

UNIVERSIDADE LUTERANA DO BRASIL

ULBRA – *CAMPUS* GUAÍBA

CURSO DE SISTEMAS DE INFORMAÇÃO



**PROPOSTA DE DESENVOLVIMENTO DE UMA  
METODOLOGIA DE GESTÃO DE PROJETOS  
BASEADA EM MÉTODOS ÁGEIS PARA A  
EMPRESA JÚNIOR**

GUILHERME SCHIRMER DA COSTA

ANDERSON RICARDO YANZER CABRAL  
**Professor Orientador**

Guaíba, julho de 2009.

## **DADOS DE IDENTIFICAÇÃO**

### **1. DADOS DO ALUNO**

Nome: Guilherme Schirmer da Costa  
Endereço: Rua Farroupilha, 52, Florida, Guaíba  
E-mail: guilhermescosta@gmail.com  
Fones: (51)34011933 / 98526823

### **2. DADOS DO PROFESSOR ORIENTADOR**

Nome: Anderson Ricardo Yanzer Cabral  
E-mail: yanzer@guaiba.ulbra.tche.br

## SUMÁRIO

<b>1</b>	<b>DEFINIÇÃO DO TEMA.....</b>	<b>4</b>
1.1	Tema.....	4
1.2	Delimitação do Tema.....	4
<b>2</b>	<b>MOTIVAÇÃO.....</b>	<b>4</b>
<b>3</b>	<b>OBJETIVOS.....</b>	<b>4</b>
<b>4</b>	<b>HIPÓTESES DE SOLUÇÃO .....</b>	<b>5</b>
<b>5</b>	<b>FUNDAMENTAÇÃO TEÓRICA .....</b>	<b>5</b>
5.1	Métodos Ágeis.....	6
5.2	Sobre os métodos ágeis avaliados.....	6
5.2.1	XP.....	6
5.2.2	Scrum.....	7
5.2.3	FDD.....	7
5.2.4	ASD.....	9
5.2.5	DSDM.....	10
5.2.6	Crystal.....	10
5.3	Comparação.....	11
<b>6</b>	<b>METODOLOGIA ADOTADA.....</b>	<b>18</b>
<b>7</b>	<b>SOLUÇÃO PROPOSTA.....</b>	<b>18</b>
7.1	Scrum + XP.....	20
7.2	Etapas de Desenvolvimento.....	20
7.3	Papéis.....	21
7.4	Artefatos.....	21
7.5	Sprint.....	22
7.6	Práticas de Desenvolvimento.....	22
<b>8</b>	<b>CONCLUSÃO E TRABALHO DE CONCLUSÃO DE CURSO – II.....</b>	<b>23</b>
<b>9</b>	<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>24</b>

## **1 DEFINIÇÃO DO TEMA**

### **1.1 Tema**

Estudo para apresentar uma metodologia de gestão de projetos baseada em métodos ágeis que tenha aplicação na Empresa Júnior da ULBRA Guaíba.

### **1.2 Delimitação do Tema**

O trabalho focará no estudo de métodos ágeis voltados a gestão de projetos, especialmente projetos de software aplicados a projetos no formato utilizados pela Empresa Júnior da ULBRA Guaíba.

## **2 MOTIVAÇÃO**

Atualmente, a Empresa Júnior do campus ULBRA Guaíba, denominada ADMI, atua de forma a realizar projetos para a comunidade. Muitos desses projetos são softwares desenvolvidos pelos alunos do curso de Sistemas de Informação. No entanto, a metodologia utilizada para o desenvolvimento desses projetos muitas vezes não é a mais adequada devido à falta de flexibilidade. De fato, equipes menores deveriam ter como ponto forte, a agilidade de lidar com o planejamento e execução dos projetos. Esse ponto forte é ignorado pela metodologia atual, que é baseada nos preceitos do Project Management Institute, que torna um pouco burocratizado o processo de desenvolvimento de *software*.

A utilização de métodos ágeis para a gestão de projetos tem como foco o desenvolvimento por meio de iterações de curto prazo. Essas iterações são planejadas com a presença dos desenvolvedores e cliente, fazendo com que o mesmo tenha uma presença fundamental no desenvolvimento, informando o foco do negócio.

## **3 OBJETIVOS**

O objetivo principal do trabalho será apresentar uma metodologia de gestão de projetos utilizando métodos ágeis.

Os objetivos específicos são:

1. Apresentar os métodos ágeis como solução viável para gestão de projetos.
2. Apresentar uma revisão bibliográfica da utilização dos métodos ágeis para o desenvolvimento de *software*.
3. Apresentar uma metodologia para a gestão de projetos da Empresa Júnior.
4. Realizar um estudo de caso da metodologia escolhida em um projeto da Empresa Júnior.
5. Realizar uma comparação entre as metodologias atuais e proposta após o término do projeto.

#### **4 HIPÓTESES DE SOLUÇÃO**

- Hipótese I: Propor uma metodologia de gestão de projetos baseada em métodos ágeis.
- Hipótese II: Demonstrar algumas ferramentas de auxílio ao desenvolvimento de projetos baseada em métodos ágeis.

