Análise de Gerenciamento de Projeto de Software Utilizando Metodologia Ágil XP e Scrum: Um Estudo de Caso Prático

Márcia Savoine, Lucyano Martins, Mayton Rocha, Cirlene dos Santos

Curso de Sistema de Informação – ITPAC – Inst. Toc. Presidente Antônio Carlos
Av. Filadélfia, 586 – Setor Oeste – CEP: 77.816-540 – Araguaína – TO

savoine@yahoo.com.br, lucyano@gmail.com, mayton@gmail.com, cirlene_rs@yahoo.com.br

Resumo. *Este artigo relata um estudo de caso sobre projeto de software utilizando as metodologias ágeis eXtreme Programming e Scrum. Aborda-se o uso das melhores práticas de ambas metodologias em uma equipe de desenvolvimento de software; onde esta, evoluiu do paradigma tradicional para o ágil. Isto ocorreu devido às necessidades dos modelos de negócios atuais, tais como: crescimento da equipe, melhoria na qualidade do código implementado e exigências do mercado de software pela excelência do produto. São levantadas as principais características das metodologias tradicionais e ágeis, fazendo um comparativo entre elas.*

Palavras chave: Metodologia Ágil, eXtreme Programming, Scrum.

1. Introdução

Os problemas enfrentados pelas empresas de desenvolvimento de software, como: os atrasos na entrega do projeto, produtos de baixa qualidade, aumento significativo dos custos muito além do esperado; são fatores causados pela falta de gerenciamento nos processos que envolvem a produção de software**,** que muitas vezes**,** estão ligados diretamente há uma coleta de dados realizada de forma incompleta.

O desenvolvimento de software tem falhas na entrega e nos valores entregues. Essas falhas têm impactos econômicos e humanos enormes. É necessário achar uma maneira de desenvolver software com qualidade e entregas freqüentes. (BECK, 2004)

Com a globalização a concorrência tornou-se muito acirrada; com isso**,** a tolerância à falhas, atrasos, *softwares* obsoletos e projetos cancelados estão cada vez menores, e as empresas precisam se enquadrar nesse novo contexto para poderem se destacar neste cenário.

Os métodos ágeis têm-se destacado nesse contexto de mudanças, oferecendo respostas rápidas a esse novo ambiente de desenvolvimento, onde os requisitos são mutáveis, não estão totalmente claros ou não são totalmente esclarecidos; mas exige-se a de entrega do produto com valor tangível. Com todos esses problemas envolvendo a produção de software, faz-se necessário a mudança dos métodos utilizados no desenvolvimento, através da adesão a um novo paradigma, eliminando conceitos que muitas vezes atrapalham o andamento do projeto.

Este artigo contempla um estudo das metodologias ágeis, explicando suas principais características, e realiza-se um comparativo entre as mesmas, incluindo um estudo de caso prático. Discutem-se os resultados obtidos deste *case* e são apresentadas às conclusões na última sessão.

2. Metodologias Ágeis

Os métodos ágeis de desenvolvimento de *softwares* surgiram com a necessidade de tornar o desenvolvimento de software mais leve, flexível a mudanças, sem o aumento exponencial dos custos em contrapartida dos métodos tradicionais que desperdiçavam muito tempo em análise e planejamento.

A partir de fevereiro de 2001, desenvolvedores de *software* de várias partes do mundo, insatisfeitos com as técnicas e métodos de desenvolvimento de sistemas usados até o momento, resolveram criar uma aliança chamada de *Agile Software Development Alliance*, mais conhecida como *Agile Alliance.*

Estes profissionais das diversas áreas de formação com pontos de vista diferentes sobre os modelos e métodos de desenvolvimento de software em comum, criaram um manifesto para encorajar melhores meios de desenvolvedor software (AMBLER, 2004).

