Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

“Integração de Padrões Organizacionais e de Processo ao Método Ágil Scrum”

Edes Garcia da Costa Filho

São Carlos - SP

Maio/2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C837ip

Costa Filho, Edes Garcia da.
Integração de padrões organizacionais e de processo ao método ágil Scrum / Edes Garcia da Costa Filho. -- São Carlos : UFSCar, 2006.
119 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Engenharia de software. 2. Padrões de software. 3. Melhoria de processo. 4. Linguagem de padrões. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Integração de padrões organizacionais e de processo ao método ágil Scrum"

EDES GARCIA DA COSTA FILHO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Profa. Dra. Rosângela Ap. Dellosso Penteado
(Orientadora – DC/UFSCar)

Profa. Dra. Júnia Coutinho Anacleto
(Co-Orientadora – DC/UFSCar)

Profa. Dra. Rosana Teresinha Vaccare Braga
(ICMC/USP)

Profa. Dra. Claudia Maria Lima Werner
(UFRJ)

São Carlos
Maio/2006

Dedico este trabalho especialmente ao meu pai,

Edes Garcia da Costa,

meu mestre e eterno amigo.

Agradecimentos

A Deus por me guiar em todos os dias de minha vida e por permitir alcançar mais um sonho.

A Profa. Dra. Rosângela Aparecida Delloso Penteado por possibilitar o meu retorno para universidade, pela orientação constante, companheirismo e confiança.

As professoras Dra. Júnia Coutinho Anacleto e Dra. Rosana Teresinha Vaccare Braga pela contribuição no trabalho.

Ao Prof. Dr. Fernão Stella R. Germano pelos ensinamentos e orientação.

A minha mãe Aparecida pelo amor, compreensão, incentivo incondicional e por me proporcionar uma vida maravilhosa.

Aos meus irmãos Elene e Erick, minhas fontes de inspiração, que souberam entender os motivos pelos quais muitas vezes estive ausente.

A Roberta, mulher da minha vida, pela paciência, dedicação, ajuda nos momentos difíceis e pelo amor dedicado a mim.

Aos meus avós pelo amor e ajuda nos momentos difíceis.

Ao meu amigo Richard pelo apoio durante este trabalho.

Aos amigos presentes na minha caminhada: Tio Eduardo, Tia Cristina, Diego, Luisão, Bárbara, Caio, Renan, Paulinho, Marcus, Amilton, Guilherme, Frederico, Rodrigo, Eros, Gabriel, Alexandre, Alessandro, Mariza, Leslei, Roberto, Ana Paula, Leda, Fabiano e Lukinha.

Aos alunos de Graduação e Pós-Graduação pela participação nas minhas pesquisas.

As secretárias da Pós-Graduação Cristina e Mirian.

Ao CNPq pelo apoio financeiro.

E a todos que de forma direta ou indireta contribuíram para a realização deste trabalho.

Sumário

Capítulo 1 ..... **1**
Introdução 1
1.1. Considerações Iniciais 1
1.2. Motivação 2
1.3. Objetivos 2
1.4. Organização da Dissertação 3
Capítulo 2 ..... **4**
Assuntos Relacionados 4
2.1. Considerações Iniciais 4
2.2. Processo de Software 5
2.2.1 Processo Unificado (PU) 6
2.3. Métodos Ágeis 8
2.3.1. Scrum 10
2.3.2. Outros Métodos Ágeis 14
2.4. Padrões e Linguagens de Padrões 16
2.4.1. Padrões Organizacionais e de Processo 19
2.4.2. Linguagens de Padrões Organizacionais e de Processo 20
2.5. Considerações Finais 29
Capítulo 3 ..... **32**
Experiência de Uso dos Padrões Organizacionais e de Processo com Scrum 32
3.1. Considerações Iniciais 32
3.2. Descrição da Experiência 32
3.3. Planejamento da Experiência 34
3.4. Condução da Experiência 39
3.4.1. Primeira Etapa da Experiência 39
3.4.2. Segunda Etapa da Experiência 41
3.5. Análise dos Resultados da Experiência 43
3.5.1. Sistema de Clínica Veterinária 43
3.5.2. Sistema de Controle de Equivalências 44
3.5.3. Opinião dos Membros das Organizações 47
3.6. Considerações Finais 47
Capítulo 4 ..... **49**
Uma Forma Ordenada para Utilização dos Padrões Organizacionais e de Processo com Scrum 49
4.1. Considerações Iniciais 49
4.2. Modelagem de Processo 50
4.2.1. SPEM - Software Process Engineering Metamodel 51
4.3. Modelagem do Scrum sem Padrões 54
4.3.1. Disciplina Definir Papéis 57
4.3.2. Disciplina Obter Requisitos 58
4.3.3. Disciplina Planejar Iteração 59
4.3.4. Disciplina Realizar Iteração 61
4.3.5. Disciplina Revisar Iteração 64
4.3.6. Disciplina Finalizar Versão 66
4.4. Identificação da Categoria dos Padrões 68
4.5. Associação entre os Padrões e Disciplinas do Scrum 70
4.6. Integração dos Padrões Organizacionais e de Processo para Extensão do Scrum. 72

4.6.1. Disciplina Definir Papéis (Estendida) 73
4.6.2. Disciplina Obter Requisitos (Estendida) 74
4.6.3. Disciplina Planejar Iteração (Estendida) 77
4.6.4. Disciplina Realizar Iteração (Estendida) 82
4.6.5. Disciplina Finalizar Versão (Estendida) 85
4.7. Considerações Finais 87
Capítulo 5 90
Estudo Piloto de Avaliação do Scrum Estendido 90
5.1. Considerações Iniciais 90
5.2. Descrição do Estudo Piloto 90
5.3. Sistema de Controle de Zoológico 91
5.3.1. Análise do Estudo Piloto 92
5.4. Sistema de Controle de Equivalências 96
5.4.1. Análise do Estudo Piloto 96
5.5. Análise dos Questionários Aplicados às Organizações 99
5.5.1. Análise dos Padrões Integrados ao Scrum 99
5.5.2. Análise do Scrum Estendido e da sua Modelagem com SPEM 101
5.5.3. Problemas Observados na Aplicação do Scrum Estendido 104
5.6. Considerações Finais 105
Capítulo 6 108
Conclusões 108
6.1. Considerações Iniciais 108
6.2. Resultados 108
6.3. Contribuições deste Trabalho 110
6.4. Limitação deste Trabalho 111
6.5. Trabalhos Futuros 111
Referências Bibliográficas 112
Apêndice A - Questionário 118

Lista de Figuras

Figura 2.1 – Fases e iterações do PU (adaptada de Larman, 2002) 6
Figura 2.2 – Estrutura do RUP (Rational Software Corporation, 2002) 7
Figura 2.3 – Evolução do modelo cascata para o XP e Scrum (adaptada de Beck, 1999a) 9
Figura 2.4 – Visão geral do processo Scrum (Adaptada de Schwaber, 2004) 12
Figura 2.5 – Exemplo de lista de Trabalho do Produto 13
Figura 2.6 – Exemplo de lista de Trabalho da Iteração 13
Figura 2.7 – Exemplo de gráfico de progresso 13
Figura 2.8 – Linguagem de padrões (Adaptada de Coplien e Harrison, 2004) 17
Figura 2.9 – Etapas de desenvolvimento de software apoiadas pelo Scrum (Adaptada de Abrahamsson et al., 2002) 30
Figura 3.1 – Tempo de desenvolvimento do sistema de Clinica Veterinária 43
Figura 3.2 – Tempo gasto com documentação pela Organização A 46
Figura 4.1 – Níveis de modelagem (adaptada de OMG, 2005) 51
Figura 4.2 – SPEM como UML *profile* 52
Figura 4.3 – Modelo conceitual SPEM (Adaptado de OMG, 2005) 52
Figura 4.4 – Meta-modelo Scrum 54
Figura 4.5 – Diagrama de pacotes do processo Scrum 55
Figura 4.6 – Diagrama de atividades do processo Scrum 56
Figura 4.7 – Diagrama de classes da disciplina Definir Papéis 57
Figura 4.8 – Diagrama de atividades da disciplina Definir Papéis 57
Figura 4.9 – Diagrama de classes da disciplina Obter Requisitos 58
Figura 4.10 – Diagrama de atividades da disciplina Obter Requisitos 59
Figura 4.11 – Diagrama de classes da disciplina Planejar Iteração 60
Figura 4.12 – Diagrama de atividades da disciplina Planejar Iteração 61
Figura 4.13 – Diagrama de classes da disciplina Realizar Iteração 63
Figura 4.14 – Diagrama de atividades da disciplina Realizar Iteração 64
Figura 4.15 – Diagrama de classes da disciplina Revisar Iteração 65
Figura 4.16 – Diagrama de atividades da disciplina Revisar Iteração 66
Figura 4.17 – Diagrama de classes da disciplina Finalizar Versão 67
Figura 4.18 – Diagrama de atividades da disciplina Finalizar Versão 67
Figura 4.19 – Estereótipos utilizados para modelar padrões de processo 70
Figura 4.20 – Notação do estereótipo Padrão Organizacional 70
Figura 4.21 – Diagrama de Classes disciplina Definir Papéis (estendida) 74
Figura 4.22 – Diagrama de classes da disciplina Obter Requisitos (estendida) 76
Figura 4.23 – Diagrama de atividades da disciplina Obter Requisitos (estendida) 77
Figura 4.24 – Diagrama de classes da disciplina Planejar Iteração (estendida) 80
Figura 4.25 – Diagrama de atividades da disciplina Planejar Iteração (estendida) 81
Figura 4.26 – Diagrama de classes da disciplina Realizar Iteração (estendida) 83
Figura 4.27 – Diagrama de atividades da disciplina Realizar Iteração (estendida) 84
Figura 4.28 – Diagrama de classes da disciplina Finalizar Versão (estendida) 85
Figura 4.29 – Diagrama de atividades da disciplina Finalizar Versão (estendida) 86
Figura 4.30 – Artefatos que compõem a documentação do projeto 86
Figura 5.1 – Gráfico de progresso da Equipe Scrum da Organização A 93
Figura 5.2 – Gráfico ideal de progresso 94
Figura 5.3 – Gráfico de progresso de um membro da Equipe Scrum da Organização A 95
Figura 5.4 – Gráfico de progresso de outro membro da Equipe Scrum da Organização A 95
Figura 5.5 – Gráfico de progresso da Equipe Scrum da Organização B 97

Figura 5.6 – Gráfico de progresso de um membro da Equipe Scrum da Organização B 97
Figura 5.7 – Gráfico de progresso de outro membro da Equipe Scrum da Organização B 98
Figura 5.8 – Tempo de desenvolvimento do sistema de Controle de Equivalências 99
Figura 5.9 – Qualidade da comunicação no estudo piloto 101
Figura 5.10 – Qualidade da colaboração no estudo piloto 102
Figura 5.11 – Facilidade de compreensão do processo Scrum estendido 102
Figura 5.12 – Visibilidade do processo Scrum estendido 103
Figura 5.13 – Facilidade de manutenção do processo Scrum estendido 103

Lista de Tabelas

Tabela 2.1 – Papéis do Scrum 10
Tabela 2.2 – Práticas do Scrum 11
Tabela 2.3 – Formatos de apresentação dos padrões 19
Tabela 2.4 – Padrões organizacionais (Coplien e Harrison, 2004) 22
Tabela 2.5 – Padrões para organização de equipes (Harrison, 1996) 24
Tabela 2.6 – Padrões para criação casos de uso efetivos (Bramble et al., 2002) 25
Tabela 2.7 – Padrões para construção e demonstração de protótipos (Coram, 1996) 25
Tabela 2.8 – Padrões para testes de sistemas (DeLano e Rising, 1998) 26
Tabela 2.9 – Padrões para gestão de configuração de software (Berczuk e Appleton, 2002) 27
Tabela 2.10 – Padrões para introduzir novas idéias em organizações (Manns e Rising, 2004) 28
Tabela 2.11 – Pontos positivos e negativos de alguns métodos ágeis (adaptada de Abrahamsson et al., 2002) 29
Tabela 2.12 – Linguagens de padrões organizacionais e de processo 31
Tabela 3.1 – Padrões organizacionais e de processo relacionados às práticas e papéis Scrum 34
Tabela 3.2 – Mapeamento dos padrões relacionados a práticas e papéis do Scrum 37
Tabela 3.3 – Padrões utilizados para planejar e conduzir a experiência 38
Tabela 3.4 – Problemas observados nas organizações e os padrões selecionados para resolvê-los 40
Tabela 3.5 – Padrões selecionados pelas organizações 42
Tabela 4.1 – Estereótipos definidos no SPEM (OMG, 2005) 53
Tabela 4.2 – Categoria dos padrões integrados ao Scrum 68
Tabela 4.3 – Disciplina dos padrões selecionados pelas organizações 71
Tabela 4.4 – Padrões integrados a disciplina Definir Papéis 74
Tabela 4.5 – Padrões integrados a disciplina Obter Requisitos 75
Tabela 4.6 – Padrões integrados a disciplina Planeja Iteração 78
Tabela 4.7 – Padrões integrados a disciplina Realizar Iteração 82
Tabela 5.1 – Vantagens da integração dos padrões organizacionais e de processo 99
Tabela 5.2 – Resultados obtidos sobre a modelagem do Scrum estendido 106
Tabela 5.3 – Resultados obtidos sobre a modelagem do Scrum estendido 107

Resumo

Devido à importância cada vez maior do software na sociedade, a área de Engenharia de Software vem se esforçando ao longo dos anos para melhorar o processo de desenvolvimento de produtos de software. Hoje, mesmo com o avanço da tecnologia, as organizações enfrentam problemas para construir software nos prazos e custos estabelecidos. Os padrões organizacionais e de processo fornecem soluções comprovadas para problemas recorrentes no desenvolvimento de software. Essas categorias de padrões possibilitam o reúso em níveis mais altos de abstração, ou seja, em nível organizacional e de processo. Porém, além de melhorar e organizar o processo de desenvolvimento de software, as organizações precisam acelerar esse processo para atender às demandas de mercado. Métodos ágeis de desenvolvimento como Scrum e Extreme Programming (XP) estão sendo cada vez mais utilizados, devido à sua proposta de liberar software com qualidade em prazos menores. Nesse contexto, este trabalho propõe uma forma ordenada para utilizar padrões organizacionais e de processo em conjunto com o método ágil Scrum, para possibilitar sua extensão ou adaptação, de acordo com as necessidades da organização que o utiliza. Com base em uma experiência de uso e com a utilização do meta-modelo SPEM (*Software Process Engineering Metamodel*) as etapas de integração dos padrões organizacionais e de processo com o Scrum foram elaboradas e utilizadas para criar uma extensão desse método. O Scrum, como outros métodos ágeis, apresenta alguns pontos fracos que necessitam de alternativas para melhorá-los. Assim, por meio das etapas elaboradas, alguns padrões organizacionais e de processo foram integrados ao Scrum para tratar as questões não abordadas por ele. O resultado de um estudo piloto de avaliação conduzido para verificar as vantagens dessa integração e da modelagem realizada com SPEM indicou a conveniência de fazê-los. Após a aplicação das etapas de integração com alguns padrões e o Scrum, notou-se a possibilidade da criação de diretrizes de integração de padrões organizacionais e de processo a outros métodos ágeis.

Abstract

Due to the increasing importance of software in society, the software engineering area has devoted continuous effort to improve the software products development process. Today, even with the technological progress, organizations are facing problems to build software within establishing time and costs. Organization and process patterns supply proven solutions to recurring software development process. These pattern categories allow reuse in higher abstract levels, that is, in organizational and process levels. However, else than improve and organize the software development process, organizations must speed up that process to attend market demands. Agile development methods such as Scrum and Extreme Programming (XP) are being often utilized due to the proposal of delivering quality software in less time. In that context, this paper proposes an organized form to use organizational and process patterns together with the Scrum agile method, to allow its extension or adaptation according to the needs of the organization that will use it. Based in use experience and using the SPEM metamodel, phases of integration of organizational and process patterns to Scrum have been proposed and used to create a Scrum extension. Scrum, as other agile methods, presents some weak points that need alternatives to improve them. So, using the phases devised, some organization and process patterns have been integrated to Scrum in order to deal with questions not approached by it. The result of a pilot evaluation study conducted to verify the advantages of that integration and of the SPEM modeling, has indicated the convenience to do them. After the application of the integration phases with some patterns and Scrum, the possibility of guidelines creation to integrate organizational and process patterns to other agile methods has been noted.

Capítulo 1

Introdução

1.1. Considerações Iniciais

Com o aumento da demanda por produtos de software, existe necessidade cada vez maior de que esses sejam liberados de forma rápida, porém com qualidade. Mesmo com a evolução de técnicas, ferramentas, métodos e processos, a entrega de software em prazo e custos estabelecidos nem sempre é conseguida.

Os métodos ágeis de desenvolvimento vêm ganhando atenção na indústria de software, devido à sua proposta de liberar o produto de forma mais rápida do que os modelos prescritivos de processo (Pressman, 2005). Diferentemente desses modelos, que tentam planejar tudo no início do desenvolvimento para controlar possíveis modificações de requisitos, os métodos ágeis defendem o planejamento contínuo, para que adaptações a essas modificações possam ocorrer. Nos últimos anos, métodos ágeis como Extreme Programming (XP) (Beck e Andres, 2004), Scrum (Schwaber, 2004) e Crystal (Cockburn, 2002) passaram a ser amplamente utilizados.

Porém, a utilização de um método ágil pode não ser suficiente para atender às necessidades específicas de um projeto ou organização, pois apesar de compartilharem os mesmos princípios e valores (Beck et al., 2001), possuem características próprias com seus pontos fortes e fracos (Abrahamsson et al., 2002) e, assim, adaptações ou extensões podem ser necessárias.

Dentre os métodos ágeis existentes, Scrum é o mais flexível, pois define um conjunto de práticas de gerenciamento e permite que a organização utilize as práticas de engenharia que julgar mais convenientes.

Métodos ágeis como o Scrum e XP vêm sendo refinados ao longo dos anos. Em sua última publicação, Beck e Andres (2004) apresentam as novas práticas que foram incorporadas ao XP. Da mesma forma, Schwaber (2004) apresenta uma nova prática que foi introduzida no Scrum, a Reunião de Retrospectiva de Iteração (*Sprint Retrospective Meeting*).

Padrões de software (*software patterns*) fornecem soluções para problemas recorrentes em um determinado contexto (Coplien e Harrison, 2004). Dentre as várias categorias de padrões existentes, os padrões organizacionais e de processo documentam soluções comprovadas para problemas no desenvolvimento de software e, assim, nada impede que esses padrões sejam utilizados para estender ou adaptar métodos ágeis de acordo com as necessidades de um projeto ou organização, como é feito neste trabalho.

Padrões normalmente formam uma linguagem de padrões, que é um sistema de padrões organizados em uma estrutura que guia a sua aplicação (Cunningham et al., 1997). Desde o surgimento da primeira linguagem de padrões organizacionais e de processo (Coplien, 1995), várias outras surgiram (Harrison, 1996; Berczuk e Appleton, 2002; Bramble et al., 2002; Coplien e Harrison, 2004; Manns e Rising, 2004).

1.2. Motivação

Com base nesse contexto, a motivação para a realização deste trabalho pode ser sintetizada nos itens seguintes:

- O Scrum é um método ágil flexível, que pode ser adaptado ou estendido de acordo com as necessidades de um projeto ou organização;
- Embora o Scrum seja amplamente utilizado, é fato que, como outros métodos ágeis, apresenta alguns pontos fracos (Abrahamsson et al., 2002) que necessitam de alternativas para melhorá-los;
- Padrões organizacionais e de processo documentam soluções comprovadas para problemas no desenvolvimento de software que podem ser utilizadas por quaisquer organizações;
- Existe hoje grande variedade de linguagens de padrões organizacionais e de processo disponíveis na literatura, que tratam de vários problemas de desenvolvimento;
- Projetos e organizações podem ter características particulares e, assim, adaptações no Scrum podem ser necessárias para atendê-las;
- A demanda cada vez maior por produtos de software requer que esses sejam liberados de forma mais rápida e com qualidade;

1.3. Objetivos

Dada a importância de melhorar constantemente o processo de desenvolvimento de software para que produtos de qualidade sejam entregues em prazo e custo estabelecidos, este

trabalho procura uma forma de racionalizar a utilização conjunta do método ágil Scrum com padrões organizacionais e de processo, para facilitar o reúso (Houaiss, 2002) desses padrões em um contexto ágil.

Para que as práticas propostas pelos padrões possam ser utilizadas de forma efetiva para adaptar ou estender o Scrum, é essencial entender como elas se relacionam com as práticas desse método, de forma que a agilidade não seja afetada. Assim, é necessária a utilização de um meta-modelo para descrição de processo, como por exemplo, o SPEM (*Software Process Engineering Metamodel*) (OMG, 2005).

1.4. Organização da Dissertação

Este trabalho está organizado em seis capítulos, sendo que este apresenta o contexto no qual a proposta de mestrado está inserida, bem como os objetivos e a motivação para o seu desenvolvimento.

O Capítulo 2 apresenta o levantamento bibliográfico considerando os assuntos relevantes para este trabalho. São feitas considerações sobre métodos ágeis, com destaque para o Scrum e também são discutidos os conceitos de padrões organizacionais e de processo e algumas linguagens de padrões organizacionais e de processo.

O Capítulo 3 apresenta uma experiência de uso dos padrões organizacionais e de processo com o Scrum. Nela foi observada a necessidade da elaboração de uma forma ordenada para racionalizar essa utilização conjunta.

O Capítulo 4 mostra a racionalização da utilização dos padrões organizacionais e de processo com Scrum, usando o meta-modelo SPEM, que possibilitou a criação de uma extensão do Scrum e a proposta de diretrizes que podem ser usadas para estender outros métodos ágeis.

No Capítulo 5, o resultado da aplicação do Scrum estendido em duas organizações fictícias é apresentado. Além disso, esse capítulo também apresenta análises realizadas com base em questionários aplicados a essas organizações para verificar a conveniência da integração dos padrões ao Scrum e as vantagens da modelagem realizada com SPEM.

No Capítulo 6 encontram-se as conclusões que comentam os resultados obtidos, as contribuições e limitação deste trabalho, além de sugestões para trabalhos futuros.

Capítulo 2

Assuntos Relacionados

2.1. Considerações Iniciais

A área de Engenharia de Software busca constantemente melhorar o processo de desenvolvimento de software. Para isso, ao longo dos últimos 30 anos, técnicas, ferramentas, métodos e modelos para tratar e controlar o desenvolvimento de software têm sido criados. Esse esforço se deve ao fato de que a qualidade do software desenvolvido depende diretamente da qualidade do processo de desenvolvimento utilizado para construí-lo. Assim, melhorar a qualidade do processo significa melhorar a qualidade do software.

Para satisfazer às pressões do mercado, além de qualidade, as organizações precisam cada vez mais de agilidade no processo de desenvolvimento. Os métodos ágeis de desenvolvimento têm por objetivo melhorar e agilizar a produção do software. Já os padrões organizacionais e de processo fornecem soluções comprovadas para o desenvolvimento de software e, se utilizados em conjunto com um método ágil, podem agregar qualidade ao processo de desenvolvimento e, consequentemente, ao produto construído.

A necessidade de melhorar e acelerar o processo de desenvolvimento de software motivou esta pesquisa sobre a utilização de padrões organizacionais e de processo para adaptar ou estender métodos ágeis. Assim, neste capítulo são apresentados conceitos de processo de software, métodos ágeis e padrões, com enfoque nos organizacionais e de processo, evidenciando alguns padrões utilizados para estender o método ágil Scrum.

Este capítulo está organizado da seguinte forma: a Seção 2.2 apresenta conceitos de processos de software; a Seção 2.3 aborda conceitos sobre métodos ágeis, com destaque para o Scrum; a Seção 2.4 apresenta os conceitos de padrões e linguagens de padrões, juntamente com alguns exemplos de linguagens de padrões organizacionais e de processo e, na Seção 2.5, estão as considerações finais.

2.2. Processo de Software

Problemas de qualidade, produtividade, prazos, custos e manutenção, que afligem a comunidade de software até hoje, tornaram evidente a necessidade de se planejar e gerenciar o processo de desenvolvimento de software. Deste modo, a Engenharia de Software surgiu com objetivo de melhorar o processo de desenvolvimento de software, e conseqüentemente, aumentar a qualidade do produto de software produzido, melhorar as previsões de prazos e custos e diminuir os custos de manutenção.

O IEEE (*Institute of Electrical and Electronic Engineers*) (1990) define processo como uma seqüência de passos realizados para um determinado propósito. Na Engenharia de Software, processo é um conjunto de atividades, práticas, métodos e tecnologias utilizadas para desenvolver e manter um software de qualidade. Processo de software pode ser compreendido como um conjunto de atividades e resultados associados que geram um produto de software Sommerville (2003).

Modelos prescritivos de processo definem um conjunto distinto de atividades, tarefas, marcos e produtos de trabalho que são necessários para fazer engenharia de software com alta qualidade (Pressman, 2005). Existem atividades comuns aos modelos prescritivos, que segundo Sommerville (2003) são:

- **Especificação de software**: é necessário definir a funcionalidade do software e as restrições em sua operação.
- **Projeto e implementação de software**: o software deve ser produzido de modo que cumpra sua especificação.
- **Validação de software**: o software precisa ser validado para garantir que ele faz o que o cliente deseja.
- **Evolução de software**: o software precisa evoluir para atender as necessidades mutáveis do cliente.

Esses modelos de processo não são perfeitos, mas procuram direcionar efetivamente o trabalho de engenharia de software. Como exemplos de processos prescritivos podem ser citados: Cascata (Royce, 1970), Espiral (Boehm, 1988), Processo Unificado (Larman, 2002), Processo Unificado Rational (Kruchten, 2003) e dentre os citados por Pressman (2005) estão também Prototipagem, Desenvolvimento Concorrente, Desenvolvimento Baseado em Componentes, Modelo de Métodos Formais e Desenvolvimento Orientado a Aspectos.

O Processo Unificado (PU) e o Processo Unificado Rational (RUP) são apresentados resumidamente a seguir.

2.2.1 Processo Unificado (PU)

Desenvolvido por Jacobson, Booch, e Rumbaugh, o Processo Unificado (PU) (*Unified Process*) (Jacobson et al., 1999) é um processo iterativo para desenvolvimento de sistemas orientados a objetos. O PU organiza o trabalho e as iterações em quatro fases principais. Essas fases são (Larman, 2002):

- **Concepção** (*Inception*): tem por objetivo desenvolver uma visão aproximada do sistema, criar o caso de negócio, definir o escopo e produzir um esboço de estimativas para custo e cronograma.
- **Elaboração** (*Elaboration*): tem por objetivo refinar a visão do sistema, descrever todos os requisitos do sistema, implementar a arquitetura e produzir estimativas realísticas para custo e cronograma.
- **Construção** (*Construction*): tem por objetivo desenvolver e integrar os componentes e as funções ao produto.
- **Transição** (*Transition*): tem por objetivo realizar testes e implantação.

O que difere o PU do modelo em cascata é que cada uma das suas quatro fases é composta por uma ou mais iterações, como mostra a Figura 2.1.

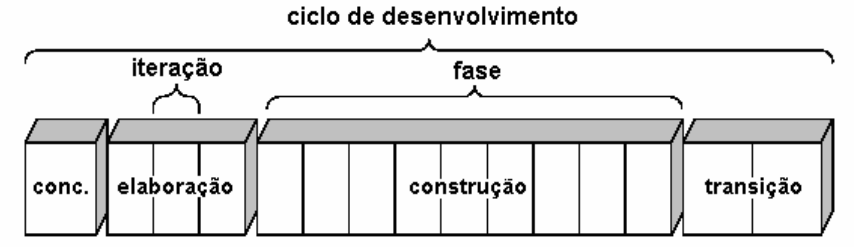


Figura 2.1 – Fases e iterações do PU (adaptada de Larman, 2002)

Uma iteração envolve conjuntos de atividades de trabalho, chamadas de disciplinas. Durante uma iteração, o trabalho ocorre na maior parte ou em todas essas disciplinas.

O PU foi refinado e detalhado pela Rational Corporation, que criou o Processo Unificado Rational (*Rational Unified Process* - RUP) (Kruchten, 2003) para complementar a UML (*Unified Modelling Language*) (Larman, 2002), que é uma linguagem de modelagem.

Kruchten (2003) define o RUP como um *framework* de processo que pode ser adaptado e estendido de acordo com as necessidades da organização. Segundo Kruchten (2003), o RUP captura as melhores práticas de desenvolvimento de software: desenvolvimento iterativo de software; gestão de requisitos; uso de arquiteturas baseadas em componentes; uso de software de

modelos visuais; verificação contínua da qualidade do software; controle de mudanças do software.

O RUP possui quatro fases compostas de uma ou mais iterações. Essas iterações são geralmente curtas, nas quais poucas funções do sistema são desenvolvidas. Cada uma dessas fases é concluída por um marco, que fornece pontos para que decisões críticas de projeto possam ser tomadas. Cada iteração trata de forma mais ou menos intensiva as disciplinas, que contém as atividades do processo.

O processo RUP é dividido em duas dimensões, como mostra a Figura 2.2. O eixo horizontal representa a estrutura dinâmica do processo, ou seja, o tempo e as perspectivas do modelo de processo à medida que se desenvolve. O eixo vertical representa a estrutura estática do processo e contém as disciplinas, que agrupam as atividades de maneira lógica.

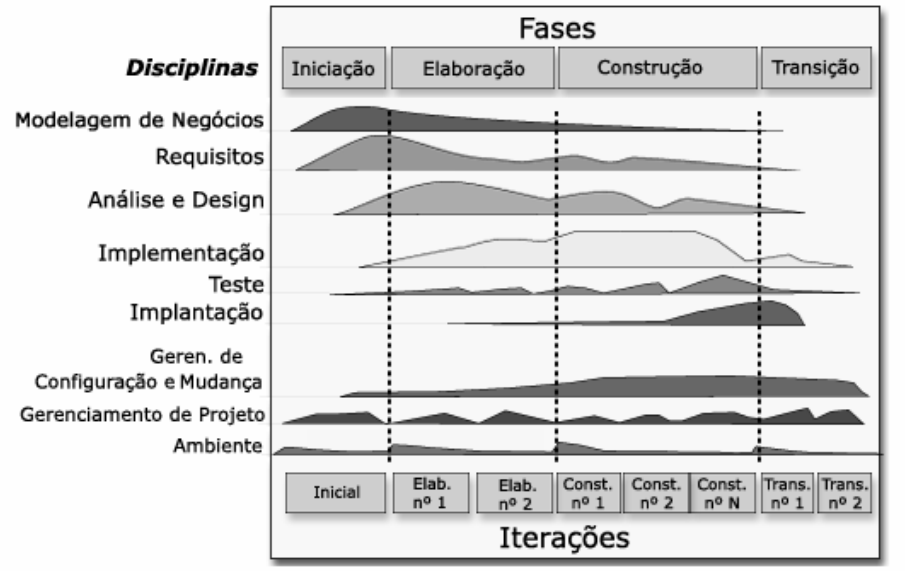


Figura 2.2 – Estrutura do RUP (Rational Software Corporation, 2002)

No eixo dinâmico do processo estão as fases do RUP, que são as mesmas do PU, enquanto no eixo estático estão as disciplinas, que são um dos cinco elementos principais do processo RUP. Os outros quatros elementos são: papéis, atividades, artefatos e fluxos de trabalho. A definição desses quatro elementos, de acordo com a Rational Software Corporation (2002) é:

- **Papel** (*Role*): define o comportamento e as responsabilidades de um indivíduo ou de um conjunto de indivíduos que trabalham juntos como uma equipe, no contexto de uma organização de engenharia de software.
- **Atividade** (*Activity*): é algo que um papel faz, produzindo um resultado significativo no contexto do projeto.

- **Artefato** (*Artifact*): é um produto de trabalho do processo; os artefatos são utilizados, modificados e produzidos na execução das atividades.
- **Fluxo de trabalho** (*Workflow*): é uma seqüência de atividades que produzem um resultado de valor observável.

As disciplinas são "recipientes" usados para organizar as atividades do processo (Kruchten, 2003). Existem nove disciplinas no RUP, que são divididas em seis técnicas e três de suporte. As disciplinas técnicas são: Modelagem de Negócios (*Business Modeling*), Requisitos (*Requirements*), Análise e Projeto (*Analysis and Design*), Implementação (*Implementation*), Teste (*Test*) e Implantação (*Deployment*). As disciplinas de suporte são: Gerenciamento de Configuração e de Modificação (*Configuration and Change Management*), Gerenciamento de Projeto (*Project Management*) e Ambiente (*Environment*).

A Seção 2.3 apresenta os métodos ágeis, que tratam o processo de desenvolvimento de software de forma diferente da dos modelos de processo prescritivos.

2.3. Métodos Ágeis

Para uma organização de desenvolvimento de software, agilidade é a habilidade de se adaptar e reagir prontamente e apropriadamente às modificações. Um método ágil é aquele que facilmente permite e dá suporte a esse tipo de adaptabilidade (Kruchten, 2001). Por esse motivo, existe hoje grande interesse em métodos ágeis de desenvolvimento.

Os modelos de processo de software prescritivos comentados anteriormente são burocráticos. Segundo Fowler (2003), a grande quantidade de atividades que devem ser realizadas nesses processos causa lentidão no desenvolvimento de produtos de software.

Os métodos ágeis abordam o processo de desenvolvimento de software de forma diferente dos modelos de processo prescritivos. A principal diferença está na forma como as modificações são tratadas durante o desenvolvimento do software. Os modelos de processo prescritivos adotam a estratégia de previsibilidade. Eles utilizam técnicas para compreender o domínio do problema e levantar todos os requisitos antes de iniciar o desenvolvimento. Depois de levantados os requisitos, é feito um planejamento para que as modificações possam ser controladas no decorrer do processo de desenvolvimento do software. Por outro lado, os métodos ágeis optam pela adaptabilidade. Os requisitos são levantados aos poucos e o planejamento é contínuo, para que a adaptação às mudanças possa ocorrer. Por ser um *framework* de processo, não existe consenso quanto à classificação do RUP como método ágil ou modelo prescritivo.

A Figura 2.3 retrata a evolução do modelo de processo em Cascata para o XP e Scrum, mostrando como os modelos prescritivos optam pela previsibilidade enquanto os métodos ágeis optam pela adaptabilidade.

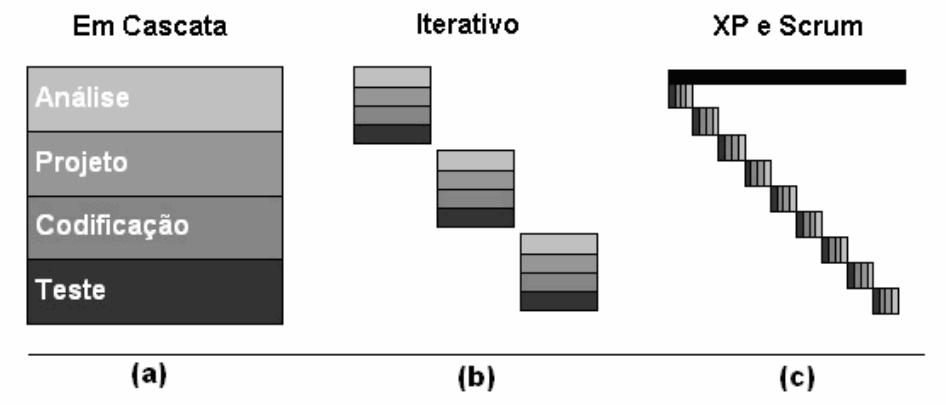


Figura 2.3 – Evolução do modelo cascata para o XP e Scrum (adaptada de Beck, 1999a)

A legenda a ser seguida nessa figura é a do modelo em Cascata (a) com as atividades diferenciadas pelas cores em tons de cinza, que mostra um ciclo de longa duração. Já em (b), os ciclos são mais curtos. A evolução do desenvolvimento iterativo para métodos ágeis pode ser observada no XP e no Scrum (c), que combinam as atividades de análise, projeto, codificação e teste em períodos de curta duração ao longo de todo projeto.

A popularização dos métodos ágeis ocorreu com a realização do "Manifesto Ágil" (Beck et al., 2001), realizado por um grupo de pesquisadores interessados em melhorar o desenvolvimento de software. Esse grupo definiu um conjunto de valores que serve como base para os métodos ágeis e que, segundo Cohn et al (2003), estabelecem uma estrutura comum para eles:

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

Esses valores enfatizam os aspectos humanos do desenvolvimento de software ao invés dos aspectos de Engenharia (Lycett et al., 2003). Cockburn e Highsmith (2001) afirmam que a maior dificuldade para os gerentes trabalharem de forma ágil é que muito mais ênfase é atribuída aos fatores humanos no projeto: cordialidade, talento, habilidade e comunicação. Segundo Highsmith e Cockburn (2001), o que existe de novo nos métodos ágeis não são as práticas que eles usam, mas o reconhecimento das pessoas como principais condutoras do sucesso do projeto.

Os métodos ágeis optam por uma documentação essencial para evitar redundâncias e excessos, que não auxiliam efetivamente no desenvolvimento do software.

Dentre os métodos ágeis pesquisados, o Scrum (Schwaber, 2004) é o que mais se relaciona com este trabalho, e será apresentado de forma mais detalhada na próxima subseção.

2.3.1. Scrum

O termo "Scrum", que surgiu de uma tática utilizada no jogo *rugby* (Schwaber e Beedle, 2002), descreve um tipo de processo de desenvolvimento de software inicialmente utilizado no Japão (Takeuchi e Nonaka, 1986).

O Scrum é iterativo, incremental e defende o uso de equipes pequenas, com até dez integrantes. Esse método ágil, que foi criado para gerenciar e controlar o processo de desenvolvimento de software em ambientes em que os requisitos estão em constante mudança, fornece um conjunto de práticas de gerenciamento, possibilitando à organização a escolha das práticas específicas de engenharia de software (Abrahamsson et al., 2002). Assim, o Scrum é um método ágil flexível que pode ser utilizado para envolver práticas de engenharia, possibilitando que as etapas de análise, projeto, codificação e teste sejam abordadas de acordo com as necessidades de cada organização.

Schwaber e Beedle (2002) identificam cinco papéis no Scrum, como mostra a Tabela 2.1. A primeira coluna apresenta o nome do papel e a segunda a sua descrição.

Tabela 2.1 – Papéis do Scrum

Papel	Descrição
Mestre Scrum (*Scrum Master*)	É o líder do projeto, responsável por garantir que as práticas Scrum sejam seguidas corretamente e por remover os impedimentos que dificultam o progresso do projeto, além de proteger os desenvolvedores de interrupções externas na etapa de desenvolvimento do software.
Proprietário do Produto (*Product Owner*)	Responsável por controlar a Lista de Trabalho do Produto (*Product Backlog*) e assegurar que ela esteja visível para todos. O Proprietário do Produto, que é selecionado pelo Mestre Scrum, Cliente e Gerência, é a pessoa responsável pelo projeto. Qualquer pessoa que queira acrescentar ou mudar a prioridade de um item da Lista de Trabalho do Produto tem que ter a aprovação do Proprietário do Produto.
Equipe Scrum (*Scrum Team*)	É a equipe de desenvolvimento que tem autoridade para tomar decisões e se auto-organizar para atingir os objetivos de cada iteração. A Equipe Scrum é responsável por desenvolver o software, além de participar da criação da Lista de Trabalho do Produto e estimativas de esforço.
Cliente (*Customer*)	Participa das tarefas relacionadas à Lista de Trabalho do Produto, como identificação e priorização dos requisitos que vão compor essa lista, além de participar da Reunião de Revisão de Iteração.
Gerência (*Management*)	Responsável por tomar as decisões críticas do projeto. A gerência participa da seleção do Proprietário do Produto e da Equipe Scrum.

O Scrum é baseado em um conjunto de valores: Comprometimento (*Commitment*), Foco (*Focus*), Sinceridade (*Openness*), Respeito (*Respect*) e Coragem (*Courage*). Os valores servem de base para as práticas de gerenciamento definidas no Scrum. As práticas Scrum são descritas, conforme Schwaber e Beedle (2002), na Tabela 2.2. A primeira coluna apresenta o nome da prática e a segunda a sua descrição.

Tabela 2.2 – Práticas do Scrum

Prática	Descrição
Lista de Trabalho do Produto (*Product Backlog*)	É uma lista priorizada e constantemente atualizada que contém os requisitos do sistema que está sendo construído ou melhorado. Essa lista contém todas as informações necessárias para construir o produto, como funções, otimizações e correções. Ela é dinâmica, ou seja, durante todo o projeto, itens podem ser adicionados, removidos e podem ter sua prioridades modificada.
Estimativa de Esforço (*Effort Estimation*)	É um processo iterativo no qual estima-se o esforço para desenvolver os itens da Lista de Trabalho do Produto. Essa estimativa se modifica à medida que novas informações surgem sobre o item em questão. O Proprietário do Produto trabalha com a Equipe Scrum para realizar estimativa de esforço.
Iteração (*Sprint*)	Período fixo de tempo (trinta dias), no qual a Equipe Scrum trabalha para atingir o objetivo estipulado para a iteração e produzir um incremento do produto de acordo com o que foi planejado.
Reunião de Planejamento de Iteração (*Sprint Planning Meeting*)	Reunião de planejamento de iteração, realizada pelo Proprietário do Produto, Equipe Scrum e Mestre Scrum, que consiste de duas etapas consecutivas. Na primeira, o objetivo da iteração deve ser oficializado e os itens da Lista de Trabalho do Produto que devem ser construídos na iteração são identificados. Na segunda, a Equipe Scrum trabalha para definir quais tarefas devem ser realizadas para atingir o objetivo da iteração. Essas tarefas são reunidas em uma lista chamada de Lista de Trabalho da Iteração.
Lista de Trabalho da Iteração (*Sprint Backlog*)	É uma lista com as tarefas que devem ser realizadas na iteração. Essa lista contém as partes detalhadas do trabalho necessário para converter os itens da Lista de Trabalho do Produto em um software.
Reuniões Diárias Scrum (*Daily Scrum Meetings*)	São reuniões de aproximadamente quinze minutos realizadas para verificar o progresso e identificar as dificuldades na realização do trabalho da Equipe Scrum. O Mestre Scrum é responsável por conduzir as Reuniões Diárias, identificar e remover os problemas encontrados pela Equipe Scrum para realizar seu trabalho. Apenas a Equipe Scrum e Mestre Scrum podem interagir nessa reunião. Qualquer outra pessoa interessada no projeto pode participar da reunião como observador. Nessa reunião os membros da Equipe Scrum devem responder a três questões: 1) O que você fez desde a última reunião diária?; 2) O que você fará até a próxima reunião diária?; 3) Quais foram os problemas encontrados para realizar o seu trabalho?
Reunião de Revisão de Iteração (*Sprint Review Meeting*)	É uma reunião realizada no final de uma iteração, na qual a Equipe Scrum apresenta para Gerência, Cliente e Proprietário do Produto o incremento do produto que foi construído durante a iteração. O Mestre Scrum é responsável por conduzir a Reunião de Revisão de Iteração. Essa reunião consta de perguntas, observações, discussões e sugestões.
Reunião de Retrospectiva de Iteração (*Sprint Retrospective Meeting*)	Reunião realizada pelo Mestre Scrum em conjunto com a Equipe Scrum e Proprietário do Produto, para discutir as seguintes questões: 1) O que ocorreu de bom na última iteração? 2) O que poderia ser melhorado para a próxima iteração?