#### **5 FUNDAMENTAÇÃO TEÓRICA**

Para a elaboração do trabalho serão estudados métodos ágeis (SCHWABER 2004), planejamento e estimativas de projetos ágeis (COHN 2005) e casos de estudos de aplicação métodos ágeis na gestão de projetos (KNIBERG 2008).

Após estudo dos tópicos citados, a pesquisa irá definir os parâmetros para a criação da metodologia buscada. Os estudos de caso serão de grande importância, pois darão uma noção real do que se deve atentar ao criar esse modelo específico para a Empresa Júnior.

Ao final do período de pesquisas, o modelo deverá entrar em testes através de experimentos práticos que conduzirão a um momento de avaliação quanto ao que se esperava e o que foi alcançado.

## 5.1 Métodos Ágeis

Métodos (ou metodologias) ágeis podem ser definidos como aqueles que seguem os ideais do chamado “Manifesto Ágil” (BECK et al., 2001). Esse manifesto foi formalizado em 2001 por alguns dos criadores de algumas das metodologias ágeis. Existe quatro máximas que são consideradas o centro de qualquer metodologia ágil. São elas:

- Indivíduos e interações são mais importantes que processos e ferramentas
- *Software* funcionando é mais importante que documentação completa
- Colaboração com cliente é mais importante que negociação de contratos
- Adaptar-se a mudanças é mais importante que seguir um plano inicial

Embora os principais métodos ágeis tenham sido criados anteriormente ao manifesto, sua utilização passou a ser maior após a criação do mesmo.

Segundo Cohn (COHN, 2005), uma das principais diferenças entre métodos ágeis em relação aos métodos tradicionais é a produção de documentação enxuta para cada tarefa. Cohn ainda fala que as iterações constantes levam a uma aproximação maior sobre o negócio e que os *feedbacks* constantes fazem o cliente e a equipe encontrarem um consenso sobre o que será desenvolvido.

## 5.2 Sobre os métodos ágeis avaliados

Serão demonstradas as características dos métodos ágeis avaliados antes da sua avaliação com o 4-DAT.

### 5.2.1XP

Segundo Beck (BECK, 2000), *Extreme Programming* (XP) é uma metodologia de desenvolvimento leve para times de tamanho pequeno a médio, que desenvolvem software em face de requisitos vagos que se modificam rapidamente. Existem seis fases dentro do processo de XP.

Exploração: Nessa fase as tarefas são divididas entre os participantes do projeto.

Planejamento: Onde são adquiridos os requisitos do projeto.

Iterações: Processo de produção do *software* em pequenas partes para ser apresentado ao cliente.

Produção: Processo de entrega de um produto funcional e já testado ao cliente.

Manutenção: É um ciclo de aperfeiçoamento do produto que visa eliminar falhas não previstas no projeto inicial.

Morte: O fim do projeto.

### 5.2.2 Scrum

Schwaber define Scrum como:

“...um *framework* e um conjunto de práticas ágeis que mantém tudo visível”.

(SCHWABER 2004)

Scrum é definido como um *framework* e não uma metodologia porque representa um conjunto de idéias e práticas que podem ser seguidas ou adaptadas.

“O melhor e o pior do *Scrum* é que você é forçado a adaptar o processo para sua situação específica.”

(KNIBERG 2008)

Schwaber defende que através de ciclos de curto prazo (*Sprints*) é possível desenvolver um produto dentro das necessidades do cliente, gastando menos tempo em manutenção após a entrega do produto final. Dentro desses ciclos, existem várias práticas diárias que podem ser seguidas para maximizar a produção. Uma das práticas comuns é o uso de *backlogs*. *Product Backlogs* basicamente uma lista de requisitos, histórias, coisas que o cliente deseja, descritas utilizando a terminologia do cliente (KNIBERG 2008). Já os *Sprints Backlogs* é um conjunto de histórias inclusas no *Sprint*.

### 5.2.3 FDD

*Feature Driven Development* é um processo de desenvolvimento de *software* para produção freqüente e tangível (PALMER 2002). Ao contrário de outros métodos ágeis, FDD pode ser usada praticamente da mesma forma por equipes pequenas, de até 10 integrantes, como para equipes gigantescas de mais de 50 ou 100 integrantes. A FDD possui 5 processos demonstrados na figura abaixo.

