CMMI guiando a integração de padrões organizacionais e de processo ao método ágil Scrum

Edes G. Costa Filho
Universidade Federal de São Carlos, Departamento de Computação
São Carlos, Brasil
edes_filho@dc.ufscar.br

Rosângela Ap. D. Penteado
Universidade Federal de São Carlos, Departamento de Computação
São Carlos, Brasil
rosangel@dc.ufscar.br

Júnia C. Anacleto
Universidade Federal de São Carlos, Departamento de Computação
São Carlos, Brasil
junia@dc.ufscar.br

Abstract

Practices proposed by agile methods such as Scrum and Extreme Programming (XP) satisfy many CMMI goals. However, additional practices are needed to complement these agile methods to satisfy other CMMI goals. Organizational and process patterns provide proven solutions to recurring software development process problems and can be used to adapt Scrum and other agile methods according to CMMI. This article proposes the integration of some organizational and process patterns to Scrum so that more CMMI practices can be satisfied when Scrum are used.

Keywords: Scrum, Extreme Programming (XP), CMMI, Organizational and Process Patterns

Resumo

Práticas propostas por métodos ágeis como Scrum e Extreme Programming (XP) satisfazem muitos objetivos do modelo de melhoria de processo CMMI. Porém, práticas adicionais são necessárias para complementar esses métodos para satisfazer outros objetivos do CMMI. Os padrões organizacionais e de processo propõem soluções comprovadas para problemas recorrentes no desenvolvimento de software e podem ser utilizados para adaptar Scrum e outros métodos ágeis de acordo com o modelo CMMI. Este artigo propõe a integração de alguns padrões organizacionais e de processo ao método ágil Scrum para que mais práticas do CMMI possam ser atendidas quando se utiliza Scrum.

Palabras claves: Scrum, Extreme Programming (XP), CMMI, Padrões Organizacionais e de Processo

1. INTRODUÇÃO

Um desafio constante da área de Engenharia de Software é melhorar o processo de desenvolvimento de software. Mesmo com a constante evolução de métodos, técnicas e ferramentas, a entrega de software com qualidade, em prazos e custos estabelecidos nem sempre é conseguida.

Os métodos ágeis de desenvolvimento vêm ganhando mais atenção na indústria de software, devido à sua proposta de liberar o produto de forma mais rápida do que os modelos de processo convencionais. Nos últimos anos, métodos ágeis como XP [4], Scrum [28] e Crystal [7] passaram a ser amplamente utilizados. Contudo, as organizações que desenvolvem software não têm se limitado ao uso de métodos ágeis ou modelos convencionais para desenvolvê-lo, cada vez mais elas estão interessadas em avaliar e melhorar seus processos. As organizações estão atentas ao fato de que a qualidade do produto desenvolvido depende diretamente do processo utilizado para seu desenvolvimento [16].

Processos de desenvolvimento de software podem ser avaliados e melhorados por meio de modelos de melhoria de processo, como, por exemplo, CMM (*Capability Maturity Model*) [25] e CMMI (*Capability Maturity Model Integration*) [29].

Estudos mostram que muitas das práticas propostas pelos métodos ágeis Scrum e XP satisfazem alguns objetivos dos modelos CMM e CMMI [21, 26, 27, 32]. Para Glazer [17] e Paulk [26], CMM e XP podem ser considerados complementares, pois CMM mostra às organizações o que fazer, mas não como fazer, e as práticas do XP contêm informações claras e específicas de como fazer. Porém, apesar de algumas práticas serem atendidas por esses métodos, existem práticas não abordadas e objetivos não satisfeitos.

Nesse contexto é que os padrões organizacionais e de processo podem ser úteis. Esses padrões cobrem problemas de desenvolvimento e podem ser utilizados para modelar uma nova organização e seu processo de desenvolvimento [10]. Assim, eles podem ser integrados a métodos ágeis para adequá-los ao modelo CMMI.

Somente Scrum e CMMI são abordados neste trabalho, pois CMMI é derivado do CMM, Scrum é um método ágil flexível e ambos são amplamente utilizados. Este artigo tem por objetivo ajudar as organizações que utilizam Scrum a se adequarem ao modelo CMMI por meio da utilização de padrões organizacionais e de processo.

Aqui são apresentados alguns padrões organizacionais e de processo, propostos por diferentes autores, que podem ser integrados ao Scrum, para que ele atenda algumas das práticas do modelo CMMI ainda não atendidas por esse método ágil. Na Seção 2, são apresentados alguns trabalhos relacionados encontrados na literatura. Na Seção 3, são apresentados conceitos básicos sobre métodos ágeis, Scrum e padrões organizacionais e de processo. A Seção 4 propõe a integração de alguns padrões organizacionais e de processo ao Scrum para atender mais práticas sugeridas pelo CMMI. Finalmente, na Seção 5, estão as considerações finais.

2. TRABALHOS RELACIONADOS

A relação entre os modelos de melhoria de processo e métodos ágeis pode ser encontrada em trabalhos que comparam as práticas fornecidas pelos métodos ágeis e as práticas das áreas chave de processo - KPAs *(Key Process Area*) do CMM e as áreas de processo do CMMI [21, 26, 27, 32].

