção 3 aborda o modelo de melhoria MPS-BR e seus níveis. A 4ª seção apresenta os materiais e métodos do artigo. Seguido da 5ª seção que contém os resultados e discussão. E assim, a 6ª seção como conclusão do trabalho, seguida de referências.

2. Métodos Ágeis

Com a globalização da informação, o uso do software nas tarefas das mais diversas empresas passou de um caráter mais diferencial para quase total estratégico e exigido [Sutherland 2016], esse ritmo era ditado por mudanças rápidas - principalmente no ambiente web - com pouca possibilidade de prever os requisitos no planejamento inicial de construção do software. As metodologias tradicionais não conseguiam acompanhar esse contexto acelerado [Pressman 2011], pois quando utilizado, o nível de incerteza da área que o produto era focado afetava diretamente a chance de atrasos e erros, assim como, aumento nos custos de implementação (Figura 1).

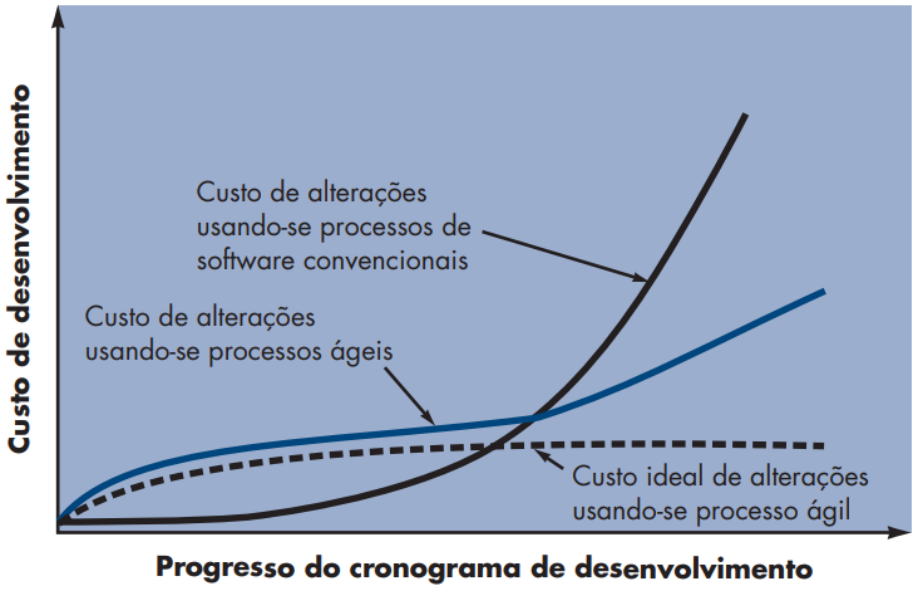

Figura 1. Custo mudança x tempo de desenvolvimento [Pressman 2011]

A partir dessas necessidades de melhorias, nasceu a filosofia das abordagens ágeis, com seus princípios formulados por desenvolvedores e profissionais da área pelo manifesto ágil de software [Beck et al. 2001], o qual afirma: "Estamos descobrindo melhores maneiras de desenvolver softwares; fazendo-o e ajudando outros a fazê-lo, valorizando mais: (I) Indivíduos e interações do que processos e ferramentas; (II) Software em funcionamento do que documentação abrangente; (III) Colaboração do cliente do que negociação de contrato; e (IV) Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda."

Diferente das abordagens tradicionais, a abordagem ágil não tenta prever todas as dependências do produto a ser criado, focando nas mudanças necessárias também na fase de desenvolvimento. Esse foco é reforçado no princípio de resposta rápida à mudanças no produto, com entregas constantes de partes do software - assim, apresenta diferenciais estratégicos ao cliente [Pressman 2011].

Assim, segundo [Sommerville 2011] as aplicações modernas são mais consistentes seguindo as abordagens ágeis, com maior satisfação do cliente, já que o software final está atualizado para realizar a tarefa para qual foi planejado.