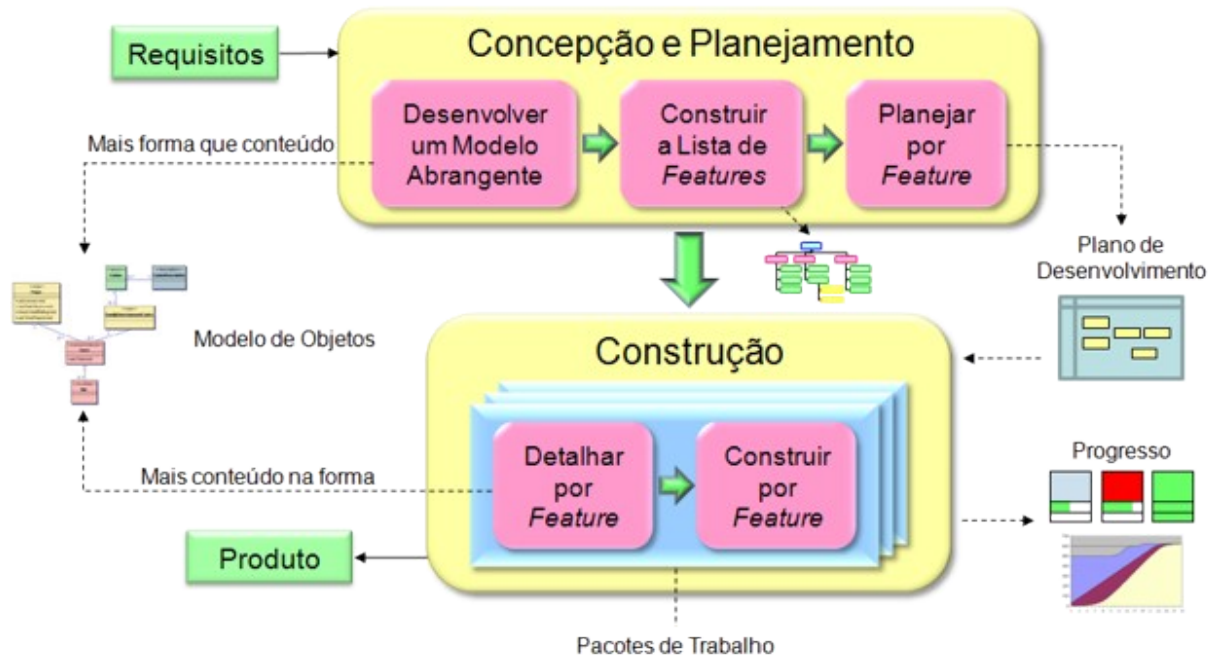


Figura 1. Processos do FDD (RETAMAL, 2009).

**Desenvolvimento de Modelo Abrangente:** Processo de arquitetar o projeto como um todo. Nessa fase são definidos o escopo e a equipe. Também é realizado um estudo sobre o domínio do projeto pela equipe de modelagem. Para isso, pequenas equipes são formadas (com não mais que 3 participantes) e desenvolvem seus próprios modelos que são discutidos pelo grupo geral. O modelo com melhor aceitação será refinado pela equipe e, se necessário, avaliado pelo cliente. O processo se encerra quando o modelo do projeto é aceito por todas as partes. Nessa fase participam gerentes de projeto, arquitetos de software e equipes de modelagem.

**Listagem de funcionalidades:** É a fase de levantamento de requisitos. Estudam-se todas as funcionalidades que foram pedidas pelo cliente e criam-se soluções viáveis para as mesmas

**Planejamento por funcionalidade:** São definidas as equipes de desenvolvimento (raramente com mais de 10 participantes por time) e as funcionalidades a serem construídas.

**Projetar por funcionalidade:** Nesse processo são definidas quais funcionalidades cada equipe construirá. Cada equipe deve criar o diagrama de seqüência de suas funcionalidades e repassar ao programador chefe do projeto. Nessa fase o programador chefe deve criar um repositório das funcionalidades (geralmente diretórios em um servidor de arquivos) e também avaliar os projetos das

funcionalidades. Após o término do projeto das funcionalidades, as mesmas são avaliadas pelo grupo geral de desenvolvimento. Se todas as funcionalidades são aprovadas, o projeto passar para o próximo processo, caso contrário será refeito até o design do projeto agradar a equipe.

Produção por funcionalidade: As equipes devem implementar suas funcionalidades conforme o projetaram. É importante que haja um consenso em relação aos comentários de objetos e métodos, já que as funcionalidades são avaliadas por outras equipes e pelo programador líder. Testes unitários devem ser realizados para confirmar o sucesso, ou não, de cada funcionalidade criada. O programador líder é o responsável pela integração das funcionalidades, sendo também responsável pelas versões de avaliação do cliente.

#### 5.2.4 ASD

Desenvolvido por Jim Highsmith após algumas experiências com projetos RAD (*RADical Application Development*, não confundir com *Rapid Application Development*), o *Adaptative Software Development* tem foco na produção de softwares complexos. O método encoraja a produção incremental, o desenvolvimento iterativo e a prototipagem constante. O ciclo de ações no ASD é constituído de três fases:

Especulação: É a fase de planejamento, onde são realizadas reuniões e é definido o que deverá ser produzido.

Colaboração: Fase de produção. As equipes devem colaborar para a produção do produto proposto.

Aprendizado: Fase de avaliação. A equipe é reunida para discutir o que foi aprendido de bom e de ruim durante a iteração.

Segundo Highsmith (HIGHSMITH 2000), o ciclo de vida é baseado em 6 propriedades adaptativas:

- Orientação a missões: Cada atividade possui uma missão.
- Baseado em componentes: Construção do sistema por partes.
- Iterativo: Refazer um componente até ficar realmente bom.

- Limitação de prazo (*Time-Boxed*): Força os participantes a tomarem decisões importantes nas fases iniciais do projeto
- Tolerância a mudanças: A avaliação constante incentiva os participantes a realizar mudanças ao invés de tentar manter o que parece errado ou inadequado.
- Orientado a riscos: Itens que parecem mais arriscados são produzidos antes dos menos complexos.