Este documento, definido como o **Manifesto Ágil ou *Agile Manifesto*** para as novas metodologias de desenvolvimento de software; foi criado por vários desenvolvedores, mas teve como idealizadores Kent Beck, Ken Schwaber, Martin Fowler e Jim Highsmith. E, abrange um conjunto de princípios, que definem critérios para o processo de desenvolvimento de software ágil. Tais princípios ágeis são:

- Indivíduos e iterações são mais importantes que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano; ou seja, mesmo havendo valor nos itens à direita, valoriza-se mais os itens à esquerda.

As metodologias ágeis têm conseguido maior espaço devido principalmente a mudanças no perfil dos projetos de *software*. As abordagens tradicionais são pouco propícias às mudanças para atender as necessidades do projeto. No entanto, a realidade atual é composta por alterações a todo o momento e requisitos em constante evolução; ou seja, projetos mutantes, por isso surge à necessidade de metodologias que se adaptem a esse paradigma.

2.1 Metodologia Ágil eXtreme Programming

Os fundamentos da metodologia *eXtreme Programming - XP*; ou ainda Programação eXtrema, teve início na década de 80, tomando como base o desenvolvimento em *SmallTalk*, que já naquela época praticava o desenvolvimento com programação pareada, *refactoring*, testes constantes, desenvolvimento iterativo, que hoje são utilizados nas práticas da XP.

Segundo Beck (2004), XP é uma metodologia leve para equipes de tamanho pequeno a médio, que desenvolve software em face a requisitos vagos que modificam rapidamente. É uma maneira leve, eficiente, de baixo risco, flexível, previsível, científica e divertida de desenvolver software.

A Programação eXtrema possui como característica importante ser adaptativa, usada principalmente quando se tem um ambiente no qual os requisitos funcionais do

sistema não estão totalmente claros, ou não são conhecidos pela equipe de desenvolvimento; e quando os mesmos são mutáveis ao longo do ciclo de vida do projeto.

É uma metodologia que encoraja a equipe a enfrentar as mudanças, como algo natural, sendo importante se adaptar aos contratempos que acontecem no projeto, contornando assim as dificuldades do modo mais simples possível sem perder o foco básico: produzir um produto de qualidade, almejando excelência.

A satisfação do cliente é uma preocupação constante na XP, pois entregar um produto de qualidade em tempo hábil que satisfaça as exigências e expectativas tanto do cliente como da equipe de desenvolvimento é o objetivo principal. Esta metodologia segue rigorosamente quatro princípios de fundamental importância para seu embasamento: a *comunicação*, o *feedback*, a *simplicidade* e a *coragem*.

A Programação eXtrema enfatiza o uso de 12 práticas, utilizadas de forma harmônica e complementar para a entrega de um software de alta qualidade, que são mostradas na tabela 1, apontados seus pontos fortes e fracos.

Tabela 1. Características da metodologia ágil XP.

Referência	Pontos Fortes	Pontos Fracos
1	Reuniões diárias: tiram-se dúvidas de estórias complexas, planejam-se as iterações do dia, discutem-se as dificuldades encontradas.	Não indicada para sistemas complexos, pois a análise não é detalhada.
2	Escopo do projeto não bem definido: os requisitos não são totalmente conhecidos ou são muito vagos.	Não indicada para equipes muito grandes, pois o escopo da equipe tem que ser bem definido, e as pessoas, geralmente, estão separadas geograficamente.
3	Foco na codificação, sendo que a documentação fica para segundo plano.	Não foca muito a gerência do projeto, o que interessa a equipe são os resultados, o código rodando.
4	Cliente presente é a chave para o sucesso.	Não é recomendada em ambientes onde os clientes não pode estar presente.
5	Flexível a mudanças encoraja as pessoas a enfrentarem com naturalidade.	Não é recomendado em projetos que o cliente exige uma extensa documentação do software.
6	Simplicidade: executar tarefas com simplicidade e qualidade, desenvolver funcionalidades realmente útil ao usuário.	Não recomendada em equipes que possuem resistências às práticas como *refactoring*, programação pareada, etc.
7	Iterações curtas, com isso o *feedback* do usuário é rápido e constante, realimentando o desenvolvimento das funcionalidades.	Não indicada em projetos em que o cliente não possa participar integralmente do desenvolvimento.
8	Os testes fazem parte do ciclo de desenvolvimento, aumentando assim a qualidade do software.	Se o *feedback* entre o cliente e a equipe não for constante, ou houver uma demora de respostas entre às partes, a XP não será produtiva.
9	Integração diária do código produzido.	Quando houver resistência em modificar o código já concluído, com o objetivo de torná-lo simples, estruturado.
10	Código coletivo: todos são responsáveis, e aptos a realizares melhorias, caso necessário.	Não é recomendada em ambientes de trabalhos exaustivos, a XP enfatiza o trabalho de 8 horas diárias.
11	Padronização do código: evitar problemas na hora da refatoração e da manutenção;	Não indicada para projetos que necessitem de ciclos de iterações longos;
12	Refatorar: simplificar o código, sem alterar sua funcionalidade.	A XP não é uma metodologia flexível em relação as suas práticas, que devem ser seguidas rigorosamente.