Paulk [26] analisa como as práticas do XP satisfazem as práticas das KPAs dos níveis 2 e 3 do CMM. Schwaber [27] evidencia quais KPAs dos níveis 2 e 3 do CMM são atendidas pelas práticas do Scrum.

Kane e Ornburn [21] apresentam as áreas de processo CMMI que são totalmente, parcialmente ou não atendidas pelas práticas do XP e Scrum. Os autores afirmam que práticas complementares podem ser utilizadas para auxiliar aquelas áreas do CMMI parcialmente ou não atendidas.

Zanatta e Vilain [32] apresentam uma análise do Scrum para as áreas de processo Gestão de Requisitos (*Requirements Management - REQM*) e Desenvolvimento de Requisitos (*Requirements Development - RD*), comparando as práticas do Scrum às práticas específicas dessas duas áreas de processo. Essa análise mostra que, com relação à área Gestão de Requisitos, 40% das práticas específicas são atendidas, 40% são parcialmente atendidas e apenas 20% não são atendidas e, com relação à área de processo Desenvolvimento de Requisitos, 59% das práticas específicas são atendidas, 8% são parcialmente atendidas e 33% não são atendidas. Os autores propõem, ainda, um conjunto de soluções de forma que o Scrum atenda totalmente as áreas de processo Gestão e Desenvolvimento de Requisitos.

Para Turner e Jain [31], embora existam diferenças entre a abordagem do CMMI e dos métodos ágeis, existem semelhanças entre elas e seus pontos fortes e fracos são complementares.

Pode-se inferir que a maior parte desses trabalhos limita-se a comparar os métodos ágeis XP e Scrum com modelos de melhoria de processo CMM e CMMI. Poucas soluções são propostas para adaptar esses métodos ágeis para que atendam as práticas das áreas de processo parcialmente ou não tratadas por eles.

3. VISÃO GERAL SOBRE MÉTODOS ÁGEIS, SCRUM E PADRÕES ORGANIZACIONAIS E DE PROCESSO

As organizações constantemente necessitam de rapidez no processo de desenvolvimento para satisfazer às exigências de seus clientes. Assim, os métodos ágeis de desenvolvimento tornam-se mais atrativos por abordarem o processo de desenvolvimento de software de forma diferente da dos modelos preconizados anteriormente pela Engenharia de Software. A principal diferença está na forma como as modificações são tratadas durante o desenvolvimento do software. Os modelos de processo convencionais, que adotam a estratégia de previsibilidade, tentam levantar todos os requisitos antes de iniciar o desenvolvimento. Depois é feito um planejamento para que as modificações possam ser controladas no decorrer do processo de desenvolvimento do software. Já os métodos ágeis optam pela adaptabilidade, o levantamento de requisitos ocorre gradualmente e o planejamento é contínuo, para que a adaptação às modificações possa ocorrer.

Os valores compartilhados pelos métodos ágeis, definidos pelo "Manifesto Ágil" [3], são:

- Indivíduos e interações são mais importantes que processos e ferramentas;
- Software funcionando é mais importante do que documentação detalhada;
- Colaboração dos clientes é mais importante do que negociação de contratos;
- Adaptação às mudanças é mais importante do que seguir um plano.

Os métodos ágeis enfatizam os aspectos humanos do desenvolvimento de software ao invés dos aspectos de Engenharia [23]. Segundo Highsmith e Cockburn [20], o que existe de novo nos métodos ágeis não são as práticas que eles usam, mas o reconhecimento de que as pessoas são os principais condutores de sucesso do projeto. Outra característica, desses métodos, é que eles não são orientados à documentação. Porém, utilizar um método ágil não significa ausência de documentação, mas sim registrar o essencial que auxilie efetivamente o desenvolvimento do software, evitando redundâncias e excessos.

XP, que é o mais popular dos métodos ágeis, define um conjunto de práticas, definidas com base em cinco valores que são: comunicação, simplicidade, *feedback,* coragem e respeito [4]. Essas práticas apóiam umas às outras e devem ser utilizadas em conjunto, para se obter agilidade no processo. Aplicar as práticas de forma isolada pode não produzir a agilidade desejada.

O Scrum é outro método ágil largamente utilizado, apresentado resumidamente na Seção 3.1, por ser o utilizado nessa proposta.

3.1. Scrum

Scrum foi desenvolvido para gerenciar o processo de desenvolvimento de software em ambientes em que os requisitos estão em constante mudança. Esse método é apropriado para equipes pequenas, com até dez integrantes [1] e não fornece práticas específicas de desenvolvimento de software, mas sim certas práticas de gerenciamento [28], que são descritas a seguir:

- **Lista de Tarefas do Produto (*Product Backlog*):** lista priorizada e constantemente atualizada que contém os requisitos do sistema que está sendo construído ou melhorado.
- **Estimativa da Lista de Tarefas do Produto (*Product Backlog Estimation*):** é obtida por meio de um processo iterativo para cada item da Lista de Tarefas do Produto.
- ***Sprint*:** são intervalos fixos de tempo, nos quais todo o trabalho é realizado. No Scrum um *Sprint* tem duração de trinta dias, período em que a equipe se organiza para produzir um incremento do produto.
- **Reunião de Planejamento do *Sprint* (*Sprint Planning Meetings*):** reunião realizada para determinar o objetivo do *Sprint* e definir as funções do sistema que serão desenvolvidas.
- **Lista de Tarefas do Sprint (*Sprint Backlog*):** lista que contém os itens da Lista de Tarefas do Produto que devem ser realizados no próximo *Sprint* e que foram selecionados a partir da Reunião de Planejamento do *Sprint.*
- **Reuniões Diárias (*Daily Scrum Meetings*):** reuniões de aproximadamente quinze minutos realizadas diariamente para verificar o progresso do projeto e para discutir questões como: o que foi feito desde a última reunião, o que precisa ser feito até a próxima e quais foram os problemas encontrados para realizar o trabalho.
- **Reunião de Revisão de *Sprint* (*Sprint Review Meeting*):** reunião realizada ao final do *Sprint*, na qual os resultados nele alcançados são apresentados para o cliente e para gerência.
- **Reunião de Retrospectiva de *Sprint* (*Sprint Retrospective Meeting*):** reunião realizada ao final do *Sprint*, depois da reunião de revisão, na qual deve ser discutido o que pode ser melhorado para o próximo *Sprint* [27].

Ao não fornecer práticas específicas de desenvolvimento, o Scrum se torna um método flexível, possibilitando que a organização tenha liberdade para utilizar as práticas de desenvolvimento já adotadas por ela.

Segundo Abrahamsson et al. [1] os papéis identificados no Scrum são:

- **Mestre Scrum** (*Scrum Master*): responsável por garantir que as práticas Scrum estão sendo seguidas corretamente.

- **Proprietário do Produto** (*Product Owner*): responsável por controlar a lista de tarefas do produto (*Product Backlog*) e assegurar que ela está visível para todos.
- **Equipe Scrum** (*Scrum Team*): responsável por desenvolver o software, além de participar da criação da lista de tarefas do produto e estimativas de esforço.
- **Cliente** (*Customer*): responsável por identificar e priorizar os requisitos que vão compor lista tarefas do produto e avaliar o produto nas reuniões de revisão de *Sprint*.
- **Gerência** (*Management*): responsável por tomar as decisões críticas do projeto.

3.2. Padrões Organizacionais e de Processo

Coplien e Harrison [11] definem padrão como uma configuração estrutural recorrente que resolve um problema em um determinado contexto. Independente do autor, todas as definições mostram o principal objetivo dos padrões, o seu reúso, ou seja, a reaplicação das soluções de sucesso na solução dos problemas que estão em um mesmo domínio.

O reúso de software é uma atividade comum durante o processo de desenvolvimento. Padrões de software (software *patterns*) auxiliam e contribuem para o reúso em níveis mais altos de abstração, como por exemplo, de análise, arquitetura, projeto, processo e organização. Das diversas categorias de padrões que surgiram, os organizacionais e de processo são os que têm por objetivo apoiar a construção do software e melhorar o seu desenvolvimento.

Além de estarem divididos em categorias, os padrões podem ser agrupados em linguagens de padrões: um sistema de padrões organizados em uma estrutura que guia a aplicação dos mesmos [15].

O formato de apresentação de um padrão varia de acordo com seu autor. Porém, existem elementos essenciais que devem estar bem claros na apresentação de um padrão [2]:

- **Nome:** identificação do padrão.
- **Contexto**: descrição da pré-condição dentro da qual o problema e a solução ocorrerem. O contexto mostra a aplicabilidade do padrão.
- **Problema**: descrição do que o padrão se propõe a solucionar.
- **Forças:** descrição das forças relevantes e restrições para o problema, e de como elas interagem umas com as outras.
- **Solução**: descrição da solução proposta pelo padrão para resolver o problema e alcançar o resultado desejado.
- **Contexto Resultante**: descrição do estado ou configuração que se obterá após aplicação do padrão, incluindo as conseqüências da sua aplicação e padrões envolvidos no contexto resultante.
- **Padrões Relacionados**: descrição da relação entre o padrão aplicado e outros padrões existentes.
- **Usos Conhecidos**: mostra ocorrências da utilização do padrão.

O padrão Scenarios Define Problem (Cenários Definem o Problema) exemplifica o formato de apresentação de um padrão organizacional e de processo, documentado por Coplien e Harrison [11]:

- **Nome:** *Scenarios Define Problem* (Cenários Definem o Problema)
- **Contexto:** necessita-se de um mecanismo para que o cliente e os desenvolvedores estejam unidos e engajados.
- **Problema:** documentos de projeto são freqüentemente meios ineficazes para mostrar ao cliente a visão de como o sistema deve funcionar.
- **Forças:** distância e desconfiança entre clientes e desenvolvedores são questões inerentes ao desenvolvimento de software. A comunicação entre eles é crucial para o sucesso do projeto.
- **Solução:** utilize casos de uso para obter os requisitos funcionais do sistema. Os casos de uso obtêm todos os cenários que o sistema deve tratar e os requisitos não funcionais podem e devem ser acrescentados aos casos de uso.
- **Contexto Resultante:** pode-se utilizar os casos de uso para comunicação e esclarecimento de requisitos, uma vez que o problema já está definido.