#### 5.2.5 DSDM

Semelhante ao *Scrum*, DSDM ou *Dynamic System Development Method*, é um *framework* altamente adaptável. O *framework* DSDM pode ser implementado tanto para métodos tanto ágeis quanto para tradicionais (Voigt 2004). Existem 9 princípios que são imprescindíveis para a implementação do DSDM:

- Envolvimento dos usuários
- Os times devem ter liberdade para fazer suas próprias decisões
- Foco na entrega freqüente.
- Adaptação ao negócio é o critério para as entregas
- Desenvolvimento iterativo e incremental
- Mudanças são reversíveis utilizando pequenas iterações
- Requisitos são acompanhados em alto nível
- Testes integrados ao ciclo de vida

#### 5.2.6 Crystal

É um conjunto de metodologias que devem ser utilizadas conforme quatro dimensões do projeto, o tamanho da equipe, recursos disponíveis, localização e prazo, sendo a metodologia *Crystal Clear* voltada para projetos menores, em times de no máximo 10 componentes, e a *Crystal Orange* para projetos de maior escala.

Tanto para a metodologia *Crystal Clear* quanto a *Crystal Orange* é sugerido a mesma política de padronização (Cockburn 2002):

- Entregas incrementais de forma periódica
- Progresso medido por *milestones* baseadas em entregas de *software* e grandes decisões ao invés de documentação escrita
- Envolvimento direto do usuário
- Testes de regressão automatizados das funcionalidades
- Duas visualizações dos usuários por lançamento
- Reuniões para ajustar o produto e metodologia no início e no meio de cada incremento

### **5.3 Comparação**

O *Framework* 4-DAT analisa os métodos ágeis através de 4 dimensões: Escopo, Características, Valores ágeis e Processos. Através das mesmas se pode avaliar os principais métodos ágeis em relação ao ambiente onde se deseja implementá-los.

## Dimensão 1: Escopo

Tamanho do Projeto	O método possui suporte específico para projetos de pequeno, médio ou grande porte?
Tamanho da Equipe	O método especifica suporte para equipes pequenas ou grandes (times únicos ou múltiplos)?
Tipo de Desenvolvimento	Qual o estilo de desenvolvimento adotado (Iterativo ou rápido)?
Tipo de Código	O método especifica um estilo de código (simples ou complexo)?
Ambiente Tecnológico	Qual ambiente tecnológico (ferramentas, compiladores) o método especifica?
Ambiente Físico	Qual ambiente físico ( <i>co-located</i> ou distribuído) é especificado pelo método?
Cultura de Negócio	Qual cultura de negócio (colaborativa, competitiva ou não-colaborativa) o método especifica?
Mecanismo de Abstração	O método especifica um mecanismo de abstração (orientado à objetos, orientado à agentes)?

## Dimensão 2: Características

Flexibilidade	O método acomoda mudanças esperadas ou inesperadas?
Velocidade	O método produz resultados rapidamente
<i>Leanness</i>	O método busca o menor tempo de duração, uso econômico, instrumentos de simplicidade e qualidade para a produção?
Aprendizado	O método pode ser atualizado conforme conhecimento prévio e experiência para criar um ambiente de aprendizado?
Responsividade	O método exibe sensibilidade?

## Dimensão 3: Valores Ágeis

Indivíduos e Iterações sobre Processos e Ferramentas	Que prática valoriza pessoas e iterações sobre processos e ferramentas?
<i>Software</i> Funcional sobre Documentação Compreensiva	Que prática valoriza o <i>software</i> funcional sobre documentação compreensiva?
Colaboração do Cliente sobre Renegociação de Contratos	Que prática valoriza a colaboração do cliente sobre renegociação de contratos?
Responder a Mudanças sobre Seguir um Planejamento	Que prática valoriza responder a mudanças sobre seguir um planejamento?
Manter Processo Ágil	Que prática ajuda a manter o processo ágil?
Manter Processo Rentável	Que prática ajuda a manter o processo rentável?
Dimensão 4: Processos	
Processo de Desenvolvimento	Que práticas cobrem o principal processo de ciclo de vida e testes ( <i>Quality Assurance</i> )?
Processo de Gestão de Projeto	Que práticas cobrem a gestão geral do projeto?
Configuração de <i>software</i> de controle de processos / Suporte processos	Que práticas cobrem os processos que possibilitam a gestão de configuração?
Processo de Gestão de Processos	Que práticas cobre o processo que é requerido para a gestão de processos?

Tabela 1. As 4 dimensões do 4-DAT (QUMER et al., 2008)

No caso da Empresa Júnior, serão descartadas as características dos métodos ágeis (já detalhados no capítulo anterior) na tabela 2 e os valores ágeis referentes à tabela 3, já que os mesmos são irrelevantes para avaliar a utilidade dos métodos estudados. Para melhor comparação com o ambiente da Empresa Júnior, foi criada uma coluna adicional nas tabelas demonstrando as características atuais