2.1. Scrum

A partir desses princípios descritos pelo manifesto, *Jeff Sutherland* que é um dos autores envolvidos no Manifesto Ágil de Software [Beck et al. 2001], profissional da área de desenvolvimento e especializado em gerenciamentos de equipes, criou o processo de gerenciamento de software Scrum - também descrito como *framework*. Segundo Sutherland [2016], o nome Scrum foi inspirado no esporte *Rugby*, que contém uma reunião rápida depois de uma falta, antes da próxima jogada - que tem o mesmo nome. Como comparativo, no *Rugby* cada jogador pode ter uma função (ou posição) diferente no time, mas as equipes devem agir em conjunto para chegar ao objetivo, de forma unificada.

Portanto, uma equipe de Scrum deve ter seus membros autogerenciados, com número de integrantes pequeno (de 5 a 9), trabalhando em união para alcançar objetivos

claros, exigindo para isso flexibilidade para adaptações nas mudanças durante o projeto, mas também revisões frequentes [Pressman 2011].

O Scrum é focado em gerenciamento de equipes em torno de um desenvolvimento interativo e incremental, de produtos complexos, com requisitos de difícil prevenção ao longo do projeto - já que a eliminação de incerteza em um ambiente altamente dinâmico é considerada improvável [Sutherland 2016]. Então, para garantir resultados sobre esse contexto, o Scrum segue cinco valores principais: foco, coragem, franqueza, compromisso e respeito [Alliance 2012].

O Scrum é composto por práticas e artefatos que podem ser definidos como (Figura 2) [Sabbagh 2014]:

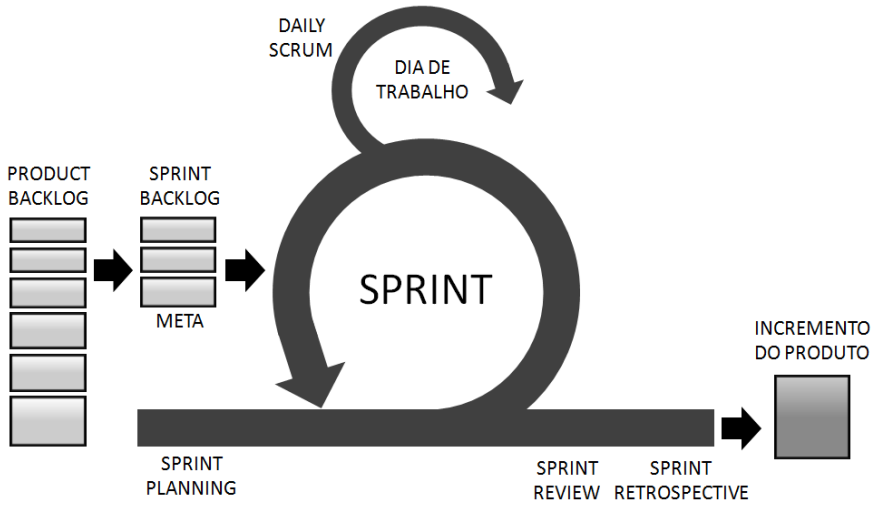

Figura 2. O ciclo de vida do Scrum

Documento de visão geral - Para iniciar o desenvolvimento, é realizada uma reunião com toda a equipe, para formar a visão geral do produto, pouco detalhada, apenas para servir de guia inicial. Importante ressaltar que tanto essa como as outras reuniões buscam incluir, de forma direta ou indiretamente, os clientes envolvidos, os chamados *Stakeholders*.

Product Backlog - é uma lista de tarefas criada pela equipe a partir do documento de visão geral, com intuito de definir as reais demandas do produto.

Sprint Planning Meeting - uma reunião com todo o time, destacando o *Product Owner*, que é o representante dos diretamente interessados ao projeto, para planejar esse próximo ciclo de desenvolvimento. A partir dos requisitos do *Product Backlog*, é discutido e pontuado cada tarefa a ser realizada com *Story Points* [Carvalho and Mello 2012], que serve para a equipe definir o *Sprint Backlog*.