- **Padrões Relacionados:** Catalytic Scenarios da linguagem de padrões "Demo Prep" [12], Mercenary Analyst, da linguagem de padrões "Project Management Pattern Language" [11].
- **Usos Conhecidos:** Cockburn [9] é uma das referências mais conhecidas sobre casos de uso.

4. CMMI, MÉTODOS ÁGEIS E PADRÕES ORGANIZACIONAIS E DE PROCESSO

A partir de estudos realizados foi possível detectar que padrões organizacionais e de processo podem ser aplicados para satisfazer aos objetivos de algumas áreas de processo do CMMI que não são satisfeitos quando somente as práticas do Scrum são utilizadas. Os trabalhos disponíveis na literatura, que comparam CMMI e Scrum, propõem poucas soluções para cobrir as práticas do CMMI não atendidas por esse método. Padrões organizacionais e de processo podem ser integrados ao Scrum para adaptá-lo às exigências do CMMI, como mostrado mais adiante.

No CMMI, cada área de processo possui objetivos específicos e práticas específicas (*Specific Practices - SP*) que devem ser realizadas para satisfazer os objetivos. Por exemplo, a área de processo Desenvolvimento de Requisitos (*Requirements Development*), cujo objetivo é analisar e produzir os requisitos do produto e cliente [30], determina que as necessidades do cliente devem ser extraídas (SP 1.1-2 Extrair as Necessidades) e as funcionalidades requeridas devem ser estabelecidas e mantidas (SP 3.2-1 Estabelecer Definição das Funcionalidades Requeridas).

Para Zanatta e Vilain [32], as práticas específicas SP 1.1-2 Extrair as Necessidades e SP 3.2-1 Estabelecer Definição das Funcionalidades Requeridas, citadas acima, não são atendidas pelo Scrum. Para permitir que organizações que utilizam o Scrum realizem essas práticas, três padrões organizacionais e de processo, que podem ser integrados ao Scrum, são sugeridos na Tabela 1. A primeira coluna apresenta o nome do padrão e seu autor, e a segunda, um resumo contendo o problema e a solução proposta.

Tabela 1. Padrões para obter requisitos.

Padrão - Autor	Resumo do Padrão
Scenarios Define Problem (Cenários Definem o Problema) [11]	**Problema**: Documentos de projeto são meios ineficientes para mostrar ao cliente a visão de como o sistema deve funcionar. **Solução**: Utilizar casos de uso para extrair todos os requisitos do sistema. Os casos de uso obtêm todos os cenários que o sistema deve tratar. Os requisitos não funcionais devem ser acrescentados aos casos de uso.
Build Prototypes (Construa Protótipos) [11]	**Problema**: Obter todos os requisitos necessários para construir o sistema. Embora muitos desses requisitos sejam fornecidos pelos clientes, alguns são decisões de projeto que surgem da solução. **Solução**: Construir protótipos para extrair e entender os requisitos.
Use Requirements Index Cards (Use Cartões de Índice de Requisitos) [18]	**Problema**: Especificar os requisitos de forma única. **Solução:** Utilizar cartões de índice para especificar os requisitos, cada um contendo uma especificação com informações adicionais para classificação, como atributos para gestão de requisitos e rastreamento do progresso do projeto.

Uma questão importante que deve ser observada é a criação de novos artefatos, propostos pelos padrões, no Scrum. Ao contrário do que, muitas vezes, é deduzido, os métodos ágeis não são contra documentação, eles são a favor de uma documentação essencial que pode agregar valor ao produto. O Scrum define apenas três artefatos que devem ser utilizados no desenvolvimento do software: a lista de tarefas do produto (*Product Backlog*); a lista de tarefas do *Sprint* (*Sprint Backlog*); um gráfico de acompanhamento, que mostra a quantidade de trabalho remanescente ao longo do *Sprint* (*Burndown Chart*). Porém, o Scrum permite a criação e utilização de qualquer outro artefato, mas foge do seu escopo mostrar como e quais outros artefatos podem ser utilizados. Se a equipe identificar que outros diagramas de trabalho podem dar mais qualidade ao produto, eles podem ser utilizados [22].

Este trabalho propõe a utilização dos padrões discutidos na Tabela 1, gerando novos artefatos, mas que não engessam o método ágil, mas sim, auxiliam para que o Scrum possa ser utilizado em uma organização interessada em certificação CMMI.

Outros padrões podem ser identificados e novas soluções podem ser aplicadas para adaptar e melhorar o processo da organização que utiliza o Scrum. A Tabela 2 apresenta cinco padrões que estão relacionados com os padrões apresentados na Tabela 1 e que fornecem diretrizes para melhor utilização de casos de uso e protótipos. Suas colunas contem o mesmo tipo de informação que o da Tabela 1.

Tabela 2. Padrões relacionados a elaboração de casos de uso e protótipos.