Segundo Beck (2004), as práticas apóiam umas às outras. O ponto fraco de uma é compensado pelos pontos fortes das outras. As peças individuais são simples. A riqueza vem da interação entre as partes.

2.1 Metodologia Ágil Scrum

A utilização da metodologia *Scrum* está crescendo muito ultimamente no mundo de desenvolvimento de *software*. Isto, por ser uma metodologia empírica, adaptativa, inovadora, usada no desenvolvimento de sistemas de modo incremental, onde os requisitos podem sofrer mudanças durante o processo de fabricação do produto.

É uma metodologia com o foco no gerenciamento da equipe, preocupada na organização dos processos, no modo como as atividades devem ser executadas, deixando a cargo dos participantes do projeto escolher a melhor maneira de concluir com sucesso essas etapas.

Os requisitos do software são levantados e organizados em uma lista de funcionalidades, chamada *product backlog*, que contém todo o escopo do projeto definido juntamente com o cliente, e suas respectivas prioridades de desenvolvimento, sendo que o *product backlog* dever ser constantemente revisado, validado e atualizado.

O desenvolvimento é dividido em iterações, chamado de *sprints,* de duração entre duas a quatro semanas. Cada *sprint* é composta de uma lista de tarefas, funcionalidades com prioridades retiradas do *product backlog*, chamada de *sprint backlog.* Definido as atividades a serem realizadas em cada *sprint,* a equipe foca o desenvolvimento de mais um ciclo que se repete ao longo do projeto. Após o encerramento das *sprints* é realizada a *sprint retrospective*, uma das mais importantes práticas dentro da *Scrum,* que são discutidos os pontos positivos e os negativos. A figura 1 ilustra o ciclo de vida da Scrum.

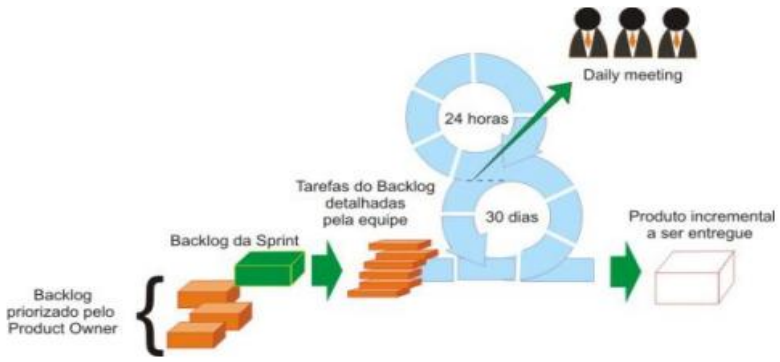

Figura 1. Ciclo de vida da *Scrum* - Fonte: PEREIRA, 2007.

Durante a fase das *sprints*, acontecem reuniões diárias, de no máximo 15 minutos chamadas *daily scrum meetings***,** com todos os integrantes da equipe. Onde se coloca em pauta perguntas sobre as atividades realizadas no dia anterior. No encerramento de cada ciclo da *sprint,* ocorre a apresentação das tarefas concluídas, também conhecidas como *sprint review*, onde o grupo apresenta os resultados atingidos e as funcionalidades 100% prontas e testadas a toda a equipe.