A Reunião de Retrospectiva de Iteração é uma prática nova no Scrum, que foi introduzida por Schwaber (2004). Nessa reunião a Equipe Scrum discute as melhorias que podem ser realizadas para a próxima Iteração. Rising e Derby (2003) consideram a retrospectiva uma oportunidade de aprender com o passado. Kerth (2001) afirma que para uma retrospectiva ser efetiva, todos os participantes tem que se sentir seguros para discutir seu trabalho e admitir que podem existir melhores formas de realizá-lo. Para ele esse ambiente seguro só é construído se a seguinte regra primordial for seguida: "Independentemente do que for descoberto em uma retrospectiva, nós devemos entender e acreditar verdadeiramente que todos fizeram o melhor trabalho possível, dadas as informações conhecidas, experiências, habilidades e recursos disponíveis na situação".

As práticas do Scrum estão organizadas de forma coerente e definem um processo para gerenciamento e controle do desenvolvimento de software. A Figura 2.4 apresenta uma visão geral do processo Scrum, que tem início com a criação da lista de Trabalho do Produto, cujos itens são priorizados pelo cliente. A Figura 2.5 apresenta um exemplo de lista de Trabalho do Produto.

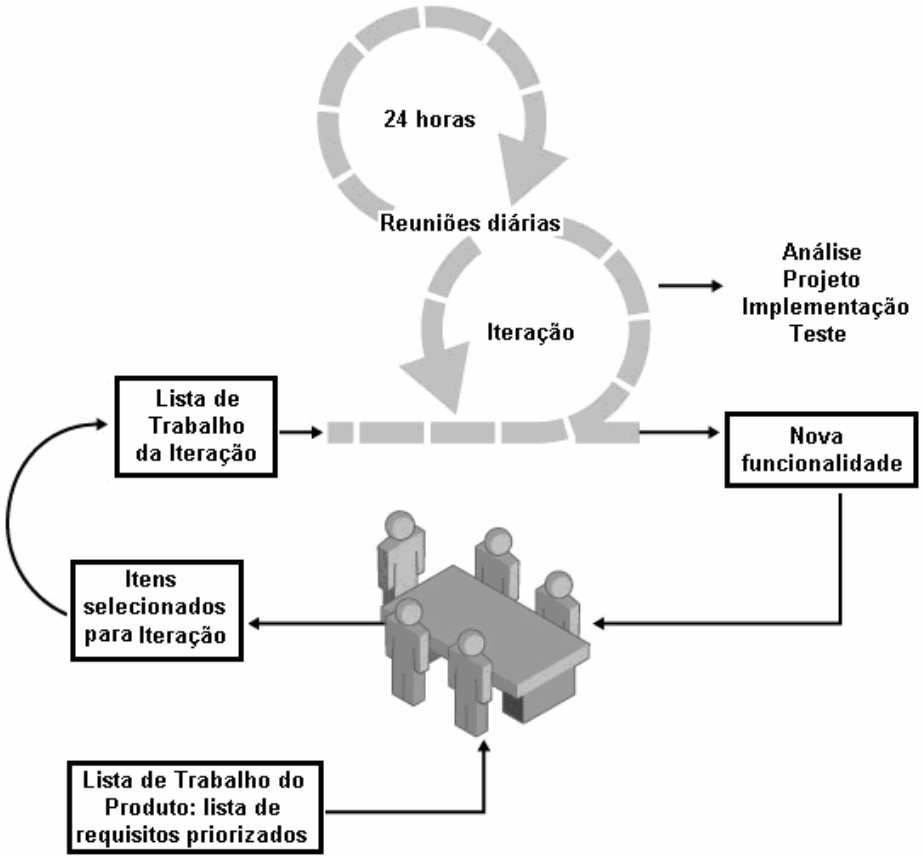


Figura 2.4 – Visão geral do processo Scrum (Adaptada de Schwaber, 2004)

Lista de Trabalho do Produto					
Iteração	ID		Item da lista de Trabalho do Produto	Proprietário	Estimativa (dias)
1	1	Cadastro	Cadastrar Funcionário	RIC	1
1	2	Web	Criar Área do Administrador	DEN	2
1	3	Banco de Dados	Banco de Dados	LUI	1
1	4	Interface	Criar Interface de Cadastro de Cliente	LUI	3

Figura 2.5 – Exemplo de lista de Trabalho do Produto

A lista de Trabalho da Iteração, também representada na Figura 2.4, nada mais é do que parte da lista de Trabalho do Produto com detalhes do trabalho a ser realizado. A Figura 2.6 apresenta um exemplo de lista de Trabalho da Iteração.

Iteração1								
16/1/2006				**Dia da iteração**	**16**	**17**	**18**	**19**
16/2/2006					**te**	**qu**	**qu**	**se**
				Horas Restantes	**24**	**24**	**24**	**24**
		Item da lista de Trabalho do Produto	**Proprietário**	**Estimativa**				
1	**Cadastro**	**Cadastrar Funcionário**	**RIC**	**8**				
		criar interface de cadastro		3	3	3	3	3
		desenvolver a função de cadastro		4	4	4	4	4
		testar cadastro		1	1	1	1	1
2	**Web**	**Criar Área do Administrador**	**DEN**	**16**				
		criar interface da área de administrador		8	8	8	8	8

Figura 2.6 – Exemplo de lista de Trabalho da Iteração

Além das listas de Trabalho do Produto e da Iteração, o Scrum sugere a criação de um Gráfico de Progresso de Iteração (*Sprint Burndown Chart*), que mostra a quantidade de trabalho remanescente ao longo de uma iteração, e é uma excelente forma de acompanhar o progresso da Equipe Scrum. A Figura 2.7 mostra um exemplo desse gráfico. No eixo horizontal estão os dias da iteração e no eixo vertical as horas que restam para desenvolver o incremento do sistema.

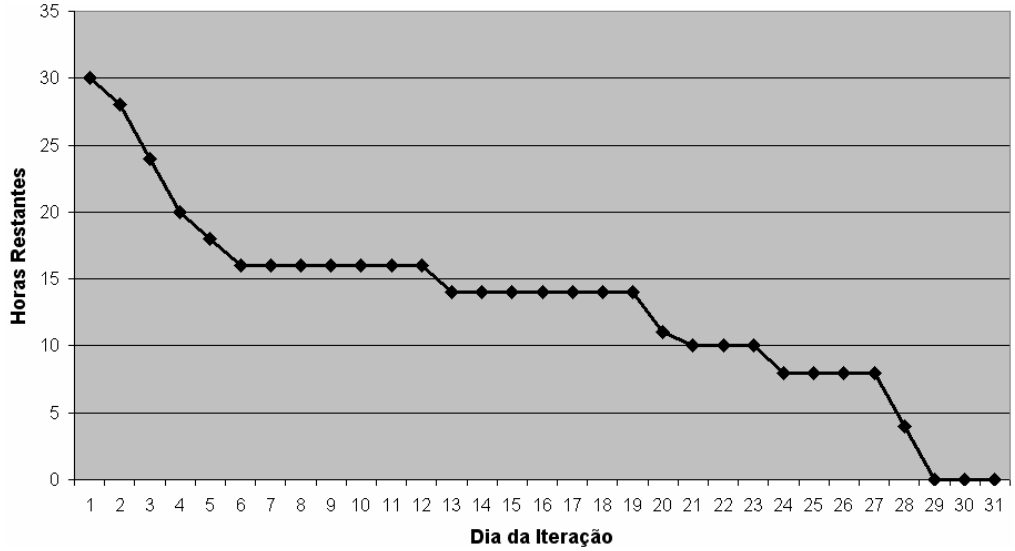


Figura 2.7 – Exemplo de gráfico de progresso

O Scrum não é contra documentação. Ele permite a criação e utilização de qualquer outro artefato que possa agregar valor ao projeto, porém, a discussão de outros produtos de trabalho além dos três definidos está fora do seu escopo (Larman, 2003).

Uma característica importante do Scrum é que suas práticas atendem a várias práticas propostas pelos modelos de melhoria de processo CMM (*Capability Maturity Model*) (Paulk et al., 1993) e CMMI (*Capability Maturity Model Integration*) (SEI, 2002) (Schwaber, 2004; Kane e Ornburn, 2002).

Neste trabalho, os nomes das práticas, papéis e artefatos do Scrum estão em português. A seguir são comentados outros métodos ágeis existentes, mas que não são objetivo de estudo deste trabalho.

2.3.2. Outros Métodos Ágeis

***Extreme Programming* (XP)**: criado por Beck (1999b), é o mais popular dos métodos ágeis. Originalmente, o XP definia um conjunto de doze práticas que foram escolhidas com base nos quatro valores: *comunicação, simplicidade, feedback* e *coragem*. Entretanto, Beck e Andres (2004) apresentam algumas alterações realizadas no XP. Além do novo valor respeito (*respect*), que foi acrescentado, práticas foram renomeadas e outras foram incorporadas. Outra mudança é que agora as práticas estão separadas em práticas primárias (*Primary Practices*) e práticas resultantes (*Corollary Practices*). As práticas no XP apóiam umas às outras e devem ser utilizadas em conjunto para se obter agilidade no processo. O ciclo de desenvolvimento XP inicia-se com o planejamento da primeira versão e das iterações que irão constituir essa versão. O cliente define e prioriza as funções que irão compor a primeira versão do software, por meio de cartões (estórias do usuário), e os desenvolvedores estimam o tempo para implementar cada estória. Após o planejamento da versão, a equipe planeja cada iteração, identificando quais estórias serão implementadas na próxima iteração. Em seguida, a equipe identifica as tarefas que devem ser realizadas para se desenvolver cada estória. Cada tarefa é implementada em duplas e, antes de cada tarefa ser implementada, devem ser criados os casos de teste para essas tarefas. Todos os dias uma reunião com toda equipe e o cliente é feita para discutir o que foi feito no dia e o que será realizado no outro. Novas iterações se seguem, e outras versões são geradas.

***Agile Modeling* (AM)**: introduzida por Ambler (2004), AM é uma nova abordagem para executar as atividades de modelagem. Ambler (2004) afirma que a modelagem é realizada para entender o que está sendo construído e para melhorar a comunicação dentro da equipe ou com os clientes do projeto. O objetivo da AM é definir e mostrar como colocar em uso um conjunto de valores, princípios e práticas relativas a uma modelagem eficaz e simples. Para Ambler (2004),

um modelo ágil é apenas bom o suficiente, ou seja, ele cumpre seu propósito, é compreensível, é suficientemente preciso, consistente e detalhado e é o mais simples possível.

Crystal: é uma família de métodos criada por Cockburn (2002). Nos métodos Crystal, o desenvolvimento de software é visto como um jogo cooperativo de invenção e comunicação, com o principal objetivo de entregar um software útil e funcionando (Cockburn, 2002). Dois projetos diferentes devem ser executados de forma diferente. A abordagem Crystal inclui princípios para a adaptação dos métodos em diversas circunstâncias de projetos. Cada método da família Crystal é nomeado com uma cor: branca, amarela, laranja e vermelha, que será apropriada para um projeto, baseando-se no seu tamanho e aspecto crítico. Apesar de diferentes, existem regras, valores e características comuns nas metodologias Crystal.

***Adaptive Software Development* (ASD)**: desenvolvido por Highsmith (2000), defende o desenvolvimento incremental e iterativo com constante prototipação. O ciclo ASD possui três fases: Especular (*Speculate*), Colaborar (*Collaborate*) e Aprender (*Learn*). As fases são assim nomeadas para enfatizar o papel da mudança no processo (Abrahamsson et al., 2002). O ASD tem por objetivo promover as partes difíceis do desenvolvimento adaptativo, em particular, a colaboração e o aprendizado em um projeto (Fowler, 2003).

***Dynamic Systems Development Method* (DSDM)**: criado por uma associação de empresas do Reino Unido, o DSDM é um *framework* para desenvolvimento RAD (*Rapid Application Development*) (Abrahamsson et al., 2002). Segundo Tuffs et. al (1999), o DSDM é mais um *framework* do que um método, por fornecer uma estrutura de processo e descrições de produto que devem ser adaptadas para acomodar um projeto ou uma organização. A idéia fundamental do DSDM é que ao invés de ajustar a quantidade de funções em um produto e então ajustar o tempo e os recursos para alcançá-la, deve-se primeiro ajustar o tempo e os recursos, para depois ajustar a quantidade de funções (Abrahamsson et al., 2002). O DSDM contém cinco fases: Estudo da Viabilidade (*Feasibility Study*), Estudo do Negócio (*Business Study*), Iteração do Modelo Funcional (*Functional Model Iteration*), Iteração de Construção e Design (*Design and Build Iteration*) e Implementação (*Implementation*).

Feature Driven Development (FDD): é uma abordagem ágil e adaptativa para o desenvolvimento de sistemas que enfatiza pontos de qualidade durante todo o processo e inclui entregas freqüentes, com monitoramento cuidadoso ao longo do progresso do projeto. O FDD consiste de cinco processos, durante os quais o projeto e a construção do sistema são realizados: Desenvolver um Modelo Abrangente (*Develop an Overall Model*), Construir uma Lista de Características (*Build a Features List*), Planejar por Característica (*Plan by Feature*), Projetar por Característica (*Design by Feature*) e Construir por Característica (*Buid by Feature*). Coad et

al. (1999) apresentam detalhadamente esses cinco processos do FDD, juntamente com os artefatos que são produzidos.

Embora os métodos ágeis estejam sendo utilizados na sua forma original, pode ser necessário adaptá-los ou estendê-los com outras práticas devido a particularidades de um projeto ou organização.

A próxima seção aborda conceitos de padrões organizacionais e de processo, que neste trabalho, são utilizados para estender o método ágil Scrum.

2.4. Padrões e Linguagens de Padrões

O conceito de padrões (*patterns*) surgiu no final dos anos 70 com o arquiteto Christopher Alexander, que criou vários padrões para construção de ambientes (Alexander, 1977; Alexander, 1979). Não existe um consenso quando se trata da definição de padrões. Segundo Alexander (1977) padrão é a descrição de uma solução para um problema recorrente em um determinado contexto. Coplien e Harrison (2004) afirmam que padrão é uma configuração estrutural recorrente que resolve um problema em um determinado contexto. Buschmann et al. (1996) definem padrão como sendo um par problema-solução em um determinado contexto, que apresenta a situação que dá origem ao problema. O problema é o conjunto de forças que ocorrem repetidamente no contexto e a solução é a maneira de resolver esse problema.

Independentemente do autor, todas as definições mostram como principal objetivo dos padrões o reúso, ou seja, a reaplicação das suas soluções na solução dos problemas que estão em um mesmo domínio.

Os padrões de Alexander foram desenvolvidos para a disciplina de arquitetura, entretanto, sua idéia foi aplicada em outras disciplinas. Assim, a comunidade de software, em particular a comunidade de programação orientada a objetos, a adotou. Na Engenharia de Software, os que se tornaram mais conhecidos foram os padrões de projeto de sistemas da "GoF" (*Gang of Four*) (Gamma et al.,1995). A partir desses, outros foram disponibilizados, como por exemplo, os padrões de arquitetura de Buschmann et al. (1996), os padrões organizacionais e de processo de Coplien (1995), entre outros.

Com o intuito de facilitar o reúso dos padrões de software, eles são classificados em diversas categorias. Porém, não deve haver ortogonalidade nessa classificação, pois podem existir padrões que se encaixam em mais do que uma categoria (Braga, 2003).

Algumas das categorias de padrões descritas por Braga (2003) são:

- **Padrões de processo**: definem soluções para os problemas encontrados nos processos envolvidos na Engenharia de Software, por exemplo, desenvolvimento, controle de configuração e testes.
- **Padrões arquiteturais**: expressam o esquema ou organização estrutural fundamental de sistemas de software ou hardware;
- **Padrões de análise**: descrevem soluções para problemas de análise de sistemas, embutindo conhecimento sobre o domínio de aplicação específico;
- **Padrões de projeto**: definem soluções para problemas de projeto de software;
- **Padrões de programação**: descrevem soluções de programação particulares de uma determinada linguagem ou regras gerais de estilo de programação.

Outra categoria de padrões, que é discutida por Coplien (1995), Harrison (1996) e outros é a dos padrões organizacionais, que apóiam o desenvolvimento do software. Este trabalho trata mais especificamente dos padrões organizacionais e de processo, que são apresentados de forma mais detalhada na próxima seção.

A classificação dos padrões em níveis de abstração facilita a sua aplicação de acordo com a fase de desenvolvimento. Além desse tipo de classificação, eles podem ser agrupados em linguagens de padrões (*pattern language*). Uma linguagem de padrões é uma coleção estruturada de padrões que se apóiam uns nos outros para transformar requisitos e restrições em uma arquitetura (Coplien, 1998). Uma linguagem de padrões contém regras e diretrizes que explicam quando e como aplicar os padrões para resolver um problema que não pode ser resolvido por um único padrão (Appleton, 1997). A Figura 2.8 mostra um exemplo de linguagem de padrões.

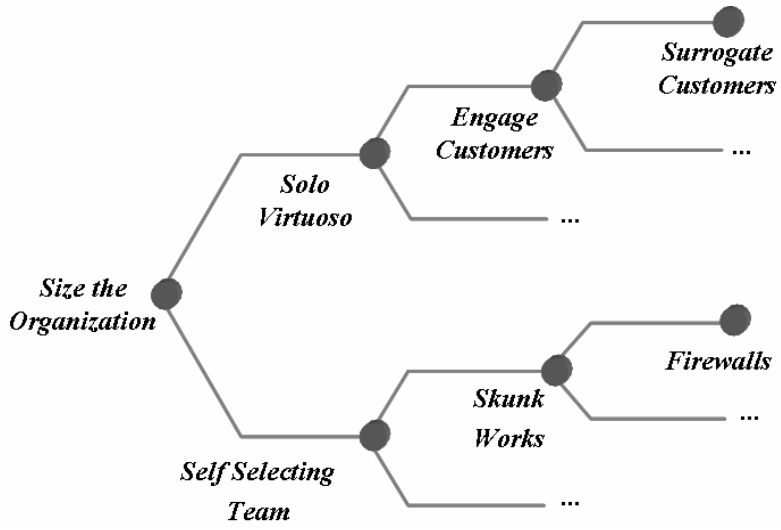


Figura 2.8 – Linguagem de padrões (Adaptada de Coplien e Harrison, 2004)

Para que o uso do padrão seja feito de forma correta, é necessário que a apresentação dos padrões esteja clara. A descrição do padrão deve facilitar sua leitura e entendimento. O formato de apresentação dos padrões varia de acordo com o autor e não existe um formato considerado certo ou errado. O formato usado nos trabalhos de Alexander é chamado de "Forma Alexandrina" e o usado nos trabalhos de Gamma et al.(1995) é chamado de "Formato GoF (*Gang of Four*)". Para a descrição dos padrões, existem elementos essenciais que devem estar bem especificados na sua apresentação. Segundo Appleton (1997) esses elementos são:

- **Nome**: identificação do padrão. Se o padrão tiver mais de um nome na literatura, o que é comum, deve-se documentar na sua descrição. Os apelidos ou sinônimos do padrão podem ser referenciados pelos elementos "Nomes alternativos" (*Aliases*) ou "Também conhecido como" (*Also Known As*).
- **Problema**: é a descrição do problema que o padrão se propõe a solucionar, na qual são apresentados os objetivos e metas que se deseja alcançar no contexto dado.
- **Contexto**: pré-condição dentro da qual o problema e a solução normalmente ocorrem. O contexto mostra a aplicabilidade do padrão e pode ser considerado como a configuração inicial do sistema, antes do padrão ser aplicado.
- **Forças**: descrição das forças relevantes e restrições para o problema, como elas interagem umas com as outras e como são conflitantes entre si e com os objetivos que se quer alcançar.
- **Solução**: relacionamentos estáticos e regras dinâmicas que descrevem como alcançar o resultado desejado. Isso é equivalente a dar instruções que descrevem como resolver o problema. A descrição pode conter figuras, diagramas e trechos que descrevem a estrutura do padrão, os participantes e colaborações, para mostrar como o problema é resolvido.
- **Exemplos**: um ou mais exemplos da aplicação do padrão, mostrando o contexto inicial específico, como o padrão é aplicado, as transformações no contexto e o contexto resultante. Os exemplos ajudam o leitor a entender o uso e a aplicabilidade do padrão.
- **Contexto Resultante**: estado ou configuração do sistema após aplicação do padrão, incluindo as conseqüências da sua aplicação, outros problemas e padrões envolvidos no contexto resultante. Esse elemento do padrão é muitas vezes chamado de "resolução de forças" porque descreve que forças foram solucionadas, quais ainda permanecem sem solução e quais padrões podem ser aplicados agora.

- **Análise Racional**: explicação dos passos ou regras no padrão, em termos de como e porque ele soluciona as forças para alcançar os objetivos desejados.
- **Padrões Relacionados**: relação estática e dinâmica entre o padrão e outros da mesma linguagem de padrões ou sistema. Freqüentemente compartilham as mesmas forças e têm um contexto inicial ou resultante compatível com o contexto inicial ou resultante de outros padrões.
- **Usos Conhecidos**: mostra ocorrências do padrão e de sua aplicação em sistemas existentes. Isso ajuda a validar o padrão, pois é possível verificar que o padrão é realmente uma solução provada para um problema recorrente.

A Tabela 2.3 mostra alguns formatos de apresentação dos padrões. A primeira coluna apresenta o nome do formato e a segunda mostra os elementos de apresentação que compõem cada formato.

Tabela 2.3 – Formatos de apresentação dos padrões

Forma	Elementos
Canônica	Nome; nome alternativo; problema; contexto; forças; solução; exemplo; contexto resultante; análise racional; usos conhecidos e padrões relacionados.
Alexandrina	Nome; confiança; contexto; resumo do problema; problema detalhado; solução; contexto resultante e análise racional.
GoF	Nome do padrão; intenção; também conhecido como; motivação; aplicabilidade; estrutura; participantes; colaborações; conseqüências; implementação; código exemplo; usos conhecidos e padrões relacionados.
Appleton	Nome; contexto; problema; forças; solução; contexto resultante; análise racional, padrões relacionados e usos conhecidos.

Exemplos de padrões e linguagens de padrões organizacionais e de processo são apresentadas a seguir.

2.4.1. Padrões Organizacionais e de Processo

Existem diversas definições para padrões organizacionais e de processo. A seguir são apresentadas algumas delas. Ambler (1998) cita que os padrões organizacionais descrevem técnicas comuns de gerenciamento ou estruturas organizacionais, enquanto que os padrões de processo descrevem uma abordagem e/ou séries de ações comprovadas e de sucesso para o desenvolvimento de software. Quando aplicados juntos de forma organizada, padrões de processo podem ser usados para construir um processo de software para uma organização. Coplien (1995) afirma que os padrões organizacionais e de processo captam práticas bem sucedidas de desenvolvimento de software e podem ser usados para modelar uma nova organização e seu processo de desenvolvimento.

Embora não exista um consenso em relação à definição dos padrões organizacionais e de processo, é fato que eles estão relacionados. Ambler (1998) afirma que os padrões organizacionais e de processo "andam" lado a lado, isto é, devem ser usados juntos, de forma complementar. Coplien (1995) afirma que as organizações que têm alta produtividade, em sua maioria, usam os mesmos ou semelhantes padrões de organização e de processo.

Exemplos de linguagens de padrões organizacionais e de processo são apresentados de forma mais detalhada na Seção 2.4.2.

2.4.2. Linguagens de Padrões Organizacionais e de Processo

Uma linguagem de padrões é um sistema de padrões organizados em uma estrutura que guia a sua aplicação (Cunningham e Kerth, 1997), diferente de um catálogo de padrões, em que os padrões são subdivididos em um pequeno número de categorias abrangentes (Buschmann et al., 1996).

Linguagens de padrões organizacionais e de processo definem soluções para os problemas encontrados nos processos envolvidos na Engenharia de Software, como por exemplo, desenvolvimento, organização, controle de configuração e testes.

A linguagem de padrões "*A Generative Development-Process Pattern Language*" (Coplien, 1995), publicada em *Pattern Languages of Program Design* (Coplien e Schmidt, 1995) é a primeira referência sobre padrões organizacionais e de processo.

Desde o surgimento da linguagem de padrões organizacionais e de processo de Coplien (1995), outras surgiram para apoiar e organizar a construção do software, como por exemplo, a linguagem de padrões para desenvolvimento "*Episodes*" de Cunningham (1996), a linguagem de padrões organizacionais para equipes "*Organizational Patterns for Teams*" de Harrison (1996), a linguagem de padrões para construção de protótipos "*Demo Prep*" de Coram (1996), a linguagem de padrões para testes "*System Test Pattern Language*" de DeLano e Rising (1998), as linguagens de padrões organizacionais Coplien e Harrison (2004), a linguagem de padrões para introduzir novas idéias em organizações (Manns e Rising, 2004), entre outras.

Coplien e Harrison (2004) descrevem quatro linguagens de padrões que combinam estruturas organizacionais com as melhores práticas de desenvolvimento de software, que devem ser usadas em conjunto para solucionar os problemas da organização e fortalecê-la:

- **Linguagem de Padrões de Gerenciamento de Projeto** (*Project Management Pattern Language*) (vinte e cinco padrões): trata da estruturação do trabalho na organização, concentrando-se no cronograma, processo, tarefas e em estruturas para apoiar o progresso do trabalho.

- **Linguagem de Padrões de Desenvolvimento Gradativo** (*Piecemeal Growth Pattern Language*) (vinte e nove padrões): descreve como criar a organização e o processo ao mesmo tempo.
- **Linguagem de Padrões de Estilo Organizacional** (*Organizational Style Pattern Language*) (vinte e um padrões): trata da estrutura de relacionamentos dos papéis na organização.
- **Linguagem de Padrões de Pessoas e Código** (*People and Code Pattern Language*) (dezessete padrões): trata do relacionamento entre a estrutura da organização e os artefatos que são construídos.

Essas quatro linguagens são resultado de um esforço coletivo na pesquisa sobre padrões organizacionais e de processo, pois, embora novos padrões tenham sido documentados, alguns dos contidos nessas linguagens pertencem a autores como Harrison (1996) e Cockburn (1998) e alguns padrões foram herdados da linguagem de padrões de Coplien (1995).

Para exemplificar um padrão organizacional e de processo de Coplien e Harrison (2004), cuja forma de apresentação é a Alexandrina, descrita na Tabela 2.1, o padrão *Fire Walls* ("Corta Fogo") é apresentado a seguir.

- **Nome**: *Fire Walls* ** (Os asteriscos ao lado do nome do padrão representam a confiança no padrão. O número de asteriscos varia de zero a duas, dependendo de quantas vezes o padrão foi aplicado.)
- **Contexto**: uma organização de desenvolvedores é formada em um contexto corporativo ou social, em que esses são freqüentemente distraídos por pessoas externas que sentem necessidade de oferecer informações e fazer críticas.
- **Resumo do Problema**: é importante proteger os desenvolvedores de outras pessoas envolvidas no projeto, que não participam do desenvolvimento, mas sentem necessidade de ajudar por meio de comentários ou críticas.
- **Problema Detalhado**: o isolamento não funciona: o fluxo da informação é importante. Mas, o excesso de comunicação aumenta de forma não linear em relação ao número de colaboradores externos.
- **Solução**: crie um cargo de gerente, que proteja os desenvolvedores de interações com cargos externos, para evitar interrupções durante desenvolvimento.
- **Contexto Resultante**: A nova organização isola os desenvolvedores de interrupções externas insignificantes. Para evitar o isolamento, esse padrão deve ser utilizado em conjunto com outros, como *Engage Customers* (Empregar o Cliente) e *Gate Keeper* (Porteiro).

- **Análise Racional**: O padrão *Fire Walls* restringe o fluxo de informações. O padrão *Gate Keeper* facilita o fluxo de informações úteis. É necessário balancear esses dois padrões.

O padrão *Fire Walls* fornece uma solução para um problema pertinente às organizações que desenvolvem software: o excesso de solicitações e informações desnecessárias que são passadas aos desenvolvedores. Interrupções como essas afetam diretamente a produtividade (Schwaber e Beedle, 2002). O *Fire Walls* deve ser utilizado para manter os desenvolvedores concentrados nas suas tarefas, pois a comunicação em excesso pode sobrecarregar o projeto. Assim, é necessário dar suporte para comunicação, que é fundamental para o sucesso da organização (Aye, 2004).

A Tabela 2.4 mostra doze padrões organizacionais de Coplien e Harrison (2004) que foram utilizados neste trabalho. A primeira coluna apresenta a linguagem a qual o padrão pertence, a segunda tem o nome do padrão e a terceira apresenta um resumo contendo o problema e a solução proposta para ele.

Tabela 2.4 – Padrões organizacionais (Coplien e Harrison, 2004)

Linguagem	Padrão	Resumo do Padrão
Piecemeal Growth Pattern Language	*Scenarios Define Problem*	**Problema**: Extrair os requisitos e mostrar ao cliente como o sistema deve funcionar. **Solução**: Utilizar casos de uso para extrair os requisitos do sistema. Isso pode ajudar a ajustar problemas de limite entre cliente e desenvolvedor.
	Application Design is Bounded By Test Design	**Problema**: Projetar e criar os planos de teste baseados em casos de uso de forma a não interferir nos testes. **Solução**: Iniciar o projeto dos testes direcionados a casos de uso quando o cliente concordar com o caso de uso, antes do início do desenvolvimento. O projeto de teste deve evoluir paralelamente ao projeto do software.
	Surrogate Customers	**Problema**: Comunicar-se com um cliente em situações em que ele se encontra indisponível. **Solução**: Criar um papel de substituto do cliente, que deve ser atribuído a alguém que deverá tentar pensar como o cliente.
	Self Selecting Team	**Problema**: Selecionar a equipe de desenvolvimento. **Solução**: Permitir que as pessoas selecionem a equipe na qual elas querem trabalhar. As equipes de pior dinâmica são aquelas estabelecidas sem levar em consideração os interesses individuais.
People And Code Pattern Language	*Architect Controls Product*	**Problema**: Dar ao produto elegância e coesão. **Solução**: Criar um cargo de arquiteto no projeto, responsável por definir um estilo de arquitetura e estabelecer consenso entre os desenvolvedores.
	Architect Also Implements	**Problema**: Fornecer direção técnica aos desenvolvedores. **Solução**: Atribuir ao arquiteto a responsabilidade de implementar o software junto com os desenvolvedores.

	Code Ownership	**Problema**: Atribuir tarefas entre os vários desenvolvedores. **Solução**: Atribuir cada módulo de código a um único desenvolvedor, que só pode ser modificado pelo seu proprietário, exceto em condições extremas.
	Standards Linking Locations	**Problema**: Fazer com que o código de cada desenvolvedor interaja e se comunique corretamente em projetos geograficamente distribuídos. **Solução**: Utilize normas (*standards*) e convenções para representar tudo o que está relacionado à arquitetura.
	Generics And Specifics	**Problema**: Distribuir as tarefas entre principiantes e especialistas. **Solução**: Separar as partes genéricas e específicas dos problemas. Use um especialista para projetar as partes genéricas e principiantes para implementar as partes específicas.
Project Management Pattern Language	*Build Prototypes*	**Problema**: Extrair e analisar os requisitos e decisões de projeto para diminuição de risco. **Solução**: Construir um protótipo para entender e validar requisitos com o cliente; explorar interações humano-computador; e explorar o custo e benefício de soluções de projeto.
	Named Stable Bases	**Problema**: Integrar o software frequentemente para que a base não fique desatualizada, de forma que não atrapalhe o entendimento de qual função já está desenvolvida na base de software que ainda está evoluindo. **Solução**: Integrar o sistema uma vez por semana e dar à base estável um nome que identifique as funções que a compõem.
	Mercenary Analyst	**Problema**: Manter a documentação do projeto atualizada e disponível para todos. **Solução**: Nomear uma pessoa responsável por escrever a documentação do projeto e manter essa documentação disponível para todos.
Organizational Style Pattern Language	*Face To Face Before Working Remotely*	**Problema**: Dar uniformidade e diminuir os problemas de comunicação em projetos distribuídos. **Solução**: Iniciar um projeto distribuído com uma reunião cara a cara com todos os envolvidos no projeto. Essa reunião deve estabelecer uniformidade no projeto, além de permitir que as pessoas se conheçam.
	Organization Follows Location	**Problema**: Atribuir tarefas e cargos de acordo com a distribuição geográfica. **Solução**: Repartir as tarefas de forma a refletir a distribuição geográfica. Responsabilidades devem ser atribuídas para que as decisões possam ser tomadas localmente, ou seja, as pessoas que possuem tarefas semelhantes devem estar no mesmo local.

Esses padrões mostram estruturas organizacionais e práticas de desenvolvimento de sucesso que podem ser reutilizadas em outras organizações. Mas, além dos padrões pertencentes a essas quatro linguagens, outros também serviram de base para esta pesquisa.

Harrison (1996) apresenta a linguagem de padrões organizacionais para equipes (*Organizational Patterns for Teams*) (quatro padrões) para auxiliar equipes a projetar e desenvolver softwares. Harrison (1996) argumenta que não se pode agrupar um conjunto de desenvolvedores e esperar que eles trabalhem como equipe automaticamente e, para que isso aconteça, é necessário organizar os desenvolvedores. A Tabela 2.5 apresenta dois padrões dessa linguagem. A primeira coluna apresenta o nome do padrão e a segunda, um resumo contendo o problema e a solução proposta para ele.

Tabela 2.5 – Padrões para organização de equipes (Harrison, 1996)

Padrão	Resumo do Padrão
Unity Of Purpose	**Problema**: Estabelecer consenso entre os membros da equipe que estão começando a trabalhar juntos, cada um com seu conhecimento e experiência. **Solução**: Atribuir ao líder do projeto a responsabilidade de expor a todos os membros da equipe uma visão comum e o propósito do projeto.
Lock 'Em Up Together	**Problema**: Criar uma arquitetura única e coerente, em uma equipe formada por diferentes pessoas. **Solução**: Agrupar os membros da equipe para elaborar a arquitetura. Todos devem se comprometer a participar totalmente até que a arquitetura esteja completa ou pelo menos suficientemente clara para todos.

Os padrões organizacionais para equipes devem ser utilizados para ajudar desenvolvedores, que têm diferentes idéias e opiniões, a trabalharem juntos, compartilhando informações como equipe.

Embora Harrison (1996) descreva soluções para projetar software, existe uma atividade crítica que deve ser realizada antes do projeto, a especificação dos requisitos do software. Um dos maiores causadores dos problemas nas especificações de requisitos é a comunicação entre clientes e desenvolvedores. Para minimizar esse problema, casos de uso e/ou protótipos podem ser utilizados para extrair requisitos do cliente. Os casos de uso capturam todos os cenários que o sistema deve tratar. Já os protótipos facilitam a extração, entendimento e validação dos requisitos junto ao cliente. Bramble et al. (2002) descrevem uma linguagem de padrões para criar casos de usos efetivos (*Patterns for Effective Use Cases*) (trinta e um padrões). Essa linguagem cobre três tópicos: a composição da equipe, técnicas para criação dos casos de uso e técnicas para melhorar os casos de uso existentes. Três padrões dessa linguagem são apresentados na Tabela 2.6, que contém o nome do padrão e um resumo contendo o problema e a solução proposta para ele.

Tabela 2.6 – Padrões para criação casos de uso efetivos (Bramble et al., 2002)

Padrão	Resumo do Padrão
Small Writing Team	**Problema**: Utilizar muitas pessoas para escrever os casos de uso. **Solução**: Limitar a três o número de pessoas que vão escrever os casos de uso, pois é mais fácil para uma equipe pequena chegar a um consenso.
Breadth Before Depth	**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 adicionar os detalhes gradualmente, trabalhando lado a lado com um conjunto de casos de uso relacionados.
Spiral Development	**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.

Os padrões da Tabela 2.6 mostram como criar uma equipe para escrever casos de uso e apresentam técnicas para gerá-los e aperfeiçoá-los ao longo do projeto. Bramble et al. (2002) afirmam que um padrão documenta solução para um problema específico, sem abranger completamente o assunto.

Além dos casos de uso, protótipos também podem ser utilizados para capturar e validar requisitos com o cliente. Coram (1996) descreve a linguagem de padrões para preparação de demonstrações de software (*Demo Prep*) (sete padrões) para guiar a construção, administração e demonstração de protótipos. A Tabela 2.7 apresenta três padrões dessa linguagem. A primeira coluna apresenta o nome do padrão e a segunda mostra um resumo contendo o problema e a solução proposta para ele.

Tabela 2.7 – Padrões para construção e demonstração de protótipos (Coram, 1996)

Padrão	Resumo do Padrão
Element Identification	**Problema**: Selecionar as funções do software que devem ser demonstradas ao cliente para manter sua confiança. **Solução**: Identificar as funções do software que preocupam os clientes. Converse com ele e o escute atentamente.
Judicious Fireworks	**Problema**: Tranqüilizar o cliente quanto ao software que está sendo construído. **Solução**: Apresentar o que vai ser desenvolvido sem criar expectativas ao cliente de características que o produto final não terá.
Light Weight User Interfaces	**Problema**: Demonstrar a interatividade do protótipo ao cliente, que espera indícios de que os desenvolvedores estão construindo o que ele necessita. **Solução**: Utilize ferramentas que facilitem a criação e modificação da interface.

Os padrões da Tabela 2.7 mostram o que deve ser levado em consideração na criação e demonstração de protótipos para clientes. Os protótipos podem ser utilizados na especificação e validação dos requisitos e também em demonstrações para tranqüilizar o cliente com relação ao que está sendo desenvolvido.

Depois de criar a especificação e desenvolver o produto, é necessário verificar se ele atende às exigências do cliente, contidas na especificação. Para isso são realizados testes. Os testes servem para verificar se o sistema possui ou não erros e corroborar se o sistema satisfaz ao objetivo definido em sua especificação. DeLano e Rising (1998) apresentam uma linguagem de padrões para teste de sistema (*System Test Pattern Language*) (vinte padrões), resultante de pesquisas realizadas com testadores. Ao longo dos anos, a preocupação com a qualidade do produto e processo aumenta e o papel do testador tem se tornado vital para o ciclo de desenvolvimento do produto. Três padrões dessa linguagem são apresentados na Tabela 2.8, que contém o nome do padrão e um resumo contendo o problema e a solução proposta para ele.

Tabela 2.8 – Padrões para testes de sistemas (DeLano e Rising, 1998)

Padrão	Resumo do Padrão
Get Involved Early	**Problema**: Aumentar ao máximo o apoio dos projetistas aos testadores. **Solução**: Estabelecer, desde o início do projeto, um bom relacionamento de trabalho com os projetistas, de forma que esse relacionamento não interfira no julgamento do testador.
Time to Test	**Problema**: Iniciar os testes **Solução**: Iniciar os testes quando uma área de funcionalidade estiver disponível. Esperar para testar todo sistema quando ele estiver pronto é arriscado, pois o tempo pode não ser suficiente para realizar todos os testes.
Busy System	**Problema**: Determinar as condições nas quais os testes de sistema devem ser executados para que a maioria dos problemas sejam encontrados. **Solução**: Testar o sistema em um ambiente que simule um sistema ocupado. O nível de atividade não precisa pressionar muito o sistema, mas deve se aproximar de um nível que o sistema vai praticar regulamente.

Os padrões de teste de DeLano e Rising (1998) foram divididos em quatro categorias, de acordo com a sua utilidade no processo de teste de sistema: Organização do Teste (*Test Organization*), Eficiência do Teste (*Testing Efficiency*), Estratégia de Teste (*Testing Strategy*) e Resolução do Problema (*Problem Resolution*). Essas categorias tratam, respectivamente, de questões como o relacionamento do testador com os outros membros da organização, eficiência dos testes, estratégia de testes e comunicação e resolução dos problemas.

Existem ainda padrões que tratam do gerenciamento de configuração do software. Berczuk e Appleton (2002) descrevem práticas de sucesso para gerenciamento de configuração

de software e outros produtos de trabalho. Segundo eles o gerenciamento de configuração de software (*Software Configuration Management - SCM*) é um conjunto de processos que uma pessoa utiliza para criar e manter um conjunto consistente de componentes de software, sendo o controle de versão a parte principal do mecanismo de comunicação. Berczuk e Appleton (2002) afirmam que, embora pareça óbvio que o controle de versões é importante para o processo de desenvolvimento de software, muitas empresas não o utilizam, mesmo com ferramentas de controle disponíveis. A Tabela 2.9 apresenta três padrões da linguagem de padrões para gerenciamento de configuração (*Software Configuration Management Patterns*) (dezesseis padrões) de Berczuk e Appleton (2002) que foram utilizados nesta pesquisa. A primeira coluna apresenta o nome do padrão e a segunda, um resumo contendo o problema e a solução proposta para ele.

Tabela 2.9 – Padrões para gestão de configuração de software (Berczuk e Appleton, 2002)

Padrão	Resumo do Padrão
Private Workspace	**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 e componentes em que estão trabalhando.
Repository	**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. O repositório deve fornecer todos os componentes necessários para que a área privada seja criada.
Smoke Test	**Problema**: Verificar se a base estável do sistema está funcionando após a sua criação. **Solução**: Submeta toda base estável do sistema a um teste de "fumaça", que verifica se a aplicação está funcionando. O teste não precisa ser exaustivo, ele deve testar apenas as funções básicas do sistema. Esse teste não deve tirar dos desenvolvedores a responsabilidade de testar suas mudanças antes de enviá-las para o repositório.

Existem várias ferramentas que implementam os padrões de controle de versões citados acima, como por exemplo o CVS (*Concurrent Versions System*) (CVS, 2006), mas foge do escopo deste trabalho propor a utilização de uma ferramenta ou outra. Berczuk e Appleton (2002) argumentam que algumas organizações necessitam de processos e ferramentas de controle de versão mais simples que outras, mas todas precisam ter uma forma de realizar o controle de versões e comunicar as mudanças de código entre os envolvidos no projeto.

Padrões organizacionais e de processo são melhores práticas de outras organizações, documentadas de forma a facilitar o seu reúso. Aplicar um padrão organizacional e de processo

significa utilizar uma nova idéia na sua organização e, introduzi-la pode ser uma tarefa difícil. Sabendo dessa dificuldade, Manns e Rising (2004) coletaram estratégias de sucesso de pessoas que, na tentativa de introduzir novas idéias em organizações, as documentaram na forma de padrão. A Tabela 2.10 apresenta alguns padrões dessa linguagem de padrões que também foram considerados nesta pesquisa. A primeira coluna apresenta o nome do padrão e a segunda, um resumo contendo o problema e a solução para ele.