Padrão - Autor	Resumo do Padrão
Element Identification (Identificação de Elementos) [12]	**Problema**: Selecionar as funções do software que devem ser demonstradas para o cliente para manter sua confiança. **Solução**: Identificar as funções do software que preocupam os clientes. Converse com o cliente e escute atentamente.
Judicious Fireworks (Fogos de Artifício Sensatos) [12]	**Problema**: Tranqüilizar o cliente quanto ao software que está sendo construído. **Solução**: Apresentar o que vai ser desenvolvido sem criar expectativas do cliente de características que o produto final não terá.
Small Writing Team (Equipe Pequena para Escrita) [6]	**Problema**: Ineficiência na escrita de casos de uso em equipes com muitos membros. **Solução**: Limitar a equipe em três membros para criar os casos de uso. É mais fácil de conseguir consenso.
Breadth Before Depth (Largura Antes de Profundidade) [6]	**Problema**: Detalhar os casos de uso antes de conhecer todo o escopo do sistema. **Solução**: Desenvolver uma visão geral dos seus casos de uso, e depois adicione os detalhes gradualmente, trabalhando lado a lado com um conjunto de casos de uso relacionados.
*Spiral Development (*Desenvolvimento Espiral) [6]	**Problema**: Desenvolver casos de uso de uma única vez dificulta a adição de novas informações, e ainda pode atrasar a descoberta de novos fatores de risco**.** **Solução**: Aperfeiçoar os casos de uso iterativamente, aumentando progressivamente a precisão de um conjunto de casos de uso em cada iteração.
Mercenary Analyst (Analista Mercenário) [11]	**Problema**: Criar e manter documentação técnica pertinente ao sistema em desenvolvimento. **Solução**: Atribuir a um membro da equipe a responsabilidade de escrever a documentação necessária e manter essa documentação disponível para todos.

Os padrões *Element Identification* e *Judicious Fireworks* mostram que o interesse do cliente deve ser levado em consideração na demonstração de um protótipo. Além desses, existem outros que podem ser utilizados para guiar a construção, administração e demonstração de protótipos [12]. Se a organização decidir utilizar o padrão *Build Prototypes*, os padrões de Coram [12] podem ser úteis, mas se a organização optar pela utilização de casos de uso, os padrões *Small Writing Team, Breadth Before Depth* e *Spiral Development* mostram como criar e desenvolver os casos de uso de forma eficiente. Já o padrão *Mercenary Analyst* [11] indica que boa documentação deve fazer parte do projeto do sistema. Dessa forma, se um dos padrões da Tabela 1 for utilizado pela organização, o *Mercenary Analyst* pode ser utilizado para manter os artefatos gerados atualizados.

Os padrões organizacionais e de processo estão relacionados uns aos outros. Além disso, existem padrões relacionados diretamente com práticas propostas pelo Scrum e XP, pois alguns padrões organizacionais e de processo são utilizados como prática dos métodos ágeis [13]. Da mesma forma, as áreas de processo do CMMI interagem entre si e influenciam umas às outras, independente da categoria a qual pertencem (*Process Management, Project Management, Engineering* e *Support*) [29]. Isso possibilita que um padrão atenda a mais de uma prática. Por exemplo, as práticas específicas SP 1.1-2 Extrair as Necessidades e SP 3.2-1 Estabelecer Definição das Funcionalidades Requeridas, que pertencem à área de processo Desenvolvimento de Requisitos, podem ser atendidas por meio da aplicação do padrão *Scenarios Define Problem.* Outro exemplo é o padrão *Build Prototypes*, que, pode atender a práticas específicas das áreas de processo Desenvolvimento de Requisitos e Solução Técnica. Esse padrão, além de poder ser utilizado para extrair e esclarecer requisitos (SP 1.1-2 Extrair as Necessidades), também pode ser usado para facilitar o projeto do produto (SP 2.1-1 Projetar o Produto ou Componente do Produto).

Além da área de processo de Desenvolvimento de Requisitos (RD), outras áreas podem ser tratadas por meio da utilização dos padrões organizacionais e de processo. O Scrum não fornece práticas que tratam de outras áreas de processo, como por exemplo, Gestão de Configuração (*Configuration Management*), Integração de Produto (*Product Integration*), Solução Técnica (*Technical Solution*) e Treinamento Organizacional (*Organizational Training*). A Tabela 3 mostra outros padrões que podem ser integrados ao Scrum para que ele atenda às práticas dessas áreas de processo. A primeira coluna contém a área de processo à qual o padrão se adapta. A segunda coluna apresenta o objetivo dessa área

de processo de acordo com Turner et al. [30]. A terceira e quarta colunas mostram, respectivamente, o nome do padrão, seu autor e um resumo do padrão apresentando o problema que se propõe a resolver e a solução sugerida.