Critério	Empresa Júnior	XP	Scrum	FDD	ASD	DSDM	Crystal
Tamanho do escopo do projeto	Pequeno	Pequeno, médio	Pequeno, médio e escalável para grande	Pequeno, médio e grande (projetos e aplicações de negócio)	Grande e complexos	Pequenos ou grandes projetos (aplicações de negócio)	Pequeno, médio
Tamanho da equipe	<10	<10	<10 e múltiplos times	Sem limites – escalável de pequenos a grandes times	Não mencionado	Mínimo 2 e máximo 6 (múltiplos times)	Time único no Crystal Clear com no máximo 6 pessoas em um time. Múltiplos times com 40 pessoas no máximo no Crystal Orange e 80 pessoas no máximo no Crystal Red.
Estilo de desenvolvimento	Não mencionado	Iterativo, rápido	Iterativo, rápido	Design iterativo e construtivo	Desenvolvimento iterativo e rápido – desenvolvimento distribuído	Desenvolvimento rápido, iterativo e cooperativo	Iterativo e rápido
Estilo do código	Não mencionado	Limpo e simples	Não especificado	Não especificado	Não mencionado	Não mencionado	Não mencionado
Ambiente tecnológico	Não mencionado	Feedback rápido é requerido	Não especificado	Não especificado	Não mencionado	Não mencionado	Não mencionado
Ambiente físico	Times co-locados	Times co-locados e times distribuídos (iteração limitada)	Não especificado	Não especificado	Times co-locados e times distribuídos	Não mencionado	Times co-locados – sem suporte para times distribuídos
Cultura de negócios	Não especificado	Colaborativa e cooperativa	Não especificado	Não especificado	Não especificado	Colaborativa e cooperativa	Não mencionado
Mecanismo de abstração	Orientado a objetos	Orientado a objetos	Orientado a objetos	Orientado a objetos	Orientado a objetos / Orientado a componentes	Orientado a objetos / Orientado a componentes	Orientado a objetos

Tabela 2. Escopo (QUMER et al 2008)

Como visto na tabela 2, as metodologias estudadas atendem a diversos tipos de projetos e equipes.

Critério	Empresa Júnior	XP	Scrum	FDD	ASD	DSDM	Crystal
Processo de Desenvolvimento	Não mencionado	<ol style="list-style-type: none"> <li>1. Releases curtas</li> <li>2. Metáforas</li> <li>3. Design simples</li> <li>4. Testes</li> <li>5. Refactoring</li> <li>6. Programação em pares</li> <li>7. Posse coletiva</li> <li>8. Integração contínua</li> <li>9. Cliente on-site</li> <li>10. Padrão de código</li> </ol>	<ol style="list-style-type: none"> <li>1. Times Scrum</li> <li>2. Product backlog</li> <li>3. Sprint</li> <li>4. Sprint review</li> </ol>	<ol style="list-style-type: none"> <li>1. Domain object modeling</li> <li>2. Developing by feature</li> <li>3. Individual class ownership</li> <li>4. Feature teams</li> <li>5. Inspection</li> <li>6. Regular builds</li> </ol>	<ol style="list-style-type: none"> <li>1. The project mission development</li> <li>2. Desenvolver por componente.</li> <li>3. Times colaborativos</li> <li>4. Joint application development</li> <li>5. Customer focus group reviews</li> <li>6. Software inspection</li> </ol>	<ol style="list-style-type: none"> <li>1. Envolvimento ativo do cliente</li> <li>2. Empowered teams</li> <li>3. Entrega frequente do produto</li> <li>4. Fitness for business purpose</li> <li>5. Iterative and incremental development</li> <li>6. Mudanças reversíveis</li> <li>7. Requirements are baselined at high level</li> <li>8. Testes integrados</li> <li>9. Colaboração e cooperação junto aos stakeholders</li> </ol>	<ol style="list-style-type: none"> <li>1. Preparação</li> <li>2. Holístico, diversificado e estratégico</li> <li>3. Parallelism and Flux</li> <li>4. User Viewings.</li> <li>5. Revision and review</li> </ol>
Processo de Gestão de Projeto	Características de PMI	1. Jogo de planejamento (Poker do planejamento )	<ol style="list-style-type: none"> <li>1. Scrum master</li> <li>2. Reunião ao começo/fim do sprint</li> <li>3. Reunião diária</li> </ol>	1. Relatórios/visibilidade dos resultados	<ol style="list-style-type: none"> <li>1. Planejamento de ciclo adaptativo</li> <li>2. Gestão de modelo adaptativo</li> </ol>	Não especificado	1. Monitoramento do progresso
Configuração de <i>software</i> de controle de processos / Suporte processos	Não existente	Não especificado	Não especificado	1. Gestão de configuração	Não especificado	Não especificado	Não especificado
Processo de Gestão de Processos	Não existente	Não especificado	Não especificado	Não especificado	1. Postmortem do projeto	Não especificado	1. Reflection workshops Methodology tuning

Tabela 3. Processos (QUMER et al 2008)

Observando a tabela de processos, nota-se que as metodologias ágeis estudadas possuem foco em diferentes fases do processo de produção de *software*, mas são voltadas especialmente para os processos de produção e gestão de projetos.

Fazendo um comparativo com a Empresa Júnior, podemos concluir que a metodologia a ser aplicada deve se focar em projetos com escopo pequenos, equipes pequenas, processos de desenvolvimento dinâmicos e processos de gestão de projetos que dêem suporte a mudanças freqüentes.

## **6 METODOLOGIA ADOTADA**

Para realização do estudo, foi feito o levantamento das principais técnicas e métodos ágeis e sua aplicabilidade em casos reais. Os mesmos foram estudados de forma que se pode fazer um paralelo do que poderia e o que não poderia ser aplicado na Empresa Junior.