Story Points - pontuação atribuída para cada tarefa a ser realizada.

Sprint Backlog - lista de demandas para desenvolver no *Sprint* atual, equivalente ao *Product Backlog* para uma *Sprint*.

Sprint - é o ciclo de trabalho do Scrum, que é repetido - com objetivos de entregas

logicamente diferentes - até o produto final entregue ao cliente. Uma *Sprint* é dividida em uma medida de tempo mínima viável (por volta 2 a 4 semanas) para entrega de um incremento estratégico ao produto - como exemplo uma funcionalidade de software - ou, pelo menos, em potencial.

Daily Scrum Meeting - é incentivada uma reunião diária, com intuito de esclarecer o realizado, o foco em desenvolvimento e se há dificuldades/bloqueios que atrapalham a execução de cada membro da equipe. Essa reunião tem objetivo de ser simples e direta (não durar mais de 15 minutos), apenas para atualizar a equipe com a troca de informações entre os desenvolvedores. Mais um ponto indicando a constante comunicação defendida de pelo *framework* [Rosa et al. 2015].

Sprint Review Meeting - reunião feita após o fim do *Sprint* para o time apresentar os resultados diante do *Product Owner* alcançados no mesmo.

Sprint Retrospective Meeting - uma segunda reunião após o fim da *Sprint*, dirigida pelo *Scrum Master* - que é um profissional do time eleito com intuito de gerenciar a equipe de desenvolvimento no uso correto do Scrum - para discutir os resultados e revisar melhorias estratégicas em geral, encaminhando o projeto para a próxima *Sprint*. Essa nova *Sprint* segue o mesmo ciclo descrito anteriormente, até o resultado final.

Incremento do produto - é o trabalho desenvolvido durante a *Sprint*, uma parte incremental para adicionar ao produto de software.

3. MPS - BR

Apesar da utilização do *framework* Scrum ser amplamente descrito como eficaz ao processo de desenvolvimento de software [Pressman 2011], como toda metodologia há casos que o uso isolado não é o suficiente para entrega de um produto com qualidade [Santos 2011]. Como exemplo, empresas com os membros do time distribuídos geograficamente, que dificulta duas das principais práticas do Scrum: o time como unidade e sua comunicação [Soares et al. 2007]. Para maior garantia de qualidade a partir dessas necessidades, o Scrum pode ser utilizado em conjunto com outras ferramentas [Szimanski et al. 2009].

Entre essas possibilidades, há os modelos de maturidade de software, criados para auxiliar e aprimorar os processos de construção de software das empresas, que é descrito como maturidade de processo de software [Pressman 2011]: esses possuem níveis de maturidade - exigindo assim regras para sua implementação - com maiores obrigações conforme mais alto o nível. Portanto, os modelos servem como regulamentação: quanto maior o nivel implementado pela empresa, mais evoluído e padronizado é o seu processo de software, e dessa forma, serve como métrica tanto às empresas de software quanto aos potenciais clientes [MPS-BR 2012].

Um ponto a destacar é que em um dos princípios ágeis - inclusive o Scrum - reprovam a prática de documentação abrangente do software, focando no seu funcionamento [Beck et al. 2001]. Por sua vez, os modelos de melhoria acrescentam documentação ao projeto, porém, a abordagem ágil não exclui a documentação por inteiro, o exagero dela que é evitado [Lima and Vendramel 2016].

Dentre esses modelos de maturidade, existe o MPS - BR (Melhoria do Processo de Software Brasileiro) criado pela Softex que é um modelo de maturidade que nasceu base-

ado nos modelos já existentes - em especial o internacional *CMMI (Capability Maturity Model Integration)* [Weber et al. 2004].