Tabela 2.10 – Padrões para introduzir novas idéias em organizações (Manns e Rising, 2004)

Padrão	Resumo do Padrão
Evangelist	**Problema**: Começar a introduzir uma nova idéia em uma organização. **Solução**: Fazer tudo o que for possível para compartilhar a sua paixão pela idéia. A primeira pessoa que deve ser convencida é você.
External Validation	**Problema**: Aumentar a credibilidade da nova idéia. **Solução**: Apresentar informações de fontes externas para a organização. As pessoas querem garantias de que a nova idéia tem validade fora da organização.
Fear Less	**Problema**: Reverter a resistência a nova idéia a seu favor. **Solução**: Pedir ajuda aos céticos, ou seja, escute o que eles têm a dizer e aprenda com eles.
Just Enough	**Problema**: Introduzir uma nova idéia. **Solução**: Concentrar-se nos fundamentos da nova idéia. Dar aos aprendizes uma breve descrição dos conceitos mais difíceis e fornecer mais informações quando eles estiverem prontos.
Next Steps	**Problema**: Mostrar aos aprendizes como eles podem aplicar a nova idéia. **Solução**: Utilizar um tempo após a apresentação da nova idéia para identificar o que os participantes podem fazer depois da introdução da idéia, como eles podem aplicar a nova informação.
Personal Touch	**Problema**: Convencer as pessoas do valor da nova idéia. **Solução**: Mostrar aos aprendizes como a nova idéia pode ser útil e valiosa para eles. Mudar um paradigma em uma organização significa convencer as pessoas da organização.
Tailor Made	**Problema**: Costurar a sua mensagem sobre inovação de acordo com as necessidades da organização. **Solução**: Estudar o processo de desenvolvimento da organização e seus objetivos e encontrar uma necessidade na organização que a inovação pode atender.
Time For Reflection	**Problema**: Aprender com o passado. **Solução**: Reservar um tempo, em intervalos regulares, para avaliar o que está funcionando bem e o que deveria ser feito de forma diferente. Essa é uma forma de aprender com os problemas do passado.

Em um mundo de mudanças rápidas, cada vez mais as organizações são pressionadas a estar em constante mudança para progredir. Porém, introduzir novas idéias em organizações é um desafio e para se ter sucesso ao compartilhar uma nova idéia é necessário acreditar nela (Manns e Rising, 2004).

2.5. Considerações Finais

Neste capítulo foram apresentados os conceitos de métodos ágeis que, diferentemente dos modelos prescritivos, abordam o desenvolvimento de software de forma adaptativa. Os diferentes métodos ágeis disponíveis na literatura possuem suas características próprias, com suas vantagens e desvantagens. Abrahamsson et al. (2002) destacam alguns pontos positivos e negativos que podem ser observados nos métodos ágeis. Na Tabela 2.11, que contém o nome do método ágil na primeira coluna e seus pontos positivos e negativos na segunda e terceira colunas respectivamente, são comparados somente os métodos ágeis citados na Seção 2.3.2.

Tabela 2.11 – Pontos positivos e negativos de alguns métodos ágeis (adaptada de Abrahamsson et al., 2002)

Método Ágil	Pontos Positivos	Pontos Negativos
Scrum	Uso de equipes pequenas; Autonomia das equipes para se auto-organizar para atingir o objetivo de uma iteração; Iterações curtas, com duração de trinta dias.	Embora forneça práticas para gerenciar e controlar uma iteração de trinta dias, não fornece detalhes de como as atividades de integração e testes devem ser realizadas.
XP	Uso de equipes pequenas; Desenvolvimento direcionado ao cliente; Desenvolvimento iterativo com integração diária.	Suas práticas individuais são adequadas para várias situações, porém pouca atenção é dada à visão global delas e às práticas de gerenciamento.
AM	Aplicação dos princípios ágeis para a modelagem: simplicidade, cultura ágil, organização do trabalho para apoiar a comunicação.	Concentra-se apenas na modelagem e documentação e, assim, deve ser utilizada em conjunto com outro método ágil.
Crystal	Compartilhamento dos mesmos valores e princípios por todos os quatro métodos da família Crystal.	Não avaliado, pois existem apenas dois dos quatro métodos sugeridos.
ASD	Cultura adaptativa; Enfoque na colaboração; Desenvolvimento iterativo.	Aborda mais conceitos e cultura do que práticas de desenvolvimento de software.
DSDM	Aplicação de controles para RAD (*Rapid Application Development*); Desenvolvimento iterativo; Existência de uma associação ativa para guiar o desenvolvimento do método.	Embora o método esteja disponível, apenas os membros da associação de empresas que o criou têm acesso aos documentos que fornecem instruções e explicações gerais sobre o método.
FDD	Processo baseado em cinco passos; Desenvolvimento orientado a componentes e funções; Iterações curtas, com duração de duas semanas.	Concentra-se apenas no projeto e implementação e necessita de outras práticas de apoio.

Embora os métodos ágeis tenham características particulares, eles compartilham os mesmos valores e princípios (Beck et al., 2001).

Dentre os métodos ágeis existentes, o Scrum é o método que mais se relaciona com o trabalho aqui proposto, por ser mais flexível que os outros. O Scrum não exige práticas

específicas de engenharia de software e, ao invés disso, fornece um conjunto de práticas de gerenciamento que controlam práticas de engenharia de software, como apresenta a Figura 2.9.

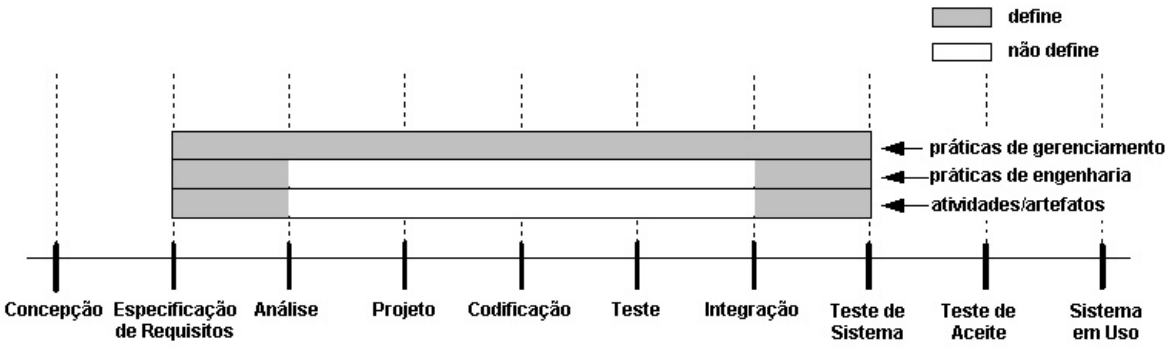


Figura 2.9 – Etapas de desenvolvimento de software apoiadas pelo Scrum (Adaptada de Abrahamsson et al., 2002)

A Figura 2.9 mostra que o Scrum fornece práticas de gerenciamento que controlam o desenvolvimento do software desde a etapa de especificação de requisitos até a etapa de teste de sistema, porém não define atividades, artefatos e práticas para as etapas de análise, projeto, codificação, teste e integração de software. Isso mostra que o Scrum pode ser utilizado com outros métodos, como por exemplo, com o XP (Mar e Schwaber, 2002; Schwaber e Beedle, 2002).

Os métodos ágeis possuem lacunas e, assim, adaptações ou extensões podem ser necessárias devido a particularidades de um projeto ou organização. Os padrões organizacionais e de processo fornecem soluções comprovadas para problemas recorrentes no processo de desenvolvimento de software e, assim, nada impede que sejam utilizados para adaptar ou estender o Scrum.

Os padrões organizacionais e de processo, normalmente, não aparecem de forma isolada, mas sim na forma de linguagens de padrões, embora possam ser aplicados de forma individual. Existem várias linguagens de padrões organizacionais e de processo disponíveis na literatura, porém apenas as que estão relacionadas com este trabalho foram comentadas na Seção 2.4.2, cujos padrões propõem práticas de sucesso para desenvolvimento de software, que podem ser aplicadas para tratar questões não abordadas pelo método ágil Scrum. A Tabela 2.12 apresenta o nome dessas linguagens de padrões, sua categoria, seus respectivos autores e o ano de sua publicação.

Tabela 2.12 – Linguagens de padrões organizacionais e de processo

Linguagem de Padrões	Categoria	Autores	Ano
A Generative Development-Process Pattern Language	Organizacional e de Processo	Coplien, J.	1995
Organizational Patterns For Teams	Organizacional	Harrison, N.	1996
Pattern Language for Preparation of Software Demonstrations	Processo	Coram, T.	1996
Patterns For System Testing	Processo	DeLano e Rising	1998
Software Configuration Management Patterns	Processo	Berczuk e Appleton	2002
Patterns for Effective Use Cases	Processo	Bramble et al.	2002
Project Management Pattern Language	Organizacional	Coplien e Harrison	2004
Piecemeal Growth Pattern Language	Organizacional	Coplien e Harrison	2004
Organizational Style Pattern Language	Organizacional	Coplien e Harrison	2004
People and Code Pattern Language	Organizacional	Coplien e Harrison	2004
Patterns for Introducing New Ideas	Organizacional	Manns e Rising	2004

O Capítulo 3 apresenta uma experiência de uso dos padrões organizacionais e de processo com o Scrum.

Capítulo 3

Experiência de Uso dos Padrões Organizacionais e de Processo com Scrum

3.1. Considerações Iniciais

A necessidade constante de melhorar o processo de desenvolvimento de software com relação a custo, prazo e qualidade de software motivou a elaboração de uma experiência de uso dos padrões organizacionais e de processo junto ao Scrum, que é o método ágil mais flexível dentre os disponíveis na literatura, como mencionado na Seção 2.5.

Como todos os métodos ágeis, o Scrum possui pontos fracos e, uma alternativa para tratar essas franquezas é utilizar padrões organizacionais e de processo para melhorá-lo. Desse modo, este capítulo apresenta a condução de uma experiência de uso dos padrões organizacionais e de processo com o Scrum, para adaptá-lo ou estendê-lo de acordo com as necessidades da organização que o utiliza.

Este capítulo está organizado da seguinte forma: a Seção 3.2 apresenta uma breve descrição da experiência e do ambiente na qual foi realizada; na Seção 3.3 é apresentada a forma como a experiência foi planejada; a Seção 3.4 mostra como ocorreu a condução de cada etapa da experiência; a Seção 3.5 apresenta a análise realizada sobre a experiência, e, por fim, na Seção 3.6, estão as considerações finais.

3.2. Descrição da Experiência

Para que avaliar a reutilização dos padrões organizacionais e de processo por uma organização, uma experiência foi conduzida em duas etapas descritas a seguir.

1. **Primeira Etapa**: desenvolvimento de um sistema sem o uso do Scrum e dos padrões organizacionais e de processo, mas seguindo os princípios do desenvolvimento ágil (Beck et al., 2001) e da Modelagem Ágil (Ambler, 2004);

2. **Segunda Etapa**: desenvolvimento de outro sistema com o uso de alguns padrões organizacionais e de processo e das práticas Scrum. A partir dessa experiência realizada com dois grupos, foi possível observar a necessidade de criar uma forma ordenada para utilizar os padrões organizacionais e de processo com Scrum.

Essa experiência foi realizada com alunos do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos (UFSCar), da disciplina "Tópicos em Engenharia de Software" no segundo período letivo de 2005.

Essa disciplina é cursada por alunos com conhecimento em desenvolvimento e projeto de software, adquiridos nas disciplinas "Paradigmas de Linguagens de Programação" e "Projeto e Gerência de Sistemas de Software" do primeiro período letivo de 2005. Assim, os alunos possuíam a base necessária para o desenvolvimento das atividades da experiência. Assuntos como padrões organizacionais e de processo e Scrum foram apresentados por meio de treinamento realizado antes e durante a experiência.

Os alunos foram inicialmente divididos em duas equipes (chamadas de organizações e que são assim referenciadas neste trabalho) com cinco elementos em cada uma, denominadas "Organização A" e "Organização B".

A Organização A foi formada por alunos recém graduados e com pouca experiência no mercado de trabalho, enquanto a Organização B foi formada por alunos com experiência no desenvolvimento de sistemas, atuando no mercado de trabalho.

Os sistemas solicitados para as organizações foram:

- **Sistema de Clínica Veterinária**: Controle de serviços e venda de produtos (*pet shop*). O sistema deve controlar todos os serviços oferecidos pela clínica, como consultas, banho, tosa e vacinação, além da venda de produtos. O sistema deve possibilitar o agendamento de serviços, bem como cadastro, alteração e remoção de animais, raças, espécies, proprietários e veterinários, além de permitir cadastro, alteração e remoção de produtos vendidos na clínica veterinária.
- **Sistema de Controle de Equivalências**: Controle de equivalências entre as disciplinas oferecidas pelo Departamento de Computação (DC) aos cursos existentes na Universidade Federal de São Carlos (UFSCar) e as disciplinas de computação cursadas por alunos de outras instituições de ensino superior, que se transferem para os cursos da UFSCar. O sistema deve auxiliar na avaliação de equivalências entre disciplinas oferecidas pelo DC-UFSCar com as de outras universidades. Por meio de avaliação da carga horária de cada disciplina e comparação de ementa, o sistema deve sugerir se as disciplinas são ou não equivalentes, e deve deixar a cargo do usuário a

confirmação ou não dessa equivalência entre as disciplinas. O sistema deve possibilitar o cadastro, alteração e remoção de outras universidades, cursos, disciplinas, ementas e bibliografia, bem como cadastro, alteração e remoção de cursos e disciplinas oferecidas pelo DC-UFSCar.

Os sistemas solicitados têm diferente número de funções, porém apresentam a mesma complexidade.

Para a execução e condução correta da experiência foi realizado um planejamento, que é comentado na próxima seção.

3.3. Planejamento da Experiência

Primeiramente, antes da realização da experiência, o Scrum e vários padrões organizacionais e de processo foram investigados. Com essa pesquisa foi possível observar a relação entre alguns padrões organizacionais e de processo com os métodos ágeis Scrum e XP. Sutherland (2003) afirma que a publicação de Coplien (1995) influenciou o Scrum. Beck (1999b) declara que muitas da idéias no XP provêm da linguagem de padrões Episodes (Cunningham, 1996). Porém, não existem na literatura referências sobre quais padrões especificamente foram utilizados. Essa ausência de referência motivou a realização de um estudo na tentativa de verificar quais padrões organizacionais e de processo estão relacionados com o Scrum (Costa Filho et al., 2005). A Tabela 3.1 apresenta os padrões organizacionais e de processo que estão relacionados às práticas e papéis Scrum. A primeira coluna mostra o nome e autor(es) do padrão, a segunda apresenta um resumo sobre o padrão e a terceira mostra seu relacionamento com as práticas e/ou papéis do Scrum.

Tabela 3.1 – Padrões organizacionais e de processo relacionados às práticas e papéis Scrum

Padrão e Autor(es)	Resumo do Padrão	Relação com Scrum
Developer Controls Process (Coplien, 1995)	Como os desenvolvedores contribuem diretamente no desenvolvimento dos artefatos visíveis para o usuário final, faça do desenvolvedor o ponto foco de informação do processo.	A Equipe Scrum produz o software e participa das reuniões diárias, de planejamento e de revisão de Iteração. Assim, seus membros têm domínio do que está sendo desenvolvido e são as principais fontes de informação no processo.
Engage Customers (Coplien, 1995)	É importante para a organização garantir e assegurar a satisfação do cliente por meio da comunicação entre clientes e desenvolvedores. Assim, junte o cliente aos desenvolvedores e arquitetos, não somente ao QA (Quality Assurance) ou marketing.	O cliente participa do desenvolvimento do produto desde o início do projeto. Ele identifica os requisitos junto à Equipe Scrum e Proprietário do Produto, os prioriza, participa das reuniões de revisão de Iteração e fica em contato constante com o Proprietário do Produto.

Few Roles (Coplien e Harrison, 2004)	As pessoas devem se comunicar para que o projeto progrida. Mas, essa comunicação pode impedir o verdadeiro progresso. Portanto, tente manter o número de papéis da organização abaixo de dezesseis.	Apenas cinco papéis são definidos no Scrum: Mestre Scrum, Proprietário do Produto, Equipe Scrum, Cliente e Gerência. Assim, o número de papéis na organização de desenvolvimento de software é pequeno.
Fire Walls (Coplien, 1995)	É importante proteger os desenvolvedores de outras pessoas envolvidas no projeto, que podem atrapalhar o desenvolvimento por meio de solicitações, comentários ou críticas. Portanto, crie um cargo de gerente para proteger os desenvolvedores de interações externas durante o desenvolvimento.	O Mestre Scrum é responsável por proteger a Equipe Scrum de interações externas durante as iterações. Ele deve garantir que nenhum requisito novo será solicitado para os desenvolvedores e que os mesmos não serão interrompidos por outras pessoas nas iterações.
Get On With It (Cockburn, 1998; Coplien e Harrison, 2004)	Não espere até que todos os requisitos tenham sido identificados para iniciar o desenvolvimento. Assim, inicie desenvolvendo as áreas que você tem mais confiança.	Não é necessário coletar todos os requisitos para iniciar o desenvolvimento. A Equipe Scrum inicia o desenvolvimento depois de identificar os requisitos conhecidos pelo cliente. Novos requisitos são acrescentados na lista de Trabalho do Produto, à medida que são identificados ao longo do projeto.
Implied Requirements (Cunningham, 1996)	O problema é definir as necessidades do cliente de forma significativa para os desenvolvedores. Portanto, selecione e nomeie partes de funcionalidade e crie uma lista com elas.	Os requisitos são identificados e divididos em itens que são listados e que vão compor a lista de Trabalho do Produto.
Work Queue (Cunningham, 1996)	O problema é conceder tempo para realizar tudo. Portanto, crie um cronograma que é simplesmente uma lista priorizada de trabalho. Utilize a lista do *Implied Requirement* e ordene-a, de forma que os itens mais urgentes tenham maior prioridade.	A lista de Trabalho do Produto é constantemente priorizada e atualizada com adição ou remoção de itens, de acordo com as necessidades do cliente.
Informal Labor Plan (Cunningham, 1996)	Um cronograma das tarefas dos desenvolvedores pode ajudá-los a planejar o seu tempo e assegurar as expectativas dos envolvidos no projeto. Assim, deixe os desenvolvedores criarem seus planos de curto prazo.	Para cada iteração, a Equipe Scrum identifica um objetivo, seleciona os itens da lista de Trabalho do Produto, divide esses itens entre seus membros e se compromete a atingir esse objetivo. Assim, eles criam planos de curto prazo, que são as listas de Trabalho das Iterações do projeto.
Recommitment Meeting (Cunningham, 1996)	Se o cronograma não puder ser satisfeito, convoque os interessados para revisá-lo e modificá-lo de modo que haja menor impacto e o objetivo seja atingido.	A Equipe Scrum se compromete a realizar o trabalho selecionado para a iteração, porém se não for possível desenvolver todos os itens da lista de Trabalho da Iteração, a equipe deve se reunir com o Mestre Scrum e Proprietário do Produto para diminuir o número de tarefas de forma a impactar o menos possível o objetivo da iteração.

Patron Role (Coplien, 1995)	É importante dar continuidade ao projeto. Assim, eleja um "Patrono" para remover as barreiras que impedem progresso da equipe de desenvolvimento.	Nas Reuniões Diárias Scrum são relatados os problemas que impedem o progresso da Equipe Scrum e o Mestre Scrum é o responsável por removê-los.
Size The Organization (Coplien, 1995)	Equipe com muitos membros raramente entregam os projetos dentro do prazo e orçamento previstos. Equipes pequenas podem não ser muito produtivas. Por isso, escolha aproximadamente dez pessoas para compor a equipe de desenvolvimento.	O Scrum sugere que a Equipe Scrum, tenha no máximo dez membros. Porém, outras equipes podem ser criadas em um projeto, desde que respeitem número máximo de integrantes.
Standup Meetings (Coplien e Harrison, 2004)	Em tempos de mudanças rápidas é essencial que todos os membros da organização recebam as mesmas informações. Portanto, realize reuniões diárias, de aproximadamente quinze minutos, para trocar informações sobre o projeto.	Durante as iterações, reuniões diárias, de aproximadamente quinze minutos, são realizadas para acompanhamento do projeto. Nessa reunião, a Equipe Scrum e o Mestre Scrum trocam informações sobre o projeto. Outros envolvidos no projeto também podem participar da reunião como observadores.
Gate Keeper (Coplien, 1995)	As pessoas devem se comunicar para que o projeto progrida, mas é importante balancear essa comunicação. O excesso de comunicação prejudica o projeto, mas o isolamento também. Assim, nomeie uma pessoa que deve ser responsável por levar as informações relevantes para os desenvolvedores.	O Proprietário do Produto, que fica em contato direto com o cliente, é responsável por levar as informações relevantes para a Equipe Scrum e esclarecer suas dúvidas ao longo da iteração.
Time For Reflection (Manns e Rising, 2004)	É importante aprender com o passado. Assim, reservar um tempo, em intervalos regulares, para avaliar o que está funcionando bem e o que deveria ser feito de forma diferente.	Após a Reunião de Revisão de Iteração deve ser realizada a reunião de retrospectiva, para discussão do que pode ser melhorado para próxima iteração.

Alguns padrões organizacionais e de processo de Coplien (1995), Cunningham (1996), Cockburn (1998) e Manns e Rising (2004) estão relacionados com o Scrum, porém salienta-se que apesar de existir essa relação não foi possível determinar quais desses padrões organizacionais e de processo influenciaram o Scrum. Este estudo propõe que outros padrões, além dos já relacionados, sejam utilizados junto ao Scrum para que as organizações obtenham melhor desenvolvimento de seus softwares, cuidando não só de pontos técnicos, mas também de pontos organizacionais.

A Tabela 3.2 apresenta um mapeamento entre as práticas e papéis definidos pelo Scrum e os padrões organizacionais e de processo apresentados na Tabela 3.1. A Tabela 3.2 apresenta os nomes dos padrões na escritos direção horizontal e os nomes das práticas e papéis definidos no Scrum escritos na direção vertical.

Tabela 3.2 – Mapeamento dos padrões relacionados a práticas e papéis do Scrum

Padrões Organizacionais e de Processo	**Práticas e Papéis do Scrum**												
	Mestre Scrum	Proprietário do Produto	Equipe Scrum	Cliente	Gerência	Lista de Trabalho do Produto	Estimativa de Esforço	Iteração	Reunião de Planejamento de Iteração	Lista de Trabalho da Iteração	Reuniões Diárias Scrum	Reunião de Revisão de Iteração	Reunião de Retrospectiva de Iteração
Developer Controls Process			■			■	■	■	■	■	■	■	■
Engage Customers		■	■	■	■	■						■	
Few Roles	■	■	■	■	■								
Fire Walls	■		■					■					
Get On With It			■			■							
Implied Requirements						■							
Work Queue						■							
Informal Labor Plan			■				■	■	■	■			
Recommitment Meeting	■	■	■					■		■			
Patron Role	■							■					
Size The Organization			■										
Standup Meetings	■	■	■	■	■			■			■		
Gate Keeper		■	■					■					
Time For Reflection													■

O objetivo da Tabela 3.2 é facilitar o entendimento de como os padrões identificados estão relacionados às práticas e papéis Scrum. Por exemplo, o padrão *Fire Walls* está relacionado com os papéis Mestre Scrum e Equipe Scrum e com a prática Iteração, pois durante uma Iteração o Mestre Scrum é responsável por proteger a Equipe Scrum de interações externas. Outro exemplo é o padrão *Standup Meetings*, que está relacionado com as práticas Iteração e Reuniões Diárias Scrum e com os papéis Mestre Scrum, Equipe Scrum, Proprietário do Produto, Cliente e Gerência, pois durante uma Iteração são realizadas as Reuniões Diárias Scrum, executadas pelo Mestre Scrum e Equipe Scrum, com a possível participação do Proprietário do Produto, Cliente e Gerência.

Após investigação do Scrum e de vários padrões, deu-se início ao planejamento da experiência. Cada organização tinha suas características e necessidades particulares, com pouco conhecimento sobre padrões organizacionais e de processo. Para introduzir esses conceitos,

foram utilizados alguns padrões de Manns e Rising (2004), apresentados na Tabela 3.3, que contém o nome do padrão na primeira coluna e a descrição de como ele foi aplicado na segunda.

Tabela 3.3 – Padrões utilizados para planejar e conduzir a experiência

Padrão	Aplicação do Padrão
Evangelist	Para cada uma das etapas da experiência, foi realizado um treinamento, ministrado pelo autor deste trabalho, para que as equipes pudessem entender os princípios da modelagem ágil, métodos ágeis, Scrum e os padrões organizacionais e de processo. Na primeira etapa, foram apresentados os conceitos sobre métodos ágeis e modelagem ágil para as organizações. Já na segunda etapa, Scrum e vários padrões organizacionais e de processo foram abordados.
Just Enough	Em cada treinamento apenas os conceitos fundamentais dos assuntos foram apresentados para as organizações. Os detalhes de cada padrão foram fornecidos após cada organização ter um contanto inicial com essas idéias.
Personal Touch	Antes da realização do treinamento sobre Scrum e padrões organizacionais e de processo na segunda etapa da experiência, os problemas de desenvolvimento de cada organização, que ocorreram na primeira etapa da experiência, foram observados. Esse estudo foi realizado para que as necessidades e problemas de cada organização fossem identificados e, de acordo com esses problemas, alguns padrões organizacionais e de processo foram selecionados para depois serem escolhidos pelas organizações e aplicados junto ao Scrum na segunda etapa da experiência. Essa identificação prévia de alguns padrões serviu como uma pré-seleção para facilitar a escolha das organizações.
External Validation	No treinamento sobre Scrum e padrões organizacionais e de processo, foram apresentados os padrões pré-selecionados e seus usos conhecidos foram reforçados. Além dos exemplos de utilização dos padrões, relatos sobre a aplicação do Scrum em organizações, encontrados em Schwaber e Beedle (2002), também foram apresentados.
Next Steps	No final do treinamento sobre padrões organizacionais e de processo, uma discussão foi realizada para que as organizações analisassem quais padrões poderiam ajudar a melhorar o processo de desenvolvimento delas. Uma lista com todos os padrões pré-selecionados foi fornecida para os membros das organizações para que eles justificassem quais deles seriam os mais apropriados para sua organização. Alguns padrões que poderiam ser integrados não foram escolhidos, sendo induzidas perguntas sobre esses para que as organizações percebessem que eles poderiam solucionar alguns de seus problemas.
Tailor Made	Durante a discussão, ocorrida no final do treinamento sobre padrões organizacionais e de processo e Scrum, cada organização ficou responsável por escolher quais seriam utilizados junto ao Scrum, de acordo com suas necessidades. Assim, as novas idéias puderam ser aplicadas sob medida para cada organização, de acordo seus problemas.
Fear Less	Durante a discussão, ocorrida no final do treinamento sobre padrões organizacionais, de processo e Scrum, cada membro das organizações emitiu sua opinião sobre os padrões apresentados. Isso foi realizado para que os membros céticos das organizações pudessem ser identificados e para que fosse possível entender e aprender com a resistência desses membros. As diferentes opiniões contribuíram para a seleção dos padrões por parte das organizações.

Time For Reflection	Ao final de cada iteração do Scrum, foram realizadas reuniões de retrospectivas de projeto para que as organizações discutissem questões como: Os padrões selecionados estão sendo aplicados de forma correta junto ao Scrum? Outros padrões não selecionados poderiam ser aplicados?

Manns e Rising (2004) afirmam que cada organização pode utilizar os padrões apresentados da Tabela 3.3 de acordo com as suas necessidades, sem levar em consideração a sua ordem na linguagem, o que ocorreu nesta experiência de uso dos padrões organizacionais e de processo com Scrum.

Após o planejamento da experiência, deu-se início à sua realização com as duas organizações. A seção seguinte apresenta algumas considerações sobre ela.

3.4. Condução da Experiência

Na primeira etapa, nenhum método ou processo foi fornecido ou exigido para que as duas organizações realizassem o desenvolvimento, porém, foi solicitado apenas que seguissem os princípios da agilidade (Beck et al., 2001) e da modelagem ágil (Ambler, 2004). Na segunda etapa foi solicitada a utilização do método ágil Scrum e de alguns padrões organizacionais e de processo selecionados pelas próprias organizações. Para cada uma das organizações, foi atribuído um sistema diferente em cada uma das etapas, assim a organização que desenvolveu o sistema de Clínica Veterinária na primeira etapa, desenvolveu o sistema de Equivalências na segunda e vice-versa.

Para não influenciar o desenvolvimento dos sistemas, as equipes não trocaram informações durante a experiência. As tecnologias escolhidas para desenvolvê-los foram: Java (Sun, 2006a) para a camada de negócios e JDBC (*Java Database Connectivity*) (Sun, 2006b) para a camada de dados. Contudo, as duas organizações ficaram livres para utilizar outras tecnologias que julgassem necessárias ou que possuíssem maior conhecimento. Não foi solicitada nenhuma tecnologia específica para a camada de apresentação.

3.4.1. Primeira Etapa da Experiência

A Organização A ficou responsável pelo desenvolvimento do sistema de Clínica Veterinária e optou pelo uso da tecnologia JSP (*JavaServer Pages*) (Sun, 2006c) para a camada de apresentação, enquanto que a Organização B ficou responsável pelo desenvolvimento do sistema de Controle de Equivalências e optou pela tecnologia Swing e pela utilização do *framework* Hibernate (Hibernate, 2006), para auxiliar na persistência de dados.

Como comentado na seção 3.3, o treinamento da primeira etapa abordou conceitos sobre métodos ágeis e modelagem ágil. Sobre modelagem, foi apresentada a linguagem UML (*Unified Modeling Language*) (Larman, 2002), com detalhes sobre seus diagramas: de casos de uso, de classes, de seqüência, de atividades e de estado.

Durante todo o desenvolvimento, as organizações foram observadas e problemas organizacionais e de processo foram identificados pelo autor desta dissertação. Deste modo, visando a segunda etapa da experiência, foram selecionados vinte e seis padrões que poderiam auxiliar na resolução dos problemas observados na primeira etapa. Os principais problemas detectados e os padrões organizacionais e de processo selecionados para solucioná-los são descritos na Tabela 3.4. A primeira coluna apresenta o problema identificado, a segunda mostra os padrões selecionados para resolvê-lo e a terceira apresenta o motivo pelo qual eles foram selecionados.

Tabela 3.4 – Problemas observados nas organizações e os padrões selecionados para resolvê-los

Problema	Padrões Selecionados	Motivo
Organização das equipes	*Unity Of Purpose, Lock 'Em Up Together, Architect Controls Product e Architect Also Implements.*	Para auxiliar as organizações a criar a arquitetura do produto em grupo e refiná-la iterativamente.
Levantamento e Validação de Requisitos	*Scenarios Define Problem, Build Prototypes, Small Writing Team, Breadth Before Depth, Spiral Development e Surrogate Customers.*	Para auxiliar as organizações na comunicação com o cliente para identificar e validar os requisitos do produto.
Distribuição geográfica dos membros das organizações	*Standards Linking Locations, Organization Follows Location e Face To Face Before Working Remotely.*	Para que o impacto da comunicação e colaboração no projeto, devido à distribuição geográfica, fosse diminuído.
Distribuição de tarefas	*Generics And Specifics e Code Ownership*	Para auxiliar aos membros das organizações, que possuem diferentes níveis de conhecimento, a realizar a divisão das tarefas no projeto.
Qualidade do software (integração, teste e validação do software):	*Get Involved Early, Time to Test, Busy System, Document the Problem, Named Stable Bases, Private Workspace, Repository e Application Design is Bounded By Test Design.*	Para auxiliar as organizações a desenvolver um software de qualidade que atenda seus requisitos.
Apresentação para os clientes	*Element Identification, Judicious Fireworks e Light Weight User Interfaces.*	Para auxiliar as organizações a fazer as apresentações do produto que está sendo desenvolvido para o cliente.

Esses padrões, que pertencem a várias linguagens de padrões (Coplien e Harrison, 2004; Bramble et al., 2002; Berczuk e Appleton, 2002; DeLano e Rising, 1998; Coram, 1996; Harrison, 1996) e que documentam práticas comprovadas para solucionar os problemas de desenvolvimento observados, foram selecionados sem participação dos membros das organizações.

A Organização A gastou um total de 387 horas para desenvolver o sistema de Clínica Veterinária, porém esse sistema apresentou problemas de interface, consistência de dados e usabilidade. Já a Organização B conseguiu desenvolver apenas 40% da funcionalidade do sistema de Controle de Equivalências em 147 horas. O sistema foi desenvolvido utilizando a tecnologia Java para a camada de negócios e JDBC para a camada de dados e, os problemas citados na Tabela 3.4 influenciaram diretamente seu desenvolvimento.

Uma vez identificados os problemas ocorridos no desenvolvimento na primeira etapa e realizada a seleção dos padrões, deu-se início a segunda etapa da experiência, que é descrita na próxima subseção.

3.4.2. Segunda Etapa da Experiência

Após o primeiro contato das organizações com os princípios da agilidade na primeira etapa, os sistemas foram trocados entre as organizações. Diferentemente da primeira etapa, a Organização A optou pela tecnologia Swing para a camada de apresentação, enquanto a Organização B continuou utilizando a tecnologia Swing para a camada de apresentação e, além de utilizar *framework* Hibernate, optou pela utilização das ferramentas iReport (iReport, 2006) e JasperReports (JasperReports, 2006) para edição visual e geração dos relatórios.

Nessa etapa, o treinamento abordou o método ágil Scrum e os padrões organizacionais e de processo apresentados na Tabela 3.4, que foram selecionados com base nos problemas observados na primeira etapa da experiência. Após o treinamento, cada organização realizou uma retrospectiva da primeira etapa da experiência , respondendo a duas questões:

- O que poderia ser melhorado (problemas)?
- O que ocorreu de bom no desenvolvimento do primeiro sistema?

Essa retrospectiva teve por objetivo auxiliar as organizações a resolver os problemas por elas enfrentados anteriormente, com o uso dos padrões apresentados. Dessa forma, as organizações escolheram, dentre os padrões selecionados ao final da primeira etapa da experiência, apresentados na Tabela 3.4, um conjunto de padrões de acordo com suas necessidades, como apresentado na Tabela 3.5, que mostra a organização na primeira coluna e os padrões selecionados por ela na segunda.

Tabela 3.5 – Padrões selecionados pelas organizações

Organização	Padrões Selecionados
Organização A	*Organization Follows Location, Face To Face Before Working Remotely, Surrogate Customers, Scenarios Define Problem, Generics And Specifics, Time to Test, Lock 'Em Up Together, Named Stable Bases, Private Workspace, Repository, Smoke Test, Small Writing Team, Breadth Before Depth* e *Spiral Development.*
Organização B	*Organization Follows Location, Face To Face Before Working Remotely, Scenarios Define Problem, Surrogate Customers, Application Design is Bounded By Test Design, Architect Controls Product, Architect Also Implements, Code Ownership, Named Stable Bases, Time to Test, Unity Of Purpose, Private Workspace, Repository, Smoke Test, Small Writing Team, Breadth Before Depth* e *Spiral Development.*

Após essa seleção, deu-se início à atribuição de papéis Scrum e, consequentemente, ao desenvolvimento dos sistemas da seguinte forma: Mestre Scrum: autor desta dissertação; Proprietário do Produto: um membro de cada organização escolhido por ela própria; Equipe Scrum: todos os elementos de cada organização, inclusive o que tem papel de Proprietário do Produto; Cliente: papel desempenhado pela professora da disciplina. Os papéis gerados pela integração dos padrões como Arquiteto e Substituto do Cliente foram realizados por membros das organizações.

Todas as práticas Scrum foram seguidas, com exceção das Reuniões Diárias, que foram realizadas semanalmente, devido à distribuição geográfica dos membros das organizações que não estavam em tempo integral na universidade e se dedicavam parcialmente a essa disciplina, devido às outras atividades do programa de mestrado ou profissionais. Essa prática não é exclusiva desse estudo, pois Rising e Janoff (2000) relatam suas experiências na aplicação do Scrum em que as reuniões diárias também foram adaptadas.

O Scrum e os padrões foram aplicados juntos, porém não foi realizada uma integração entre eles, ou seja, não foi feito um estudo mais detalhado para identificar como as práticas propostas pelos padrões estão relacionadas com as práticas propostas pelo Scrum. Isso foi realizado para verificar se a simples aplicação dos padrões não integrados ao Scrum é suficiente ou se é necessário racionalizar a utilização.

Com o Scrum e os padrões, diferentemente da primeira etapa, cada sistema foi entregue com sua funcionalidade completamente implementada. Nessa etapa, a Organização A desenvolveu o sistema de Controle de Equivalências em 381 horas, enquanto a Organização B desenvolveu o sistema de Clínica Veterinária em apenas 195 horas. O que indicou uma melhor organização das Organizações A e B, que conseguiram cumprir o cronograma e entregar os sistemas com as funcionalidades solicitadas pelos clientes.

A próxima seção apresenta a análise realizada sobre os resultados obtidos na experiência.

3.5. Análise dos Resultados da Experiência

Há indícios de que, com a utilização do Scrum juntamente com padrões organizacionais e de processo, os resultados obtidos foram melhores no desenvolvimento do sistema de Clínica Veterinária do que no sistema de Controle de Equivalências, considerando o cronograma e a qualidade do software desenvolvido. As próximas subseções apresentam uma comparação entre esses resultados.

3.5.1. Sistema de Clínica Veterinária

Na primeira etapa da experiência, a Organização A desenvolveu o sistema, em 387 horas, atendendo completamente a funcionalidade solicitada. Já na segunda etapa, com a utilização dos padrões junto com o Scrum, esse mesmo sistema também foi completamente desenvolvido, agora pela Organização B, em 195 horas, ou seja, aproximadamente 53% do tempo gasto pela Organização A. Isso indica uma evolução em direção à agilidade no processo de desenvolvimento, como pode ser observado no gráfico da Figura 3.1. O eixo vertical mostra a quantidade de horas gastas para desenvolver o sistema e o eixo horizontal mostra a etapa na qual o sistema foi desenvolvido.

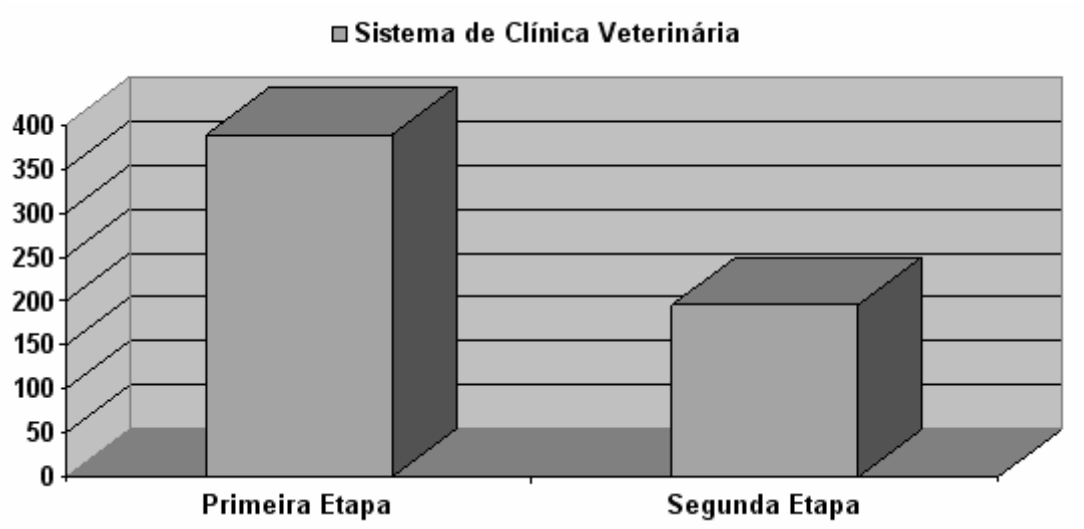


Figura 3.1 – Tempo de desenvolvimento do sistema de Clinica Veterinária

Embora o sistema tenha sido completamente desenvolvido pelas duas organizações, na primeira etapa foram observados problemas de interface, consistência de dados e usabilidade, o que não ocorreu na segunda.

A forma como Scrum foi utilizado, juntamente com os padrões organizacionais e de processo, indica melhorias de qualidade e prazo de desenvolvimento. O Mestre Scrum observou a organização B, na segunda etapa, para que pudesse identificar problemas com a aplicação dos padrões organizacionais e de processo juntamente com o Scrum. Notou-se que a aplicação ocorreu de forma adequada, pois a Equipe Scrum inicialmente entendeu a relação entre as práticas propostas pelos padrões com as práticas Scrum. Porém, esse entendimento não foi imediato, pois a organização apresentou dificuldades para identificar essa relação.

Salienta-se que o padrão *Application Design is Bounded By Test Design*, anteriormente selecionado pela organização B, não foi aplicado, o que acarretou um mau planejamento das atividades de testes.

Dessa forma, no desenvolvimento do sistema de Clínica Veterinária os resultados foram satisfatórios e mostraram que, com a aplicação dos padrões organizacionais e de processo junto com o Scrum, o sistema foi desenvolvido mais rapidamente e com melhor qualidade. Porém, isso não pôde ser observado no desenvolvimento do sistema de Controle de Equivalências, apresentado na próxima subseção.

3.5.2. Sistema de Controle de Equivalências

O sistema de Controle de Equivalências foi desenvolvido na primeira etapa pela Organização B em 147 horas, sendo entregue apenas 40% de sua funcionalidade devido aos problemas de organização e distribuição de tarefas entre desenvolvedores. Já na segunda etapa, com a utilização dos padrões junto com o Scrum, esse mesmo sistema foi desenvolvido completamente, pela Organização A, em 381 horas. Nesse caso, torna-se difícil analisar a efetividade da utilização ou não de padrões junto com o Scrum, uma vez que apenas uma organização desenvolveu o sistema completamente.

Supondo que, na primeira etapa, a Organização B continuasse no mesmo ritmo de trabalho, proporcionalmente, o tempo gasto para desenvolver completamente o sistema de Controle de Equivalências seria de aproximadamente 368 horas, ou seja, 221 horas a mais do que já tinham sido gastas. Isso indica que, ainda assim, o tempo para desenvolver esse sistema sem o Scrum e padrões seria menor do que aplicando-os.

Esse resultado é conseqüência da aplicação inadequada dos padrões junto ao Scrum. O Mestre Scrum observou a organização A, na segunda etapa, para que pudesse identificar

problemas com a aplicação dos padrões organizacionais e de processo juntamente com o Scrum. Notou-se que essa organização aplicou os padrões de forma desorganizada e descuidada, ou seja, faltou entendimento de como as práticas propostas pelos padrões devem ser relacionadas com as práticas propostas pelo Scrum. Dois erros foram cometidos pela Organização A na aplicação dos padrões:

1) De acordo com o padrão *Lock 'Em Up Together*, a reunião para definição da arquitetura deve ser realizada apenas no início do projeto e a arquitetura deve evoluir naturalmente durante o desenvolvimento nas iterações. Porém, a Organização A realizou várias reuniões para definição da arquitetura do produto ao longo das iterações, o que não é sugerido pelo padrão e muito menos pelo Scrum. No Scrum, as reuniões que devem ocorrer durante as iterações são diárias (semanais, neste caso), de aproximadamente quinze minutos, para acompanhamento do projeto e não para discussão da arquitetura do produto. Essa utilização inadequada do padrão junto ao Scrum impactou diretamente o cronograma, pois foram gastas 33 horas a mais do que deveriam ser gastas com as reuniões para elaboração da arquitetura.