Foi aplicado o *framework* 4-DAT (QUMER et al. 2007) para fazer um estudo aprofundado das técnicas e ferramentas de cada método a fim de buscar as características mais fortes e as mais fracas de cada metodologia.

## **7 SOLUÇÃO PROPOSTA**

Após a revisão bibliográfica sobre métodos ágeis, do ambiente da Empresa Júnior e de casos de uso de metodologias ágeis, foram escolhidos o *framework* Scrum e a metodologia de desenvolvimento XP para suprir as demandas da Empresa Júnior.

A escolha se deve principalmente pelo alto número de casos de uso de Scrum e XP sendo utilizados juntos de forma que um complementa o outro em vários sentidos. Outro motivo da escolha se deve pelo número de bibliografias encontradas sobre Scrum e XP.

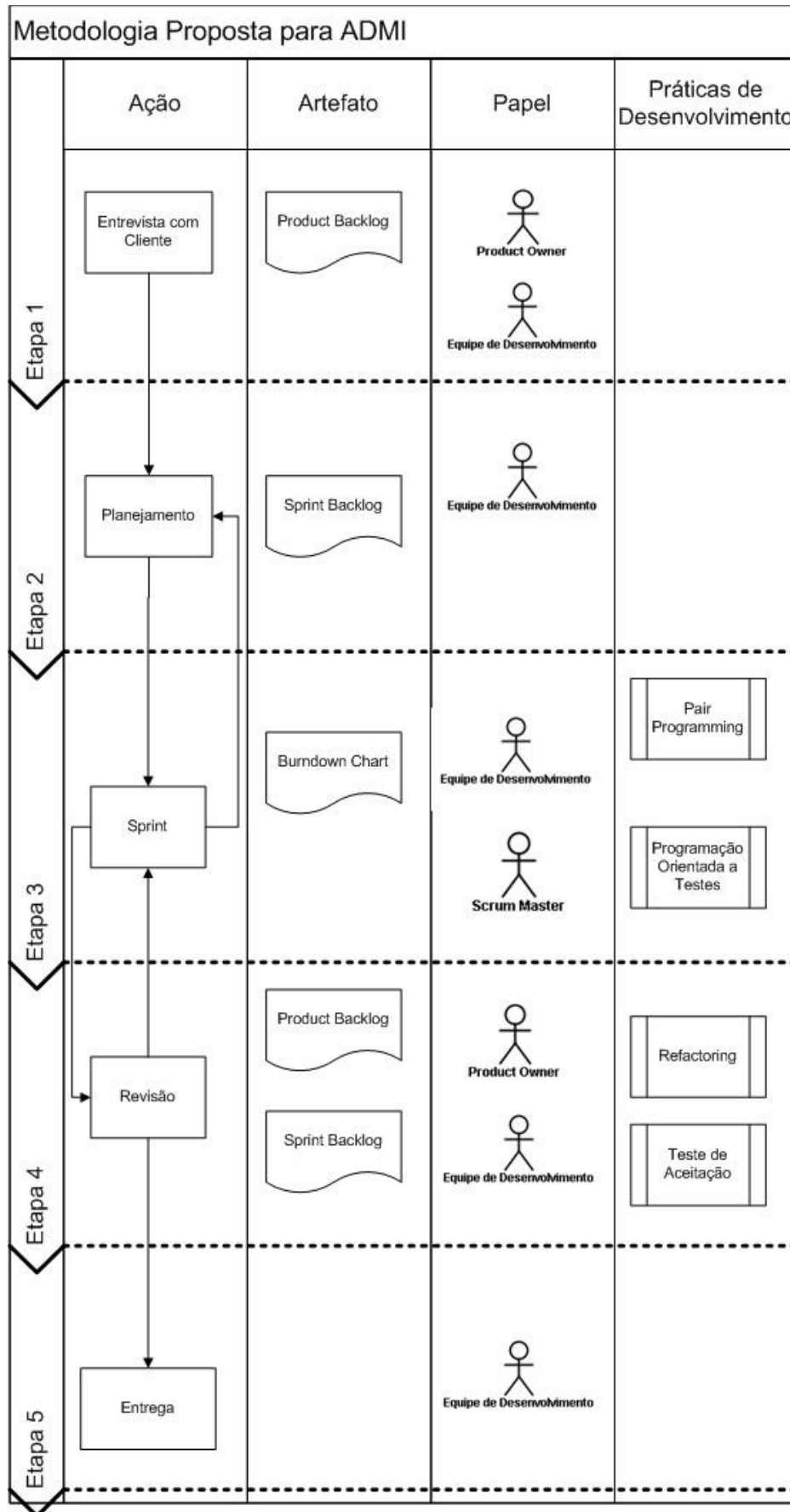


Figura 2. Metodologia Proposta para ADMI.

A figura 2 demonstra as etapas, assim com os papéis e artefatos previstos dentro da metodologia proposta. As etapas serão detalhadas a seguir.

### 7.1 Scrum + XP

Embora tanto Scrum quanto XP tenham aplicabilidade limitada quando usados separados, são altamente complementares quando usados em união. Scrum é usado para suprir tarefas de organização e planejamento do projeto. XP em geral é usado para padronizar tarefas de desenvolvimento. Existem diversos casos que relatam o sucesso dessa união, como o caso da Crisp, de Estocolmo, relatado por Kniberg em Scrum e XP Direto das Trincheiras (KNIBERG 2008).