O MPS-BR é composto por 7 níveis, onde sua estrutura é dividida em 3 partes: o Modelo de Referência (MR-MPS), o Modelo de Avaliação (MA-MPS) e o Modelo de Negócio (MN-MPS). Essa divisão possibilita a inclusão de empresas menores, principalmente pelo nível G, tornando a adoção do MPS mais gradual ao contexto nacional [Weber et al. 2004]. Conforme o MPS - BR, os sete níveis de maturidade são (figura 3):

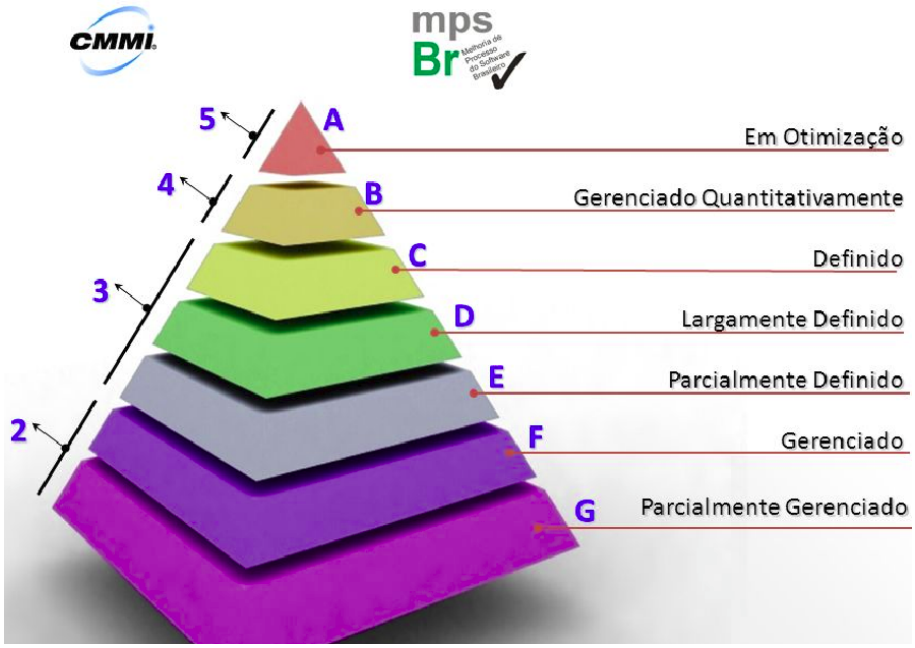

Figura 3. Níveis MPS.
Disponível em: https://www.oficinadanet.com.br/artigo/desenvolvimento/melhoria-de-processos-do-software-brasileiro–mpsbr. Acesso em nov. 2019.

A aplicação do MPS-BR é de natureza gradual, e seu primeiro nível (de baixo para cima) é o G: consiste nas práticas de gerência de projetos (GPR) ou também chamado de gerência de trabalhos e gerência de requisitos (GRE) [MPS-BR 2012]. De formas gerais, a gerência de trabalhos define tarefas essenciais ao processo de construção de software, como monitoramento das atividades; e a gerência de requisitos trata as mudanças e ajustes nos requisitos de desenvolvimento ao longo do projeto. Tanto o monitoramento das atividades, como as reuniões e o *feedback* do cliente, quanto a prática de aceitação das alterações nos requisitos, já são práticas implícitas no Scrum, facilitando assim a aplicação do nível G do MPS-BR [Lima and Vendramel 2016].

3.1. Scrum com MPS-BR

O modelo MPS-BR divide também as tarefas incluídas em cada nível em práticas menores, dessa forma criando um processo mais simples de sua aplicação, com intuito de evolução gradual [MPS-BR 2012]. Para aplicação da gerência de projetos, e também já discutindo os resultados esperados com as práticas e princípios do Scrum, são elas (Tabela 1):