2) O Scrum permite a criação de artefatos além dos definidos por ele (Larman, 2003), porém esses artefatos devem agregar valor ao produto e devem ser criados para auxiliar o desenvolvimento, e não para serem colocados em um repositório para possível utilização futura, pois software funcionando é mais importante que documentação detalhada (Beck et al., 2001). De acordo com os padrões selecionados pela Organização A, apenas três novos artefatos deveriam ser gerados durante o desenvolvimento do sistema: casos de uso, arquitetura e bases estáveis do software. Porém, apesar de nenhum outro artefato ter sido solicitado pelo cliente, essa organização optou pela criação de diagramas de seqüência e de entidade relacionamento do banco de dados, além dos artefatos determinados pelo Scrum e padrões. Isso é não incorreto, desde que não sejam gastas muitas horas na sua criação e manutenção. Entretanto, foram gastas 26 horas na criação e manutenção desses artefatos, o que mostra que os princípios do Scrum não formam respeitados. Além disso, a Organização A gastou 12 horas para realizar a revisão da arquitetura do produto, no final do desenvolvimento, o que não era necessário, pois o produto já tinha sido desenvolvido.

Foram gastas um total de 71 horas com reuniões e documentação excessivas, ou seja, o sistema ao invés de ter sido desenvolvido em 381 horas, poderia ter sido desenvolvido em aproximadamente 310 horas. Considerando esses valores, o tempo para desenvolver o sistema de Controle de Equivalências com Scrum junto com padrões seria aproximadamente 15,5% menor em relação à primeira etapa. A Figura 3.2 mostra, em porcentagem, a quantidade de horas

utilizadas pela Organização A para criar todos os artefatos durante o desenvolvimento do sistema.

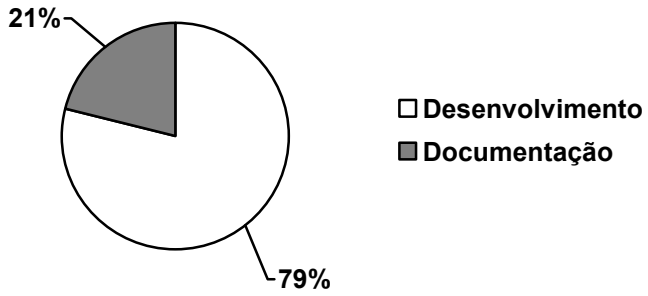


Figura 3.2 – Tempo gasto com documentação pela Organização A

Na segunda etapa da experiência, ambas organizações entregaram os sistemas de forma completa e funcionando para o cliente e isso indicou um progresso, pois software funcionando é a medida básica de progresso (Beck et al., 2001).

Pôde-se inferir também que os resultados não foram melhores devido à forma como as organizações foram divididas. Não foi considerada a afinidade entre os alunos na divisão das organizações. Isso gerou alguns problemas de relacionamento dentro das organizações, o que impactou na produtividade. A princípio, os alunos seriam divididos de acordo com o padrão *Self Selecting Team* (Coplien e Harrison, 2004), que sugere que os interesses das pessoas devem ser considerados na formação das equipes, e que elas devem selecionar a equipe na qual querem trabalhar. Segundo esse padrão, as equipes de pior dinâmica são aquelas estabelecidas, porém, por falta de conhecimento do perfil de cada membro, eles foram divididos aleatoriamente. Além disso, outro fator relevante que impactou o resultado da experiência foi o comprometimento dos membros das organizações. Alguns se esforçaram para melhorar o processo e produto desenvolvido, porém, nas duas organizações, alguns alunos não se envolveram com a disciplina de forma satisfatória e assim, essa falta de comprometimento afetou a produtividade das organizações, pois para que a aplicação do Scrum adaptado com os padrões seja realizada corretamente é necessário comprometimento de todas as pessoas envolvidas no processo.

A distribuição geográfica dos membros das organizações também impediu uma melhor organização e produtividade, já que nem todos os padrões que tratam de problemas de comunicação e colaboração em projetos distribuídos (*Standards Linking Locations, Organization Follows Location e Face To Face Before Working Remotely*) foram utilizados.

3.5.3. Opinião dos Membros das Organizações

Produtividade e qualidade do produto são importantes, porém também é importante obter o *feedback* dos membros das organizações envolvidas na experiência. Assim, um questionário foi aplicado ao término da segunda etapa às organizações para obter suas opiniões sobre o Scrum e os padrões utilizados. O questionário abordou questões como a utilização do Scrum e a utilidade dos padrões. Alguns resultados importantes são comentados a seguir:

- Todos os membros se mostraram satisfeitos com a utilização dos padrões organizacionais e de processo junto ao Scrum e mencionaram que os padrões foram úteis para o desenvolvimento.
- 100% dos membros citaram que a divisão dos requisitos em itens priorizados, que compõe a lista de Trabalho do Produto (*Product Backlog*), e a divisão desses itens em pequenas listas de Trabalho da Iteração (*Sprint Backlog*) que devem ser desenvolvidas nas iterações, são as práticas mais relevantes do Scrum.
- 80% dos membros mencionaram que os padrões organizacionais e de processo contribuíram para uma melhor organização das equipes e divisão das tarefas entre seus membros.

Apesar da melhoria na produtividade das organizações e da qualidade do produto final, problemas foram identificados na aplicação dos padrões organizacionais e de processo juntamente com o Scrum. Para que essa utilização seja realizada de forma apropriada, é necessário entender o relacionamento das práticas Scrum com as práticas propostas pelos padrões. Por esse motivo, observou-se a necessidade de racionalizar essa aplicação dos padrões organizacionais e de processo ao Scrum para facilitar a utilização conjunta dessas duas abordagens.

3.6. Considerações Finais

O Scrum pode ser adaptado de acordo com as necessidades da organização que o utiliza, devido a sua flexibilidade. Uma forma efetiva de realizar essas modificações no Scrum, observada neste capítulo, é aplicar padrões organizacionais e de processo a esse método. Esses padrões fornecem soluções comprovadas que agregam qualidade ao processo, como mostrado na Seção 2.4.2. Existem vários padrões disponíveis na literatura e, embora alguns tenham influenciado métodos ágeis como Scrum e XP, desde sua criação, novos padrões surgiram e outros ainda irão surgir, pois organizações diferentes podem necessitar de novos padrões ou de

diferentes versões dos já existentes. Assim, aplicar padrões ao método ágil Scrum significa ajustá-lo às necessidades da organização, por meio de soluções de sucesso.

Com a introdução das Reuniões de Retrospectiva de Iteração no Scrum (Schwaber, 2004), detalhada na Seção 2.3.1, foi possível realizar o uso dos padrões com o Scrum de forma mais real, pois nessas reuniões as organizações tentaram aprender com os erros do passado, o que permitiu que elas identificassem os problemas enfrentados para desenvolver o software e selecionassem padrões organizacionais e de processo que os resolvessem. Assim, a Reunião de Retrospectiva de Iteração é o período ideal para selecionar um padrão e melhorar o processo de desenvolvimento de software e a organização.

Um problema verificado nas duas organizações, porém de forma mais nítida na Organização A, foi a dificuldade de pensar de forma ágil, ou seja, apesar das duas organizações terem desenvolvido os sistemas completamente, utilizando o Scrum e os padrões organizacionais e de processo em conjunto, foi difícil mudar a concepção de desenvolvimento tradicional, que é voltado para documentação, para o ágil. Com a utilização dos padrões organizacionais e de processo com o Scrum, novos artefatos, além dos definidos por ele, foram gerados e utilizados, o que facilitou a construção do software. Porém, pôde-se observar que a Organização A gastou muito tempo com documentação desnecessária, como mostrado na Seção 3.5.2.

Pôde-se observar também que a Organização B, formada por membros com experiência no mercado de trabalho, conseguiu entender melhor o relacionamento entre as práticas do Scrum e as práticas propostas pelos padrões do que a Organização A, formada por membros recém graduados. Isso indica que o perfil da organização pode ter influenciado o resultado da aplicação dos padrões organizacionais e de processo junto ao Scrum.

Embora a experiência tenha indicado que o uso de padrões organizacionais e de processo com o Scrum pode agilizar o processo de desenvolvimento e melhorar a qualidade do produto final, para se utilizar as práticas propostas por esses padrões de forma efetiva com às do Scrum, é necessário entender o relacionamento entre elas. Desta forma, observou-se a necessidade de racionalizar o uso conjunto dos padrões organizacionais e de processo com o Scrum. A simples aplicação do padrão não integrado ao método não é suficiente para que essa utilização seja realizada de forma efetiva, como foi observado na Organização A, como mostrado na Seção 3.5.2. Além disso, foi observado que mesmo a Organização B, que utilizou os padrões de forma adequada, teve dificuldades de entender os relacionamentos entre as práticas propostas pelos padrões e as práticas do Scrum.

O Capítulo 4 apresenta uma forma ordenada para utilizar padrões organizacionais e de processo com o Scrum, cuja necessidade foi observada como mencionado na Seção 3.5.2.

Capítulo 4

Uma Forma Ordenada para Utilização dos Padrões Organizacionais e de Processo com Scrum

4.1. Considerações Iniciais

Uma questão que surgiu durante a experiência de uso dos padrões organizacionais e de processo com Scrum, apresentada no Capítulo 3, é como utilizá-los em conjunto de forma eficiente.

O principal objetivo dos padrões organizacionais e de processo é o reúso de soluções de sucesso para solução de problemas de desenvolvimento de software. Para que esse reúso possa ser realizado de forma efetiva junto ao Scrum, é necessário entender como as práticas propostas por esses padrões se relacionam com as práticas desse método. A utilização incorreta dos padrões pode prejudicar a organização e atrasar o projeto ao invés de agilizá-lo, como foi observado na experiência apresentada no Capítulo 3.

Esse fato motivou a criação de uma forma ordenada para facilitar a utilização de padrões organizacionais e de processo ao método ágil Scrum. Essa utilização pode ser realizada de forma efetiva integrando os padrões organizacionais e de processo ao Scrum. A integração aqui proposta é organizada em quatro etapas: modelagem do Scrum sem padrões com a utilização do SPEM; identificação da categoria dos padrões que podem ser integrados ao Scrum; associação entre esses padrões e as disciplinas do Scrum; integração dos padrões organizacionais de processo para extensão do Scrum.

Este capítulo trata na Seção 4.2 de conceitos de modelagem de processo, em especial, o meta-modelo SPEM. As Seções de 4.3 a 4.6 descrevem as etapas da integração proposta. Assim, na seção 4.3 é apresentada a modelagem do Scrum sem padrões, utilizando o meta-modelo SPEM; na Seção 4.4 a identificação da categoria dos padrões que são integrados ao Scrum é abordada; na Seção 4.5 é apresentada a associação entre os padrões e as disciplinas do Scrum; na Seção 4.6 é apresentada a integração dos padrões organizacionais de processo para extensão do Scrum e, na Seção 4.7 estão as considerações finais.

4.2. Modelagem de Processo

Processo de software é um conjunto de atividades parcialmente ordenadas, executadas para gerenciar, desenvolver e manter sistemas de software (Acuña e Ferré, 2001). O processo de software é formado por diversos elementos e pela definição específica de seus relacionamentos. Os elementos comuns em processos de software, de acordo com Conradi et al. (1994) e Acuña e Ferré (2001) são:

- **Ator ou Agente**: é a entidade que executa o processo. Atores podem ser divididos em dois grupos, de acordo com seu relacionamento com o computador, atores humanos e atores de sistema. Os atores humanos são as pessoas que estão envolvidas no processo de desenvolvimento de software e que estão, possivelmente, organizadas em equipes. Os atores de sistema ou ferramentas do sistema são componentes de software ou hardware.
- **Papel**: descreve as responsabilidades e habilidades necessárias para realizar uma atividade específica do processo de software.
- **Atividade**: representa o trabalho realizado por um ator ou grupo de atores. É o estágio do processo que produz mudanças externas visíveis de estado no software que está sendo construído. Uma atividade pode necessitar de um artefato de entrada, e pode gerar um artefato de saída.
- **Artefato:** é o subproduto produzido e mantido no processo. Os artefatos podem ser gerados, consumidos e modificados em uma atividade, e, assim, podem ter várias versões ao longo do processo. Os artefatos que devem ser entregues ao cliente ou usuário, são chamados de produtos de software.

Um modelo de processo de software é uma representação abstrata da definição do processo de software (Acuña e Ferré, 2001), que representa o relacionamento entre os elementos do processo. As vantagens de se modelar um processo, segundo Curtis et al. (1992), são:

- **Entendimento e comunicação**: permite a organização dos elementos do processo e a representação de seus relacionamentos, possibilitando que todos os envolvidos no processo tenham a mesma visão dele.
- **Gestão e controle do processo**: permite e facilita o monitoramento, gerenciamento e coordenação das atividades do processo.
- **Suporte para melhoria de processo:** todos os envolvidos têm a possibilidade de sugerir adaptações e melhorias devido ao maior conhecimento do processo.

- **Suporte para automação:** permite a identificação de partes do processo que podem ser automatizadas.

Para Finkelstein et al. (1994), modelo de processo é a descrição de um processo, representado em uma linguagem de modelagem adequada. Nos últimos anos, várias linguagens para modelagem de processo foram criadas. Heimann et al. (1996) definem uma linguagem baseada em conceitos UML e redes de tarefas para modelagem de processo de software, enquanto Jaccheri et al. (1998) propõem uma linguagem orientada a objetos e uma ferramenta de apoio para modelagem de processos.

Devido ao surgimento de várias linguagens de modelagem de processo de software, diversas empresas como a Alcatel, IBM, Unisys, entre outras, se uniram para elaborar um meta-modelo que pudesse atender aos requisitos definidos pelo *Object Management Group* (OMG). Um meta-modelo é um modelo que define a linguagem para expressar um modelo (Rational Software Corporation, 2002). Em 2002 o meta-modelo SPEM (*Software Process Engineering Metamodel*), baseado na UML, foi oficializado pela OMG (OMG, 2005) e é apresentado de forma resumida na próxima subseção, por ter sido utilizado para modelar o Scrum neste trabalho.

4.2.1. SPEM - Software Process Engineering Metamodel

SPEM (OMG, 2005) é um meta-modelo que estende a UML para especificação de processos de desenvolvimento de software. Sua versão atual é a 1.1, definida com base em MOF 1.3 (*Meta-Object Facility*) (OMG, 1999), que é a tecnologia adotada pela OMG para definir metadados, e na UML 1.4 (OMG, 2001).

A OMG adota uma abordagem orientada a objetos e utiliza a notação UML com um conjunto de estereótipos específicos para descrição de processos (OMG, 2005). A Figura 4.1 apresenta a arquitetura para modelagem em quatro camadas definida pela OMG.

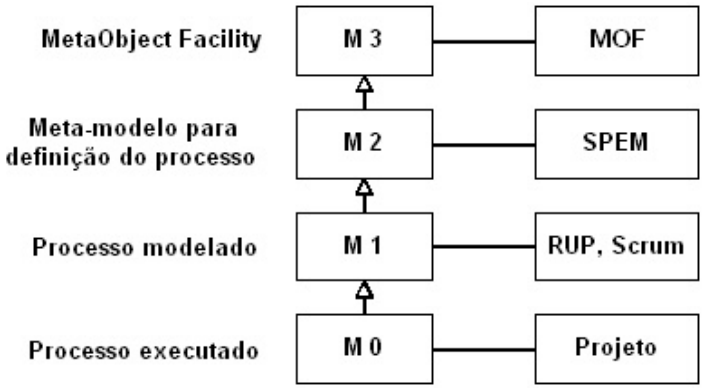


Figura 4.1 – Níveis de modelagem (adaptada de OMG, 2005)

O nível M0 representa como o processo é executado na realidade. A definição do processo está no nível M1, como, por exemplo, o RUP (Rational Software Corporation, 2002), que já é modelado com SPEM e o Scrum (Schwaber, 2004). O meta-modelo para definição do processo, nesse caso o SPEM, que está no nível M2, serve de modelo para o nível M1. O SPEM é estruturado como um UML *profile*, como mostra a Figura 4.2, e define um conjunto de estereótipos para descrição de processos que estão no nível M1. Além de ser estruturado na UML, o SPEM também é fundamentado no MOF, que se encontra no nível M3.

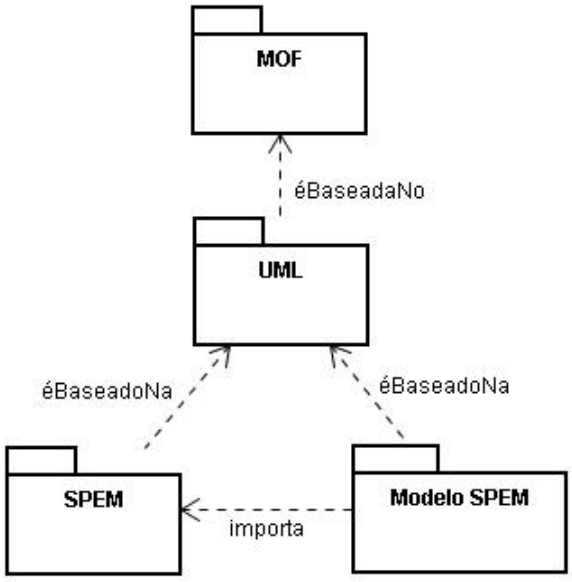


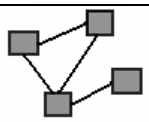
Figura 4.2 – SPEM como UML *profile*

Os princípios fundamentais do SPEM são provenientes da idéia que processo é uma colaboração entre entidades abstratas ativas, chamadas de papéis de processo (*process roles*), que realizam operações chamadas de atividades (*activities*) em entidades chamadas de produtos de trabalho (*work products*). Os relacionamentos entre esses elementos são apresentados na Figura 4.3, por meio de um diagrama de classes UML.

Figura 4.3 – Modelo conceitual SPEM (Adaptado de OMG, 2005)

Essa figura apresenta o relacionamento entre os estereótipos fundamentais de processo, porém, o SPEM define outros para modelagem de processo. Os estereótipos SPEM, utilizados neste trabalho, são apresentados de forma resumida na Tabela 4.1. A primeira coluna contém o nome do estereótipo, a segunda apresenta a sua descrição e a terceira mostra a notação utilizada para representá-lo.

Tabela 4.1 – Estereótipos definidos no SPEM (OMG, 2005)

Estereótipo	Descrição	Notação
Artefato (*WorkProduct*)	É qualquer coisa produzida, consumida ou modificada por um processo. Um artefato pode pertencer a uma categoria específica de artefato (CategoriaDeArtefato) (Figura 4.4), como, por exemplo, Documento (*Document*), Modelo UML (*UMLModel*), entre outras.	
Documento (*Document*)	É uma Categoria de Artefato (*WorkProductKind).*	
Modelo UML (*UMLModel*)	É uma Categoria de Artefato (*WorkProductKind)*.	
Conjunto de Trabalho (*WorkDefinition*)	Descreve o trabalho realizado no processo. Pode ser criado para representar partes compostas do trabalho que serão futuramente separadas, além de poder ser composto por outros conjuntos de trabalho. Atividade é a principal subclasse de conjunto de trabalho (Figura 4.4).	
Atividade (*Activity*)	Descreve parte do trabalho que deve ser realizado por um papel (Figura 4.4). Uma atividade pode ser composta de passos.	
Papel (*ProcessRole*)	Define a responsabilidade sobre produtos de trabalho específicos e papel de um determinado indivíduo no processo (Figura 4.4).	
Guia (*Guidance*)	Provê informações adicionais sobre o elemento do processo ao qual está associado. Um guia pode pertencer a uma categoria de guia (CategoriaDeGuia) (Figura 4.4), como, por exemplo: Normas, Técnicas, entre outras.	
Disciplina (*Discipline*)	É um conjunto elementos de processo agrupados de forma coerente, de acordo com um tema comum (Figura 4.4), como, por exemplo: Gestão de Requisitos, Análise e Implementação.	
Processo (*Process*)	É a descrição completa de um processo de desenvolvimento de software, em termos de papéis, artefatos, conjuntos de trabalho, atividades e guias (Figura 4.4).	

Esses estereótipos podem ser utilizados com digramas UML para representar diferentes perspectivas do processo de software. Segundo a OMG (2005), as seguintes notações UML são úteis para representar o processo de software: Diagrama de Classe; Diagrama de Pacote;

Diagrama de Atividades; Diagramas de Caso de Uso; Diagramas de Seqüência; Diagramas de Estados.

Os diagramas de classes possibilitam representar a perspectiva estática do processo, descrevendo quais artefatos devem ser gerados ou consumidos pelas atividades e quem são os responsáveis por cada atividade, enquanto que os diagramas de atividades permitem representar a perspectiva dinâmica do processo, descrevendo a seqüência em que elas devem ocorrer. Assim, no restante deste trabalho, a perspectiva estática do Scrum é modelada por meio de digramas de classes, enquanto que a perspectiva dinâmica é modelada por meio de diagramas de atividades.

A próxima seção apresenta a modelagem do processo Scrum sem padrões utilizando o meta-modelo SPEM.

4.3. Modelagem do Scrum sem Padrões

Scrum é um método ágil que define um processo para gerenciar e controlar o desenvolvimento de software, utilizando práticas iterativas e incrementais (Schwaber, 2004; Schwaber e Beedle, 2002). Esse método estabelece um conjunto de atividades, papéis e artefatos que compõem seu processo.

Para facilitar a compreensão de como os elementos do processo Scrum estão relacionados nos modelos criados com SPEM (OMG, 2005), foi definido neste trabalho o modelo básico do Processo Scrum, ou meta-modelo Scrum, como mostra a Figura 4.4.

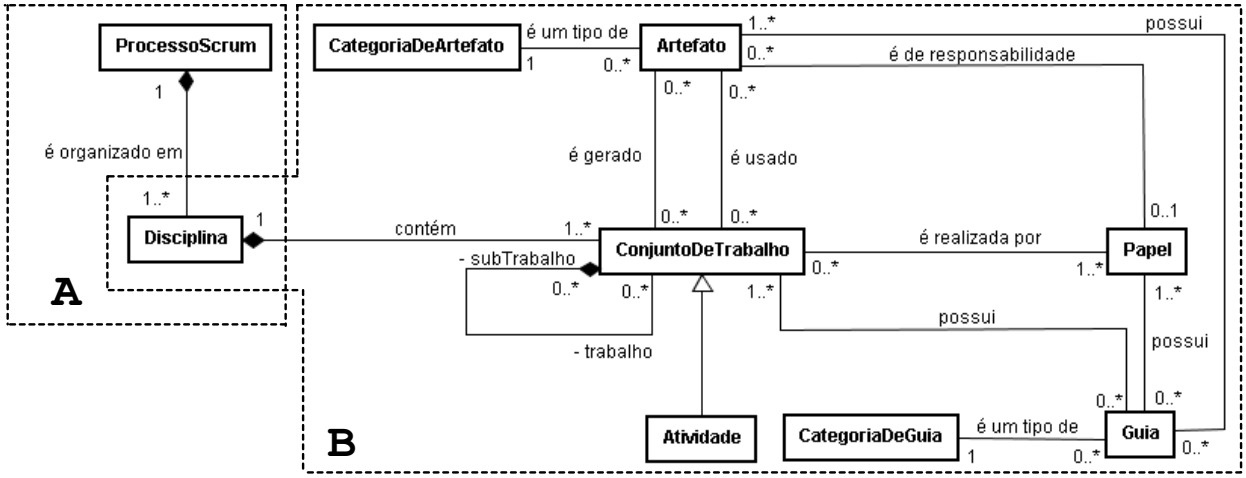


Figura 4.4 – Meta-modelo Scrum

Por meio desse meta-modelo Scrum é possível verificar como os elementos do processo Scrum estão organizados e relacionados na modelagem com SPEM.

O processo Scrum (classe ProcessoScrum) é organizado em disciplinas (classe Disciplina). Cada disciplina contém conjuntos de trabalho (classe ConjuntoDeTrabalho) e atividades (classe Atividade), que se relacionam com papéis (classe Papel), artefatos (classe Artefato) e guias (classe Guia). Um conjunto de trabalho não é uma classe abstrata e suas instâncias representam partes compostas de trabalho e, além disso, um conjunto de trabalho pode ser composto de outros conjuntos de trabalho (OMG, 2005). Os conjuntos de trabalho estão relacionados com artefatos e papéis, ou seja, eles são executados por um papel e podem gerar ou consumir artefatos. As atividades herdam do conjunto de trabalho os relacionamentos com artefatos e papéis, e apesar de cada atividade e conjunto de trabalho ser executado por um papel, papéis adicionais podem ser associados para representar auxiliares na execução do trabalho. Cada artefato pertence a uma categoria de artefato (classe CategoriaDeArtefato) e pode possuir um papel como responsável. Os guias (classe Guia) podem ser relacionados com conjuntos de trabalho, atividades, artefatos e papéis no processo, e da mesma forma que os artefatos, os guias pertencem a categorias (classe CategoriaDeGuia).

Toda a modelagem do Scrum foi realizada com base nas informações obtidas nos trabalhos de Schwaber (2004), Schwaber e Beedle (2002) e Schwaber (1995). As disciplinas identificadas no processo Scrum são: Definir Papéis; Obter Requisitos; Planejar Iteração; Realizar Iteração; Revisar Iteração; Finalizar Versão. Essas disciplinas são apresentadas, por meio de um diagrama de pacotes e dos estereótipos SPEM, na Figura 4.5, que se refere à parte A da Figura 4.4.

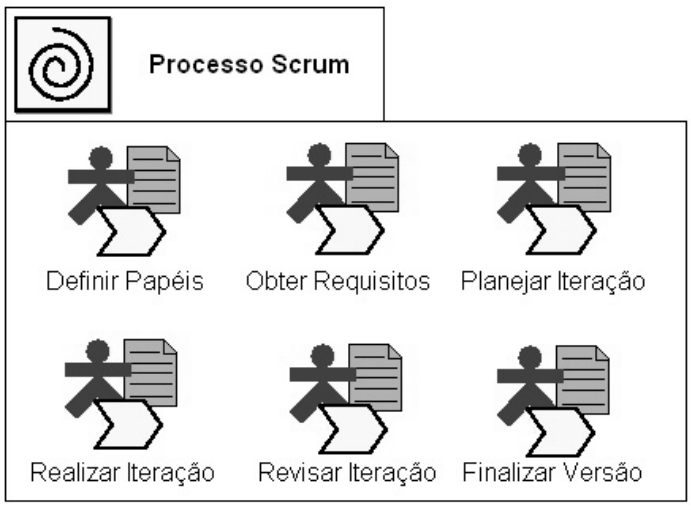


Figura 4.5 – Diagrama de pacotes do processo Scrum

A organização das disciplinas Scrum está representada no diagrama de atividades da Figura 4.6, que também se refere à parte A da Figura 4.4.

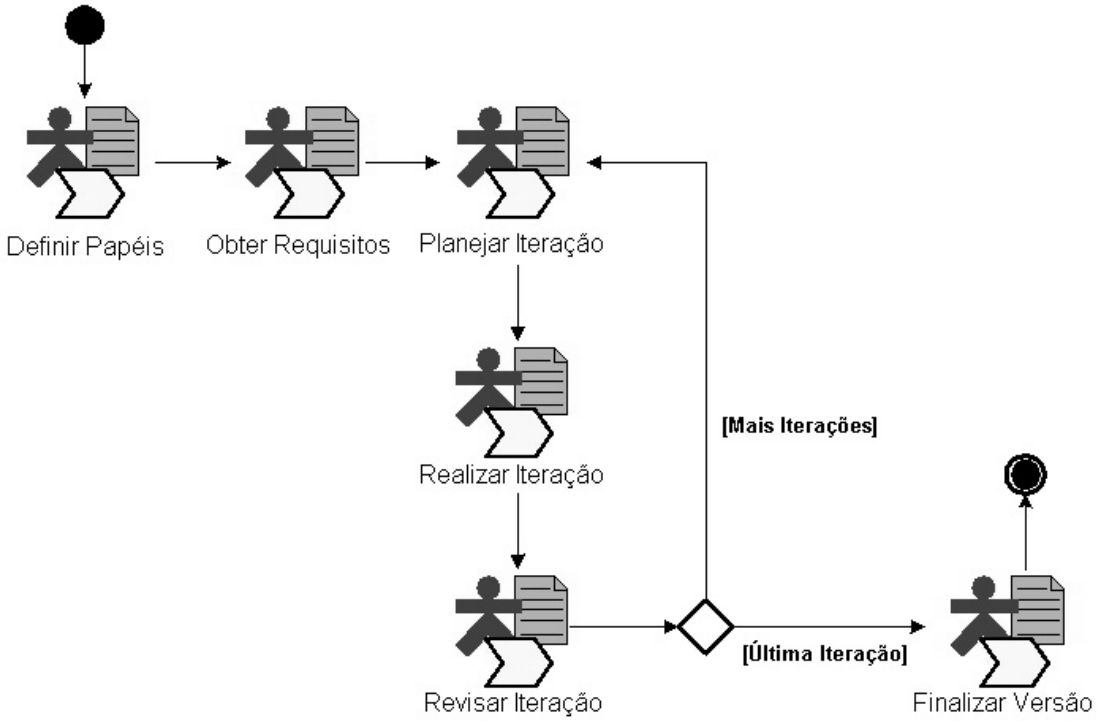


Figura 4.6 – Diagrama de atividades do processo Scrum

O processo Scrum tem início com a definição de alguns papéis que serão desempenhados durante o processo (disciplina Definir Papéis). Em seguida, inicia-se a obtenção de requisitos (disciplina Obter Requisitos) junto ao cliente, porém, coletando-se somente os requisitos conhecidos pelo cliente na fase inicial do projeto, para possibilitar o começo do desenvolvimento. O Scrum não tenta identificar todos os requisitos logo no início do projeto, ao invés disso, ele opta pela adaptabilidade, ou seja, controlar as mudanças de requisitos ao longo do projeto, por meio de um planejamento contínuo, que é realizado na etapa de planejamento das iterações. Cada iteração é planejada (disciplina Planejar Iteração) e depois executada (disciplina Realizar Iteração). No final de cada iteração é realizada uma reunião de revisão de iteração (disciplina Revisar Iteração) para apresentação do incremento do produto criado. Assim, as iterações abrangem as disciplinas de Planejar, Realizar e Revisar Iteração, como mostra a Figura 4.6. Após a realização de todas as iterações necessárias para desenvolver o produto, a versão final do software é concluída (disciplina Finalizar Versão).

As próximas subseções apresentam a modelagem estática (diagrama de classes) e dinâmica (diagrama de atividades) de cada uma das disciplinas do Scrum, utilizando os estereótipos SPEM, apresentados na Tabela 4.1. O texto dessas subseções refere-se às práticas e papéis preconizados pelo Scrum, que podem ser observadas nas Tabelas 2.1 e 2.2 do Capítulo 2. Os diagramas de classe e atividades se referem à parte B da Figura 4.4.

4.3.1. Disciplina Definir Papéis

No início do projeto, o Mestre Scrum, o Cliente e a Gerência devem definir quem será o Proprietário do Produto no projeto. Em seguida, o Mestre Scrum e a Gerência devem selecionar os indivíduos que vão compor a Equipe Scrum. A Figura 4.7 mostra o diagrama de classes da disciplina Definir Papéis com os estereótipos SPEM.

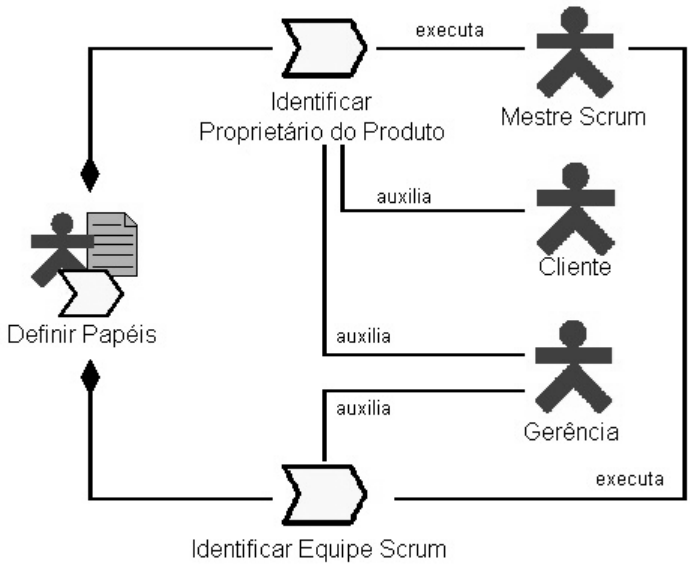


Figura 4.7 – Diagrama de classes da disciplina Definir Papéis

Como apresentado na Figura 4.4, as disciplinas no Scrum são compostas de conjuntos de trabalho e atividades, que se relacionam com artefatos e papéis. Na Figura 4.6 e nos demais diagramas de classes das disciplinas Scrum, o relacionamento de composição (◆——) é utilizado para representar os conjuntos de trabalho e atividades que compõem cada disciplina.

A Figura 4.7 mostra as atividades que compõem a disciplina Definir Papéis (Identificar Proprietário do Produto e Identificar Equipe Scrum) e os papéis relacionados com essas atividades (Mestre Scrum, Cliente e Gerência). Nesse caso, nenhum artefato é gerado ou consumido pelas atividades, que devem ser executadas em uma ordem, como mostra o diagrama de atividades da disciplina Definir Papéis com os estereótipos SPEM da Figura 4.8.

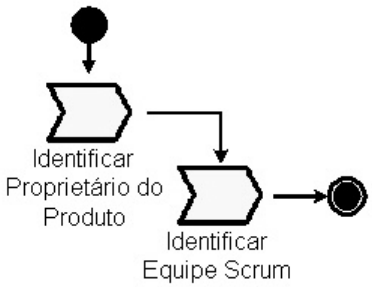


Figura 4.8 – Diagrama de atividades da disciplina Definir Papéis

Depois de realizadas as atividades de identificação de papéis, pode-se dar início ao levantamento de requisitos junto ao cliente, como mostra a próxima subseção.

4.3.2. Disciplina Obter Requisitos

O cliente deve expor os requisitos desejados do produto a ser construído para o Mestre Scrum, Equipe Scrum e Proprietário do Produto. Depois de identificados, os requisitos iniciais são divididos em itens e colocados em uma lista chamada de lista de Trabalho do Produto. Cada item da Lista de Trabalho do Produto é então priorizado pelo cliente e depois avaliado, quanto ao esforço para desenvolvê-lo. Após a priorização e estimativa do trabalho, o Proprietário do Produto pode validar as ferramentas e infra-estrutura do projeto.

A Figura 4.9 mostra o diagrama de classes da disciplina Obter Requisitos com os estereótipos SPEM.

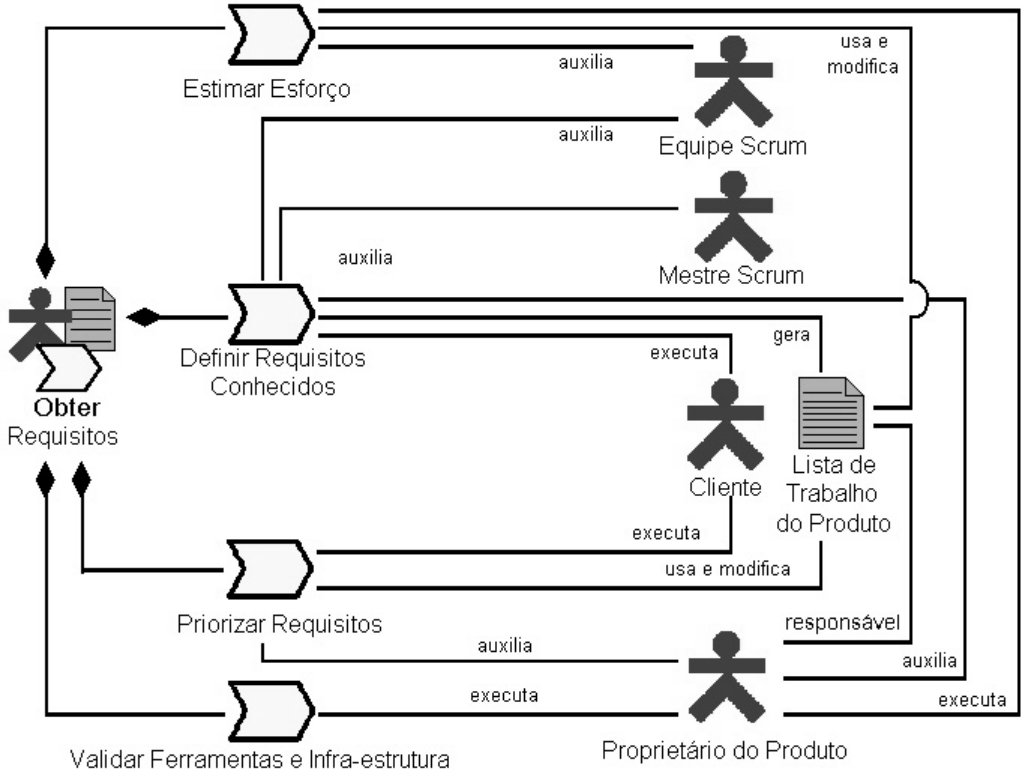


Figura 4.9 – Diagrama de classes da disciplina Obter Requisitos

Essa figura mostra as atividades que compõem a disciplina Obter Requisitos (Definir Requisitos Conhecidos, Priorizar Requisitos, Estimar Esforço e Validar Ferramentas e Infra-estrutura), os papéis relacionados com essas atividades (Proprietário do Produto, Mestre Scrum, Equipe Scrum e Cliente), e o artefato gerado pelas atividades (Lista de Trabalho do Produto), que tem como responsável o Proprietário do Produto.

A Figura 4.10 mostra o diagrama de atividades da disciplina Obter Requisitos com os estereótipos SPEM.

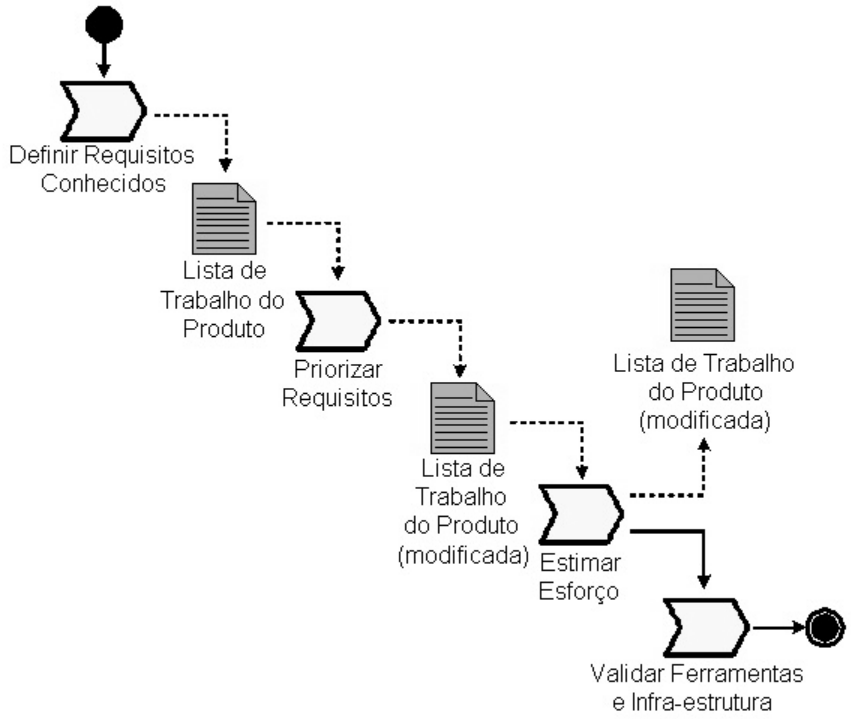


Figura 4.10 – Diagrama de atividades da disciplina Obter Requisitos

As setas contínuas indicam o encadeamento de duas atividades, enquanto as setas tracejadas indicam o encadeamento entre uma atividade e um artefato. Nesse caso, o único artefato envolvido nas atividades é a Lista de Trabalho do Produto, que é modificada ao longo das atividades e cuja criação permite dar início ao planejamento da primeira iteração no projeto, como descrito na próxima subseção.

4.3.3. Disciplina Planejar Iteração

Antes de cada iteração, ocorre uma reunião de planejamento, na qual várias atividades devem ser realizadas. Nessa reunião, executada pelo Proprietário do Produto, com a participação da Equipe Scrum e Mestre Scrum, a Equipe Scrum deve oficializar o objetivo da iteração e, com base nesse objetivo, a equipe deve selecionar os itens da Lista de Trabalho do Produto que serão desenvolvidos nela. Esses itens formam a Lista de Trabalho da Iteração, que nada mais é do que parte da Lista de Trabalho do Produto. Os itens dessa lista devem ser partilhados entre os membros da Equipe Scrum e, cada membro pode dividir seus itens em tarefas, de acordo com a suas necessidades. O esforço para o desenvolvimento de cada tarefa deve ser estimado. A Equipe

Scrum é responsável pela Lista de Trabalho da Iteração e pode inserir, remover e alterar itens dessa lista ao longo da iteração.

A Figura 4.11 mostra o diagrama de classes da disciplina Planejar Iteração com os estereótipos SPEM.

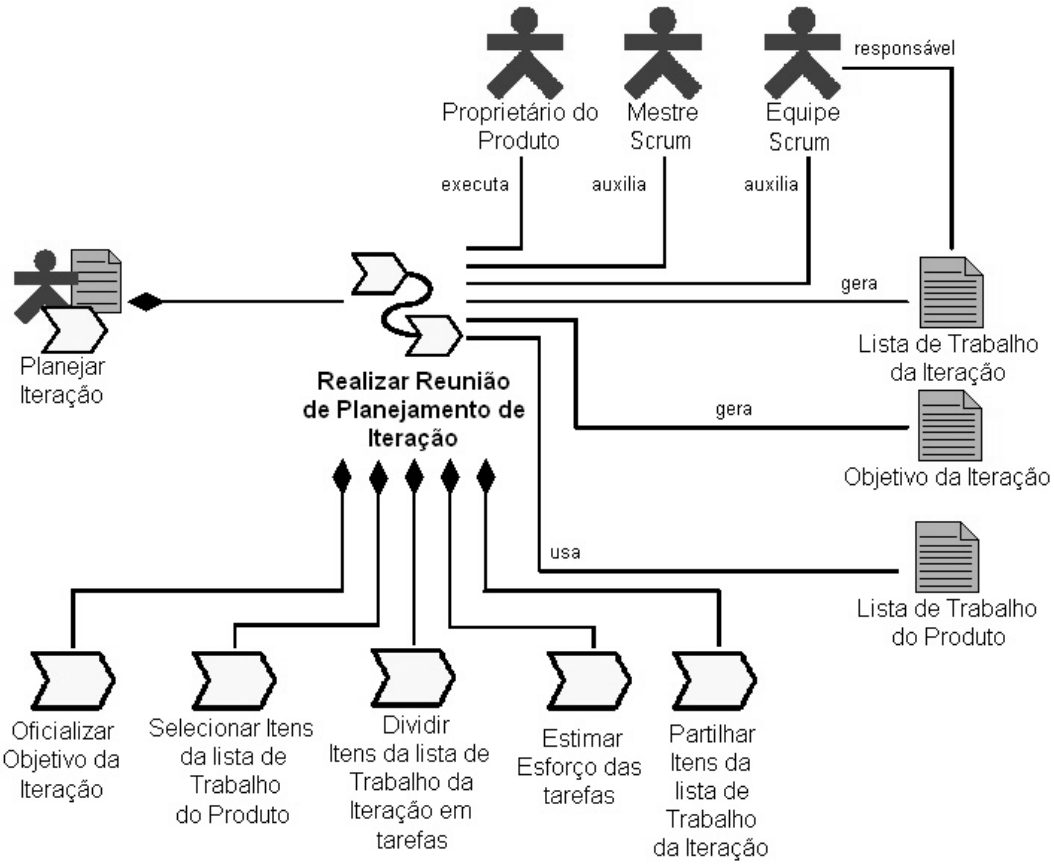


Figura 4.11 – Diagrama de classes da disciplina Planejar Iteração

Devido à característica de agrupar várias atividades, Realizar Reunião de Planejamento de Iteração foi modelada como um conjunto de trabalho (classe ConjuntoDeTrabalho) (Figura 4.4), que será dividido em tarefas quando da realização dessa reunião.