### 7.2 Etapas de Desenvolvimento

Uma etapa de desenvolvimento é uma fase dentro do ciclo de vida de um projeto. Yourdon (YOURDON, 1989) afirma que, embora existam variações conforme a metodologia adota ou tipo de projeto, existe 5 etapas de desenvolvimento de *software*: Análise de Requisitos, *Design*, Implementação, Testes, Entrega/Manutenção.

A metodologia proposta é composta por 5 etapas definidas de acordo com o *framework* Scrum, porém também possui características XP nas etapas mais voltadas ao desenvolvimento do projeto. As etapas são:

Entrevista com o cliente: A equipe da ADMI deve reunir-se com o cliente (*product owner*), a fim de discutir os detalhes do produto que se busca criar. A equipe deve criar o *product backlog* contendo os prazos, prioridades e estimativa de entrega junto ao cliente, de forma que o mesmo passa a fazer parte da equipe indiretamente, revendo prioridades e detalhando os componentes do produto quando necessário.

Planejamento: Com o *product backlog* pronto, a equipe deve criar um plano de ação, definindo detalhes de desenvolvimento. Esses detalhes, prazos, tarefas e responsabilidades devem ser anotados no *sprint backlog*.

Sprint: A etapa de desenvolvimento. Nela o Scrum máster deverá manter a equipe de desenvolvimento livre de qualquer evento ou situação que possa tirar a atenção da codificação do produto. O Scrum máster será o responsável pelo

*burndown chart*, que será o controle do projeto. Ao final do *Sprint* a equipe deve recriar as prioridades, sendo que, se o projeto for considerado completo, ele deve passar para a fase revisão junto ao cliente. Caso contrário deve ser implementado as funcionalidades que não foram criadas durante o *Sprint* anterior.

Revisão: Com as funcionalidades prontas, a equipe deve criar um teste de aceitação para o *product owner*. Esse teste pode definir se o projeto está de acordo com o que foi especificado e pode se entregue da forma apresentada, ou se deve ser refeito conforme novas especificações.

Entrega: A fase final do projeto onde a equipe se reúne para avaliar o projeto e discutir mudanças que podem melhorar a performance para os próximos projetos.

### 7.3 Papéis

Os papéis (ou roles) propostos são os mesmos descritos pelo framework Scrum:

- Product Owner – É o representante do cliente, responsável por indicar o que deve ser feito à equipe.
- Scrum Team – A equipe responsável pelo desenvolvimento das aplicações. Varia de tamanho, mas em geral é composta de 5 a 9 componentes. Alguns autores citam a necessidade de que a equipe seja formada de programadores plenos e seniores, no entanto as técnicas de XP que serão usadas devem amenizar a necessidade de programadores com vasta experiência.
- Scrum Master – Uma combinação de tutor com gestor. É responsável pelas reuniões com equipe e cliente. Também é responsável pelas avaliações dos processos e motivação da equipe.

### 7.4 Artefatos

São atividades que servem de guia para a produção do projeto. Podem ser manuais de projeto, guias para uma fase determinada ou até *softwares* com função de guiar o projeto (Eclipse Process Framework).

- Product Backlog - O primeiro passo a ser dado por projeto é a criação de um *product backlog*. Um *product backlog* nada mais é do que uma

listagem de funcionalidades exigidas pelo cliente que são indexados pela importância que a equipe e o cliente definem. Além da importância, é estabelecido um prazo para cada funcionalidade. Embora seja indicado que a equipe deva iniciar o projeto a partir das funcionalidades com maior importância e menor prazo, isso pode variar conforme o projeto.

- Sprint Backlog – É um guia de atividades de cada iteração. Basicamente ele informa as atividades que devem ser realizadas dentro do período da iteração. Também informa a prioridade de cada tarefa assim como os papéis dentro da iteração.
- Burndown Chart - É um indicador de progresso que mede como a equipe tem se comportado em relação ao tempo e tarefas por fazer dentro do *sprint*.

## 7.5 Sprint

*Sprint* é a forma como é chamado o ciclo de desenvolvimento do projeto em Scrum. Após criar um *product backlog*, é importante criar um *sprint backlog*, um conjunto de tarefas a ser feitas no *sprint*. O tempo médio de um sprint varia de duas semanas a, no máximo, um mês. Durante o *sprint* é aconselhável uma reunião diária de curta duração (média de 15 minutos) entre a equipe para ser discutido o que foi feito no último dia e o que será feito até o próximo dia. Ao fim de cada *sprint*, a equipe se reúne com o cliente com um release do projeto. Esse release é avaliado e a partir do mesmo são definidos os caminhos que o projeto deve seguir.

## 7.6 Práticas de Desenvolvimento

*Pair Programming*, ou programação em pares, é uma prática utilizada em XP.