Tabela 3. Áreas de Processo e Padrões Organizacionais e de Processo

Área de Processo	Objetivo	Padrão - Autor	Resumo do Padrão
Configuration Management (Gestão de Configuração)	Estabelecer e manter a integridade dos produtos de trabalho utilizando controles de configuração e identificação	*Private Workspace* (Área Privada) [5]	**Problema**: Trabalhar em um ambiente no qual os produtos de trabalho são compartilhados. **Solução**: Fornecer aos desenvolvedores um mecanismo para que eles possam realizar seu trabalho em uma área privada, na qual possam controlar as versões do código em que estão trabalhando.
		Repository (Repositório) [5]	**Problema**: Obter as versões corretas dos componentes corretos na sua área privada. **Solução**: Manter apenas um ponto de acesso, ou repositório, para seu código e artefatos relacionados.
Product Integration (Integração do Produto)	Integrar o produto a partir de seus componentes e assegurar que as funções do produto integrado estão corretas.	*Named Stable Bases* (Bases Estáveis Nomeadas) [11]	**Problema**: Integrar o software frequentemente para que a base não fique desatualizada. **Solução**: Integrar o sistema uma vez por semana. Dar a base estável um nome de forma que os desenvolvedores possam identificar as funções que a compõem.
Technical Solution (Solução Técnica)	Projetar, desenvolver e implementar soluções para as necessidades do cliente. Soluções, projetos e implementações produzem o produto.	*Unit Test* (Teste de Unidade) [5]	**Problema**: Testar se um módulo ou função funciona da forma adequada após uma modificação. **Solução**: Desenvolver e executar testes de unidade após uma modificação.
		Architect Controls Product (Arquiteto Controla o Produto) [11]	**Problema**: Dar ao produto que está sendo desenvolvido elegância e coesão. **Solução**: Criar um papel de Arquiteto, cujo objetivo é definir um estilo de arquitetura para o produto. O arquiteto deve aconselhar os desenvolvedores.
		Architect Also Implements (Arquiteto também Implementa) [11]	**Problema**: Fornecer direção técnica para os desenvolvedores. **Solução**: Atribuir ao Arquiteto a responsabilidade de participar da implementação, além de aconselhar os desenvolvedores.
		Lock 'Em Up Together (Agrupe o Time) [19]	**Problema**: Criar uma arquitetura única e coerente em equipe. **Solução**: Agrupar todos os membros da equipe no início do projeto para elaborar a arquitetura, de forma que se comprometam a participar totalmente até que a arquitetura esteja completa ou pelo menos suficientemente clara.
Organizational Training (Treinamento Organizacional)	Desenvolver as habilidades e conhecimento das pessoas para que elas possam exercer seus papéis efetivamente e eficientemente.	*Apprenticeship* (Aprendizado) [11]	**Problema**: Desenvolver e manter o conhecimento do domínio. **Solução**: Transformar os novos contratados em especialistas por meio de um programa de aprendizagem, no qual todo novo empregado deve trabalhar como aprendiz de especialista.

		Day Care (Dia de Atenção) [8]	**Problema**: Gastar tempo ensinando os empregados inexperientes ou novatos. **Solução**: Colocar um especialista para treinar os novatos contratados e deixar os outros especialistas desenvolverem o sistema. Permitir que os novatos contribuam em uma pequena parte do projeto para ganharem experiência.

Ressalta-se que outros padrões não apresentados aqui, podem ser também integrados ao Scrum de modo que todas ou a maioria das práticas das áreas de processo CMMI sejam cobertas. Por exemplo, a linguagem de padrões para gestão de configuração de software [5] possui dezesseis padrões, sendo que a utilização desses integrada ao Scrum pode ser também recomendada, dependendo da forma como a organização realiza suas atividades.

A seguir é comentada como a integração dos padrões apresentados nas Tabelas 1, 2 e 3 ocorre junto às práticas preconizadas pelo Scrum:

- **Definição dos Papéis**: No Scrum, antes do início do projeto ocorre a definição dos papéis, em que o mestre Scrum (*Scrum Master*) e gerência identificam o dono do produto (Product Owner) e equipe Scrum (Scrum Team) para o projeto [28]. É necessário identificar também o Arquiteto (*Architect Controls Product*), que deve ser um membro do Scrum Team (*Architect Also Implements*) e definir o(s) responsável(eis) pelos novos contratados ou novatos no projeto (*Apprenticeship e Day Care*).
- **Obtenção dos Requisitos**: Após a definição dos papéis, os requisitos devem ser obtidos junto ao cliente. Pode ser utilizado um ou um conjunto de padrões para extração e entendimento dos requisitos. Se for escolhido o padrão *Scenarios Define Problem,* casos de uso devem ser criados, primeiramente, em alto nível (*Breadth Before Depth*) para que os detalhes sejam obtidos iterativamente (*Spiral Development*), nos *Sprints*. Deve-se ter em mente que os casos de uso devem ser criados por três pessoas, no máximo, para que se atinja um consenso em menor tempo (*Small Writing Team*). O resultado obtido com a utilização dos casos de uso deve ser adicionado à lista de tarefas do produto.
- **Reunião de Planejamento de Sprint**: Essa reunião deve ser realizada pelo mestre Scrum, dono do produto e equipe Scrum [28]. Além de planejar o *Sprint* os membros da equipe Scrum devem elaborar uma arquitetura inicial do produto (*Lock Em Up Together*), guiados pelo Arquiteto (*Architect Controls Product*). O mestre Scrum é responsável por proteger a equipe de interrupções nessa etapa. Essa prática vai permitir que todos tenham uma visão clara e comum da arquitetura.
- ***Sprint***: Finalmente, o *Sprint* pode ser iniciado. Os desenvolvedores compartilham produtos de trabalho comuns como, por exemplo, a lista de tarefas do produto, lista de tarefas do *Sprint* e código, que podem ser modificados ao longo do projeto e, assim, controlar versões de software é essencial. Para isso, devem ser criados um repositório único para os produtos de trabalho compartilhados (*Repository*) e uma área privada para cada desenvolvedor trabalhar de forma isolada (*Private Workspace*). Além disso, os desenvolvedores devem realizar um teste de unidade sempre que uma função for modificada (*Unit Test*) e realizar a integração do software uma vez por semana (*Named Stable Bases*). Isso facilitará a preparação do incremento do produto para apresentação na Reunião de Revisão de *Sprint*, já que a equipe Scrum não deve gastar mais de uma hora para se preparar para essa reunião de revisão [28].