O conjunto de trabalho Realizar Reunião de Planejamento de Iteração está representado na Figura 4.11, bem como os papéis (Proprietário do Produto, Mestre Scrum e Equipe Scrum), as atividades (Oficializar Objetivo da Iteração, Selecionar Itens da lista de Trabalho do Produto, Partilhar Itens da lista de Trabalho da Iteração, Dividir Itens da lista de Trabalho da Iteração em tarefas, e Estimar Esforço das Tarefas) e os artefatos (Lista de Trabalho do Produto, Lista de Trabalho da Iteração e Objetivo da Iteração) a ele relacionados.

As atividades que estavam representadas como um conjunto de trabalho na perspectiva estática da disciplina, apresentada na Figura 4.11, devem ser executadas em uma ordem, para

possibilitar o planejamento da próxima iteração, como mostra o diagrama de atividades da disciplina Planejar Iteração com os estereótipos SPEM da Figura 4.12.

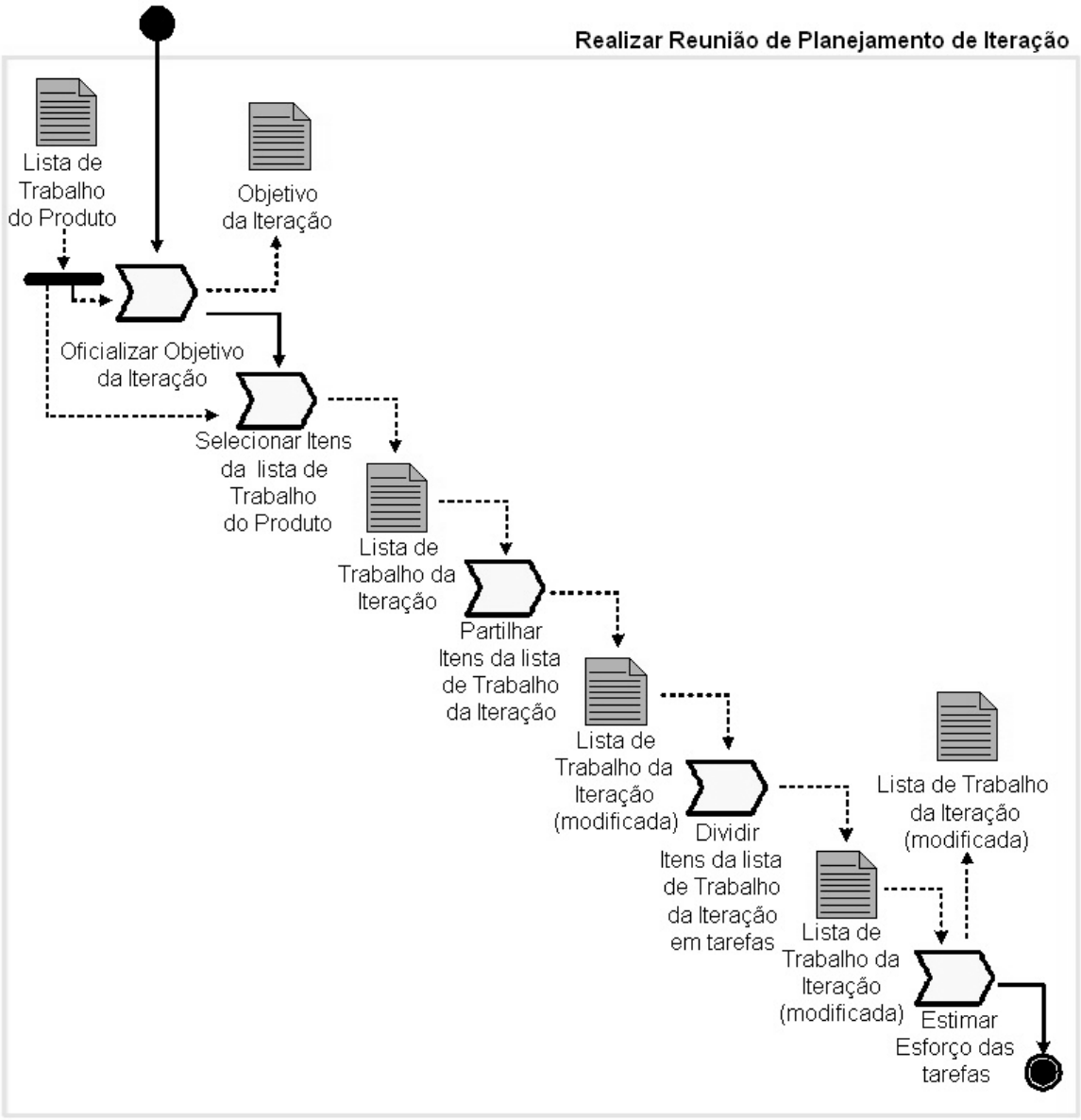


Figura 4.12 – Diagrama de atividades da disciplina Planejar Iteração

A Figura 4.12 também mostra que o artefato Lista de Trabalho da Iteração é gerado e constantemente modificado durante o planejamento da iteração. Após o planejamento, a iteração pode ser iniciada, como mostra a próxima subseção.

4.3.4. Disciplina Realizar Iteração

Na iteração, que deve durar trinta dias, ocorrem as atividades de desenvolvimento do produto, como análise, projeto, codificação e teste, realizadas pela Equipe Scrum, que deve criar

um incremento do produto até o final da iteração para apresentá-lo ao cliente e aos outros interessados na reunião de revisão de iteração.

Em paralelo com o desenvolvimento, ocorrem também as reuniões diárias, de aproximadamente quinze minutos, executadas pelo Mestre Scrum com a participação da Equipe Scrum. Nessa reunião de acompanhamento do estado do projeto, a Equipe Scrum e o Mestre Scrum devem discutir o que foi realizado desde a última reunião diária, o que será realizado até a próxima reunião e quais foram os problemas encontrados ao realizar o trabalho. O Mestre Scrum deve listar os impedimentos relatados pelos membros da Equipe Scrum e removê-los. Outros interessados no estado do projeto como o Proprietário do Produto, Cliente e Gerência podem participar da reunião diária, mas apenas como observadores, apenas o Mestre Scrum e a Equipe Scrum podem emitir opiniões durante essa reunião. Após a reunião diária, o Gráfico de Progresso, que mostra a quantidade de trabalho remanescente ao longo da iteração, deve ser atualizado pela Equipe Scrum para possibilitar o acompanhamento do progresso do projeto por todos os envolvidos.

Além do desenvolvimento e das reuniões diárias, durante a iteração, o Proprietário do Produto pode ficar em contato direto com o Cliente para levantamento de novos requisitos e esclarecimento de dúvidas que surgem da Equipe Scrum. Assim, a Lista de Trabalho do Produto pode ser alterada pelo Proprietário do Produto durante a iteração.

A Figura 4.13 mostra, por meio de um diagrama de classes e os estereótipos SPEM, a disciplina Realizar Iteração, que contém um conjunto de trabalho (Desenvolver), atividades (Fazer Análise, Fazer Projeto, Codificar, Testar, Realizar Reunião Diária, e Levantar Requisitos), papéis (Proprietário do Produto, Mestre Scrum, Equipe Scrum, Cliente e Gerência), artefatos (Lista de Trabalho da Iteração, Lista de Trabalho do Produto, Código, Gráfico de Progresso, Lista de Impedimentos, Incremento da Iteração Anterior, Novo Incremento e Lista de Melhorias – a ser discutida na próxima subseção).

Devido à sua característica de agrupar atividades de análise, projeto, codificação e teste, Desenvolver foi modelado como um conjunto de trabalho (classe ConjuntoDeTrabalho) (Figura 4.4). Porém, apesar de reconhecer que todas essas atividades podem ocorrer durante o desenvolvimento do software na iteração, o Scrum não determina quais devem ocorrer ou a ordem em que devem fazê-lo.

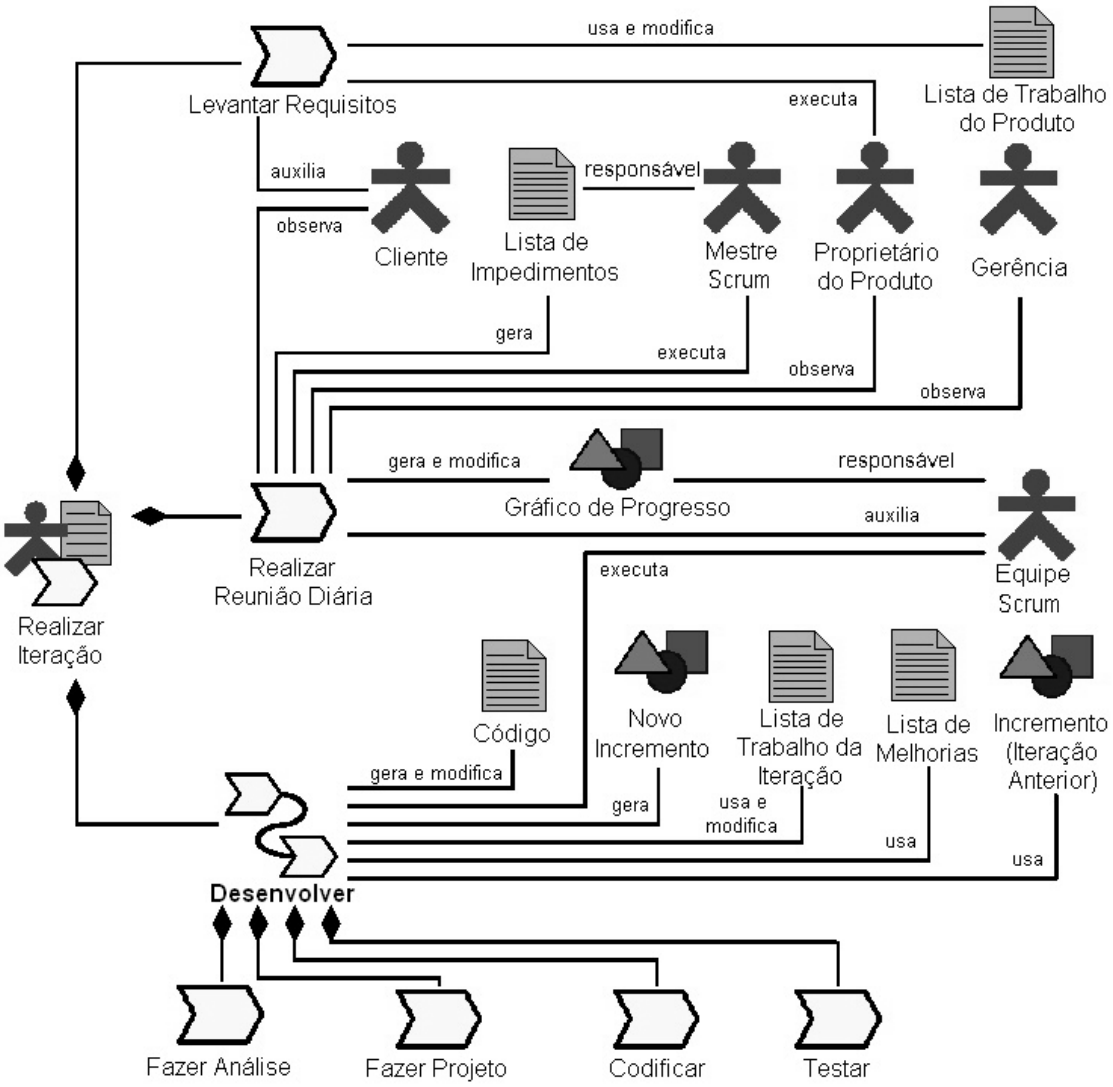


Figura 4.13 – Diagrama de classes da disciplina Realizar Iteração

As atividades de desenvolvimento que vão ocorrer em uma iteração e a sua respectiva ordem dependem da necessidade atual da Equipe Scrum. Por exemplo, ao longo de uma iteração a Equipe Scrum pode apenas Fazer o Projeto do software e Codificar na próxima iteração, ou pode Fazer Projeto, Codificar e Testar em uma mesma iteração, como mostra o diagrama de atividades da disciplina Realizar Iteração com os estereótipos SPEM da Figura 4.14.

A Figura 4.14 mostra que, em paralelo com as atividades do conjunto de trabalho Desenvolver, ocorrem as atividades Realizar Reunião Diária e Levantar Requisitos. Além disso, outra característica importante do Scrum, que está representada nessa figura, é o possível cancelamento de iteração, ou seja, a qualquer momento do desenvolvimento, a Equipe Scrum ou Proprietário do Produto podem cancelar uma iteração se acharem necessário.

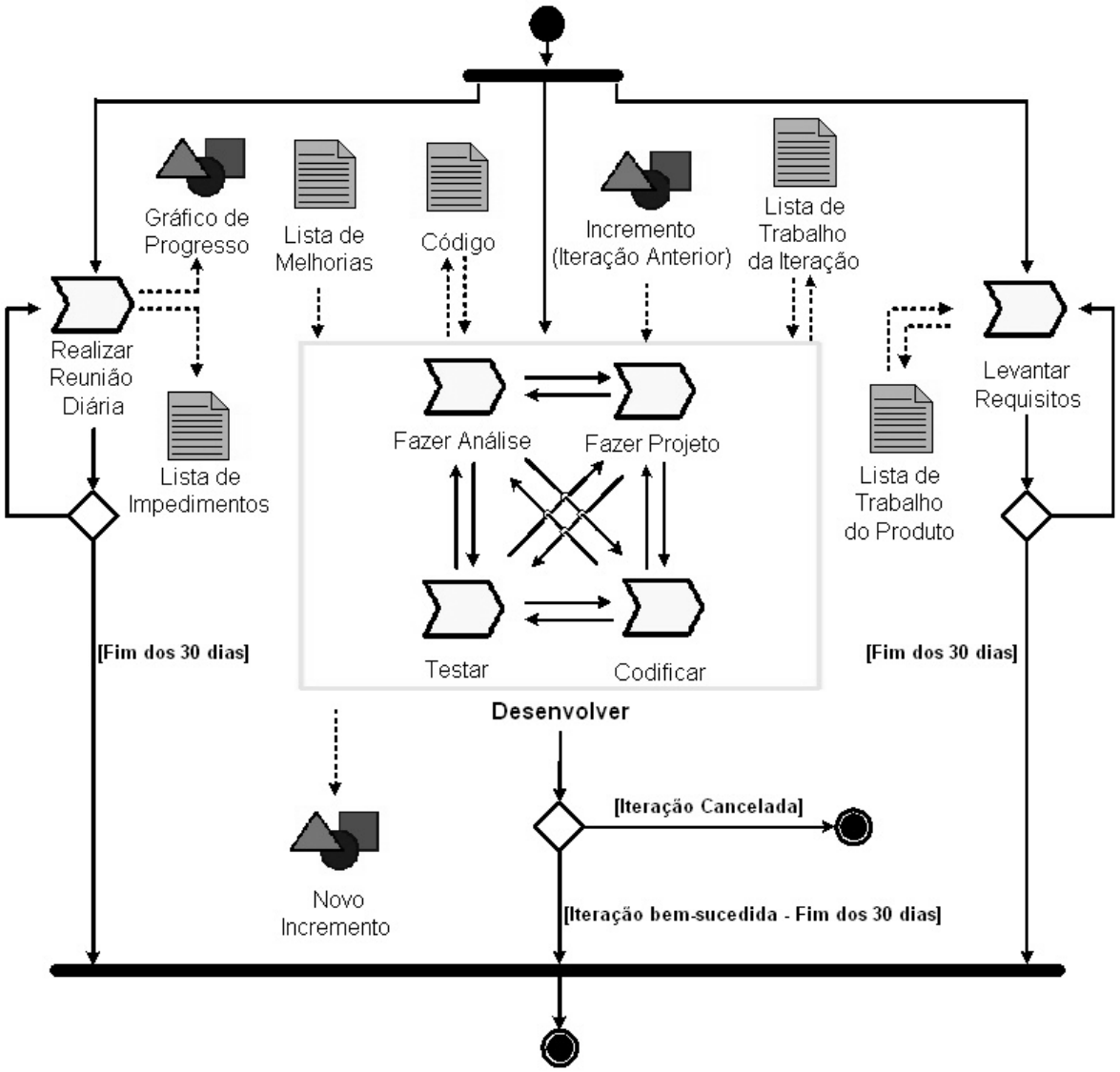


Figura 4.14 – Diagrama de atividades da disciplina Realizar Iteração

A Equipe Scrum só vai parar de Desenvolver ao final dos trinta dias da iteração ou se ela for cancelada. Ao término dos trinta dias, um incremento do produto deve ser gerado e apresentado ao cliente e a outros interessados, como é descrito na próxima subseção.

4.3.5. Disciplina Revisar Iteração

Nessa disciplina, devem ocorrer as reuniões de revisão e retrospectiva de iteração. Na reunião de revisão de iteração, executada pelo Mestre Scrum, com a participação da Equipe Scrum, os membros da Equipe Scrum devem apresentar ao cliente e a outros interessados, o incremento do produto desenvolvido na última iteração. Nessa reunião, diferentemente das reuniões diárias, todos envolvidos podem e são aconselhados a participar. Após apresentação, a

lista de Trabalho do Produto deve ser atualizada de acordo com o *feedback* fornecido pelos clientes e outros participantes.

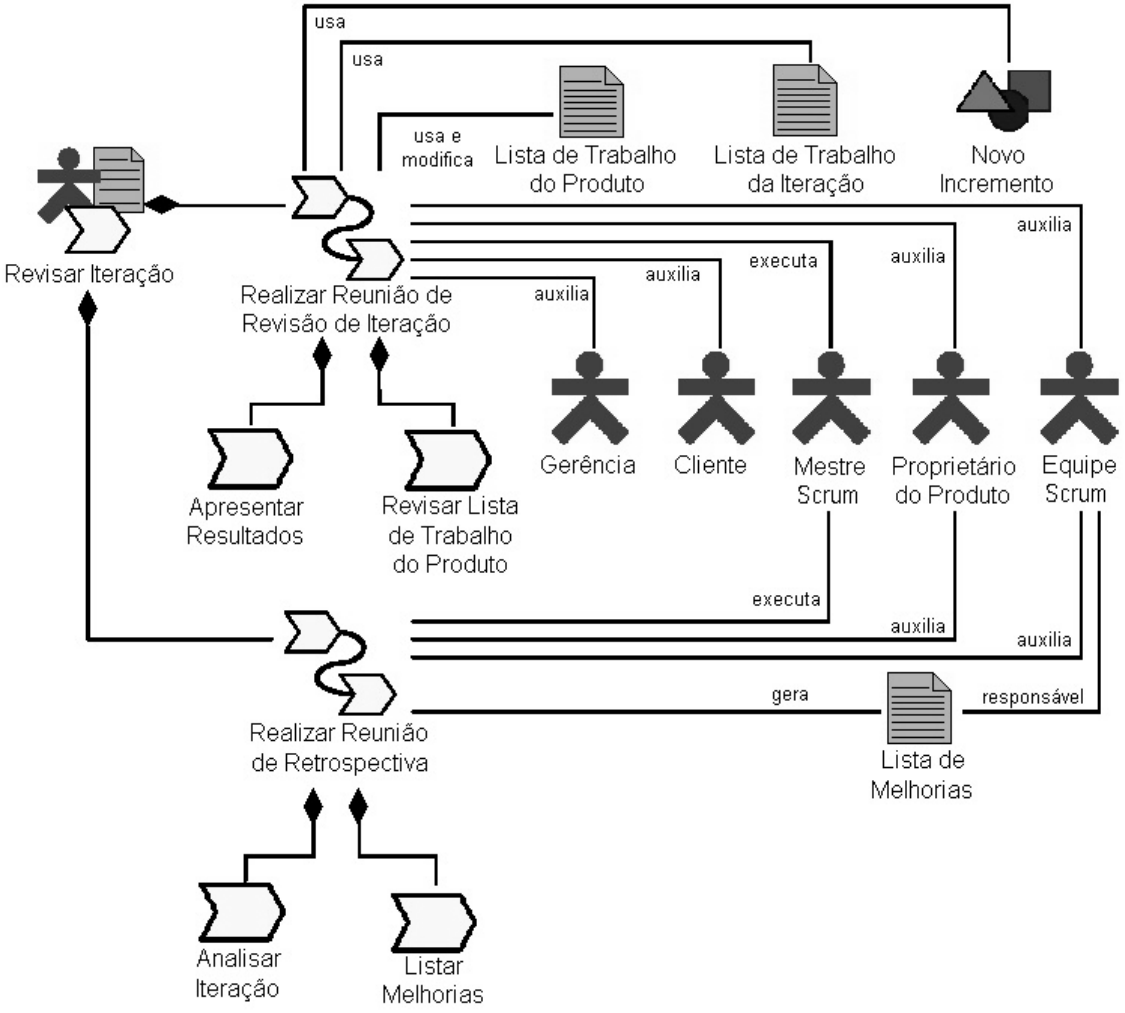


Figura 4.15 – Diagrama de classes da disciplina Revisar Iteração

Além da reunião de revisão, uma reunião de retrospectiva de iteração deve ser realizada, para discussão do que ocorreu de bom na última iteração e o que poderia ser melhorado para a próxima iteração. O resultado dessa reunião, executada pelo Mestre Scrum com a participação da Equipe Scrum, é uma lista de melhorias que devem ser realizadas na próxima iteração.

A Figura 4.15 descreve, por meio de um diagrama de classes com os estereótipos SPEM, a disciplina Revisar Iteração, com seus conjuntos de trabalho (Realizar Reunião de Revisão de Iteração e Realizar Reunião de Retrospectiva), atividades (Apresentar Resultados, Revisar Lista de Trabalho do Produto, Analisar Iteração, Listar Melhorias), papéis (Proprietário do Produto, Mestre Scrum, Equipe Scrum, Cliente e Gerência) e artefatos (Lista de Trabalho da Iteração, Novo Incremento, Lista de Trabalho do Produto e Lista de Melhorias).

Devido à característica de agrupar várias atividades, Realizar Reunião de Revisão de Iteração e Realizar Reunião de Retrospectiva foram modeladas como conjuntos de trabalho (classe ConjuntoDeTrabalho) (Figura 4.4).

A Figura 4.16 mostra o diagrama de atividades da disciplina Revisar Iteração com os estereótipos SPEM.

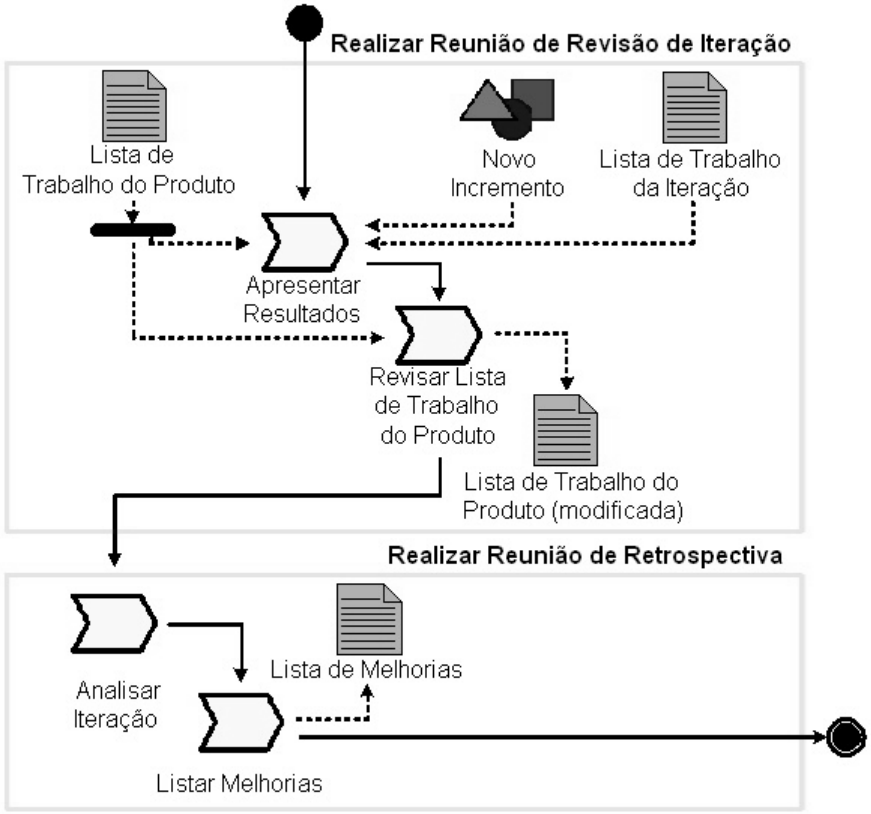


Figura 4.16 – Diagrama de atividades da disciplina Revisar Iteração

Cada projeto Scrum é formado por uma ou várias iterações. Ao final da última iteração, a versão do produto de software desenvolvida deve ser gerada, como mostra a próxima subseção.

4.3.6. Disciplina Finalizar Versão

Ao finalizar a versão, a Equipe Scrum deve realizar testes de sistema, criar a documentação que foi solicitada pelo cliente e colocar o software em produção.

A Figura 4.17 mostra, por meio de um diagrama de classes e dos estereótipos SPEM, a disciplina Finalizar Versão, com suas atividades (Fazer Teste de Sistema, Documentar e Colocar em Produção), papel (Equipe Scrum) e artefatos (Novo Incremento, Versão e Documentação).

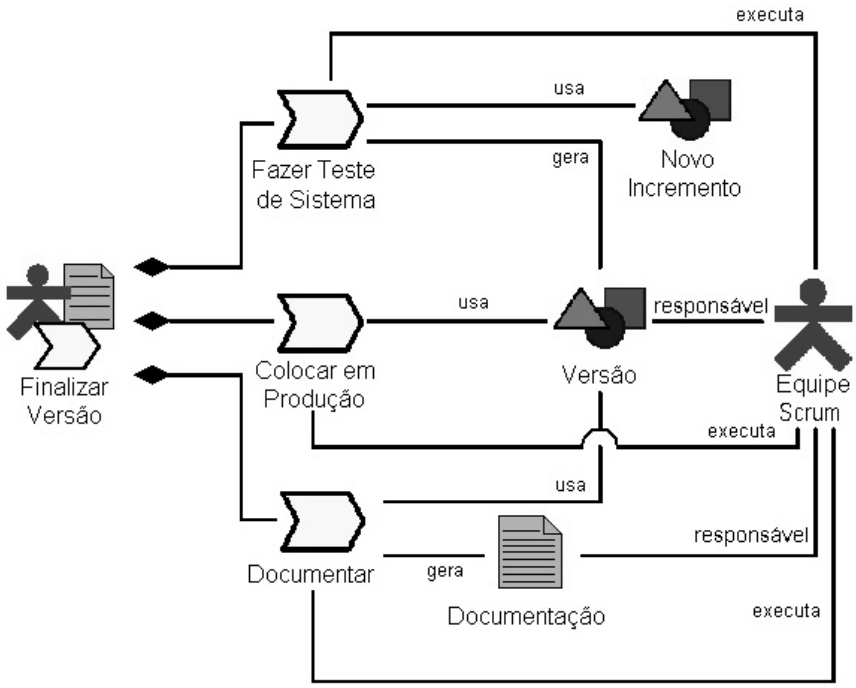


Figura 4.17 – Diagrama de classes da disciplina Finalizar Versão

Salienta-se que a atividade de documentação refere-se à documentação solicitada pelo cliente e não aos documentos de projeto utilizados durante o desenvolvimento.

A Figura 4.18 mostra o diagrama de atividades da disciplina Revisar Iteração com os estereótipos SPEM.

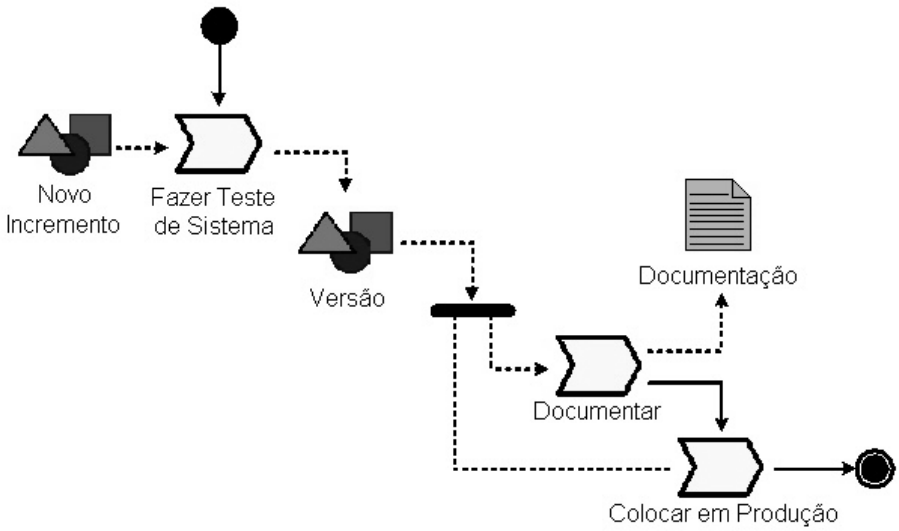


Figura 4.18 – Diagrama de atividades da disciplina Finalizar Versão

A próxima seção apresenta a classificação dos padrões, apresentados no Capítulo 2, em duas categorias, de acordo com a definição de padrão organizacional e de processo considerada neste trabalho.

4.4. Identificação da Categoria dos Padrões

Depois de modelado o processo Scrum, é necessário identificar a categoria de cada padrão que será integrado, isto é, verificar se ele é organizacional ou de processo, para possibilitar a sua representação por meio dos estereótipos SPEM.

Por não ter sido encontrada na literatura definição clara para padrões organizacionais e de processo que facilite a sua representação por meio de modelos, adotou-se neste trabalho as seguintes definições:

- **Padrões de Processo**: documentam soluções para problemas no processo de desenvolvimento de software, que prescrevem a criação de uma nova atividade, artefato ou papel no processo.
- **Padrões Organizacionais**: documentam diretrizes que devem ser respeitadas na execução de uma atividade, cumprimento de um papel ou criação de um artefato no processo de desenvolvimento de software.

Um padrão de processo sugere a criação de uma atividade, papel ou artefato no processo, enquanto um padrão organizacional mostra como realizar uma atividade ou papel no processo. Ressalta-se que existem padrões que se encaixam nessas duas categorias.

A Tabela 4.2 apresenta a categoria dos padrões que serão integrados ao Scrum, tendo na primeira coluna o nome do padrão, na segunda a categoria a qual o padrão foi associado, e na terceira a justificativa pela qual o padrão foi associado à determinada categoria.

Tabela 4.2 – Categoria dos padrões integrados ao Scrum

Padrão	Categoria	Justificativa
Self Selecting Team	Organizacional	Sugere que as pessoas devem selecionar a equipe na qual elas querem trabalhar.
Scenarios Define Problem	Processo	Propõe a criação de casos de uso para extração e esclarecimento dos requisitos.
Small Writing Team	Organizacional	Fornece informações sobre como criar casos de uso, sugerindo que eles devem ser criados por no máximo três pessoas.
Breadth Before Depth	Organizacional	Fornece informações sobre como desenvolver e refinar casos de uso, sugerindo que os desenvolvedores devem identificá-los antes de começar a refinar cada um deles.
Spiral Development	Organizacional	Fornece informações sobre como desenvolver e refinar casos de uso, sugerindo que seu refinamento seja realizado iterativamente.
Face To Face Before Working Remotely	Processo	Propõe a realização de uma reunião presencial no início de projetos geograficamente distribuídos, para que os envolvidos possam se conhecer.
Unity Of Purpose	Processo	Propõe a realização da atividade de compartilhamento da visão comum no início do projeto para mostrar aos envolvidos o propósito do projeto.

Standards Linking Locations	Processo	Propõe a realização da atividade de estabelecimento de normas e convenções em projetos geograficamente distribuídos, para possibilitar a comunicação entre as partes do software que está sendo construído.
Lock 'Em Up Together	Processo	Propõe a realização de uma reunião presencial no início do projeto para a criação da arquitetura do produto que será desenvolvido em grupo.
Organization Follows Location	Organizacional	Fornece informações para repartição do trabalho em projetos geograficamente distribuídos, sugerindo que as responsabilidades devem ser atribuídas de forma que as decisões possam ser tomadas localmente.
Code Ownership	Organizacional	Fornece informações para atribuição de tarefas no desenvolvimento do software, sugerindo que cada módulo do código deve ser atribuído a um desenvolvedor.
Generics And Specifics	Organizacional	Fornece informações para atribuição de tarefas no desenvolvimento do software, sugerindo que os especialistas e principiantes devem implementar, respectivamente, as partes genéricas e as partes específicas do software.
Surrogate Customers	Processo	Propõe a criação do papel de substituto do cliente no processo, para suprir a ausência do cliente quando o mesmo não está disponível.
Named Stable Bases	Organizacional e de Processo	Propõe a realização da atividade de integração do software no desenvolvimento, além de estabelecer a freqüência com que essa integração deve ocorrer.
Private Workspace	Organizacional	Fornece informações de como gerenciar as versões de produtos de trabalho.
Repository	Organizacional	Fornece informações de como prover as versões corretas dos produtos de trabalho para os desenvolvedores.
Smoke Test	Organizacional	Fornece informações sobre como verificar se a integração das partes do sistema não corrompeu o mesmo.
Application Design is Bounded By Test Design	Processo	Propõe a realização da atividade de criação de planos de teste baseados nos casos de uso.
Architect Controls Product	Processo	Propõe a criação do papel de arquiteto no processo para definir um estilo de arquitetura ao produto.
Architect Also Implements	Organizacional	Fornece informações sobre a execução do papel de arquiteto no processo, sugerindo que ele deve implementar o software junto com os desenvolvedores

Por meio dessa classificação, é possível representar os padrões organizacionais e de processo integrados ao Scrum utilizando o SPEM, pois a categoria do padrão identifica quais estereótipos devem ser utilizados para modelá-los. Assim, os padrões de processo serão modelados utilizando os estereótipos da Figura 4.19.

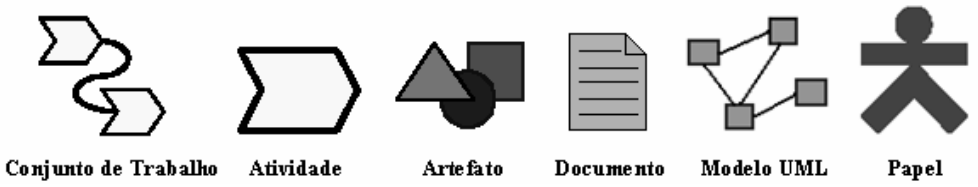


Figura 4.19 – Estereótipos utilizados para modelar padrões de processo

Para a modelagem de um padrão organizacional, é necessária a utilização de outro estereótipo, o Guia (*Guidance*), pois essa categoria de padrão fornece informações mais detalhadas sobre o elemento do processo ao qual está associado.

O SPEM permite a adição de novas categorias de guia no processo e, neste trabalho, um novo tipo de guia é definido para representar os padrões organizacionais na modelagem, o estereótipo Padrão Organizacional, cuja notação é mostrada a Figura 4.20.

Figura 4.20 – Notação do estereótipo Padrão Organizacional

Assim, devido às propriedades dos padrões organizacionais e de processo de propor a criação de novas atividades, papéis, artefatos e diretrizes para solucionar problemas de desenvolvimento de software, antes de integrar os padrões, é fundamental modelar o processo Scrum para possibilitar a integração correta das práticas propostas pelos padrões.

Além da classificação dos padrões em categorias, é necessário identificar a disciplina Scrum a qual o padrão a ser integrado se relaciona, como mostra a próxima seção.

4.5. Associação entre os Padrões e Disciplinas do Scrum

Depois de identificar a categoria do padrão, é necessário identificar a disciplina do método ágil Scrum a qual o padrão será associado. Disciplina é um pacote que divide as atividades do processo de acordo com um tema comum (OMG, 2005). Assim, de acordo com a definição de cada padrão e as disciplinas identificadas no Scrum, a Tabela 4.3 apresenta a disciplina Scrum a qual cada padrão foi associado. A primeira coluna apresenta o nome do padrão, a segunda mostra a disciplina a qual o padrão foi associado e a terceira apresenta a justificativa da associação entre eles.

Tabela 4.3 – Disciplina dos padrões selecionados pelas organizações

Padrão	Disciplina	Justificativa
Self Selecting Team	Definir Papéis	Padrão relacionado com a atividade Identificar Equipe Scrum (Figura 4.7).
Scenarios Define Problem	Obter Requisitos	Padrão relacionado com a atividade Definir Requisitos Conhecidos (Figura 4.9).
Small Writing Team	Obter Requisitos	Padrão relacionado com a atividade Definir Requisitos Conhecidos (Figura 4.9).
Breadth Before Depth	Obter Requisitos	Padrão relacionado com a atividade Definir Requisitos Conhecidos (Figura 4.9).
Spiral Development	Obter Requisitos	Padrão relacionado com a atividade Definir Requisitos Conhecidos (Figura 4.9).
Private Workspace	Obter Requisitos	Padrão relacionado com a atividade Validar Ferramentas e Infra-estrutura (Figura 4.9).
Repository	Obter Requisitos	Padrão relacionado com a atividade Validar Ferramentas e Infra-estrutura (Figura 4.9).
Face To Face Before Working Remotely	Planejar Iteração	Padrão relacionado com o conjunto de trabalho Realizar Reunião de Planejamento de Iteração (Figura 4.11).
Unity Of Purpose	Planejar Iteração	Padrão relacionado com o conjunto de trabalho Realizar Reunião de Planejamento de Iteração (Figura 4.11).
Standards Linking Locations	Planejar Iteração	Padrão relacionado com o conjunto de trabalho Realizar Reunião de Planejamento de Iteração (Figura 4.11).
Lock 'Em Up Together	Planejar Iteração	Padrão relacionado com o conjunto de trabalho Realizar Reunião de Planejamento de Iteração (Figura 4.11).
Architect Controls Product	Planejar Iteração	Padrão relacionado com o conjunto de trabalho Realizar Reunião de Planejamento de Iteração (Figura 4.11).
Code Ownership	Planejar Iteração	Padrão relacionado com a atividade Partilhar Itens da lista de Trabalho da Iteração (Figura 4.11).
Generics And Specifics	Planejar Iteração	Padrão relacionado com a atividade Partilhar Itens da lista de Trabalho da Iteração (Figura 4.11).
Organization Follows Location	Planejar Iteração	Padrão relacionado com a atividade Partilhar Itens da lista de Trabalho da Iteração (Figura 4.11).
Named Stable Bases	Realizar Iteração	Padrão relacionado com o conjunto de trabalho Desenvolver (Figura 4.13).
Smoke Test	Realizar Iteração	Padrão relacionado com o conjunto de trabalho Desenvolver (Figura 4.13).
Application Design is Bounded By Test Design	Realizar Iteração	Padrão relacionado com a atividade Testar (Figura 4.13).
Architect Also Implements	Realizar Iteração	Padrão relacionado com o conjunto de trabalho Desenvolver (Figura 4.13).
Surrogate Customers	Realizar Iteração	Padrão relacionado com a atividade Realizar Reunião Diária (Figura 4.13).

A Tabela 4.3 mostra que esses padrões fornecem soluções para as disciplinas Definir Papéis, Obter Requisitos, Planejar Iteração e Realizar Iteração e, assim, essas são estendidas pela integração das práticas propostas por esses padrões organizacionais e de processo. As disciplinas Revisar Iteração e Finalizar Versão não tiveram necessidade da integração de padrões, porém os artefatos gerados pela integração dos padrões em outras disciplinas são utilizados na disciplina Finalizar Versão para a construção de uma base de conhecimento.

A próxima seção apresenta o Scrum estendido, com as disciplinas modificadas pela integração dos padrões.

4.6. Integração dos Padrões Organizacionais e de Processo para Extensão do Scrum.

Esta seção apresenta uma proposta de extensão do Scrum por meio da integração de padrões organizacionais e de processo, utilizando as etapas descritas nas seções anteriores.

Métodos ágeis como Scrum e XP vêm sendo refinados ao longo dos anos. Em sua última publicação, Beck e Andres (2004) apresentam algumas alterações para o XP. Além de acrescentar o novo valor respeito (*respect*), algumas práticas foram renomeadas e outras foram incorporadas. Outra mudança é que as práticas estão separadas em práticas primárias (*Primary Practices*) e resultantes (*Corollary Practices*).

Da mesma forma, o Scrum vem sendo aprimorado desde a sua criação (Schwaber, 1995; Schwaber e Beedle, 2002; Schwaber, 2004). Uma alteração importante do Scrum, apresentada por Schwaber (2004), foi a introdução de uma nova prática, as reuniões de retrospectiva de iteração (*Sprint Retrospective Meeting*), que devem ser realizadas após as reuniões de revisão de iteração (*Sprint Review Meeting*).

Apesar de estar sendo refinado, o Scrum possui pontos fracos e não trata de algumas questões técnicas e organizacionais de desenvolvimento de software, observadas neste trabalho:

- **Técnicas para extração dos requisitos**: na etapa de levantamento de requisitos o Scrum propõe a criação de uma lista de Trabalho do Produto, porém não fornece técnicas para extração dos requisitos junto ao cliente. Isso dificulta o entendimento e validação dos requisitos, principalmente em equipes novas, que estão começando a trabalhar com esse método.
- **Criação da arquitetura do produto**: Não está claro no Scrum quando a arquitetura inicial deve ser criada e, com isso, não é possível definir se essa atividade deve ocorrer antes, durante ou depois da Reunião de Planejamento de Iteração (*Sprint Planning Meeting*), na primeira iteração.
- **Divisão do trabalho**: A atividade Partilhar Itens da lista de Trabalho da Iteração (Figura 4.11), que compõe o conjunto de trabalho Realizar Reunião de Planejamento de Iteração, não trata questões como a experiência dos membros da Equipe Scrum e sua localização.
- **Integração e Teste**: Abrahamsson et al. (2002) destacam que no Scrum não é definida a freqüência de integração do software e os testes devem ser realizados.

- **Distribuição geográfica**: Um problema particular das organizações que participaram da experiência, apresentada no Capítulo 3, também não abordado pelo Scrum, é a distribuição geográfica dos membros das organizações, que dificultou a organização, a divisão de trabalho, a comunicação e a colaboração. Projetos distribuídos estão se tornando comuns na indústria de software (Stotts et al., 2003; Baheti et al., 2002a; Baheti et al., 2002b; Kircher et al., 2001; Schümmer e Schümmer, 2001).