Em suma, programação em pares utiliza dois programadores num mesmo computador para desenvolver um código que é pensado por duas mentes diferentes. As duplas são escolhidas conforme a experiência dos programadores sendo que, o programador com menor experiência assume o computador e o com maior experiência supervisiona o código. Os benefícios, tanto para a equipe quanto para o projeto, são óbvios como a diminuição de erros e melhor qualidade de código. A

troca de pares deverá ser realizada no fim de cada *sprint* ou projeto. O desenvolvimento orientado a testes é uma forma que tende a criar códigos de grande qualidade e ao mesmo tempo produzir *softwares* com menos erros. Beck (BECK, 2002) definiu um ciclo simples em 5 passos para o desenvolvimento orientado a testes:

- Criar os testes – Criam-se os testes antes mesmo de criar as funcionalidades. Cada teste deve avaliar uma funcionalidade do sistema.

- Rodar os testes e verificar se falham – Todos os testes devem falhar ao serem rodados pela primeira vez, afinal, não existe código nas funcionalidades ainda. No entanto, caso já tenha sido escrito algum código, os testes devem preservar a integridade das funcionalidades que já funcionam e apenas apontar erros nas funcionalidades com problemas.

- Escrever algum código – Quanto menor a quantidade de código para uma funcionalidade rodar melhor. Esse é o foco do desenvolvimento orientado a testes.

- Fazer as funcionalidades passarem nos testes – Quando todas os testes são feitos e o resultado é positivo os programadores passam para a próxima fase.

- Refatorar – Refatorar o código é uma forma de deixá-lo mais elegante, legível e simples de ler o que torna mais simples de manter.

Existem muitas no mercado que auxiliam na prática do desenvolvimento orientado a testes, como o *jUnit* que realiza testes unitários na plataforma Java. Outra prática a ser implementada é o *story test* (ou teste de aceitação). Um *story test* nada mais é do que uma forma do cliente testar o *software* e participar de forma ativa do projeto dizendo se o *software* atende ou não o que foi especificado.

## **8 CONCLUSÃO E TRABALHO DE CONCLUSÃO DE CURSO – II**

Os projetos realizados pela ADMI, a Empresa Júnior da ULBRA Guaíba, em sua maioria são de pequeno porte, com prazo razoavelmente curto e as equipes de desenvolvimento tem sido de menos de 10 integrantes. Atualmente, são utilizados métodos tradicionais de gestão de projetos para gerir esses projetos, algo que talvez não seja a forma mais eficiente para o desenvolvimento desses projetos.

Os métodos ágeis são formas de desenvolver projetos focando o produto final, com maior aproximação do cliente no projeto e com documentação enxuta.

Após uma revisão bibliográfica sobre o tema, do ambiente da Empresa Júnior e de casos de uso de metodologias ágeis, foram escolhidos o *framework Scrum* e a metodologia de desenvolvimento XP para suprir as demandas da Empresa Júnior.

A escolha se deve principalmente pelo alto número de casos de uso de *Scrum* e XP sendo utilizados juntos de forma que um complementa o outro em vários sentidos. Outro motivo da escolha se deve pelo número de bibliografias encontradas sobre *Scrum* e XP.

A solução proposta visa dividir os projetos em 5 etapas, focando a produção dos projetos de forma simples e ágil. Porém, para não comprometer a qualidade dos projetos, serão utilizadas práticas de desenvolvimento do XP e práticas de gestão de projetos do *Scrum*.

Em TCC-II, a metodologia será experimentada em um projeto da Empresa Júnior. A meta será aplicar a metodologia por completo, buscando produzir um produto de qualidade e de forma eficiente como prega o Manifesto Ágil.

## 9 REFERÊNCIAS BIBLIOGRÁFICAS

ABRANHAMSSON, Pekka; Salo Outi; Ronkainen, Jussi; Warsta, Juhani. **Agile Software Development Methods. Reviews and Analysis.** Finlândia, VTT Publications, 2002.

BECK, Kent. **Test Driven Development: By Example.** Addison-Wesley Longman, 2002.

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: 27 de Março de 2009

BECK, Kent, **Programação Extrema (XP) Explicada. Acolha as mudanças.** São Paulo. Bookman, 2000.

COCKBURN, Alistair; **Agile Software Development**. Addison-Wesley Professional. 2001. 256p

COHN, Mike. **Agile Estimating and Planning**. Prentice Hall PTR, 2005. 368p.

HIGHSMITH, J.A. **Adaptative Software Development: A Collaborative Approach to Managing Complex Systems**. Dorset House Publishing.

KNIBERG, Henrik. **Scrum e XP Direto das Trincheiras: como nós fazemos Scrum**. InfoQ. 2008. 147p.

PALMER, S.R, Felsing, J.M.; **A Pratical Guide to Feature-Driven Development**. Prentice Hall. 2002.

QUMER, Asif, Brian *Henderson-Sellers*. **An evaluation of the degree of agility in six agile methods and its applicability for method engineering**. Elsevier. 2007.

RETAMAL, Adail. **Heptagon, Tecnologia da Informação**, 2009. Disponível em <<http://www.heptagon.com.br/fdd>>. Acesso em 24 de Maio de 2009.

SCHWABER, Ken. **Agile Project Management with Scrum**. Microsoft Press, 2004.

VOIGT, Benjamin J.J, **Dynamic System Development Method**. Zurique, Departamento de Tecnologia da Informação da Universidade de Zurique, 2004.

Yourdon, Edward, **Modern Structured Analysis**. Prentice Hall, 1988.