Tabela 1. Gerência de Projetos x Scrum

GPR1	Escopo do trabalho definido: essa etapa contém todas as necessidades e demais características para realizar o trabalho. Associa-se com a reunião sobre visão do produto no Scrum, que finaliza definindo o escopo de trabalho: o *Product Backlog*.
GPR2	As tarefas e os produtos derivados de trabalho são dimensionados utilizando métodos apropriados: nessa prática, o modelo define a decomposição do escopo de trabalho anteriormente definido, dividindo etapas em tamanhos. Inicialmente atendido com as pontuações geradas: as *Story Points* definidas pelo Scrum para cada tarefa na primeira reunião de visão do produto, e atualizadas conforme as modificações entre as *Sprints*.
GPR3	O modelo e as fases do ciclo de vida do trabalho são definidos: essa etapa define divisão do projeto em ciclos de vida, seguindo o escopo de requisitos visando maior controle gerencial com avaliações. Com a divisão do projeto de software pelo Scrum em *Sprints*, essa prática pode ser estabelecida quanto a divisão de fases. Mas também as reuniões *Daily Scrum Meeting*, *Sprint Planning Meeting*, *Sprint Review Meeting* e *Sprint Retrospective Meeting* realizando as reportações necessárias pela equipe e suas avaliações, atendendo as exigências de natureza gerencial.
GPR4	O esforço e o custo para a execução das tarefas do trabalho são estimados com base em dados históricos ou referências técnicas. Essa parte não é atendida explicitamente pelo Scrum.
GPR5	O orçamento e o cronograma do trabalho, incluindo a definição de marcos e pontos de controle, são estabelecidos e mantidos: as definições de cronograma, atribuindo tempo de execução e também o nível de importância é discutida na reunião *Sprint Planning Meeting*, para orçamento não há atendimento pelo Scrum.
GPR6	Os riscos do trabalho são identificados e o seu impacto, prioridade de tratamento E probabilidade de ocorrência são determinados e documentados: essa etapa é atendida pelas reuniões diárias *Sprint Planning Meeting*, com as dificuldades e demais informações sobre o impedimento reportados pela equipe ao *Scrum Master*, que fica como responsável pelas demais tarefas exigidas. A documentação desses riscos segue implícito no *framework*.
GPR7	Os recursos humanos para o trabalho são planejados considerando o perfil e o conhecimento necessários para executá-lo: um dos princípios do Scrum é uma equipe técnica auto gerenciável, de forma que o diferencial técnico de cada membro é realocado conforme necessitado e que os demais evoluam seu conhecimento de processo com a interação entre si [Sutherland 2016]. Esse planejamento então é de responsabilidade da equipe técnica.
GPR8	Os recursos e o ambiente de trabalho necessários para executar o trabalho são planejados: todas os impedimentos e dificuldades encontrados pela equipe técnica são de responsabilidade do *Scrum Master*. Dessa forma, o *Scrum Master* define acordos em que o *Product Owner* pode ser requisitado, como por exemplo o uso de um serviço terceirizado.
GPR9	Os dados relevantes do trabalho são identificados e planejados quanto à forma de coleta, armazenamento e distribuição. Um mecanismo é estabelecido para acessá-los, incluindo, se pertinente, questões de privacidade e segurança: essa prática não é atendida de forma clara pelo Scrum.