Essas questões motivaram esta pesquisa para estender o Scrum, utilizando padrões organizacionais e de processo. Assim, nas próximas subseções, é apresentada uma proposta para melhoria do Scrum. De acordo com essas questões, os padrões organizacionais e de processo apresentados na Tabela 4.2 e 4.3 foram integrados ao Scrum.

As disciplinas estendidas pelos padrões são apresentadas nas próximas subseções, isto é, todas as disciplinas mostradas na Figura 4.6, com exceção da Revisar Iteração, que não sofreu nenhuma modificação em sua modelagem inicial.

Os padrões são modelados de acordo com as suas categorias, apresentadas na Tabela 4.2, utilizando os estereótipos apresentados na Figura 4.19 para os padrões de processo e os apresentados na Figura 4.20 para os padrões organizacionais. Como todos os padrões organizacionais são modelados apenas com o estereótipo "Padrão Organizacional", criado neste trabalho, eles continuam com o seu nome original em inglês na modelagem, diferentemente dos padrões de processo, que são modelados utilizando vários estereótipos e, assim, seus nomes originais não são mantidos na modelagem.

As modificações do Scrum podem ser visualizadas nos retângulos cinza com chanfro externo. Esses retângulos, não existentes no SPEM, são utilizados aqui para facilitar a identificação das modificações realizadas para a criação da extensão do Scrum. Os demais estereótipos existentes nos modelos são referentes às disciplinas do Scrum modeladas anteriormente.

O texto das próximas subseções refere-se exclusivamente à integração dos padrões ao Scrum. O detalhamento dos padrões integrados está no Capítulo 2.

4.6.1. Disciplina Definir Papéis (Estendida)

Para auxiliar a identificação da Equipe Scrum que irá participar do projeto, o padrão *Self Selecting Team* foi integrado ao Scrum. A Tabela 4.4 apresenta o nome do padrão integrado na primeira coluna, a descrição da sua prática integrada ao Scrum na segunda coluna e os estereótipos SPEM utilizados para modelar esse padrão, com seus respectivos nomes e referência para a modelagem na terceira coluna.

Tabela 4.4 – Padrões integrados a disciplina Definir Papéis

Padrão	Prática no Scrum	Estereótipos
Self Selecting Team	Durante a seleção da Equipe Scrum a Gerência e o Mestre Scrum devem levar em consideração os interesses de cada indivíduo para Identificar a Equipe Scrum.	Padrão Organizacional (*Self Selecting Team*) (Figura 4.21 – retângulo A)

A Figura 4.21 mostra o diagrama de classes da disciplina Definir Papéis (estendida) modelado com estereótipos SPEM.

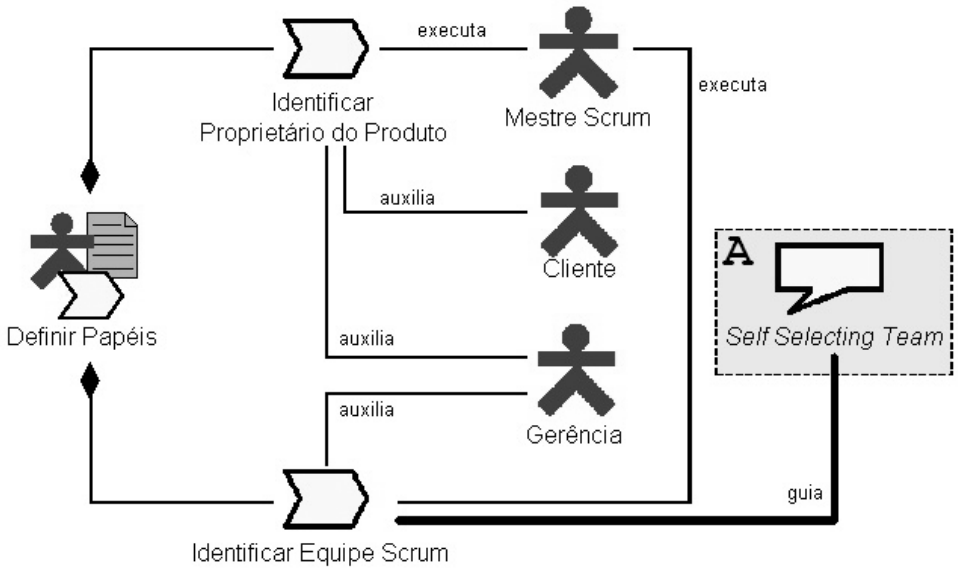


Figura 4.21 – Diagrama de Classes disciplina Definir Papéis (estendida)

Os padrões organizacionais são descritos apenas na modelagem estática (diagrama de classes) da disciplina, pois a modelagem dinâmica (diagrama de atividades) descreve somente a ordem em que as atividades devem ocorrer e os artefatos gerados e consumidos em cada atividade. Assim, a modelagem dinâmica da disciplina Definir Papéis (estendida) é a mesma do Scrum original, apresentada no diagrama de atividades da Figura 4.8.

Nas próximas disciplinas, todos os padrões integrados são descritos em tabelas que contém os mesmo tipos de informações apresentados na Tabela 4.4.

4.6.2. Disciplina Obter Requisitos (Estendida)

Para auxiliar a extração e entendimento de requisitos, os seguintes padrões foram integrados ao Scrum: *Scenarios Define Problem*, *Small Writing Team*, *Breadth Before Depth* e *Spiral Development*. Além desses, os padrões *Private Workspace* e *Repository* foram integrados

para auxiliar no estabelecimento de uma infra-estrutura adequada para a Equipe Scrum, como mostra a Tabela 4.5.

Tabela 4.5 – Padrões integrados a disciplina Obter Requisitos

Padrão	Prática no Scrum	Estereótipos
Scenarios Define Problem	Durante a exposição dos requisitos do produto por parte do cliente, na atividade Definir Requisitos Conhecidos, casos de uso devem ser criados para facilitar a extração e entendimento dos requisitos.	Modelo UML (Casos de Uso) (Figura 4.22 – retângulo B)
Small Writing Team	Na atividade Definir Requisitos Conhecidos, apenas três membros da Equipe Scrum devem definir todos os casos de uso do sistema. Isso facilita a Equipe Scrum chegar a um consenso de forma mais rápida.	Padrão Organizacional (*Small Writing Team*) (Figura 4.22 – retângulo B)
Breadth Before Depth	Na atividade Definir Requisitos Conhecidos, os casos de uso devem ser escritos de forma abrangente, assim eles vão fornecer uma visão geral do escopo do software antes do mesmo ser desenvolvido.	Padrão Organizacional (*Breadth Before Depth*) (Figura 4.22 – retângulo B)
Spiral Development	Na atividade Definir Requisitos Conhecidos, os casos de uso devem ser detalhados iterativamente para que a sua precisão aumente progressivamente ao longo das iterações. Desenvolver casos de uso de uma única vez pode dificultar a identificação e adição de novas informações.	Padrão Organizacional (*Spiral Development*) (Figura 4.22 – retângulo B)
Private Workspace	Na atividade Validar Ferramentas e Infra-estrutura, o Proprietário do Produto deve verificar se cada membro da Equipe Scrum possui uma área de trabalho privada para trabalhar.	Padrão Organizacional (*Private Workspace*) (Figura 4.22 – retângulo C)
Repository	Na atividade Validar Ferramentas e Infra-estrutura, o Proprietário do Produto também deve verificar se a Equipe Scrum têm um repositório único para obter e atualizar as versões corretas dos artefatos compartilhados.	Padrão Organizacional (*Repository*) (Figura 4.22 – retângulo C)

A utilização de casos de uso, além de facilitar o entendimento dos requisitos, pode facilitar a divisão do trabalho entre os vários membros da Equipe Scrum, já que, por meio deles, todos os cenários do sistema podem ser capturados e distribuídos entre os membros da Equipe Scrum. Cada caso de uso representa um item da lista de Trabalho do Produto.

A Figura 4.22 mostra o diagrama de classes da disciplina Obter Requisitos (estendida) modelado com estereótipos SPEM.

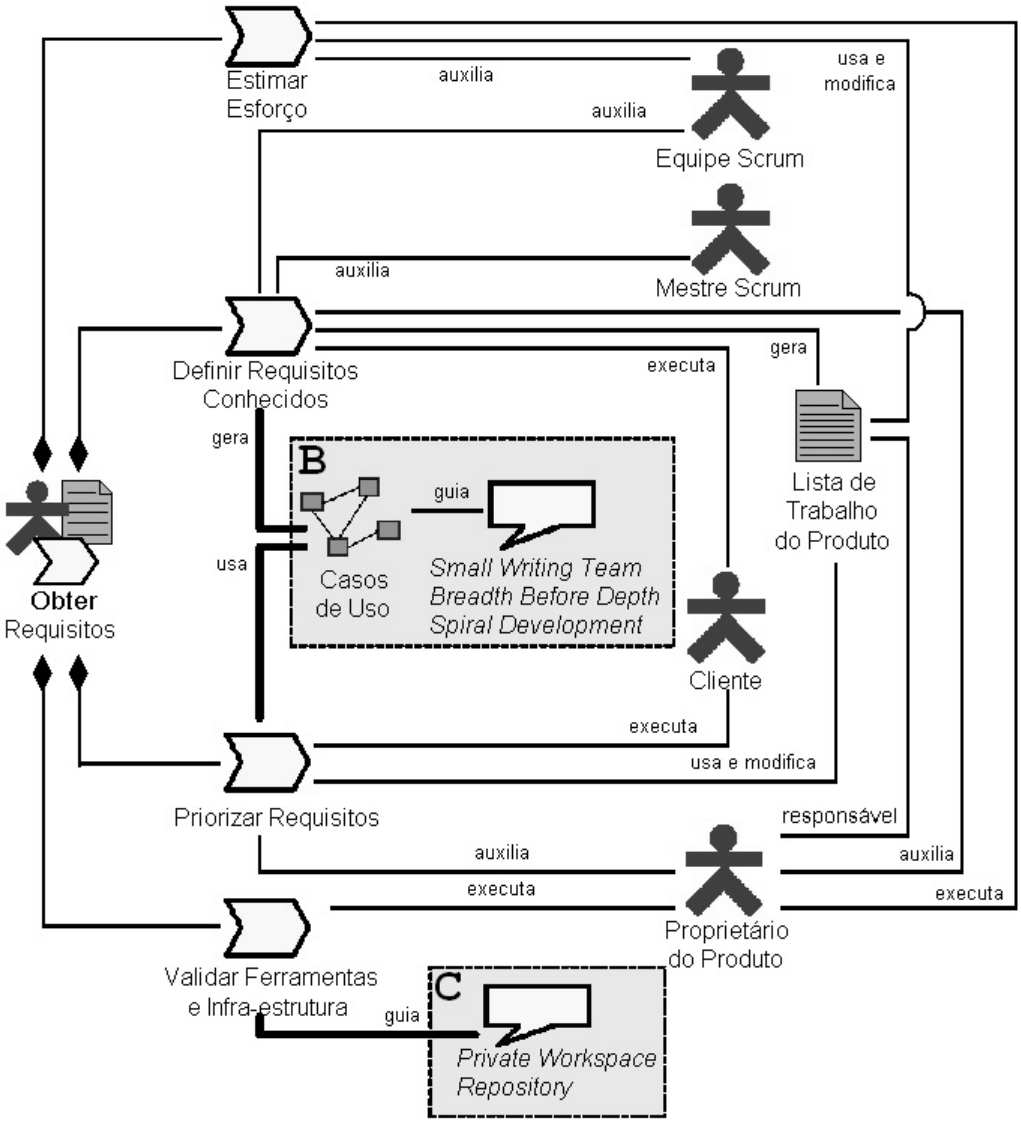


Figura 4.22 – Diagrama de classes da disciplina Obter Requisitos (estendida)

Para desenvolver o produto, com base nos casos de uso, os membros da Equipe Scrum compartilham produtos de trabalho comuns, como a Lista de Trabalho da Iteração e Código, que são constantemente modificados. Desta forma, o controle de versões de software é essencial.

Existem ferramentas como o CVS (CVS, 2006) que facilitam a aplicação dos padrões, *Private Workspace* e *Repository*, porém foge do escopo deste trabalho a discussão dessas ferramentas. Berczuk e Appleton (2002) afirmam que, apesar de ser evidente que o controle de versões possibilita a comunicação entre os membros da equipe, por meio dos produtos de trabalho, muitas organizações não controlam as versões dos seus produtos de trabalho, mesmo tendo uma ferramenta disponível.

A Figura 4.23 mostra o diagrama de atividades da disciplina Obter Requisitos (estendida) modelado com estereótipos SPEM.

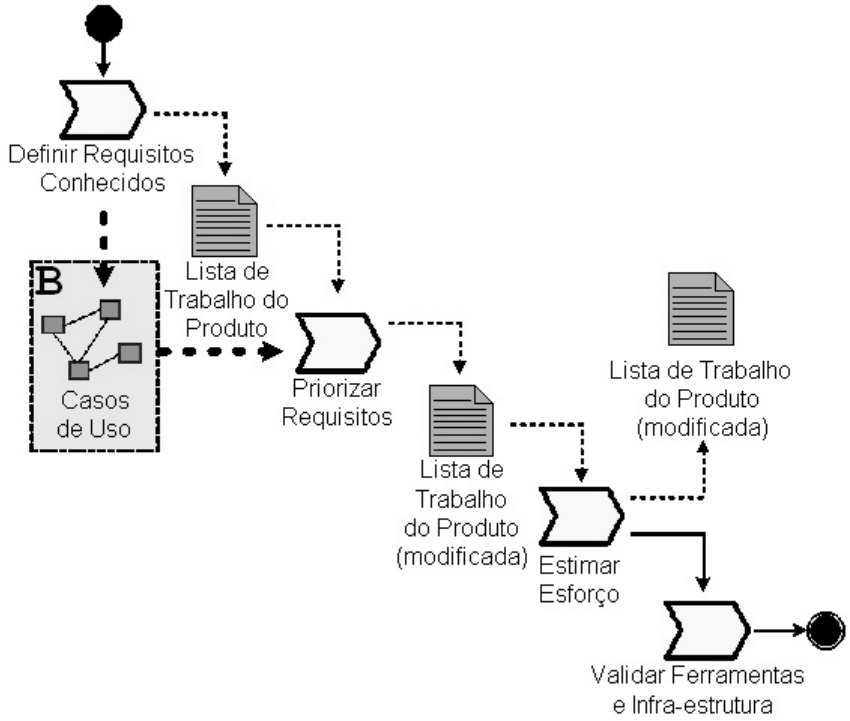


Figura 4.23 – Diagrama de atividades da disciplina Obter Requisitos (estendida)

A única diferença do diagrama de atividades da disciplina Obter Requisitos (estendida) para a original é a inserção dos casos de uso (retângulo B), pois o diagrama de atividades, utilizado para modelar a perspectiva estática das disciplinas, descreve apenas a ordem em que as atividades devem ocorrer e os artefatos gerados e consumidos nessas atividades e, por isso, as práticas propostas pelos padrões organizacionais não estão modeladas no diagrama de atividades.

A próxima subseção apresenta as modificações realizadas na disciplina de planejamento de requisitos.

4.6.3. Disciplina Planejar Iteração (Estendida)

Para auxiliar a criação da arquitetura inicial e o compartilhamento da visão comum do projeto, os seguintes padrões foram integrados ao Scrum: *Face To Face Before Working Remotely*, *Unity Of Purpose*, *Standards Linking Locations, Lock 'Em Up Together e Architect Controls Product.* Além desses, os padrões *Code Ownership, Generics And Specifics e Organization Follows Location* também foram integrados para auxiliar na atividade Partilhar Itens da lista de Trabalho da Iteração, como mostra a Tabela 4.6.

Tabela 4.6 – Padrões integrados a disciplina Planeja Iteração

Padrão	Prática no Scrum	Estereótipos
Face To Face Before Working Remotely	Antes da reunião de planejamento da primeira iteração, uma reunião técnica deve ser realizada, com a presença de todos envolvidos, para que questões como visão do projeto, convenções, normas e arquitetura, possam ser discutidas. Essa reunião, executada pelo Proprietário do Produto, com a participação do Mestre Scrum e Equipe Scrum, é dividida em três etapas: Compartilhar Visão Comum (*Unity Of Purpose*), Estabelecer Normas e Convenções (*Standards Linking Locations*) e Elaborar Arquitetura Inicial (*Lock 'Em Up Together*).	Conjunto de Trabalho (Realizar Reunião Técnica) (Figura 4.24 – retângulo D)
Unity Of Purpose	Etapa da Reunião Técnica na qual o Proprietário do Produto deve compartilhar a finalidade do produto e a visão comum do projeto com todos os membros da Equipe Scrum. A falta de visão clara sobre o sistema pode gerar indecisões, opiniões contrárias entre os envolvidos no projeto e até paralisá-lo (Bramble et al., 2002). Essa etapa permite que os membros da equipe se conheçam, o que pode facilitar a colaboração. Os casos de uso criados para identificar as necessidades dos clientes, que nesse momento ainda não estão detalhados, podem e devem ser utilizados pelo Proprietário do Produto para estabelecer a visão comum e objetivo do projeto.	Atividade (Compartilhar Visão Comum) Artefato (Casos de Uso) (Figura 4.24 – retângulo D)
Standards Linking Locations	Etapa da Reunião Técnica na qual a Equipe Scrum deve estabelecer as convenções ou normas que vão integrar as partes do software que está sendo desenvolvido. A seleção de normas e convenções é uma atividade que deve ser realizada em projetos distribuídos, pois isso pode facilitar a integração e diminuir a necessidade de comunicação. Contudo, a utilização de normas e convenções em equipes cujos membros trabalham no mesmo local pode ser prejudicial, já que esse tipo de equipe tem contexto e comunicação melhores, e as normas podem dificultar a comunicação e entendimento (Coplien e Harrison, 2004).	Atividade (Estabelecer Normas e Convenções) (Figura 4.24 – retângulo D)
Lock 'Em Up Together	Etapa da Reunião Técnica na qual a Equipe Scrum deve elaborar uma arquitetura inicial do produto. Pessoas diferentes têm idéias e experiências diferentes e, assim, todos os membros da equipe devem ser colocados juntos em uma sala para elaborar a arquitetura do produto, que não precisa ser detalhada, mas sim suficientemente clara para que o trabalho possa ser iniciado. 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.	Atividade Elaborar Arquitetura Inicial Artefato (Arquitetura) (Figura 4.24 – retângulo D)

Architect Controls Product	A atividade Elaborar Arquitetura Inicial, deve ser guiada por um membro experiente da Equipe Scrum, que deve desempenhar o papel de arquiteto, para facilitar o consenso entre os membros da equipe.	Papel (Arquiteto) (Figura 4.24 – retângulo D)
Organization Follows Location	Na atividade Partilhar Itens da lista de Trabalho da Iteração, o trabalho deve ser dividido de acordo com a localização, para que as pessoas que trabalhem com assuntos relacionados estejam no mesmo local. A divisão dos itens da Lista de Trabalho do Produto em casos de uso facilita essa atividade, pois os casos de uso relacionados podem ser atribuídos aos membros da equipe Scrum que estão no mesmo local. Essa prática pode ser adotada para projetos em que os envolvidos estão em diferentes andares de um edifício, em diferentes edifícios, ou até mesmo em diferentes localizações.	Padrão Organizacional (*Organization Follows Location*) (Figura 4.24 – retângulo E)
Code Ownership	Na atividade Partilhar Itens da lista de Trabalho da Iteração, cada localização deve ser proprietária de seu código, para que diminua a necessidade de maior comunicação com outras localizações. Porém, internamente, a propriedade do código deve ser coletiva, pois isso contribui para a agilidade no processo. Os casos de uso identificados na obtenção dos requisitos podem ser divididos entre as localizações. Cada localização fica responsável por um ou mais casos de uso e, exceto em circunstâncias extremas, o código referente a uma localização deve ser implementado apenas por membros a ela pertencentes. Salienta-se que, se todos os membros da Equipe Scrum estiverem no mesmo local, a propriedade do código deve ser coletiva, ou seja, a propriedade sobre o código deve ser aplicado a localizações e não a indivíduos, para que a agilidade não seja afetada.	Padrão Organizacional (*Code Ownership*) (Figura 4.24 – retângulo E)
Generics And Specifics	Na atividade Partilhar Itens da lista de Trabalho da Iteração, as partes genéricas do sistema devem ser atribuídas aos membros mais experientes da Equipe Scrum, enquanto as partes específicas devem ficar com os membros menos experientes. Isso facilita a divisão de trabalho entre os membros da Equipe Scrum. Os casos de uso de regra de negócio podem ser atribuídos aos membros mais experientes, enquanto que os casos de uso mais simples, como por exemplo, o de consulta e de cadastro de dados, podem ser atribuídos a membros menos experientes.	Padrão Organizacional (*Generics And Specifics*) (Figura 4.24 – retângulo E)

A Figura 4.24 mostra o diagrama de classes da disciplina Planejar Iteração (estendida) modelado com estereótipos SPEM. Como apresentado na Tabela 4.6, a Reunião Técnica é composta de três atividades Compartilhar Visão Comum, Estabelecer Normas e Convenções e

Elaborar Arquitetura Inicial. Os artefatos Caso de Uso e Arquitetura são consumidos e gerados durantes as atividades da Reunião Técnica e um membro da Equipe Scrum herda as responsabilidades de arquiteto durante essa reunião.

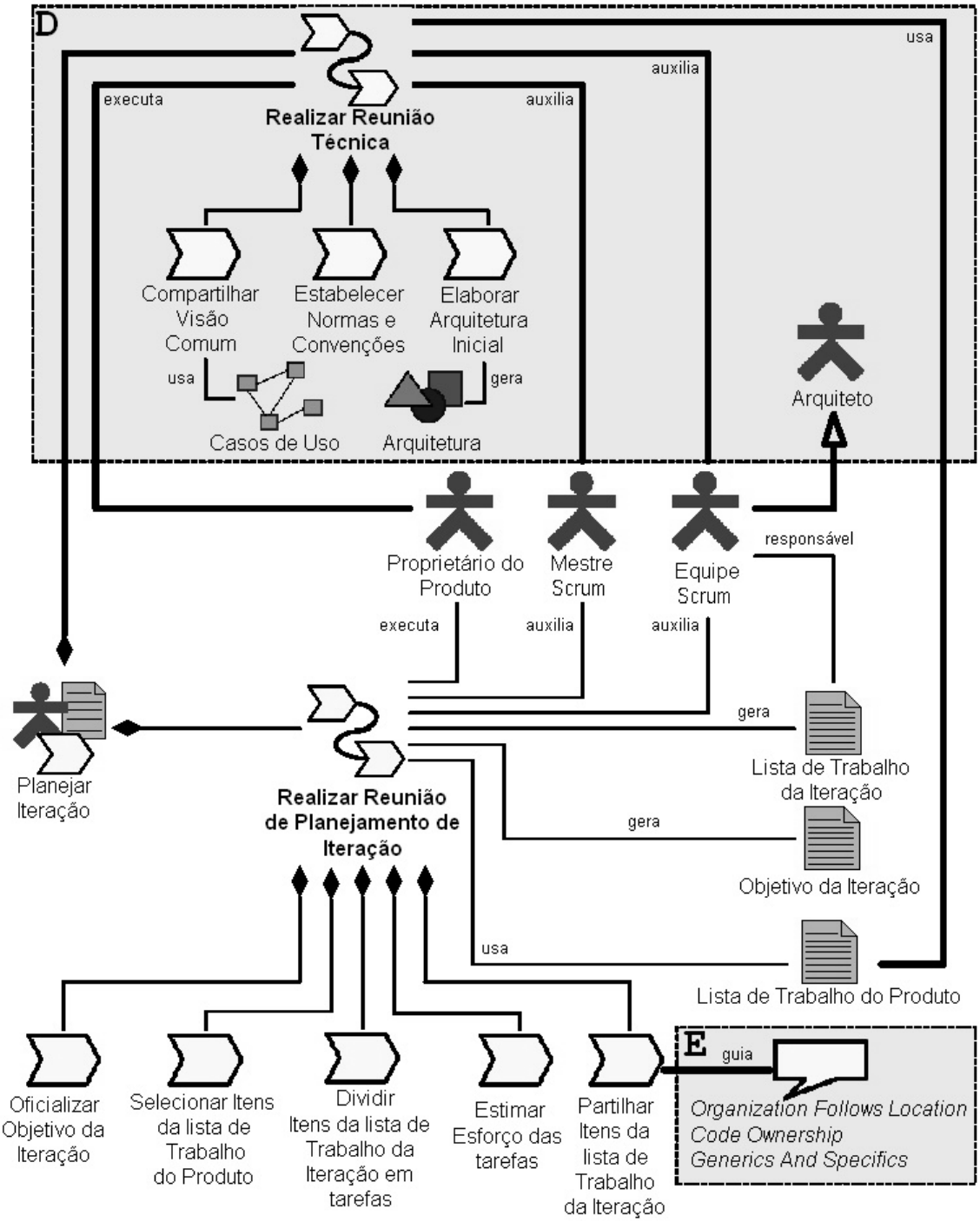


Figura 4.24 – Diagrama de classes da disciplina Planejar Iteração (estendida)

O conjunto de trabalho Realizar Reunião Técnica deve ser executado apenas antes da primeira iteração, pois a arquitetura vai evoluir naturalmente ao longo das iterações. Além disso, a atividade Estabelecer Normas e Convenções deve ser realizada apenas em projetos

distribuídos. O diagrama de atividades possibilita a representação de condições, assim, a Figura 4.25 apresenta o diagrama de atividades da disciplina Planejar Iteração (estendida) modelado com estereótipos SPEM, com as condições ([Primeira Iteração] e [Projeto Distribuído]).

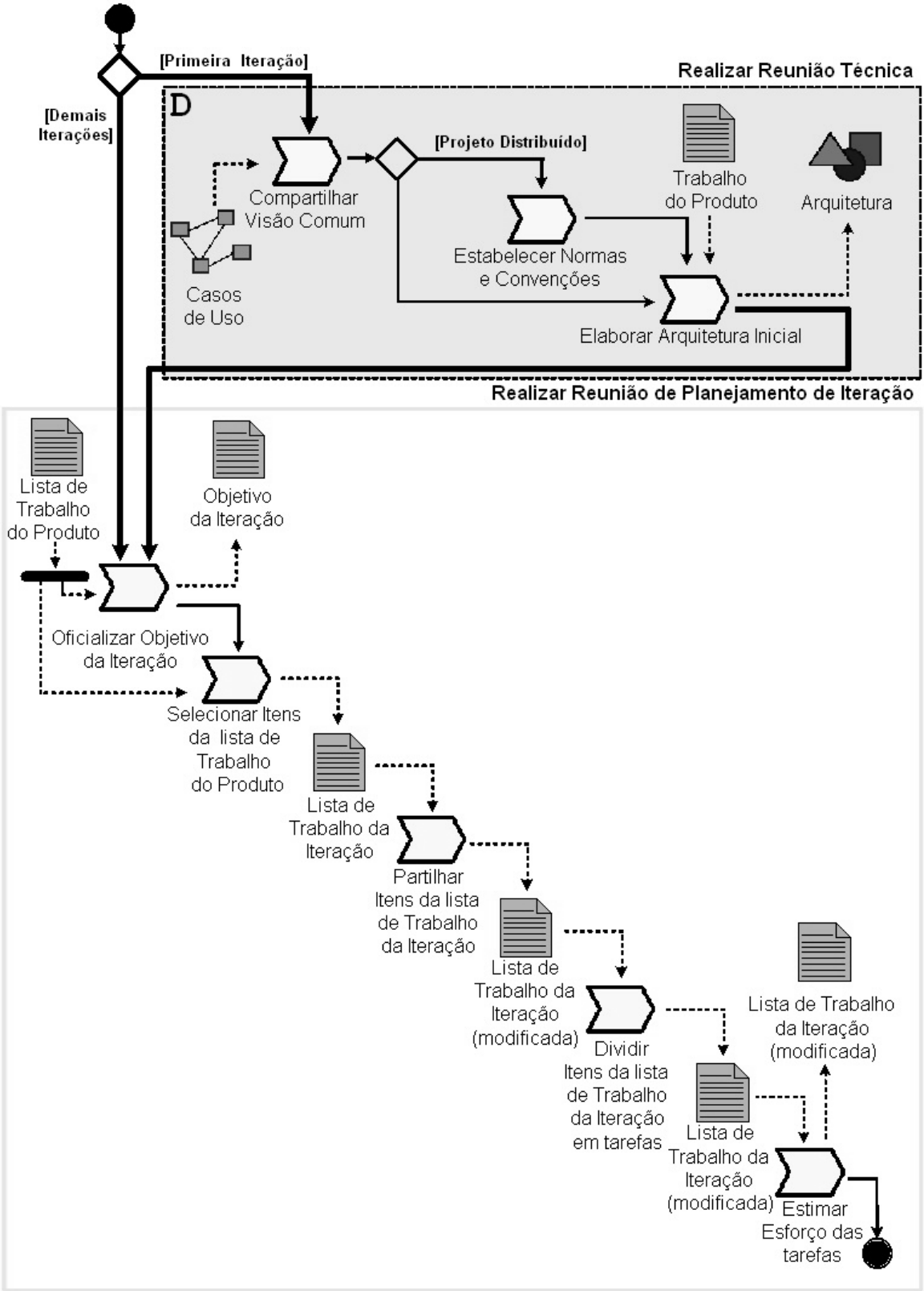


Figura 4.25 – Diagrama de atividades da disciplina Planejar Iteração (estendida)

Depois de Realizar a Reunião Técnica, deve-se Realizar a Reunião de Planejamento de Iteração, para que se possa dar início ao desenvolvimento do incremento do produto, como mostra a próxima subseção.

4.6.4. Disciplina Realizar Iteração (Estendida)

Para auxiliar o desenvolvimento do incremento do produto, foram integrados os padrões: *Named Stable Bases, Application Design is Bounded By Test Design, Smoke Test, Architect Also Implements e Surrogate Customers,* como mostra a Tabela 4.7.

Tabela 4.7 – Padrões integrados a disciplina Realizar Iteração

Padrão	Prática no Scrum	Estereótipos
Named Stable Bases	Ao longo da iteração a Equipe Scrum deve desenvolver um incremento do produto e, para isso, os membros dessa equipe compartilham produtos de trabalho comuns, como o código. Para que todos possam executar suas tarefas de forma mais segura, a Equipe Scrum deve integrar o software constantemente, uma vez por semana, para que a base não fique desatualizada, responsabilidade atribuída a um membro da equipe Scrum.	Atividade (Integrar) Artefato (Base) Padrão Organizacional (*Named Stable Bases*) (Figura 4.26 – retângulo F)
Smoke Test	Na atividade Integrar, antes de colocar a base estável atualizada no repositório, um teste de "fumaça" deve ser realizado para verificar se a base gerada não está corrompida.	Padrão Organizacional (*Smoke Test*) (Figura 4.26 – retângulo F)
Application Design is Bounded By Test Design	Antes de enviar as modificações realizadas para o repositório, os membros da Equipe Scrum devem testar as partes da funcionalidade do software que eles desenvolveram, e esses testes devem ser planejados e executados com base nos casos de uso.	Atividade (Criar Plano de Teste) Artefatos (Plano de Teste e Casos de Uso) (Figura 4.26 – retângulo G
Architect Also Implements	O membro da Equipe Scrum que desempenhou o papel de arquiteto na atividade Elaborar Arquitetura Inicial do produto, também deve Desenvolver.	Padrão Organizacional (*Architect Also Implements*) (Figura 4.26 – retângulo G)
Surrogate Customers	Ao longo da iteração, os membros da Equipe Scrum podem precisar de esclarecimentos do cliente, e o mesmo pode não estar disponível, assim, o Proprietário do Produto deve realizar o papel de substituído do cliente e esclarecer as dúvidas dos membros da Equipe Scrum.	Padrão Organizacional (*Surrogate Customers*) (Figura 4.26 – retângulo H)

A Figura 4.26 mostra o diagrama de classes da disciplina Realizar Iteração (estendida) modelado com estereótipos.

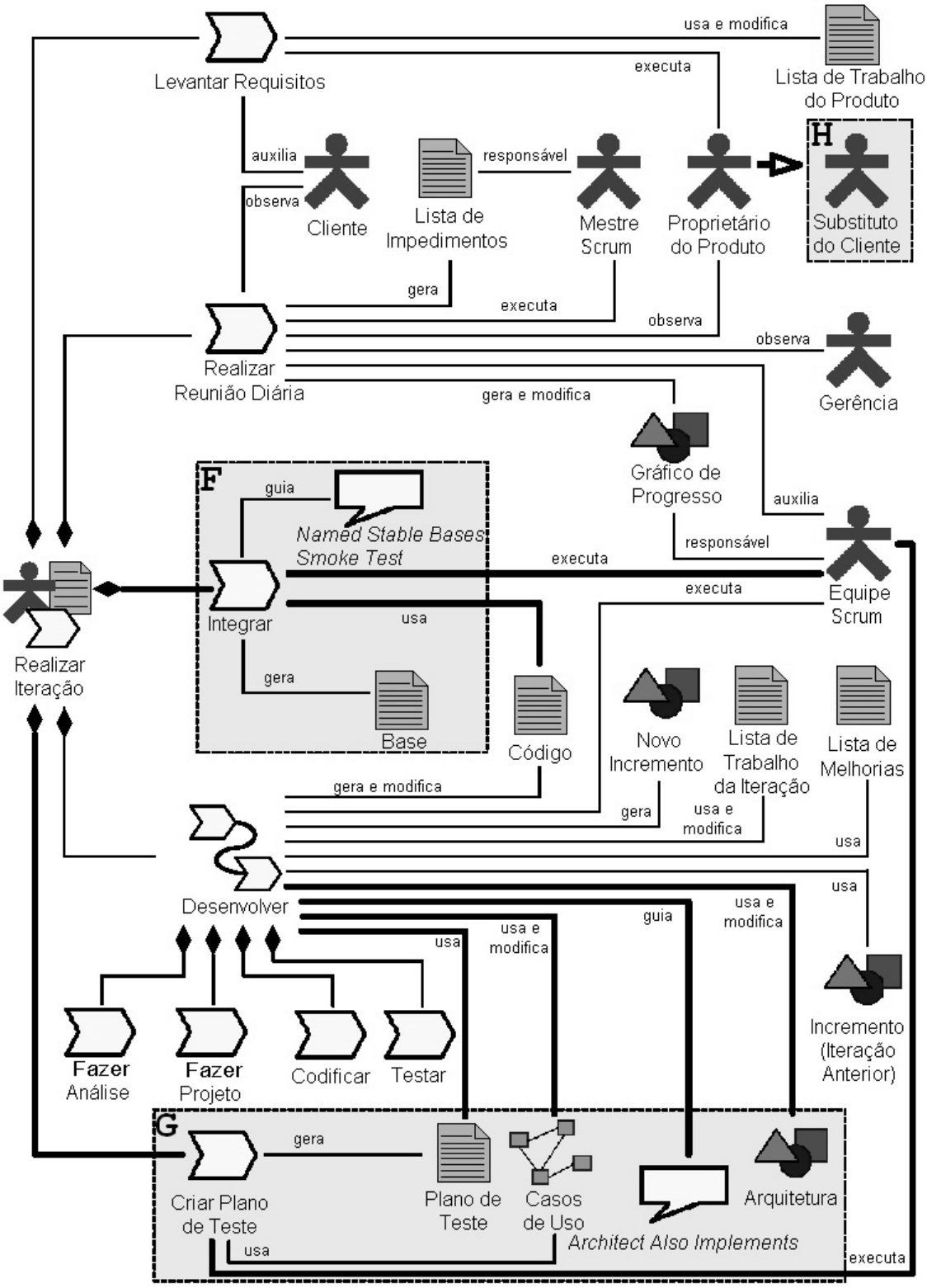


Figura 4.26 – Diagrama de classes da disciplina Realizar Iteração (estendida)

Na Figura 4.27, que mostra o diagrama de atividades da disciplina Realizar Iteração (estendida) modelado com estereótipos SPEM, encontram-se as atividades Criar de Plano de Teste e Integrar, que ocorrem junto com as atividades do conjunto de trabalho Desenvolver e com as atividades Realizar Reunião Diária e Levantar Requisitos.

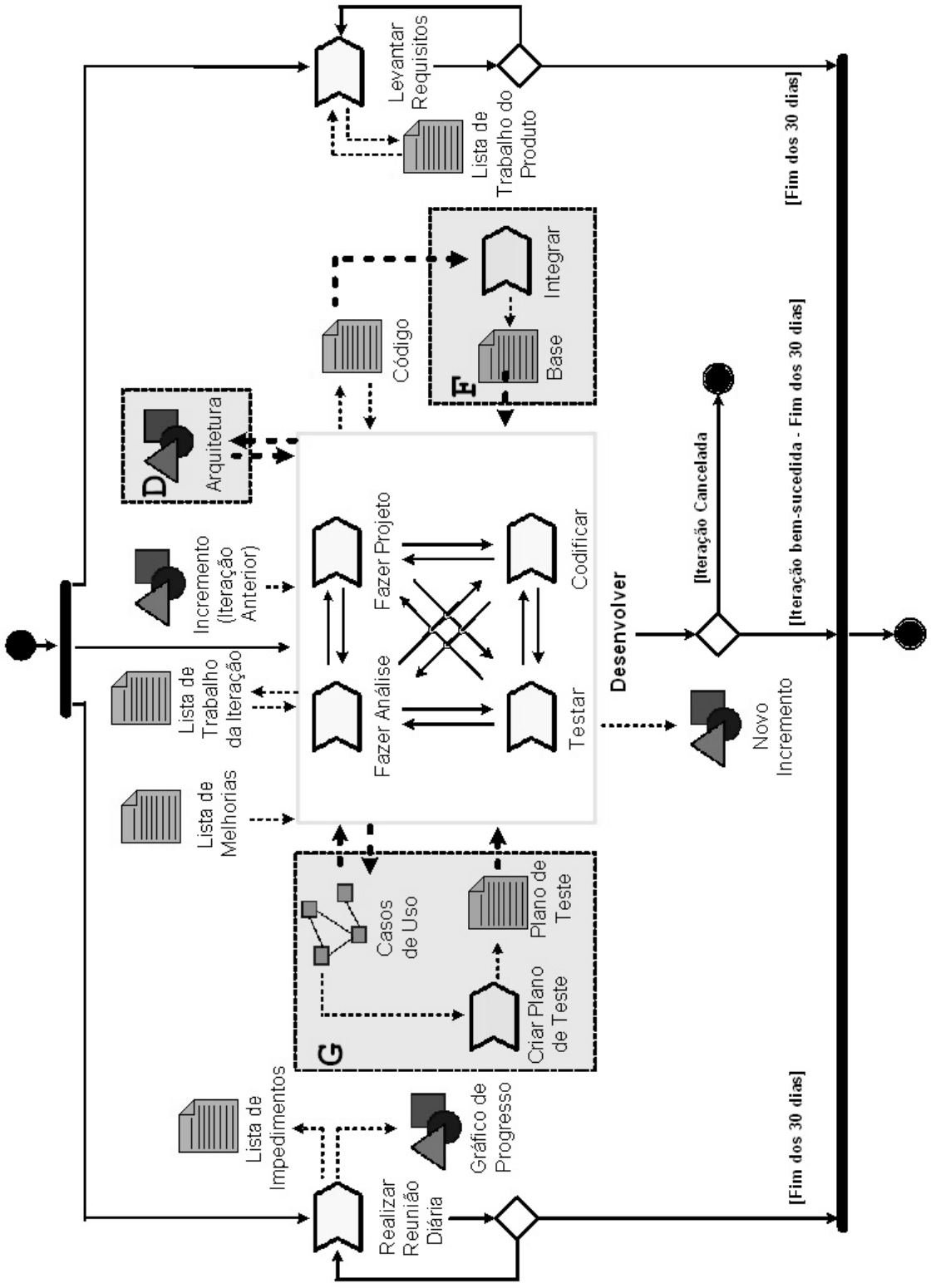


Figura 4.27 – Diagrama de atividades da disciplina Realizar Iteração (estendida)

A integração semanal do software pode facilitar a preparação do incremento do produto para a atividade Apresentar Resultados da disciplina Revisar Iteração, já que pela definição

original do Scrum, a Equipe Scrum não deve gastar mais de uma hora para se preparar para a reunião de revisão (Schwaber e Beedle, 2002).

Um problema observado ao aplicar o Scrum em projetos distribuídos é a impossibilidade de realizar as reuniões diárias com a Equipe Scrum e Mestre Scrum no mesmo local. Assim, e-mail, *instant messenger*, vídeo conferência são exemplos de tecnologias que podem ser utilizadas como alternativa para solucionar esse problema. Um sistema de vídeo conferência seria a melhor opção para permitir a comunicação e colaboração nessas reuniões.

A integração dos padrões gerou novos artefatos no processo, como a arquitetura e casos de uso, e essa documentação deve ser reunida e armazenada pela organização. Assim, apesar da disciplina Finalizar Versão não ter sido modificada por nenhum padrão, os novos documentos gerados impactaram a atividade de documentação, como mostra a próxima subseção.

4.6.5. Disciplina Finalizar Versão (Estendida)

A Equipe Scrum, além de documentar o que foi solicitado pelo cliente, deve reunir os artefatos gerados no processo e armazená-los para gerar uma base de conhecimento. Assim, dois tipos de documentação devem ser gerados, a do cliente e a do projeto, como pode ser visto na Figura 4.28, que mostra o diagrama de classes da disciplina Finalizar Versão (estendida) modelado com estereótipos SPEM.

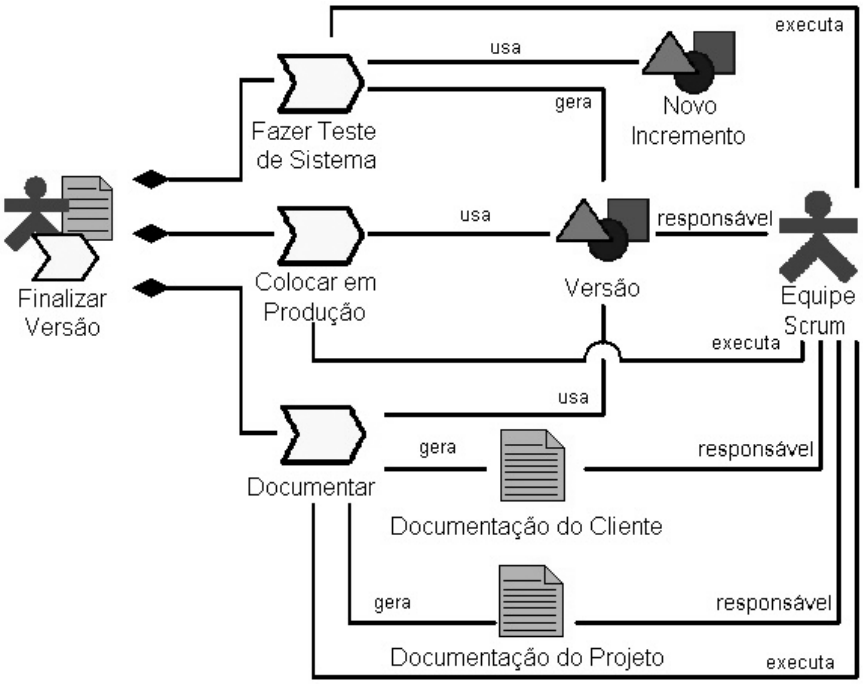


Figura 4.28 – Diagrama de classes da disciplina Finalizar Versão (estendida)

A ordem em que as atividades de finalização de versão devem ocorrer é a mesma do Scrum sem padrões (Figura 4.18), a única diferença é a criação de dois tipos de documentação: a do cliente e a do projeto.