5. CONSIDERAÇÕES FINAIS

Soluções comprovadas e de sucesso são documentadas na forma de padrões. Dessa forma, este artigo sugere uma adaptação do método ágil Scrum por meio da integração de padrões organizacionais e de processo para que as organizações possam obter certificação CMMI.

Neste artigo foram selecionados alguns padrões organizacionais e de processo, pertencentes a diferentes linguagens de padrões e/ou autores, para exemplificar essa integração. Salienta-se, entretanto, que não são atendidas todas as práticas de uma determinada área de processo, porém, como cada padrão, em sua forma de apresentação, tem padrões relacionados, esses podem também ser utilizados para completar o atendimento. Essa atividade faz parte de trabalhos futuros a serem realizados.

Para utilizar as práticas propostas pelos padrões organizacionais e de processo de forma efetiva, é necessário entender como elas se relacionam com as práticas do método ágil Scrum. Isso foi notado quando um pequeno estudo de caso foi realizado com alunos de pós-graduação compondo uma organização fictícia. Os alunos foram inicialmente divididos em duas equipes (organizações) com cinco elementos cada uma. A aplicação dos padrões foi realizada de

forma correta, porém não foi percebida, pelas equipes, a relação entre as soluções fornecidas pelos padrões e as práticas Scrum. Por exemplo, os casos de uso foram criados e detalhados de forma correta, como indicam os padrões *Scenarios Define Problem*, *Breadth Before Depth* e *Small Writing Team*, porém, após a obtenção dos requisitos, seguindo o padrão *Lock 'Em Up Together,* todos os membros da equipe devem ser colocados juntos em uma sala até elaborar a arquitetura do produto, que não precisa ser detalhada, mas sim ficar clara o suficiente para que o trabalho possa ser iniciado, o que não foi cumprido pela equipe. Em vez disso, a equipe utilizou as reuniões diárias do Scrum para definir e detalhar a arquitetura do produto, o que não é uma prática correta, pois essas reuniões, de aproximadamente quinze minutos, são para acompanhamento de projeto e não de trabalho. Além disso, segundo o padrão *Lock 'Em Up Together* a reunião para definição da arquitetura deve ser realizada apenas no início do projeto. A arquitetura deveria evoluir naturalmente durante o desenvolvimento nos *Sprints.* A conseqüência disso foi atraso no cronograma.

Nesse mesmo estudo de caso, foi proposto o desenvolvimento de dois sistemas por duas equipes em duas etapas: 1) desenvolvimento de um sistema sem o uso do Scrum e dos padrões organizacionais e de processo, mas seguindo os princípios do desenvolvimento ágil [3], já conhecidos por eles; 2) desenvolvimento de outro sistema com o uso de dos padrões organizacionais e de processo e utilizando as práticas Scrum. Para que as equipes não fossem influenciadas pelo conhecimento dos requisitos do sistema, em cada etapa, cada equipe desenvolveu um dos sistemas, já que ambos apresentavam a mesma complexidade. Pôde-se observar que os produtos provenientes da segunda etapa foram superiores aos da primeira para as duas equipes: quando o Scrum e os padrões organizacionais e de processo não foram utilizados, na primeira etapa, parte da funcionalidade do sistema não foi implementada por falta de tempo e organização na forma de trabalho adotada nas duas equipes, diferente da segunda etapa, na qual toda funcionalidade do sistema foi implementada em menor tempo.

Com a integração dos padrões, novas atividades, papéis, artefatos e diretrizes foram "costurados" às práticas Scrum. Assim, o Scrum passa a atender, parcialmente, a mais áreas de processo do CMMI, como apresentado na Tabela 3. Ou seja, práticas das áreas de processo Gestão de Configuração (*Configuration Management*), Integração de Produto (*Product Integration*), Solução Técnica (*Technical Solution*) e Treinamento Organizacional (*Organizational Training*) podem agora ser também atendidas.

SPEM (*Software Process Engineering Metamodel*) [24] é um meta-modelo para descrever processos de software, e está sendo utilizado para modelar a integração entre Scrum e padrões organizacionais e de processo para possibilitar o uso correto da integração aqui proposta [14].

6. AGRADECIMENTOS

Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. Agile Software Development Methods: Reviews and Analysis. Espoo: VTT Publications, 2002. Disponível em: <http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf>. Acesso em: 11 jan. 2006.