Os papéis da *Scrum* são: *Product Owner, ScrumMaster* e o *TeamMember***.** Esta metodologia pode ser usada em equipes pequenas e grandes, mas recomenda-se quando for um grupo com mais de dez pessoas, que faça à subdivisão do mesmo. Então, com o

projeto dividido em partes, que todos trabalhem paralelamente e de forma sincrônica. Segue a tabela 2 indicando os pontos fortes e fracos da metodologia.

Tabela 2. Características da metodologia ágil Scrum.

Referência	Pontos Fortes	Pontos Fracos
1	Metodologia destinada ao caos, onde não se tem todos os requisitos do sistema; ou eles, não estão totalmente definidos.	Não indicada para sistemas complexos, pois a análise não é detalhada.
2	Flexibilidade para mudanças rotineiras que acontecem freqüentemente durante as fases do projeto.	Não indicada para projetos que exigem uma vasta documentação.
3	O cliente faz parte da equipe em tempo integral.	Não indicada onde os clientes não tenham disponibilidade de tempo e onde são resistentes a troca de informações.
4	Reuniões diárias que guiam o andamento do projeto.	Não indicada para equipes que não fazem usa das reuniões diárias, sendo de extrema importância para o sucesso de um projeto.
5	Ciclos de desenvolvimentos curtos e constante, conhecidas como *sprints.*	Não indicada para projetos que necessitem de ciclos de iterações longos.
6	Comunicação entre os membros da equipe é freqüente.	Não recomendada em ambientes onde as tarefas são impostas as pessoas, pois as mêsmas têm liberdade em escolher tarefas que tenham afinidades de desenvolvimento.
7	Revisões das funcionalidades realizadas acontecem no final de cada ciclo.	Projetos que o líder não divide responsabilidades com a equipe, não tendo efetiva troca de idéias com o restante do grupo, a Scrum não será produtiva; já que, defende trabalho colaborativo, não apenas feito pelo líder.
8	Pode ser aplicada tanto para equipes pequenas e grandes, e também para projetos de pequeno e de grande porte.	Definindo-se a *sprint*, nenhuma mudança póderá ser acrescentada na mesma; adições de novos requisitos são feitos na próxima *sprint*.
9	Elaboração de cronogramas para a fixação de datas e horas trabalhadas.	Não indicada em projetos que não há a presença do cliente, nesta metodologia se tem o papel *do product owner* que representa a figura do cliente.
10	Testes acontecem em cada fase das *sprints*.	Equipe tem resistência em realizar testes na fase de desenvolvimento, a Scrum não é recomendada.

3. Estudo de Caso Utilizando as Metodologias XP e Scrum

O estudo de caso foi realizado em uma equipe de desenvolvimento com 15 colaboradores, que se constituiu para trabalhar em um projeto de *software* de gestão de ensino, em uma fábrica de *software* que já desenvolvia outros produtos.

Após diversas reuniões e estudos de mercado, a equipe chegou ao consenso de que seria necessário mudar a atitude em relação ao desenvolvimento de *software*, pois hoje o mercado traz novos conceitos e novas necessidades. Fez-se a análise do paradigma atual e propôs-se a adoção de práticas ágeis, pois este seria o primeiro projeto de desenvolvimento ágil e os requisitos não se apresentavam com exatidão.

3.1 Práticas Ágeis Utilizadas no Projeto

A tabela 3 mostra a compilação das melhores práticas selecionadas entre as metodologias XP e Scrum para serem utilizadas no projeto de gestão de ensino. Dividiram-se as práticas por metodologia e foco, abordando o nome pela qual a prática é

conhecida, a metodologia que utiliza a mesma, uma descrição de sua utilização dentro do estudo de caso e a justificativa da escolha das práticas para o projeto.