GPR10	Um plano geral para a execução do trabalho é estabelecido com a integração de planos específicos: a visão geral do produto no começo do projeto atende ao plano geral de execução. Os planos específicos são relacionados aos artefatos *Product Backlog* e *Sprint Backlog*.
GPR11	A viabilidade de atingir as metas do trabalho é explicitamente avaliada considerando restrições e recursos disponíveis. Se necessário, ajustes são realizados: a viabilidade de execução fica por responsabilidade do *Scrum Master* e também do *Product Owner*, que podem ser melhor medidas e discutidas na reunião de *Sprint Planning Meeting*.
GPR12	O Plano do Trabalho é revisado com todos os interessados e o compromisso com ele é obtido e mantido: essa etapa é de fácil associação com as reuniões de *Sprint Planning Meeting* e *Daily Scrum Meeting*, que define e mantém o time nos objetivos estabelecidos.
GPR13	O escopo, as tarefas, as estimativas, o orçamento e o cronograma do trabalho são monitorados em relação ao planejado: inicialmente atendido para monitoramento pela reunião de *Daily Scrum Meeting* e posteriores ao ciclo de trabalho com a *Sprint Review Meeting* e a *Sprint Retrospective Meeting*. O cronograma, conforme citado pelo autor [Sutherland 2016], pode ser atendido pelo gráfico de *Burndown*. Esse gráfico apresenta o trabalho realizado em relação ao total planejado durante a *Sprint*. O orçamento não é atendido de forma explícita.
GPR14	Os recursos materiais e humanos bem como os dados relevantes do trabalho são monitorados em relação ao planejado: como essa prática é durante a execução do plano de trabalho, pode ser considerada como responsabilidade do *Scrum Master*.
GPR15	Os riscos são monitorados em relação ao planejado: o *Scrum Master* é responsável por avaliar durante as *Daily Scrum Meetings* os riscos e impedimentos do time, e assim alcançando o resultado.
GPR16	O envolvimento das partes interessadas no trabalho é planejado, monitorado e mantido: fácil associação com o princípio de time defendido pelo Scrum. Toda a evolução do projeto é considerada e é compartilhada de maneira clara com os *stakeholders*, em especial pelo *Product Owner* que é a representação dos *stakeholders* e é incluído nas demais atividades do projeto como de fato um membro do time.
GPR17	Revisões são realizadas em marcos do trabalho e conforme estabelecido no planejamento: esse resultado é dividido entre a apresentação do realizado pela equipe técnica com a *Sprint Review Meeting* e pela avaliação dirigida pelo *Scrum Master* com a *Sprint Retrospective Meeting*.
GPR18	Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas: essa etapa é de maior associação com a *Sprint Review Meeting*, na qual o time apresenta o exigido pela prática ao *Product Owner*.
GPR19	Ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas identificados são estabelecidas, implementadas e acompanhadas até a sua conclusão: é inicialmente estimado pelas correções responsáveis ao *Scrum Master*, mas o acompanhamento até a conclusão não é definido de forma clara pelo *framework* Scrum.

Mas também em relação às práticas descritas na gerência de requisitos, da mesma forma retiradas do documento do modelo MPS-BR [MPS-BR 2012], com suas possibilidades de vinculação com o Scrum (Tabela 2):

Tabela 2. Gerência de Requisitos x Scrum

GRE1	O entendimento dos requisitos é obtido junto aos fornecedores de requisitos: esta prática procura garantir a clareza entre os requisitos necessários do projeto junto ao *Product Owner*, pode se dizer que o Scrum atende com a fase de visão do produto, com a reunião discutida entre todo o time. Aqui o modelo exige documentação dessa comprovação, o que deve ser reforçado, já que o Scrum trata de forma implícita;
GRE2	Os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com estes requisitos é obtido: especifica um filtro nos requisitos após a primeira prática, de forma a transformar os requisitos em implementações de serviços no software, envolvendo a equipe técnica para isso. A reunião *Sprint Planning Meeting* atende essa prática, ainda mais quando os requisitos sofrem modificações: o modelo descreve a necessidade de um novo planejamento com base nessas mudanças de requisitos;
GRE3	A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida: essa prática tenta estimar o impacto nas mudanças de requisitos relacionado com as tarefas da gerência de projetos, apesar de citado essa forma de avaliação, não há prática definida no Scrum.
GRE4	Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos: o modelo cita revisões e correção de problemas visualizados no processo, essa etapa pode ser tratada na reunião - *Sprint Review Meeting* - que apresenta a visão da equipe de construção junto ao *Product Owner* e também com suas correções dirigidas pelo *Scrum Master*.
GRE5	Mudanças nos requisitos são gerenciadas ao longo do projeto: essa prática é explanada pelo modelo para resolver parte das consequências do contexto de mudanças constantes na produção de software. E essa prática é diversamente reforçada pelo Scrum, desde as reuniões diárias praticadas pelo time quanto também as *Sprint Review Meetings* de revisão depois de cada *Sprint*.