[2] Appleton, B. Patterns and Software: Essential Concepts and Terminology. 1997. Disponível em: <http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html>. Acesso em: 10 jan. 2006.

[3] Beck, K.; et al. Manifesto for Agile Software Development. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 15 dez. 2005.

[4] Beck, K; Andres, C. Extreme Programming Explained: Embrace Change. 2. ed. Addison-Wesley, 2004

[5] Berczuk, S.; Appleton, B. Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Addison-Wesley, 2002.

[6] Bramble, P.; Cockburn, A.; Pols, A.; Adolph, S. Patterns for Effective Use Cases. Reading, MA: Addison-Wesley, 2002.

[7] Cockburn, A. Agile software development. Boston: Addison Wesley, 2002.

[8] Cockburn, A. Surviving Object-Oriented Projects: A Manager's Guide. Addison-Wesley, 1998.

[9] Cockburn, A. Writing Effective Use Cases (TheCrystal Collection for Software Professionals). Reading, MA: Addison-Wesley, 2000.

[10] Coplien, J. O. A Generative Development-Process Pattern Language. In: Coplien, J.; Schmidt, D. Pattern Languages of Program Design. USA: Addison-Wesley, 1995.

[11] Coplien, J. O.; Harrison N. B. Organizational Patterns of Agile Software Development. 1. ed. Prentice Hall, 2004.

[12] Coram, T. Demo Prep: A Pattern Language for the Preparation of Software Demonstrations. In: Vlissides, J.; Coplien, J.; Kerth, N. Pattern Languages of Program Design 2. Addison-Wesley, 1996.

[13] Costa Filho, E. G.; Penteado, R.; Braga, R. T.; Silva, J. C. Padrões e Métodos Ágeis: agilidade no processo de desenvolvimento de software. In: Proceedings of 5ª Conferência Latino-Americana em Linguagens de Padrões para Programação, SugarLoafPlop Agosto, 2005, Campos do Jordão, Brasil, p. 156-169.

[14] Costa Filho, E. G. Integração de Padrões Organizacionais e de Processo ao Método Ágil Scrum. Dissertação de mestrado do Programa de Pós Graduação em Ciência da Computação UFSCar, a ser apresentada em maio de 2006.

[15] Cunningham, W; Kerth, H. Using Patterns to Improve our Architectural Vision. IEEE Software, v.14, n. 1, p. 53-59, 1997.

[16] Fuggetta, A. Software Process: A Roadmap. In: Finkelstein, A. (ed.). The Future of Software Engineering. ACM Press, 2000.

[17] Glazer, H. Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativit". CrossTalk, Novembro 2001, p. 27-30.

[18] Hagge, L.; Lappe, K. Sharing Requirements Engineering Experience Using Patterns. IEEE Software, v. 22, n. 1, p.24-31, 2005.

[19] Harrison, N. Organizational Patterns for Teams. In: Vlissides, J.; Coplien, J.; Kerth, N. Pattern Languages of Program Design 2. Addison-Wesley, 1996.

[20] Highsmith, J.; Cockburn, A. Agile Software Development: The Business of Innovation. Computer, v. 34, p. 120-127, 2001.

[21] Kane, D.; Ornburn, S. Agile Development: Weed or Wildflower? Prentice Hall, 2002. Disponível em: <http://www.informit.com/articles/article.asp?p=29029>. Acesso em: 10 jan. 2006.

[22] Larman, C. Agile and Iterative Development: A Manager's Guide. Addison Wesley, 2003.

[23] Lycett, M.; Macredie, R. D.; Patel, C.; Paul, R. J. Migrating Agile Methods to Standardized Development Practice. Computer, v. 36, n. 6, p. 79-85, 2003.

[24] OMG - Object Management Group. Software Process Engineering Metamodel Specification, January, 2005.

[25] Paulk, M. C.; Curtis, B.; Chrissis, M. B.; Weber, C. Capability Maturity Model for Software, Version 1.1. Software Engineering Institute, 1993.

[26] Paulk, M. Extreme Programming from a CMM Perspective. IEEE Software, v. 18, n. 6, p. 19-26, 2001.

[27] Schwaber, K. Agile Project Management With Scrum. Microsoft Press, 2004

[28] Schwaber, K.; Beedle, M. Agile Software Development with SCRUM. Prentice-Hall, 2002.

[29] SEI CMMI Product Team. CMMI For Systems Engineering and Software Engineering. Version 1.1, Continuous Representation. Software Engineering Institute, 2002.

[30] Turner, R.; Clouse, A.; Ahern, D. CMMI Distilled: A Practical Introduction to Integrated Process Improvement. Addison Wesley, 2003.

[31] Turner, R.; Jain, A. Agile Meets CMMI: Culture Clash Or Common Cause? Proceedings of Extreme Programming and Agile Methods - XP/Agile Universe 2002, Chicago, I.L., Agosto 2002, p. 153-165.

[32] Zanatta, A.; Vilain, P. Uma análise do método ágil Scrum conforme abordagem nas áreas de processo Gerenciamento e Desenvolvimento de Requisitos do CMMI. Anais do Workshop em Engenharia de Requisitos. Junho, 2005, Porto, Portugal. p 209-220.