Tabela 3. Práticas ágeis das metodologias XP e Scrum Aplicadas no estudo de caso.

PRÁTICA	ORIGEM	DESCRIÇÃO	JUSTIFICATIVA
Práticas Focadas no Desenvolvimento			
Código Coletivo	**XP**	Código fonte disponível para qualquer desenvolvedor a qualquer momento.	Tornou-se necessário adotar essa prática devido à necessi-dade da implementação ser feita em colaboração, ou seja, vários desenvolvedores no mesmo projeto.
Refatoração	**XP**	A refatoração está presente no projeto devido ao código coletivo, quando um desenvolvedor visualiza uma possibi-lidade de melhoria, isto é feito e docu-mentado no próprio código, criando o histórico dessas alterações.	A coletividade do código tam-bém gera responsabilidades a todos os desenvolvedores de perceberem possibilidades de melhoria.
Simplicidade	**XP**	Todo o código produzido é elaborado com o máximo de simplicidade para que possa ser economizado tempo de produção e entendimento da estrutura do código.	A utilização da simplicidade justifica-se pelo fato do código ser coletivo; ou seja, todos precisam entender o que foi implementado.
Foco na Codificação	**XP**	A documentação tem a função de ori-entar o desenvolvedor as necessida-des do cliente, porém não é o foco da equipe de analistas desenvolver docu-mentação.	A prioridade da equipe é soft-ware rodando com base nos princípios ágeis; porém, a do-cumentação não é dispensada.
Integração Contínua	**XP**	Toda a produção do projeto é organizada no servidor automati-camente pela ferra-mentas utilizadas, o código produzido pelos desenvolve-dores, documentação pelos analistas, e requisitos pelos clientes e gerentes.	O desenvolvimento em cola-boração exige que tudo esteja centrado em um único lugar permitindo assim que essas informações sejam repassa-das a todos da equipe.
Testes	**XP**	Após o término de cada *sprint* semanal ou quinzenal é repassado o código compilado em executáveis para o departamento de testes, após a conclusão dos testes os desenvolve dores trabalham em rodízio para correção de *bugs.*	Para evitar a entrega de produtos com um número elevado de erros para o cliente faz-se necessário estabelecer uma rotina de testes.
Programação Pareada	**XP**	O desenvolvimento em pares ocorre quando uma funcionalidade requer conhecimento de vários desenvol-vedores; sendo assim, estabelece um par de desenvolvedores responsável para implementar essa funcionalidade.	A necessidade de desenvol-ver em pares ocorre quando o domínio sobre o problema não é total por somente uma pessoa, faz-se assim a junção de pares para a proposta do desenvolvimento em duplas.
Práticas Focadas no Gerenciamento			
Product Backlog	**SCRUM**	Volume de documentação inicial do projeto que foi extraída na fase de análise de requisitos pelos analistas contém as informações sobre o escopo do projeto, ficando no portal do projeto e, é acessível por todos os membros da equipe.	Essa documentação engloba a estrutura fundamental do projeto é composta pela lista de todos os módulos do sistema com a descrição de suas funcionalidades que foram estabelecidas junta-mente com o cliente.