A Figura 4.29 mostra o diagrama de atividades da disciplina Finalizar Versão (estendida) modelado com estereótipos SPEM.

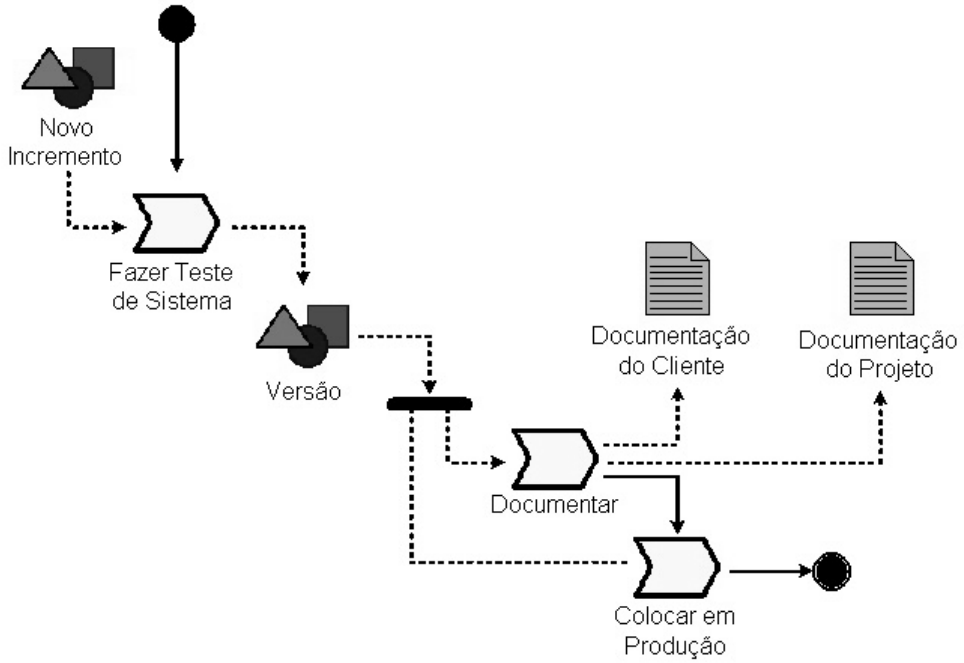


Figura 4.29 – Diagrama de atividades da disciplina Finalizar Versão (estendida)

Os artefatos que compõem a documentação do projeto são representados por meio de um diagrama de classe e de estereótipos SPEM, como mostra a Figura 4.30.

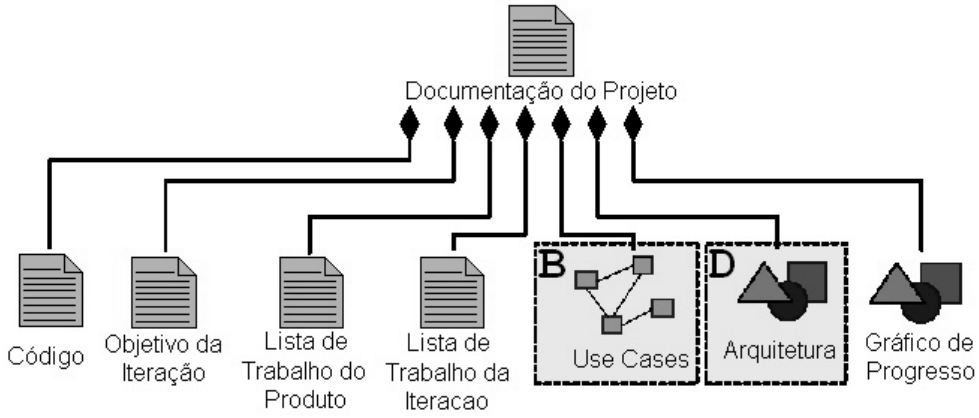


Figura 4.30 – Artefatos que compõem a documentação do projeto

Isso não significa que a Equipe Scrum deve modificar esses artefatos nessa etapa, ao invés disso, ela deve simplesmente reunir e armazenar o que foi gerado, para que eles possam ser utilizados em projetos futuros.

As demais disciplinas do Scrum não foram estendidas e podem ser vistas nos diagramas apresentados na Seção 4.2.

4.7. Considerações Finais

Para utilizar as práticas propostas pelos padrões organizacionais e de processo de forma efetiva com o Scrum, é necessário entender como elas se relacionam com as práticas desse método. A utilização incorreta dos padrões pode atrasar o projeto ao invés de agilizá-lo. Isso foi notado quando uma pequena experiência foi realizada com alunos de pós-graduação compondo duas organizações fictícias, como mencionado na Seção 3.5.

Este capítulo mostrou uma forma ordenada para realizar a utilização efetiva dos padrões organizacionais e de processo com o Scrum. Como detalhado nas Seções de 4.3 a 4.6, essa forma é composta de quatro etapas: 1) Modelagem do Scrum sem padrões, utilizando diagramas de classes e atividades com estereótipos SPEM; 2) Identificação da categoria dos padrões que podem ser integrados ao Scrum; 3) Associação entre esses padrões e as disciplinas do Scrum; 4) Integração dos padrões organizacionais de processo para extensão do Scrum.

Por meio dessas etapas foi possível criar uma extensão do método ágil Scrum que contém práticas que abordam questões organizacionais e técnicas não tratadas por ele, como apresentado na Seção 4.6.

O SPEM permitiu a integração das práticas propostas pelos padrões com as práticas Scrum, facilitando o entendimento de como os padrões estão relacionados com ele. Por meio dos diagramas de classes e de atividades da UML e dos estereótipos SPEM, foi possível representar de forma clara todo o processo Scrum estendido com as novas práticas integradas, como mostrado na Seção 4.6.

Tanto a adaptação quanto a extensão de um método ágil não podem ser realizadas de qualquer maneira, pois a característica ágil do processo deve ser preservada. O processo Scrum, que gerencia e controla o desenvolvimento de software, é baseado em um modelo de controle de processo empírico, que o torna flexível e adaptável, como mencionado na Seção 2.5. Assim, com a integração dos padrões organizacionais e de processo procurou-se melhorar o Scrum sem afetar sua flexibilidade, ou seja, não foram integradas práticas de engenharia específicas para a construção do software.

A versão estendida do Scrum, proposta neste capítulo, contém práticas adicionais para tratar os pontos fracos desse método, já destacados na Tabela 2.11. Além disso, essa extensão inclui práticas para tratar de outros problemas técnicos e organizacionais observados na Seção 4.6. Assim, além de complementar o Scrum, os padrões a ele integrados podem diminuir o impacto da distribuição geográfica na comunicação e colaboração em projetos distribuídos.

A única disciplina do Scrum não estendida é a Revisar Iteração, apresentada na Seção 4.3.5. Porém essa disciplina contém o conjunto de trabalho Realizar Reunião de Retrospectiva, no qual devem ser identificados os padrões que podem ser integrados ao Scrum para melhorar seu processo. Cada organização possui necessidades particulares e pode necessitar de versões diferentes de padrões existentes ou até mesmo de novos padrões. Se a organização não encontrou um padrão para solucionar certo problema de desenvolvimento, a solução adotada será sem a aplicação de um padrão. Contudo, se isso ocorrer, a organização deve utilizar essa reunião para documentar essa solução na forma de padrão. Os padrões documentam lições aprendidas com o passado e fornecem um nome para soluções úteis, e a reunião de retrospectiva é o lugar para procurar esses padrões (Rising e Derby, 2003).

Embora as etapas seguidas para integrar os padrões tenham sido criadas para o método ágil Scrum, é possível que elas possam ser utilizadas com outros métodos ágeis. Assim, algumas diretrizes podem ser seguidas para essa integração:

1. Modelar o método ágil sem os padrões organizacionais e de processo, utilizando o meta-modelo SPEM.
    a. Criar o meta-modelo do método ágil para possibilitar a modelagem dos relacionamentos entre os estereótipos SPEM (conjunto de trabalho, atividades, papéis, artefatos e guias) dentro das disciplinas ou fases.
    b. Dividir o método em fases ou disciplinas, que contém atividades referentes a um tema comum, para organizar e facilitar a integração dos padrões.
    c. Para cada fase, criar um diagrama de classes para representar a perspectiva estática do processo e outro, de atividades, para representar a perspectiva dinâmica do processo utilizando os estereótipos SPEM.
2. Identificar a categoria dos padrões que podem ser integrados ao método ágil, de acordo com a definição de padrões organizacionais e de processo consideradas na Seção 4.4.
3. Associar os padrões organizacionais e de processo às fases ou disciplinas definidas no método ágil.

4. Integrar os padrões organizacionais de processo ao método ágil por meio dos estereótipos SPEM.
    a. Representar os padrões organizacionais por meio do estereótipo Padrão Organizacional, definido na Seção 4.4, e os padrões de processo com os estereótipos: conjunto de trabalho, atividade, papel e artefato.
    b. Representar os padrões organizacionais somente nos diagramas de classes de cada disciplina.

O Capítulo 5 mostra a aplicação do Scrum estendido em um estudo piloto de avaliação, realizado com duas organizações.

Capítulo 5

Estudo Piloto de Avaliação do Scrum Estendido

5.1. Considerações Iniciais

Como indicado no Capítulo 3, havia a necessidade de racionalizar a utilização dos padrões organizacionais e de processo com Scrum, o que foi mostrado no Capítulo 4. Neste capítulo, é realizada a aplicação do processo Scrum estendido, definido no Capítulo 4, em um estudo piloto de avaliação, diferente da experiência realizada para a sua elaboração. Assim, o estudo piloto foi realizado com duas organizações para avaliar se as práticas integradas auxiliam efetivamente o desenvolvimento do software e a organização dos envolvidos no projeto e para verificar se a modelagem com SPEM facilitou o entendimento e aplicação do processo Scrum estendido.

Cada organização desenvolveu um dos seguintes sistemas: 1) Sistema de Controle de Zoológico; 2) Sistema de Controle de Equivalências, com os mesmos requisitos apresentados no Capítulo 3.

Este capítulo está organizado da seguinte forma: a Seção 5.2 apresenta breve descrição e organização do estudo piloto; na Seção 5.3 é apresentado projeto do Sistema de Controle de Zoológico; na Seção 5.4 é apresentado o projeto do Sistema de Controle de Equivalências; a Seção 5.5 apresenta a análise realizada sobre o questionário aplicado às organizações e, por fim, na Seção 5.6 estão as considerações finais.

5.2. Descrição do Estudo Piloto

Nesse estudo piloto procurou-se investigar os benefícios e os problemas da integração das práticas propostas pelos padrões organizacionais e de processo às práticas preconizadas pelo Scrum, bem como as vantagens da modelagem do Scrum estendido utilizando SPEM (OMG, 2005).

O estudo piloto foi realizado com alunos de Graduação dos cursos de Ciência e Engenharia da Computação e de Pós-Graduação em Ciência da Computação, todos da Universidade Federal de São Carlos (UFSCar).

Os alunos foram inicialmente divididos em duas equipes (organizações), denominadas "Organização A" e "Organização B", sendo que a Organização A foi formada por dez alunos de graduação cursando o segundo ou terceiro ano, que possuíam base necessária para o desenvolvimento das atividades do estudo piloto. A Organização B era a mesma que participou da experiência apresentada no Capítulo 3.

Cada organização desenvolveu apenas um sistema, utilizando o Scrum estendido. Os sistemas solicitados para a Organização A e B, respectivamente, foram:

- **Sistema de Controle de Zoológico**: O sistema deve controlar o fluxo dos animais, histórico clínico de cada animal, alimentação de cada setor do Zoológico e os dados relativos a falecimentos. O sistema deve possibilitar o cadastro, alteração e remoção de funcionários, animais, espécies, famílias, ordens, classes, recintos, setores, alimentos, cardápios, sobras de alimentos, históricos clínicos e laudos de necropsia.
- **Sistema de Controle de Equivalências**: Controle de equivalências entre as disciplinas oferecidas pelo Departamento de Computação (DC) aos cursos existentes na Universidade Federal de São Carlos (UFSCar) e as disciplinas de computação cursadas por alunos de outras instituições de ensino superior, que se transferem para os cursos da UFSCar. Esse sistema é o mesmo que foi solicitado na experiência.

Ambas as organizações possuíam membros em diferentes localizações e por isso, procurou-se observar se as práticas integradas ao Scrum auxiliam o desenvolvimento em projetos geograficamente distribuídos, facilitando a comunicação e colaboração.

Antes do início do desenvolvimento dos sistemas, um treinamento, ministrado pelo autor desta dissertação, foi realizado com as duas organizações para apresentação do Scrum estendido modelado com SPEM. As seções seguintes apresentam os principais resultados observados durante o desenvolvimento dos dois sistemas.

5.3. Sistema de Controle de Zoológico

Nesse projeto a Organização A, formada somente por alunos de graduação, desenvolveu o sistema de controle de Zoológico utilizando a tecnologia .NET (Microsoft, 2006a). O Sistema foi desenvolvido em C# (MSDN, 2006a), por meio do ambiente Visual Studio 2005 (MSDN, 2006b), utilizando o banco de dados SQLServer 2005 (Microsoft, 2006b). Essa tecnologia foi selecionada devido à participação de todos os alunos dessa organização no programa de células acadêmicas Microsoft (Microsoft, 2006c). Nesse programa, a Microsoft fornece material necessário para que os alunos estudem as tecnologias Microsoft de acordo com os interesses de cada célula.

A idéia de aplicar o Scrum estendido com os alunos das células surgiu devido à própria intenção da Microsoft de utilizar o Scrum para produzir e revisar seus produtos de forma rápida para atender às necessidades dos seus clientes (Taft, 2005). Assim, após contato com o responsável pelas células acadêmicas da Microsoft, a aplicação do Scrum estendido com alunos das células da UFSCar foi realizada.

Os papéis do Scrum estendido foram atribuídos da seguinte forma: Mestre Scrum: autor desta dissertação; Proprietário do Produto: um membro da organização que também realizou o papel de cliente, por ter conhecimento sobre o sistema a ser desenvolvido; Equipe Scrum: todos os elementos da organização, inclusive o que tem papel de Proprietário do Produto. Os papéis gerados pela integração dos padrões como Substituto do Cliente e Arquiteto foram realizados, respectivamente, pelo autor dessa dissertação e pelo mesmo membro da organização que realizou o papel de Proprietário do Produto.

Todas as práticas do Scrum estendido foram seguidas, com exceção das Reuniões Diárias, que foram realizadas semanalmente, devido à distribuição geográfica dos membros das organizações que não estavam integralmente na universidade e se dedicavam parcialmente as atividades do estudo piloto.

5.3.1. Análise do Estudo Piloto

Na primeira iteração do projeto, foram solicitadas as seguintes tarefas pelo cliente: criação de todas as funções de cadastro do sistema, bem como as funções de alteração e remoção de cadastros.

A Equipe Scrum estimou, devido à falta de experiência de seus membros, 400 horas para a realização dessas tarefas, considerando que a iteração Scrum deve durar um mês (vinte dias úteis) e que eles dedicavam duas horas por dia para realizar as tarefas do estudo piloto. No entanto, essas tarefas foram completadas em 196 horas, 49% do tempo total previsto, ou seja, 204 horas a menos. Isso é apresentado no gráfico de progresso da Figura 5.1, que mostra no eixo horizontal os dias da iteração e no eixo vertical as horas que restam para desenvolver o incremento do sistema.

O Scrum sugere a criação dos gráficos de progresso para acompanhamento do estado do projeto. Esses gráficos que mostram o estilo de desenvolvimento de uma Equipe Scrum (Schwaber e Beedle, 2002), confrontam o trabalho realizado com o que foi planejado.

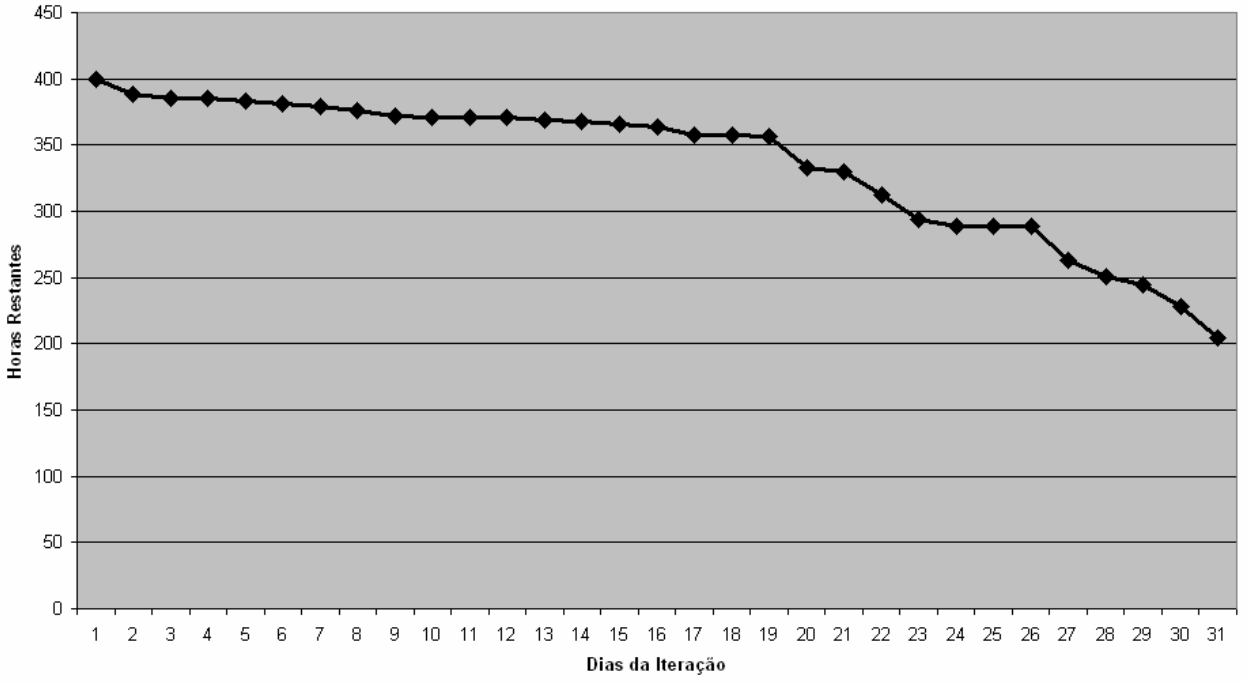


Figura 5.1 – Gráfico de progresso da Equipe Scrum da Organização A

O gráfico da Figura 5.1 também mostra que a Organização A finalizou o desenvolvimento do sistema no trigésimo dia da iteração. Porém, se ela tivesse terminado o trabalho muito antes do trigésimo dia, ela deveria se reunir com o Mestre Scrum e com o Proprietário do Produto para discutir o que deveria ser realizado: 1) a equipe poderia finalizar o incremento do produto e terminar a iteração antecipadamente; 2) A equipe poderia se aprofundar mais no projeto e na arquitetura do produto.

Além das horas da iteração, a Equipe Scrum participou da reunião realizada com o cliente, da reunião técnica e das reuniões de planejamento e de revisão de iteração. Cada uma dessas durou 4 horas, perfazendo um total de 160 horas (4 reuniões X 4 horas X 10 membros). Assim, todo o projeto foi realizado em 356 horas.

O gráfico de progresso da Figura 5.1 mostra que, além de terem realizado o trabalho dentro do prazo, a maioria dos membros da equipe atualizou constantemente esse gráfico de progresso, o que possibilitou o acompanhamento do estado do projeto por todos os envolvidos. Porém, foi possível observar também que a maior parte do trabalho foi realizada após os quinze primeiros dias do início do projeto, o que não é adequado, já que o trabalho deveria ter sido realizado de forma mais uniforme. Um exemplo de gráfico ideal de progresso é apresentado na Figura 5.2.

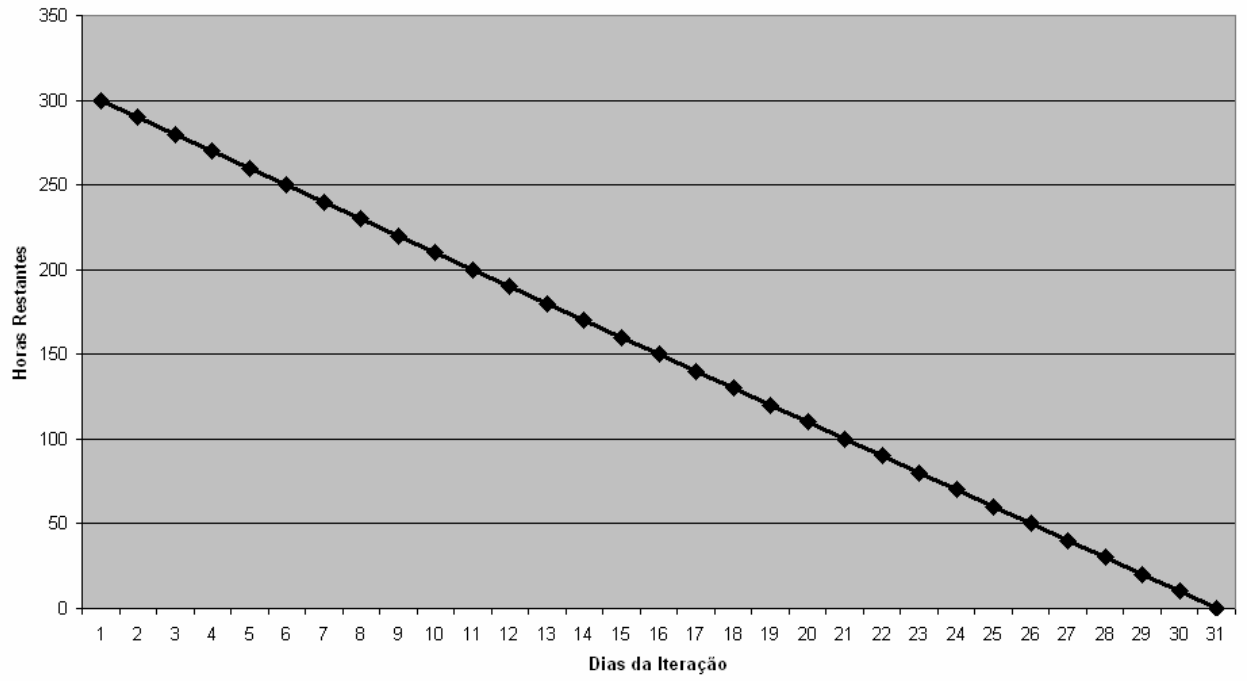


Figura 5.2 – Gráfico ideal de progresso

O gráfico da Figura 5.2 indica que o Proprietário do Produto foi capaz de predizer tudo o que o produto iria precisar anteriormente ao início das iterações, ou seja, nenhuma função foi adicionada ao longo das iterações. O gráfico mostra também que a Equipe Scrum foi capaz de estimar corretamente o esforço para realizar o trabalho do produto e seus membros trabalharam regularmente e sistematicamente sem surpresas ou complexidades inesperadas. Porém, não é comum encontrar esse tipo de gráfico, principalmente em equipes que estão começando a trabalhar com o Scrum.

Neste estudo piloto, além do gráfico de progresso da Equipe Scrum, foram criados gráficos de progresso individuais devido à distribuição geográfica dos membros das equipes, o que possibilitou acompanhar o progresso de cada membro. Esses gráficos permitem, por exemplo, observar que um membro da Equipe Scrum não o atualizou constantemente. Isso é reflexo de um desenvolvedor iniciante com o processo Scrum. Pode-se observar na Figura 5.3 que durante os dez primeiros dias da iteração o desenvolvedor atualizou o gráfico e, após o décimo dia, isso não foi mais realizado. No vigésimo sexto dia esse desenvolvedor foi lembrado dessa responsabilidade pelo Mestre Scrum, o que refletiu em grande quantidade de horas trabalhadas em um dia (quinze horas). Isso não reflete verdadeiramente o progresso desse desenvolvedor, pois ele não deixou de trabalhar após o décimo dia da iteração, apenas esqueceu de atualizar seu gráfico de progresso. Analogamente, se isso acontecesse em uma organização real, o gráfico da Figura 5.3 iria refletir que mais de oito horas foram trabalhadas em um único dia.

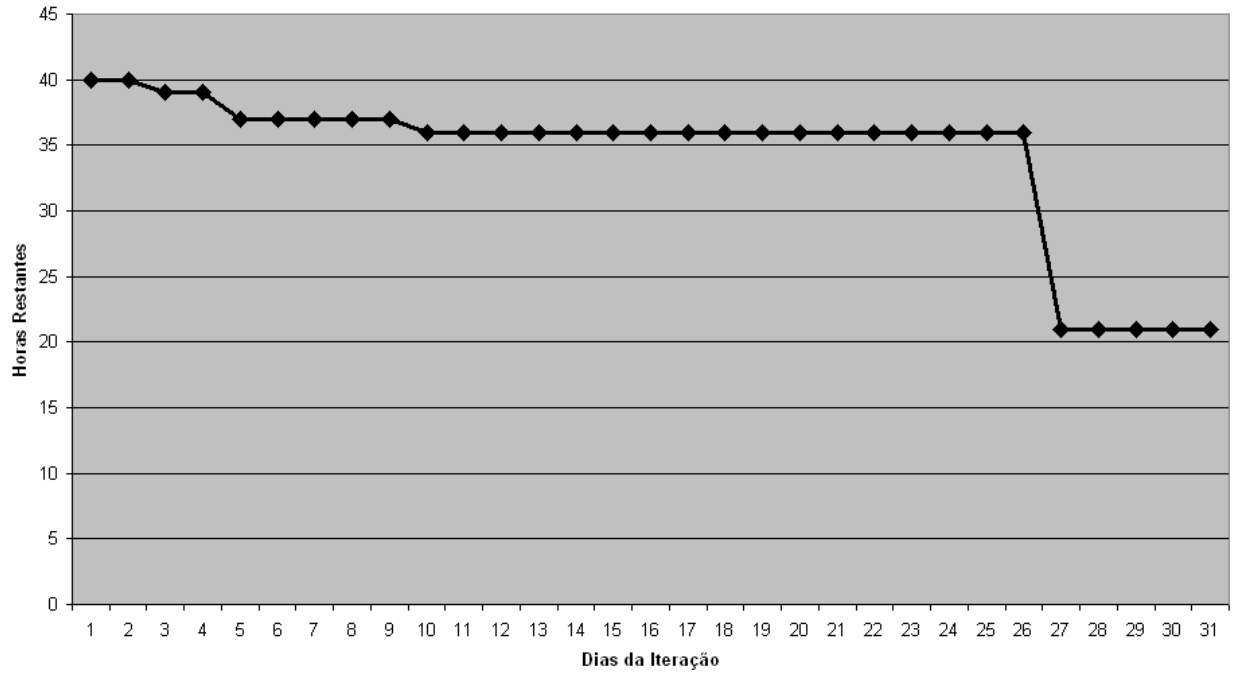


Figura 5.3 – Gráfico de progresso de um membro da Equipe Scrum da Organização A

A Figura 5.4 mostra um gráfico individual de outro membro da Equipe Scrum que atualizou constantemente seu progresso, o que permitiu o real acompanhamento desse membro pelo Mestre Scrum e pelo Proprietário do Produto.

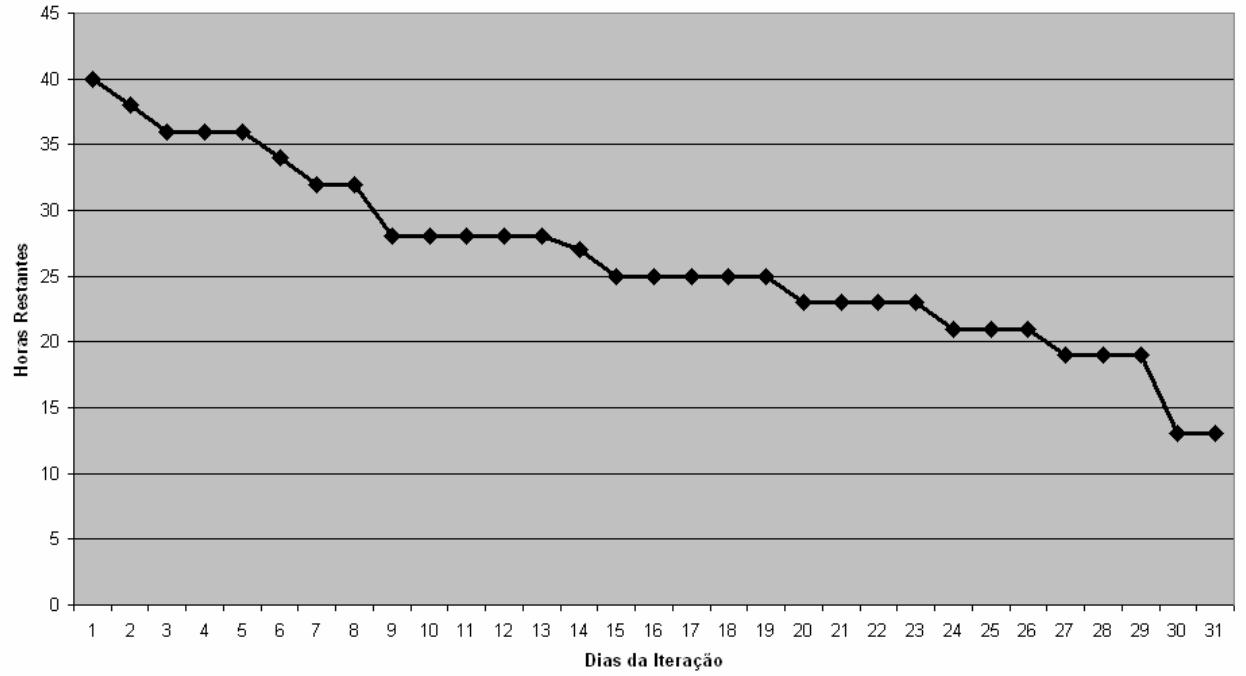


Figura 5.4 – Gráfico de progresso de outro membro da Equipe Scrum da Organização A

Todos os membros da Equipe Scrum possuíam um gráfico individual de progresso, porém só foram apresentados dois para exemplificar as diferenças entre um membro da equipe

que não o atualizou constantemente e outro que o fez, evidenciando que, no primeiro caso, há dificuldade para observação do progresso, enquanto no segundo caso é possível acompanhar o progresso real do desenvolvedor.

A próxima seção apresenta o desenvolvimento de outro sistema utilizando o Scrum estendido.

5.4. Sistema de Controle de Equivalências

Nesse projeto, foi solicitado que a Organização B concluísse o desenvolvimento do sistema de Controle de Equivalências, já iniciado por ela na primeira etapa da experiência.

Apenas 40% da funcionalidade do sistema foi entregue na primeira etapa da experiência e, assim, a organização desenvolveu os outros 60% da funcionalidade desse sistema nesse estudo piloto.

A Organização B continuou utilizando a tecnologia Swing para a camada de apresentação e o *framework* Hibernate para persistência de dados.

Os papéis do Scrum estendido foram atribuídos da seguinte forma: Mestre Scrum: autor desta dissertação; Proprietário do Produto: um membro da organização escolhido por ela própria; Equipe Scrum: todos os elementos da organização, inclusive o que tem papel de Proprietário do Produto; Cliente: papel desempenhado pela professora da disciplina. Os papéis gerados pela integração dos padrões como Substituto do Cliente e Arquiteto foram realizados, respectivamente, pelo autor dessa dissertação e por um membro da organização.

Todas as práticas do Scrum estendido foram seguidas, com exceção das Reuniões Diárias, que foram realizadas semanalmente, devido à distribuição geográfica dos membros das organizações que não estavam integralmente na universidade.

5.4.1. Análise do Estudo Piloto

O sistema foi entregue com sua funcionalidade completamente implementada. A Equipe Scrum terminou o desenvolvimento do sistema de Controle de Equivalências, no trigésimo dia da iteração, em 81 horas, 19% a menos do tempo estimado pelos membros da organização, como apresenta o gráfico da Figura 5.5.

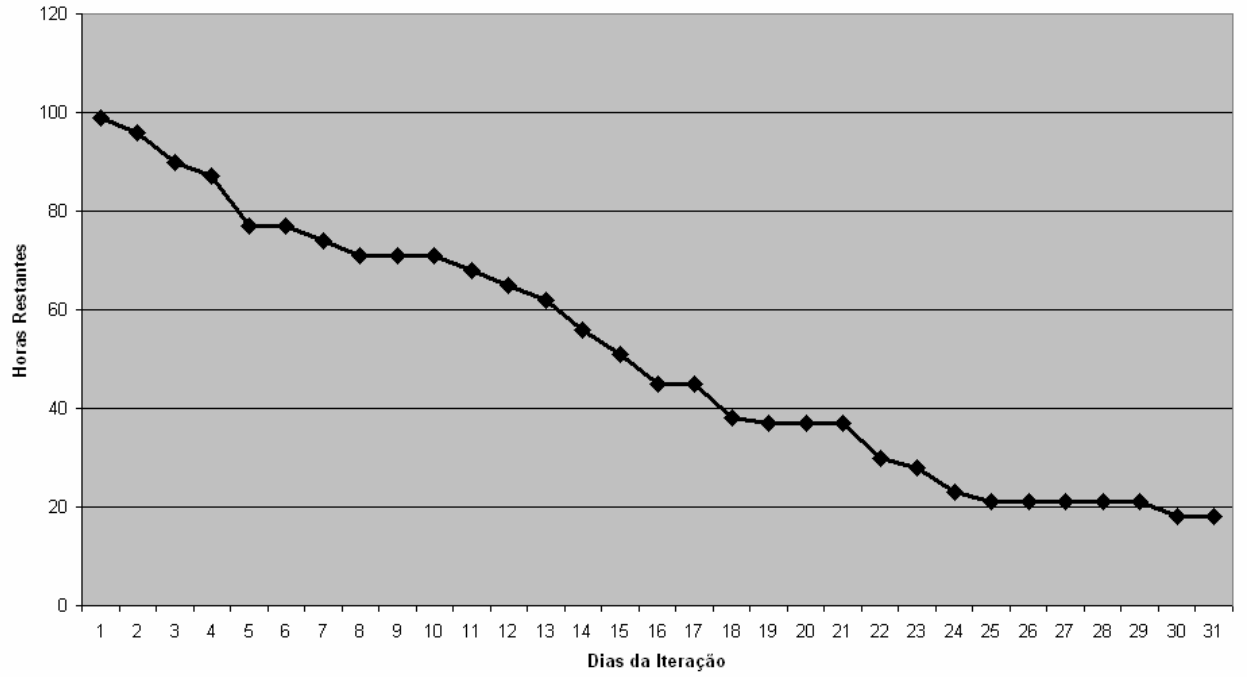


Figura 5.5 – Gráfico de progresso da Equipe Scrum da Organização B

Além das horas da iteração, a Equipe Scrum participou da reunião realizada com o cliente, da reunião técnica e das reuniões de planejamento e de revisão de iteração. Foram gastas 30 horas com essas reuniões, totalizando 111 horas para o todo projeto.

Algumas análises podem ser realizadas com gráficos de progresso individuais dos membros da Equipe Scrum. Por exemplo, a Figura 5.6 mostra que um membro da Equipe trabalhou apenas 18 horas no projeto nos dez primeiros dias da iteração.

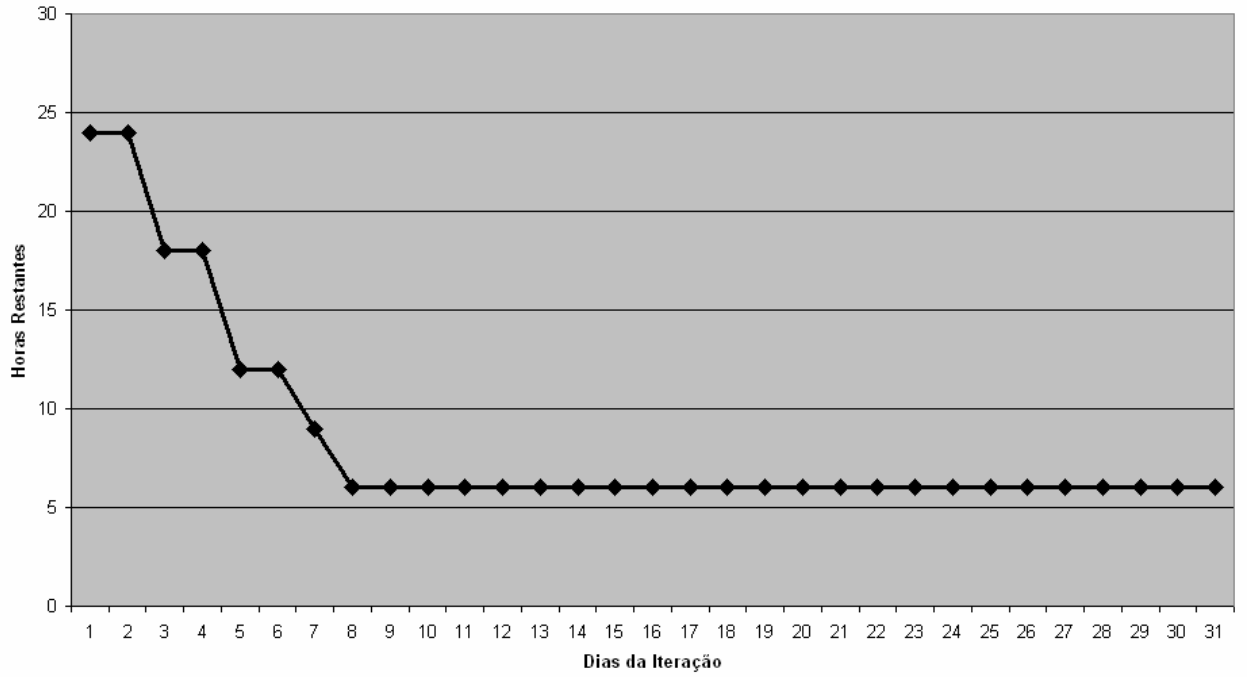


Figura 5.6 – Gráfico de progresso de um membro da Equipe Scrum da Organização B

A Figura 5.7 mostra que um outro membro da Equipe trabalhou pouco nos dez primeiros dias da iteração e 57 horas nos dias restante do projeto.

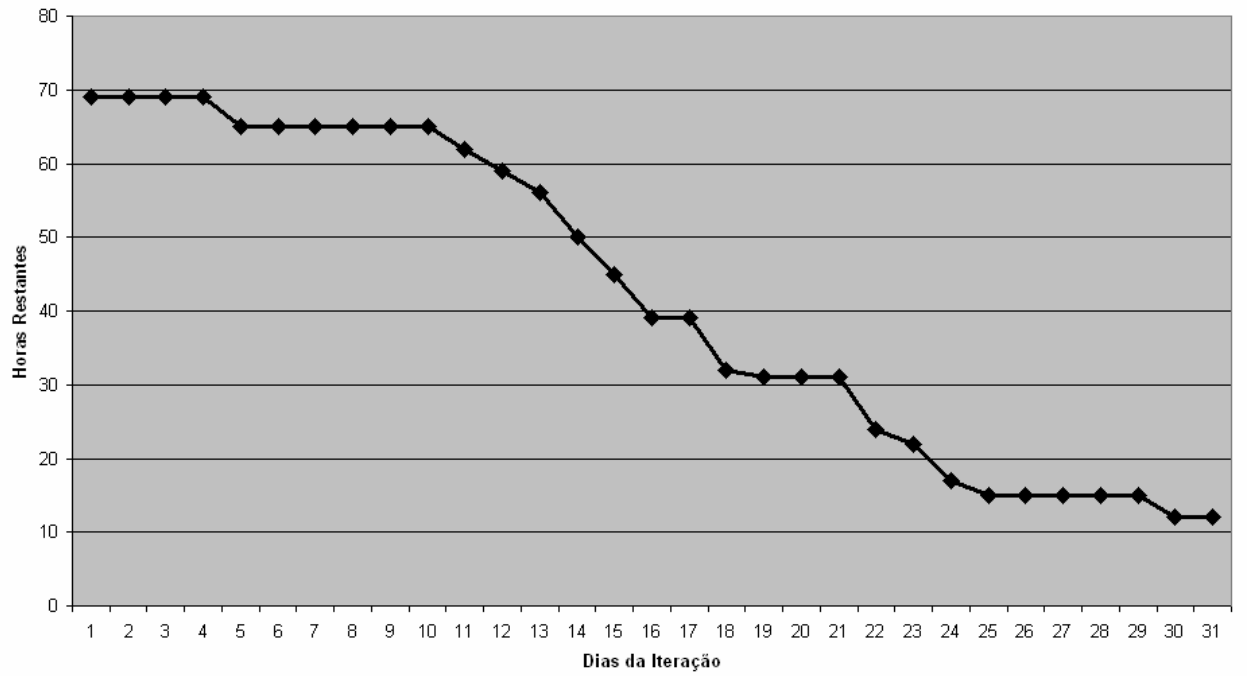


Figura 5.7 – Gráfico de progresso de outro membro da Equipe Scrum da Organização B

Esses gráficos individuais possibilitaram ao Mestre Scrum e Proprietário do Produto uma visão de como a Equipe Scrum estava se comportando com relação ao Projeto, facilitando o acompanhamento dos membros distribuídos.

Quando da realização da experiência, foi feita uma estimativa para avaliar quanto tempo seria necessário para que a Organização B implementasse os 60% restantes do sistema, sem a utilização do Scrum em conjunto com os padrões, e chegou-se a um total de 221 horas, como mostrado na Seção 3.5.2.

Esse estudo piloto indicou que, com a utilização do Scrum estendido, o tempo de desenvolvimento foi reduzido. O gráfico da Figura 5.8 mostra que, proporcionalmente, a produtividade da Organização B melhorou com a utilização do Scrum estendido.

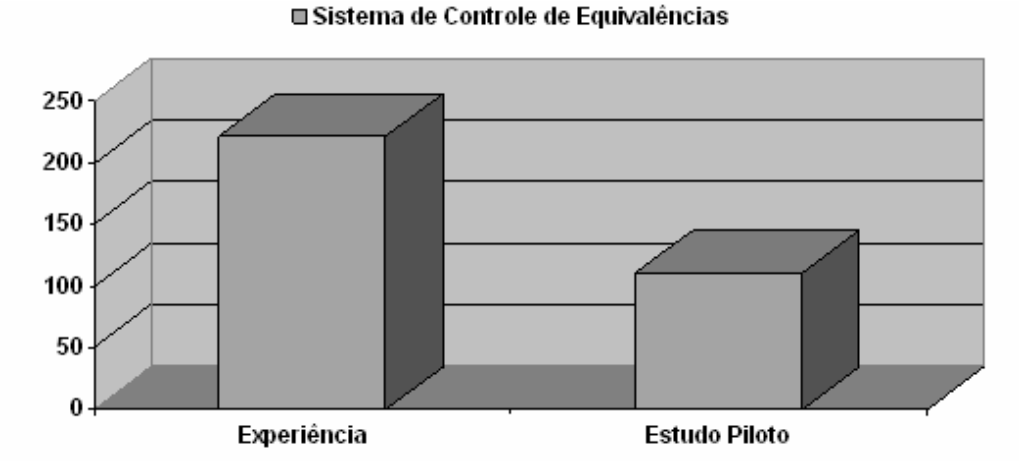


Figura 5.8 – Tempo de desenvolvimento do sistema de Controle de Equivalências

A próxima seção apresenta a análise realizada por meio do questionário aplicado às organizações.