Sprints	**SCRUM**	As iterações têm duração de duas semanas, que é a divisão do desenvolvimento, que por sua vez, compõem a lista de funcionalidades a serem desenvolvidas.	Os *sprints* são como o plano a ser seguido o mapa para a equipe possa se nortear no que desenvolver durante o prazo do ciclo.
Sprints Backlog	**SCRUM**	No planejamento de cada s*print* retira-se as funcionalidades a serem desenvolvidas durante aquela fase; é repassado para o papel e exposta no quadro atual da equipe para que seja fácil o acesso e visível a todos.	Durante cada *sprint* é feita à lista dessas funcionalidades a serem desenvolvidas para que seja possível acompanhar as tarefas da equipe.
Sprint Retrospective	**SCRUM**	Quando fecha-se uma *sprint* é feita uma retrospectiva para que os problemas encontrados pela equipe sejam discutidos, isto para não ocorrer no futuro, e também os acertos para servirem como base a todos.	Essa prática torna-se fundamental à medida que o projeto é desenvolvido, para que a equipe possa crescer com os erros e acertos.
ScrumMaster	**SCRUM**	O gerente do projeto, líder da equipe, tem papel fundamental na parte gerencial do projeto, auxilia toda a equipe, resolve problemas que possa estar prejudicando o andamento das tarefas, exige dos colaboradores, delega responsabilidades, marca reuniões e em contato direto com os patrocinadores.	A liderança máxima na equipe em relação aos participantes da linha de produção é de fundamental importância, pois o seu papel principal é facilitar o trabalho dos membros da equipe.
Daily Scrum Meeting	**SCRUM**	Pequenas reuniões diárias com a participação de todos os integrantes da equipe, para que possa ser dado sempre o status do projeto, e também para que o *ScrumMaster* possa resolver qualquer problema que impeça a equipe de desenvolver o *sprint* dentro do tempo hábil.	As reuniões são de grande importância para o projeto, pois é onde as informações sobre o status atual do projeto é dada para todos, também o líder *ScrumMaster* exerce o seu papel de facilitador das tarefas da equipe.

Pode-se observar a diferença de foco entre as metodologias; enquanto a XP é voltada para as práticas de implementação, a Scrum foca no gerenciamento e planejamento e; também, como tratar com a parte estratégica do projeto.

Apresenta-se na Tabela 4 as melhores práticas aplicadas neste estudo de caso; porém, algumas práticas previstas nas metodologias ficaram fora do projeto devido a empecilhos impostos pela empresa que subsidia a equipe de desenvolvimento.

Tabela 4. Práticas ágeis não utilizadas no projeto

PRÁTICA	ORIGEM	JUSTIFICATIVA	IMPORTÂNCIA
Cliente sempre presente	XP	Apesar de existir consultores o projeto foi iniciado sem patrocinador específico o que impediu a presença constante desse ator de papel fundamental no início do projeto.	Com o cliente sempre presente é possível ter sempre em mãos os requisitos diretos da fonte; ficando mais simples para a equipe entender os requisitos propostas.
Semana de 40 horas	XP	Os membros da equipe do projeto também atuam em outros projetos desenvolvidos pela empresa que acolhe a equipe; portanto, os desenvolvedores têm semanas de 44 horas trabalhando em expedientes de segunda a sexta-feira de 8 horas e aos sábados de 4 horas.	Trabalhar com colaboradores sempre motivados e descansados é um dos princípios do manifesto ágil, permite extrair do colaborador o máximo do seu poder criativo, fundamental para desenvolver-se software.

3.2. Ferramenta Ágil para Gerenciamento das Práticas Ágeis

Para tornar possível a utilização das práticas ágeis, utilizou-se uma ferramenta que promove a colaboração, a qualidade, a previsibilidade e o controle, permitindo também a integração de todas as equipes envolvidas no projeto, trazendo recursos que oferecem ao gerente do projeto a possibilidade de entender a aplicação dos pilares do desenvolvimento ágil.

4. Resultados Obtidos

Os resultados obtidos foram classificados com foco nos resultados de desenvolvimento e nos resultados gerenciais. São descritos nos próximos itens.

4.1 Resultados de Desenvolvimento:

a) Qualidade do Código Fonte

Na qualidade do Código Fonte: esses resultados partiram das práticas de implementação da metodologia XP. Observou-se que o código fonte produzido no projeto atende critérios de qualidade estabelecidos em métricas padrões da ferramenta; essa qualidade pode ser comprovada com a opção *Code Analysi*. Na figura 2 mostra uma avaliação de parte do código fonte.

Code Metrics Results

Filter: None

Min: Max:

Hierarchy	Maintainabilit...	Cyclomatic Complexity	Depth of I...	Class Coup...	Lines of Code
Secretaria (Debug)	56	2.875	8	296	21.247
Ren	49	358	7	142	2.421
FrmDiario	49	358	7	142	2.421
Secretaria	63	136	7	104	962
FrmCalendarioLetivo	44	90	7	61	620
FrmPrincipal	63	45	7	53	339
Program	81	1	1	3	3
Secretaria.Formularios	56	2.381	8	241	17.864
FrmCadAlergias	58	69	8	68	391
FrmCadAlunos	42	174	8	127	2.516
FrmCadAlunos.ParametrosConsulta	100	0	1	0	0
FrmCadBimestres	54	74	8	77	535
FrmCadCalendario	53	110	8	90	693
FrmCadCategDisciplina	56	73	8	68	429

Figura 2. Análise do código fonte.

Os valores das colunas da figura 2 mostram os itens:

- ***Maintainability Index*** - mostra a facilidade de manutenção do código, quanto maior o valor indicado mais fácil à manutenção**.**
- ***Cyclomatic Complexity*** – demonstra o quanto o código apresenta complexidade, quanto menor o valor menos complexo é a codificação.
- ***Depth of Inheritance*** – mostra a quantidade de herança da classe, quanto menor o valor, melhor será para esse campo.
- ***Class Coupling*** – avalia a quantidade de referências de outras classes, o valor menor é o melhor neste item.
- ***Lines of Code*** – aproximadamente o número de linhas de código do executável.

Justifica-se o alcance dessa qualidade devido às práticas ágeis voltadas para codificação como: simplicidade, refatoração e coletividade do código. Porém, o gerenciamento do projeto proporcionado pela ferramenta é foco da metodologia Scrum; e este acompanhamento das etapas desenvolvidas, é função do *ScrumMaster*. A figura 3 mostra o acompanhamento realizado pelo *ScrumMaster* na ferramenta.

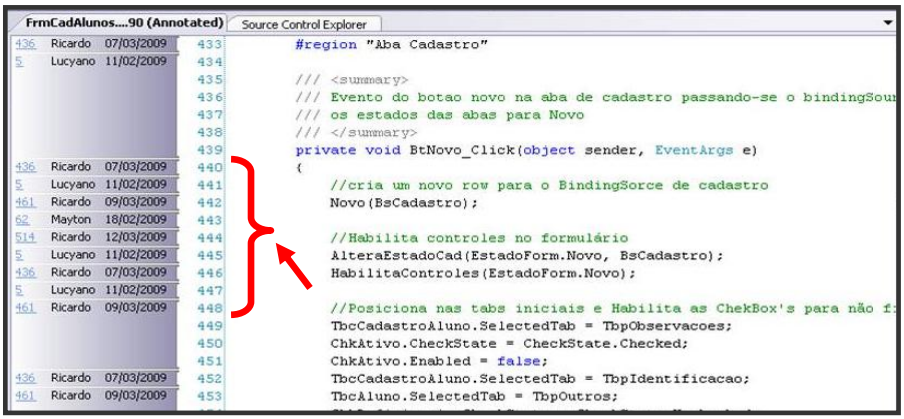

Figura 3. *Annotated* mostra a refatoração e a coletividade do código-fonte, indicando qual usuário e em qual data fez a alteração.

b) Integração Contínua

A integração contínua ocorre de forma automática através da opção *Source Control* da ferramenta. Com o uso dessa prática é possível unir o trabalho da equipe de forma coesa, permitindo assim, além de centralização a armazenagem segura.

O ganho de tempo com o uso da prática de integração contínua não pode ser mensurado de forma precisa; pois anteriormente, esse processo era totalmente manual e gastavam-se várias horas para que o gerente do projeto reunisse todo o trabalho produzido. Isto, atualmente é feito de forma automática e imperceptível pelos membros da equipe. Porém; pode-se fazer uma estimativa de que o ganho de tempo na equipe é de aproximadamente 48 horas por mês (sendo 2 horas diárias de perda), totalizando 1 mês de tempo para a equipe em produtividade.