5.5. Análise dos Questionários Aplicados às Organizações

O questionário, apresentado no Apêndice A, foi respondido pelos membros das duas organizações, para avaliação das vantagens da integração das práticas propostas pelos padrões e da utilização do Scrum estendido modelado com SPEM. As subseções a seguir apresentam as análises realizadas sobre as opiniões dos membros das organizações.

5.5.1. Análise dos Padrões Integrados ao Scrum

Por meio do questionário, os membros das organizações avaliaram as vantagens da integração dos padrões organizacionais e de processo ao Scrum. A Tabela 5.1 mostra os resultados obtidos. A primeira coluna apresenta o nome do padrão e a segunda mostra as principais vantagens da sua integração.

Tabela 5.1 – Vantagens da integração dos padrões organizacionais e de processo

Padrão	Principais Vantagens
Self Selecting Team	Facilita o desenvolvimento das atividades em equipe, pois os interesses individuais são os mesmos.
Scenarios Define Problem	Criar casos de uso abrangentes no início do projeto, por apenas três membros da Equipe Scrum, para depois refiná-los iterativamente, possibilita entendimento rápido do domínio do problema, fornece uma visão abrangente e comum do sistema e facilita o desenvolvimento, pois a complexidade de cada caso de uso pode ser aumentada aos poucos. Além disso, os casos de uso auxiliam a divisão das tarefas, já que todos os envolvidos têm conhecimento abrangente sobre domínio do problema.
Small Writing Team	
Breadth Before Depth	
Spiral Development	

Face To Face Before Working Remotely	Realizar uma reunião com todos os envolvidos no projeto possibilita que todos eles se conheçam no início do projeto, facilitando a comunicação e colaboração, principalmente em equipes geograficamente distribuídas.
Unity Of Purpose	Compartilhar a visão comum do responsável pelo projeto facilita o estabelecimento de um consenso entre os desenvolveres que têm experiências e opiniões diferentes.
Standards Linking Locations	As normas e convenções previamente estabelecidas permitem a rápida integração das partes do software, evitando possíveis ajustes por diferenças entre ambientes remotos.
Lock 'Em Up Together	Criar uma arquitetura não detalhada antes do início do desenvolvimento permite um melhor entendimento do sistema e facilita a divisão das tarefas, já que a arquitetura inicial permite aos desenvolvedores ter uma idéia da complexidade do sistema.
Organization Follows Location	Dividir o trabalho de acordo com as localizações possibilita a tomada de decisões localmente e dá maior autonomia no processo, diminuindo a necessidade de comunicação com outras localizações.
Code Ownership	Atribuir a cada localização a propriedade sobre o código possibilita a cada localização ter maior autonomia para realizar o desenvolvimento e, além disso, permite que a propriedade sobre o código seja coletiva internamente em cada localização, o que contribui para a agilidade no desenvolvimento.
Generics And Specifics	Facilita a divisão das tarefas e a agilidade no desenvolvimento. Os casos de uso e arquitetura inicial permitem a identificação das tarefas mais complexas que devem ser atribuídas aos membros mais experientes da Equipe Scrum.
Surrogate Customers	Facilita a tomada de decisões e esclarecimento de dúvidas mais rapidamente e, conseqüentemente, contribui para o progresso do projeto.
Named Stable Bases	Permite acompanhar o desenvolvimento do sistema e a descoberta de erros antes do término da iteração, o que facilita a sua correção. Além, disso a integração semanal facilita a execução iterativa dos testes.
Private Workspace	Permite que todos os membros da equipe trabalhem sobre a base de software mais atualizada, dando maior segurança para os desenvolvedores trabalharem nas suas área privadas. Além disso, facilita o acompanhamento do projeto e a obtenção rápida de artefatos compartilhados.
Repository	
Smoke Test	Permite a identificação rápida de erros após a integração do software e previne o aparecimento de novos erros que podem ser gerados ao se utilizar uma base de software corrompida.
Application Design is Bounded By Test Design	Os detalhamentos dos casos de uso fornecem um guia de testes, facilitando a identificação de todas as seqüências de ações que devem ser executadas no sistema.
Architect Controls Product	Facilita o estabelecimento de consenso na criação da arquitetura inicial e refinamento da arquitetura durante as iterações.
Architect Also Implements	

De todos os padrões integrados, os mais relevantes, citados pela maioria dos membros das organizações, foram: *Scenarios Define Problem, Breadth Before Depth, Face To Face Before Working Remotely, Lock 'Em Up Together, Surrogate Customers, Named Stable Bases* e *Repository*.

O questionário, além de abordar as vantagens da aplicação dos padrões, continha questões sobre o Scrum estendido modelado com SPEM, como é comentado na próxima subseção.

5.5.2. Análise do Scrum Estendido e da sua Modelagem com SPEM

No questionário também foram abordadas questões sobre o Scrum estendido modelado com SPEM, como: facilidade de compreensão, visualização, facilidade de manutenção e comparação com o Scrum original. Além disso, o questionário abordou perguntas sobre a qualidade da comunicação e a colaboração nas organizações, que são questões críticas em projetos geograficamente distribuídos.

Os resultados obtidos indicaram que o Scrum estendido é mais completo e mais fácil de seguir com relação ao Scrum original, segundo todos os membros das organizações. Ainda segundo eles, o Scrum estendido inclui um conjunto bem definido de artefatos e atividades adicionais que preenchem lacunas do Scrum original, possibilitando melhor organização dos envolvidos no projeto e gerenciamento do mesmo. Outras vantagens citadas foram que o Scrum estendido: facilita o direcionamento das atividades, divisão das tarefas entre os membros da Equipe Scrum, e mesmo com a integração dos padrões, continua flexível como o Scrum original, fornecendo liberdade na etapa de desenvolvimento de software. Além disso, segundo a opinião dos membros das organizações, as novas práticas são úteis para auxiliar tanto os desenvolvedores menos experientes quanto os mais experientes.

Como os membros das duas organizações estavam geograficamente distribuídos, foi necessário avaliar o impacto da comunicação e colaboração no projeto. Assim, os membros das organizações analisaram a qualidade da comunicação e colaboração no estudo piloto realizado, sendo os resultados apresentados pelos gráficos das Figuras 5.9 e 5.10, respectivamente.

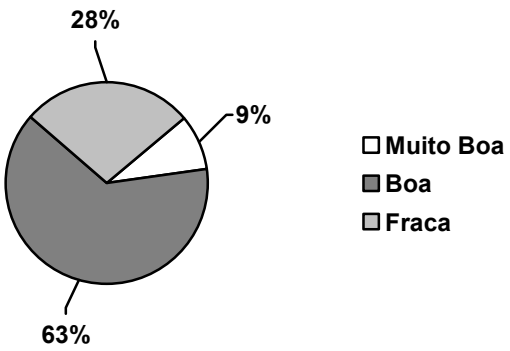


Figura 5.9 – Qualidade da comunicação no estudo piloto

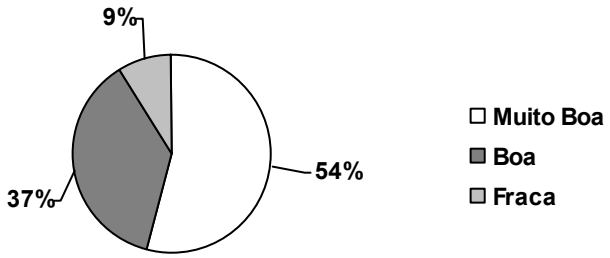


Figura 5.10 – Qualidade da colaboração no estudo piloto

Os gráficos das Figuras 5.9 e 5.10 mostram que a maioria dos membros afirmou que a comunicação no projeto foi boa e que a colaboração foi muito boa. Isso indica que os padrões *Face To Face Before Working Remotely, Organization Follows Location, Standards Linking Locations* e *Code Ownership* diminuíram o impacto da distribuição geográfica na comunicação e colaboração do envolvidos no projeto. Esses padrões facilitaram a comunicação e colaboração durante o projeto, pois todos envolvidos já se conheciam, cada localização tinha autonomia para realizar as tarefas que foram atribuídas a ela, as partes do software que estava sendo desenvolvido pelos diferentes locais seguiam as mesmas normas e convenções e, internamente a propriedade do código era coletiva. Além disso, esse resultado indica que a utilização de e-mail e *instant messenger* foi suficiente para realizar a comunicação e colaboração remotamente.

A modelagem do Scrum estendido também foi analisada pelas organizações quanto à facilidade de compreensão, visibilidade e facilidade de manutenção. Os resultados foram satisfatórios, como mostram os gráficos das Figuras 5.11, 5.12 e 5.13.

A Figura 5.11 mostra a opinião dos membros das organizações com relação à facilidade de compreensão do processo Scrum estendido.

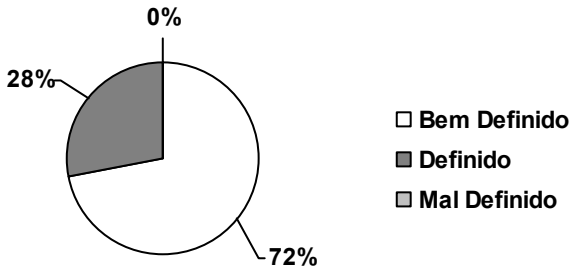


Figura 5.11 – Facilidade de compreensão do processo Scrum estendido

A maioria dos membros das organizações (72%) afirmou que o processo Scrum estendido é bem definido, com suas atividades, artefatos e papéis bem explicados no processo modelado com SPEM, ou seja, o processo define claramente as atividades que devem ocorrer no projeto, os responsáveis por cada uma delas, os artefatos gerados e utilizados em cada atividade, bem como a seqüência que em que devem ocorrer. Salienta-se que nenhum dos membros das organizações afirmou que o processo Scrum estendido é mal definido, ou seja, que não se tem idéia da responsabilidade de cada papel, artefatos que devem ser gerados ou consumidos nas atividades e a seqüência em que essas devem ocorrer.

A Figura 5.12 mostra a opinião dos membros das organizações com relação à visibilidade do processo Scrum estendido.

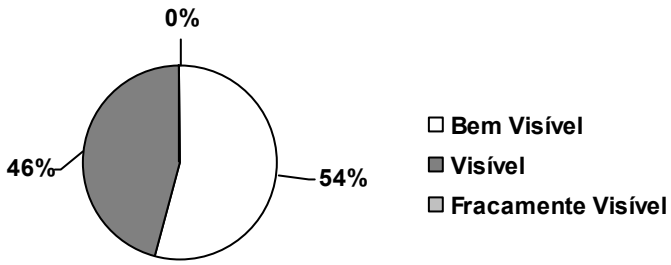


Figura 5.12 – Visibilidade do processo Scrum estendido

A maioria dos membros das organizações (54%) afirmou que o processo Scrum estendido é bem visível, com suas atividades, artefatos e papéis bem organizados no processo modelado com SPEM, ou seja, os seus relacionamentos estão visivelmente claros e bem estruturados na modelagem. Salienta-se que nenhum membro das organizações afirmou que o processo é fracamente visível, ou seja, que o processo não está claro e bem estruturado.

A Figura 5.13 mostra a opinião dos membros das organizações com relação à facilidade de manutenção do processo Scrum estendido.

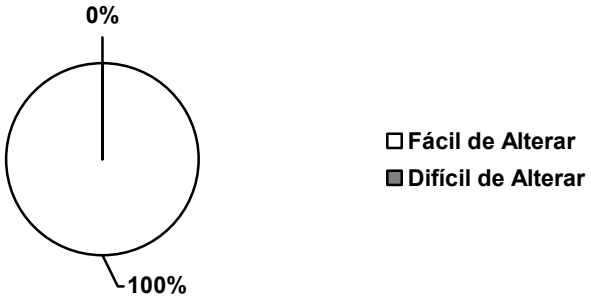


Figura 5.13 – Facilidade de manutenção do processo Scrum estendido

Como mostra o gráfico da Figura 5.13, todos os membros das organizações (100%) afirmaram que no processo Scrum estendido é simples fazer modificações, ou seja, de acordo com a opinião dos membros das organizações, devido ao processo estar bem definido seria fácil alterá-lo para atender às necessidades particulares de um projeto ou organização.

O questionário permitiu analisar as vantagens do Scrum estendido sobre o Scrum original e da sua modelagem utilizando o SPEM. Porém, alguns problemas ocorridos no estudo piloto também foram reportados no questionário e outros foram observados pelo autor desta dissertação, como comentados na subseção a seguir.

5.5.3. Problemas Observados na Aplicação do Scrum Estendido

Apesar dos resultados terem indicado a conveniência da integração dos padrões ao Scrum, alguns problemas foram destacados pelas organizações no questionário. Os principais são:

- A presença do substituto do cliente não pode garantir com segurança a tomada de decisão certa, a menos que ele esteja totalmente interado do problema do cliente.
- Apesar da comunicação ter sido boa nos projetos, ela pouco se aproximou do contato diário definido como prática no Scrum. Isso ocorreu devido à distribuição geográfica, mas também devido à freqüência das reuniões de acompanhamento.
- Só a reunião semanal não foi suficiente para o acompanhamento do projeto e, mesmo com a distribuição geográfica, as reuniões de acompanhamento deveriam ocorrer em freqüência maior, três vezes por semana.
- O processo não define atividades relacionadas ao reúso de artefatos provenientes de outros projetos como pesquisa, aquisição e integração. Essas atividades poderiam ser acrescentadas no início do processo.

Além desses problemas mencionados pelas organizações, outros foram observados pelo autor dessa dissertação, que realizou o papel de Mestre Scrum em ambas as organizações:

- Com relação à execução das novas práticas propostas pelos padrões organizacionais e de processo integrados ao Scrum, no início, as bases estáveis (*Named Stable Bases*) não foram criadas na freqüência certa, ou seja, uma vez por semana. Só após a segunda semana é que essa prática começou a ser seguida corretamente. Outro problema ocorreu com a criação dos planos de teste com base nos casos de uso (*Application Design is Bounded By Test Design*) que não foram gerados para agilizar o processo de desenvolvimento. As organizações utilizaram a própria descrição do caso de uso para realizar os testes.

- A adaptação das reuniões diárias Scrum para reuniões semanais devido à distribuição geográfica. Apesar das tecnologias disponíveis para a comunicação a distância, como *instant messenger* e e-mail, não foi possível realizar as reuniões diárias com as duas organizações em um horário comum, devido à indisponibilidade de muitos membros que se dedicavam a outras atividades profissionais, de graduação ou pós-graduação. Em organizações não fictícias isso é possível e o ideal seria a utilização de um sistema de vídeo conferência para as reuniões diárias para permitir melhor comunicação e colaboração.
- Pode-se verificar também que o caráter fictício das organizações não permitiu uma melhor análise do Scrum estendido. Como os membros das organizações não estavam totalmente dedicados ao desenvolvimento dos sistemas, as atividades não ocorreram em um ritmo normal, ou seja, os membros das organizações poderiam deixar de trabalhar um dia e trabalhar mais no outro para cumprir o cronograma. Por exemplo, foi observado na Organização B que a maior parte do trabalho foi realizada nos últimos quinze dias da iteração, e pouco foi feito nos primeiros dias, porém, como os envolvidos não estavam dedicados, foi possível cumprir o cronograma trabalhando mais horas nos últimos dias. Além disso, apesar dos membros das organizações terem sido convidados a trabalhar, alguns alunos não se envolveram no projeto de forma satisfatória e isso afetou a produtividade das organizações, pois para se utilizar o Scrum estendido e qualquer outro método ágil, é necessário comprometimento de todas as pessoas envolvidas no projeto.

Salienta-se que os problemas de comunicação e das reuniões diárias mencionados pelas organizações ocorreram devido à distribuição geográfica de seus membros. Em um ambiente com todos trabalhando no mesmo local, as reuniões seriam realizadas diariamente e a comunicação poderia ser melhor do que em um ambiente distribuído. O Scrum estendido tenta apenas minimizar o impacto da distribuição no projeto, por meio de algumas práticas integradas, porém ele não é uma adaptação para projetos distribuídos, mas sim uma extensão para melhor gerenciar tanto projetos realizados no mesmo local quanto projetos distribuídos.

5.6. Considerações Finais

Por meio do estudo piloto foi possível verificar benefícios da integração de alguns padrões organizacionais e de processo com o Scrum e da sua modelagem com SPEM, como proposto no Capítulo 4. As práticas propostas por esses padrões são úteis e auxiliam

efetivamente a organização dos envolvidos no projeto e o desenvolvimento do software, como mencionado na Seção 5.5. Mesmo com a integração dessas práticas, a característica ágil do processo foi preservada, pois procurou-se melhorar o Scrum sem afetar a sua flexibilidade. Não foram integradas práticas específicas de engenharia para a construção do software. Assim, o Scrum estendido, da mesma forma que o Scrum original, pode ser adaptado de acordo com as necessidades da organização que já utiliza práticas específicas de engenharia de software.

Os padrões organizacionais e de processo integrados ao Scrum tratam de questões técnicas e organizacionais não abordadas por esse método, como mencionado na Seção 4.6. Além disso, alguns desses padrões podem diminuir o impacto da distribuição geográfica na comunicação e colaboração em projetos distribuídos, que têm se tornado cada vez mais comuns na sociedade, uma vez que devido aos avanços da Internet, grandes corporações, empresas e instituições educacionais se beneficiam cada vez mais da distribuição geográfica para aumentar suas oportunidades. Os resultados obtidos com o questionário aplicado às organizações para avaliar a qualidade da comunicação e colaboração no estudo piloto são apresentados de forma resumida na Tabela 5.2.

Tabela 5.2 – Resultados obtidos sobre a modelagem do Scrum estendido

Processo	**Critério**					
	Comunicação			**Colaboração**		
	Muito Boa	Boa	Fraca	Muito Boa	Boa	Fraca
Scrum Estendido	9%	63%	28%	54%	37%	9%

Assim, de acordo com a opinião dos membros das organizações, os padrões integrados facilitam o desenvolvimento de software tanto em projetos locais quanto distribuídos. Além disso, os gráficos individuais de progresso, criados para todos os elementos das organizações e apresentados nas Seções 5.3 e 5.4, foram úteis para o acompanhamento remoto dos envolvidos nos projetos.

Os resultados indicaram que o Scrum estendido melhorou a organização dos envolvidos no projeto, facilitou a realização das atividades e possibilitou uma melhor divisão das tarefas entre os membros das organizações. Além disso, a modelagem do Scrum estendido facilitou o entendimento do processo, direcionando de forma mais clara os envolvidos no projeto, como mencionado na Seção 5.5. Os resultados obtidos com o questionário aplicado às organizações para avaliar a modelagem do Scrum estendido realizada com SPEM são apresentados de forma resumida na Tabela 5.3.

Tabela 5.3 – Resultados obtidos sobre a modelagem do Scrum estendido

Processo	Critério							
	Compreensão			Visibilidade			Manutenção	
	Bem definido	Definido	Mal definido	Bem visível	Visível	Fracamente visível	Fácil de Alterar	Difícil de Alterar
Scrum Estendido	72%	28%	0%	54%	46%	0%	100%	0%

Embora os resultados obtidos indiquem que as práticas propostas pelos padrões são úteis e que o Scrum estendido é fácil de seguir, devido à sua modelagem, problemas foram observados na sua aplicação, como a não realização das reuniões diárias Scrum, que foram substituídas por reuniões semanais, apesar das tecnologias disponíveis para a comunicação a distância, como *instant messenger* e e-mail. Essa adaptação nas reuniões não deveria ser realizada em uma organização real, cujos membros estão dedicados às atividades do projeto e possuem essas tecnologias para comunicação remota.

A modelagem do Scrum estendido com SPEM facilita o entendimento do processo, sua visualização e manutenção. Os relacionamentos entre todos os elementos do processo são representados pelos diagramas de classe, enquanto a ordem em que as atividades devem ocorrer são representadas pelos diagramas de atividades. Assim, com esses dois tipos de diagramas juntamente com os estereótipos SPEM é possível representar de forma clara o processo Scrum estendido com as práticas propostas pelos padrões organizacionais e de processo, como mostrado na Seção 4.6.

Capítulo 6

Conclusões

6.1. Considerações Iniciais

O Scrum, como outros métodos ágeis, possui pontos fracos que necessitam de alternativas para melhorá-los. Uma alternativa adotada neste trabalho para tratar esses pontos fracos foi utilizar padrões organizacionais e de processo, que documentam soluções de sucesso para problemas recorrentes no desenvolvimento de software.

Sendo assim, neste trabalho foi realizada uma experiência de uso dos padrões organizacionais e de processo com Scrum, em duas organizações fictícias, para verificar os problemas que surgem com essa utilização conjunta. Durante o planejamento dessa experiência, foi possível identificar uma relação direta entre alguns padrões organizacionais e de processo existentes e as práticas e papéis definidos no Scrum, como mencionado na Seção 3.3. Com base nessa experiência, observou-se a necessidade de elaborar uma forma ordenada para racionalizar a utilização conjunta dos padrões organizacionais e de processo com o Scrum, para possibilitar uma melhoria efetiva no seu processo.

As práticas de gerenciamento do Scrum estão organizadas de forma coerente e definem um processo de gerenciamento de software e, para que as práticas de sucesso propostas pelos padrões sejam utilizadas de forma efetiva com essas práticas do Scrum, é necessário entender o relacionamento entre elas e depois integrá-las. Para possibilitar essa integração, o meta-modelo para descrição de processo SPEM (OMG, 2005) foi utilizado.

6.2. Resultados

Alguns resultados importantes com relação à utilização dos padrões com o Scrum puderam ser observados na experiência apresentada no Capítulo 3. Quando o Scrum com padrões foi aplicado, inicialmente, as duas organizações entregaram os sistemas desenvolvidos no prazo estabelecido e com suas respectivas funcionalidades completas, o que não ocorreu quando eles não foram utilizados. Além disso, todos os membros das organizações se mostraram satisfeitos com a utilização dos padrões organizacionais e de processo com o Scrum e

mencionaram que os padrões foram úteis para o desenvolvimento e que eles contribuíram para melhor organização das equipes e divisão das tarefas entre seus membros.

Embora essa experiência de uso tenha fornecido resultados positivos, problemas foram observados na aplicação dos padrões com o Scrum e, assim, verificou-se a necessidade de racionalizar essa aplicação, o que possibilitou a criação de uma extensão do Scrum, apresentada na Seção 4.6.

Quatro etapas foram definidas para possibilitar a integração dos padrões organizacionais e de processo com o Scrum:

1. Modelagem do processo Scrum sem padrões, por meio do meta-modelo SPEM. As atividades do Scrum foram divididas em disciplinas, de acordo com um tema comum. Cada disciplina foi modelada utilizando os estereótipos SPEM e diagramas de classes e de atividades da UML, para representar as perspectivas estática e dinâmica do processo, respectivamente;
2. Classificação dos padrões de acordo com a definição de padrões organizacionais e de processo estabelecida neste trabalho;
3. Associação entre os padrões organizacionais e de processo e as disciplinas definidas no Scrum.
4. Integração das práticas propostas pelos padrões com o processo Scrum, por meio dos estereótipos do meta-modelo SPEM.

Por meio dessas etapas, foi possível integrar alguns padrões organizacionais e de processo ao Scrum, além de propor uma extensão desse método para tratar alguns de seus pontos fracos como as atividades de teste e integração, destacados por Abrahamsson et al. (2002) e outros como a utilização de técnicas para extração de requisitos, criação da arquitetura do produto, divisão das tarefas e distribuição geográfica, observados durante o trabalho aqui descrito.

O Scrum estendido foi aplicado em um estudo piloto com duas organizações fictícias e, em ambas, os sistemas foram desenvolvidos completamente e entregues no prazo. Além disso, os membros das organizações mencionaram que o Scrum estendido é mais completo, mais fácil de seguir, facilita o direcionamento das atividades e divisão das tarefas entre os membros da equipe, sem afetar a flexibilidade original do Scrum. Também foi mencionado que, com relação à modelagem com SPEM, o Scrum estendido está bem definido, bem visível e é de fácil manutenção, como mostrado na Seção 5.5.

Além de facilitar a integração dos padrões ao Scrum e, conseqüentemente a sua extensão, a modelagem realizada facilita a visualização e entendimento do processo. Os relacionamentos

entre todos os elementos do processo são representados pelos diagramas de classe, enquanto a ordem em que as atividades devem ocorrer é representada pelos diagramas de atividades. Assim, com esses dois tipos de diagramas e os estereótipos SPEM é possível representar de forma adequada o processo Scrum estendido com as novas práticas propostas pelos padrões.

Com a aplicação dessas etapas para a criação do Scrum estendido, percebeu-se a possibilidade de criar diretrizes para a integração de padrões organizacionais e de processo a outros métodos ágeis, como mostrado na Seção 4.7.

6.3. Contribuições deste Trabalho

Além da criação de uma extensão do método ágil Scrum, outras contribuições importantes deste trabalho são: a identificação de alguns padrões organizacionais e de processo existentes que estão relacionados com as práticas e papéis definidos no Scrum, a racionalização da utilização dos padrões organizacionais e de processo junto ao Scrum, a modelagem do processo Scrum original e a criação de diretrizes para possibilitar a integração de padrões organizacionais e de processo a outros métodos ágeis.

A extensão do processo Scrum contém práticas de sucesso para tratar alguns pontos fracos identificados nesse método. Essas práticas propostas pelos padrões foram integradas ao Scrum utilizando o meta-modelo SPEM, de forma que a flexibilidade original do Scrum fosse preservada. Assim, essa extensão é uma versão mais completa do Scrum, modelada com SPEM, cuja aplicação em um estudo piloto de avaliação com duas organizações indicou a conveniência de utilizá-lo.

Essa extensão foi criada por meio de uma forma ordenada para racionalizar a utilização dos padrões organizacionais e de processo junto ao Scrum. Essa forma ordenada, que é composta de quatro etapas, possibilita a integração efetiva de qualquer padrão organizacional e de processo ao Scrum, por meio do meta-modelo SPEM. Assim, outros padrões podem ser integrados ao Scrum para satisfazer as necessidades de um projeto ou organização.

Um subproduto gerado em uma das etapas de racionalização é a modelagem do processo Scrum original com SPEM. As práticas do Scrum estão organizadas de forma coerente e definem um processo de software e a sua modelagem facilita o entendimento, visualização e extensão desse método.

A aplicação das etapas de racionalização para a criação do Scrum estendido possibilitou a criação de diretrizes para integração de padrões organizacionais e de processo a outros métodos ágeis.

Além disso, a investigação do método ágil Scrum e de várias linguagens de padrões existentes possibilitou a identificação de alguns padrões organizacionais e de processo que estão diretamente relacionados com as práticas e papéis definidos no Scrum. Isso mostra que o Scrum contém várias práticas de sucesso, o que aumenta sua confiabilidade e de sua extensão aqui proposta.

6.4. Limitação deste Trabalho

Um ponto não abordado pela extensão do Scrum proposta neste trabalho, e que também não é tratado pelo Scrum original, é o uso de práticas de engenharia de software. O Scrum estendido fornece práticas para gerenciar o desenvolvimento do software, porém não fornece práticas específicas de engenharia. As fases de análise, projeto, codificação e teste ocorrem durante uma iteração e são controladas pelo Scrum estendido, porém fica a cargo da organização a escolha das práticas específicas de engenharia de software mais adequadas.

6.5. Trabalhos Futuros

A seguir, são apresentadas algumas sugestões para trabalhos futuros:

- Realizar estudos de caso para comparação entre o Scrum original e o Scrum estendido para verificar os pontos positivos e negativos das duas versões.
- Aplicar o Scrum estendido em organizações não fictícias, para possibilitar uma avaliação mais efetiva das novas práticas integradas.
- Verificar se é conveniente aplicar a outros métodos ágeis as diretrizes criadas para a integração de padrões organizacionais e de processo a métodos ágeis.
- Identificar padrões organizacionais e de processo relacionados a outros métodos ágeis existentes.
- Verificar se existem outros padrões organizacionais e de processo que podem ser integrados com o Scrum para melhorá-lo sem afetar sua flexibilidade.

Referências Bibliográficas

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: 18 jan. 2006.

Acuña, S.; and Ferré, X. *Software Process Modeling*. In Proceedings of The 5th World Multiconference on Systemics, Cibernetics and Informatics (SCI 2001). Orlando, Florida,USA. 1-6, July 2001.

Alexander, C. *A Pattern Language*. New York: Oxford University Press, 1977.

Alexander, C. *The Timeless Way of Building*. New York: Oxford University Press, 1979.

Ambler, S. W. Modelagem Ágil: Modelagem Ágil: Práticas Eficazes para a Programação eXtrema e o Processo Unificado. Bookman, 1 ed., 2004.

Ambler, S. W. *Process Patterns: Building Large-Scale Systems Using Object Technology*. New York: Cambridge University Press, 1998.

Appleton, B. *Patterns and Software: Essential Concepts and Terminology*. 1997. Disponível em:<http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html>. Acesso em: 30 jan. 2006.

Aye, S. S.; Zhou, Y.; Ochimizu, K. *Process Model Combining the Artifact Centered Process with Communication Path*. In: 5th International Workshop on Software Process Simulation and Modeling, 2004.

Baheti, P.; Gehringer, E.; Stotts, D. *Exploring the Efficacy of Distributed Pair Programming*, In XP/Agile Universe, 2002a, p. 208-220.

Baheti, P.; Williams, L.; Gehringer, E.; Stotts, D. *Exploring Pair Programming in Distributed Object-Oriented Team Projects*. In Proceedings OOPSLA Educator's Syposium, Seattle, WA, 2002b.

Beck, K. *Embracing Change with Extreme Programming*. IEEE Computer, v.32, n.10, p. 70-77, 1999a.

Beck, K. *Extreme Programming Explained: Embrace Change*. 1. ed. Addison-Wesley, 1999b.

Beck, K.; Beedle, M.; Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R.; Mellor, S.; Schwaber, K.; Sutherland, J.; Thomas, D. *Manifesto for Agile Software Development*. 2001. Disponível em: <http://www.agilemanifesto.org>. Acesso em: 15 jan. 2006.

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

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

Boehm, B. W. *A Spiral Model for Software Development and Enhancement*. IEEE Computer, v. 21, n.5, p.61-72, 1988.

Braga, R. T. V. Um Processo para a Construção e Instanciação de Frameworks baseado em uma Linguagem de Padrões para um Domínio Específico. Tese de Doutorado, ICMC/USP, São Carlos-SP, 2003.

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

Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. *Pattern-Oriented Software Architecture: A System of Patterns*, Volume 1. Wiley, 1996.

Coad, P.; Lefebvre, E.; De Luca, J. *Java modeling in color with UML: Enterprise Components and Process*. Upper Saddle River: Prentice Hall, 1999.

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

Cockburn, A.; Highsmith, J. *Agile Software Development: The People Factor*. Computer, v. 34, n.11, p. 131-133, 2001.

Cockburn, A. *Agile Software Developmen*t. Boston: Addison Wesley, 2002.

Cohn, M.; Ford, D. Introducing An Agile Process To An Organization. Computer, v. 36, n. 6, p.74-78, 2003

Conradi, R.; Fernström, C.; Fuggetta, A. *Concepts for Evolving Software Process, in Software Process Modeling and Technology*, Research Studies Press, 1994.

Coplien, J. O. *A Generative Development-Process Pattern Language*. In: Coplien, J.; Schmidt, D. Pattern Languages of Program Design. USA: Addison-Wesley, 1995, p. 183-237.

Coplien, J. O.; Schmidt, D. C. *Pattern Languages of Program Design*. Reading – MA, USA: Addison-Wesley, 1995.

Coplien, J. O. *Software Design Patterns: Common Questions and Answers*. In: Rising, L. The Patterns Handbook: Techniques, Strategies, and Applications. Cambridge University Press, 1998. p. 311-320.

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

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, p. 407-416.

Costa Filho, E. G.; Penteado, R. A. D.; Braga, R. T. V.; 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 2005. Agosto, 2005, Campos do Jordão, Brasil.

Cunningham, W. *Episodes: A Pattern Language of Competitive Development*. In: Vlissides, J.; Coplien, J.; Kerth, N. Pattern Languages of Program Design 2. Addison-Wesley, 1996, p. 371-388.

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

Curtis, B.; Kellner, M.; Over, J. *Process Modeling*. Communications of the ACM, p.75-90, v.35, n.9, Setembro, 1992.

CVS - *Concurrent Versions System*, 2006. Disponível em: <http://www.nongnu.org/cvs/>. Acesso em: 15 fev. 2006.

Delano, D.; Rising, L. *Patterns for System Testing*. In: Martin, R.; Riehle, D.; Buschmann F. Pattern Languages of Program Design 3. Addison-Wesley, 1998, p. 503-525.

Finkelstein, A.; Kramer, J.; Nuseibeh, B. *Software Process Modelling and Technology*. Research Studies, 1994.

Fowler, M. *The New Methodology*. 2003. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>. Acesso em: 25 jan. 2006.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

Harrison, Neil B. *Organizational Patterns for Teams*. In: Vlissides, J.; Coplien, J.; Kerth, N. Pattern Languages of Program Design 2. Addison-Wesley, 1996, p. 345-352.

Heimann, P.; Joeris, G.; Krapp, C.; Westfechtel, B., *DYNAMITE: Dynamic Task Nets for Software Process Management*, Proceedings of ICSE 18, p. 331-341, Berlin, Março de 1996.

Hibernate. *Relational Persistence for Java and .NET*. 2006. Disponível em: <http://www.hibernate.org/>. Acesso em: 19 jan 2006.

Highsmith, J. *Adaptive Software Development*. Dorset House, 2000.

Highsmith, J.; Cockburn, A. *Agile Software Development: The Business of Innovation*. Computer, v. 34, n. 9, p. 120-122, 2001.

Houaiss, A. Dicionário Eletrônico Houaiss da Língua Portuguesa. Objetiva, 2002.

Institute of Electrical And Electronics Engineers - IEEE. *IEEE Std 610.12-1990 - Standard glossary of software engineering terminology*. Piscataway: IEEE, 1990.

iReport. *iReport Designer for JasperReports*. 2006. Disponível em: <http://ireport.sourceforge.net/>. Acesso em: 19 jan. 2006.

Jaccheri, M; Picco, G.; Lago, P. *Eliciting software process models with the E3 language*. In ACM Transactions on Software Engineering and Methodology (TOSEM), v.7, n.4, Outubro, 1998.

Jacobson, I., Booch, G., And Rumbaugh, J. *The Unified Software Development Process*. Reading, MA.: Addison-Wesley, 1999.

JasperReports. *JasperReports*. 2006. Disponível em: <http://jasperreports.sourceforge.net/>. Acesso em: 19 jan. 2006.

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.

Kerth, N. *Project Retrospectives: A Handbook for Team Reviews*. Dorset House, 2001.

Kircher, M.; Jain, P.; Corsaro, A.; Levine, D. *Distributed eXtreme Programming*. In Proceedings of XP-2001, Villasimius, Itália, 2001. Disponível em: <http:// www.agilealliance.com/articles/1kirchermichaelprasha/file>. Acesso em: 12 fev. 2006.

Kruchten, P. *Agility with the RUP*. Cutter IT Journal, Arlington, v.14, n.12, p. 27- 33, Dec. 2001.

Kruchten, P. The *Rational Unified Process: An Introduction*. 3. ed. Adison-Wesley. 2003.

Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 2 ed. Prentice Hall, 2002.

Larman, C. *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley, 2003.

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.

Manns, L.; Rising L. *Fearless Change: Patterns for Introducing New Ideas*. Addison-Wesley, 2004.

Mar, K; Schwaber, K. *Scrum with XP*. Prentice Hall, 2002. Disponível em: <http://www.informit.com/articles/article.asp?p=26057&rl=1>. Acesso em: 26 jan 2006.

Microsoft. Microsoft.NET. 2006a. Disponível em: <http://www.microsoft.com/net/default.mspx>. Acesso em: 19 jan. 2006.

Microsoft. Células Acadêmicas. 2005. Disponível em <http://www.seminargroup.com.br/ celula/>. Acesso em: 3 abr. 2006c.

Microsoft. *Microsoft SQL Server*. 2006b. Disponível em: <http://www.microsoft.com/ sql/default.mspx>. Acesso em: 19 jan. 2006

MSDN. *Microsoft Visual C# Developer Center*. 2006a. Disponível em: <http://msdn.microsoft.com/vcsharp/>. Acesso em: 19 jan. 2006

MSDN. *Microsoft Visual Studio Developer Center*. 2006b. Disponível em: <http://msdn.microsoft.com/vstudio/>. Acesso em: 19 jan. 2006

OMG - Object Management Group. *Meta-Object Facility (MOF) Specification v. 1.3*. Document ad/99-06-05, 1999.

OMG - Object Management Group. *UML Specification, 1.4*. 2001.

OMG - Object Management Group. *Software Process Engineering Metamodel Specification*, January, 2005.

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

Pressman, R. S. - *Software Engineering - A Practitioner's Approach*, 6th edition, McGraw-Hill, 2005.

Rational Software Corporation. *Rational Unified Process – RUP*. Versão 2002.05.00.

Rising, L.; Janoff, N. S. *The Scrum Software Development Process for Small Teams*. IEEE Software, v. 17, n. 4 p. 26-32, 2000.

Rising, L.; Derby, E. *Singing the Songs of Project Experience: Patterns and Retrospectives*. Cutter IT Journal, v. 16, n. 9, p. 27-31, Setembro, 2003.

Royce, W. *Managing the Development of Large Software Systems*. In: Proceedings of IEEE WESCON, 1970. p. 1-9.

Schümmer, T.; Schümmer, J. *Support for Distributed Teams in Extreme Programming*. In Succi, G., Marchesi, M., eds., p. 355-377, Addison Wesley, 2001.

Schwaber, K. *Scrum Development Process*. In Proceedings of 1º Workshop on Business Object Design and Implementation (OOPSLA'95), Austin, USA, October 1995, p. 94-116.

Schwaber, K.; Beedle, M. *Agile Software Development with Scrum*. Prentice-Hall, 2002.

Schwaber, K. *Agile Project Management with Scrum*. Microsoft Press, 2004

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

Sommerville, I. Engenharia de Software. 5. ed. Addison-Wesley, 2003.

Stotts, D.; Williams, L.; Nagappan, N.; Baheti, P.; Jen, D. *Virtual Teaming: Experiments and Experiences with Distributed Pair Programming*. In XP/Agile Universe 2003, pages 129-141. Springer, 2003.

Sun. *Java Technology*. 2006a. Disponível em: <http://java.sun.com/>. Acesso em: 20 jan. 2006.

Sun. *JDBC Technology*. 2006b. Disponível em: <http://java.sun.com/products/jdbc/>. Acesso em: 19 jan. 2006.

Sun. *JavaServer Pages Technology*. 2006c. Disponível em: <http://java.sun.com/products/jsp/>. Acesso em: 19 jan 2006.

Sutherland, J. Scrum: *Another Way to Think about Scaling a Project*. 2003. Disponível em <http://jeffsutherland.org/scrum/2003_03_01_archive.html>. Acesso em: 06 fev. 2006.

Taft, D. *Microsoft Lauds Scrum Method for Software Projects*, Novembro, 2005. Disponível em <http://www.eweek.com/article2/0,1895,1885883,00.asp>. Acesso em: 15 mar. 2006.

Takeuchi, H.; I. Nonaka, The New *New Product Development Game*. Harvard Business Review (Janeiro 1986), p. 137-146, 1986.

Tuffs, D.; Stapleton, J.; West, D.; Eason, Z. *Inter-operability of DSDM with the Rational Unified Process*. 1999. Disponível em: <http://www.agilealliance.com/articles/tuffsdavidstapletonje/file>. Acesso em: 26 jan. 2006.

Apêndice A - Questionário

Questionário – Estudo Piloto de Avaliação do Scrum Estendido

1) Scrum Estendido:

a. Quais as principais vantagens que as práticas propostas pelos padrões integrados ao Scrum proporcionaram para desenvolvimento do sistema? Justifique sua resposta.
b. As práticas inseridas complementam o Scrum? Justifique sua resposta.
c. Faça uma comparação entre Scrum original e Scrum estendido. Você acredita que as práticas inseridas são necessárias? Justifique sua resposta.
d. Quais das práticas propostas pelos padrões integrados ao Scrum foram mais relevantes para desenvolvimento? Justifique sua resposta.

2) Distribuição Geográfica:

a. Como foi a comunicação entre os membros da equipe?
    i. ( ) Muito Boa
    ii. ( ) Boa
    iii. ( ) Fraca
b. Como foi colaboração entre os membros da equipe?
    i. ( ) Muito Boa
    ii. ( ) Boa
    iii. ( ) Fraca

3) Modelagem do Scrum Estendido (SPEM):

a. Facilidade de Compreensão: Com que facilidade se pode compreender o processo por meio da modelagem realizada com SPEM?
    i. ( ) Mal Definido: As atividades do Scrum Estendido estão mal definidas. Não se tem idéia da responsabilidade de cada papel e nem da seqüência em que as atividades devem ocorrer.

ii. ( ) Razoavelmente Definido: As atividades do Scrum Estendido estão razoavelmente definidas e explicadas. Não se tem idéia detalhada da responsabilidade de cada papel e da seqüência em que as atividades devem ocorrer.

iii. ( ) Bem definido: As atividades do Scrum Estendido estão bem definidas e explicadas. Tem-se idéia clara das responsabilidades de cada papel e da seqüência em que as atividades devem ocorrer.

b. Visibilidade: Analise o Scrum Estendido quanto aos artefatos gerados, consumidos e modificados em cada atividade.

i. ( ) Fracamente Visível: As atividades do Scrum Estendido não definem de forma clara os artefatos gerados e consumidos. Não se tem idéia clara dos artefatos gerados, modificados e consumidos em cada atividade e nem do seu responsável.

ii. ( ) Visível: As atividades do Scrum Estendido definem de forma razoavelmente clara os artefatos gerados e consumidos e seus responsáveis. Tem-se idéia não muito clara dos artefatos gerados, modificados e consumidos em cada atividade e nem do seu responsável.

iii. ( ) Bem Visível: As atividades do Scrum Estendido definem de forma clara os artefatos gerados e consumidos e seus responsáveis. Tem-se idéia clara dos artefatos gerados, modificados e consumidos em cada atividade e nem do seu responsável.

c. Facilidade de Manutenção: Seria fácil realizar alterações no processo para integrar novas atividades, artefatos e papéis?

i. ( ) Difícil de Alterar: Devido a sua definição inadequada seria difícil alterá-lo para atender às necessidades particulares de um projeto ou organização.

ii. ( ) Fácil de Alterar: Devido ao processo estar bem definido seria fácil alterá-lo para atender às necessidades particulares de um projeto ou organização.

4) Retrospectiva:

a. Quais problemas ocorreram no desenvolvimento do sistema?

b. O que poderia ser melhorado no processo?

c. Alguma prática não fornecida deveria ser integrada? Justifique sua resposta.