4.2 Resultados Gerenciais:

a) Cumprimentos de Prazos

Foi possível medir a eficiência no cumprimento das datas pré-estabelecidas, após a utilização das práticas de planejamento e delegação de tarefas conforme as iterações e com estabelecimento de prazos a serem cumpridos pela ferramenta. Na Tabela 5 são mostrados os dados obtidos a partir do primeiro *sprint* de funcionalidades desenvolvidas dentro e fora do prazo.

Tabela 5 – Dados de itens desenvolvidos dentro e fora do prazo.

Descrição	Quantidade	Média da Quantidade
Dentro do prazo previsto	38	58%
Fora do prazo previsto	28	42%
TOTAL	**66**	**100%**

b) Resultados dos Testes

Analisaram-se os resultados dos testes com base nos *Work Itens* (itens de trabalho) desenvolvidos durante a iteração, ao final foram realizados os testes considerando cada funcionalidade para verificar quantos e quais itens seriam necessários serem revistos pelos desenvolvedores. Esses resultados são apresentados na Tabela 6 abaixo.

Tabela 6. Resultados dos testes aprovados e a serem revistos pelos desenvolvedores.

Descrição	Quantidade	Média da Quantidade
Itens Aprovados	45	68%
Itens Revistos	21	32%
TOTAL	**66**	**100%**

Observou-se que o índice de revisão foi baixo, devido a gerência anteriormente aplicada aos itens e; também as técnicas de desenvolvimento propocionarem um código com maior qualidade e, então consequentemente as funcionalidades atenderam as necessidades propostas.

5. Conclusão

A padronização dos processos de desenvolvimento de software é essencial para o alcance do sucesso; sem o mesmo, o desenvolvimento comercial fica inviável, ou tende ao fracasso. O mercado de *software* está cada vez mais competitivo, exigindo produtos de qualidade, equipes dinâmicas, cliente sempre presente; sendo de fundamental importância uma metodologia que possa proporcionar a construção de artefatos aplicáveis a soluções de problemas corriqueiros.

Os resultados obtidos com a utilização das metodologias ágeis foram fundamentais para o sucesso do projeto e da equipe de desenvolvimento, causando impactos determinantes para o alcance dos objetivos propostos; tais como: a mudança de conceito de desenvolvimento tradicional para o ágil, alcançando as exigências de mercado e excelência dos produtos desenvolvidos.

Como mostrado nesse trabalho, o uso das metodologias ágeis *XP* e *Scrum* não é aplicável em todas as situações, e frequentemente requer uma adaptação específica. Estas adaptações mostram os aspectos positivos em relação a flexibilidades desse tipo de metodologia, apresentando um leque de soluções sugeridas a serem usadas em cada problema encontrado.

A equipe absorveu com naturalidade o uso das práticas da metodologia XP, não seguindo a risca todas elas; mas acolhendo as que se adequavam melhor na situação atual da empresa. O uso da ferramenta de gerenciamento foi imprescindível para o alcance dos objetivos iniciais, auxiliando o *ScrumMaster a* mensurar todo o trabalho realizado pela equipe e a demonstrar os resultados a todos os participantes do projeto, onde sem esta mudança de paradigma seria um pouco mais árdua e complicada de se conseguir.

Contudo, pode-se notar que o foco desse tipo de metodologia não está na análise de riscos, nem tão pouco em uma documentação detalhada, deixando a desejar em projetos grandes, e com complexidade um pouco mais elevada, que necessitam de um planejamento mais aprofundado. Assim, o grande desafio das metodologias ágeis é encontrar formas de se detalhar a análise de riscos sem torná-las pesadas; mantendo acima de tudo a flexibilidade, que é uma das grandes características que as diferenciam das demais.

6. Referências Bibliográficas

AMBLER, Scott W. Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado. In: Porto Alegre, Bookman, 2004.

BECK, Kent. Programação Extrema (XP) explicada: acolha as mudanças. In: Porto Alegre: Bookman, 2004.

PEREIRA, Paulo; TORREÃO, Paula; MARÇAL, Ana Sofia. Entendendo Scrum para Gerenciar Projetos de Forma Ágil. Mundo PM. v. 1.8, p3-11, 2007.