Desenvolvimento de Software: Processos Ágeis ou Tradicionais? Uma visão crítica.

Mainart, Domingos de A[1]. Santos, Ciro M.[1,2]

[1]Faculdade Presidente Antônio Carlos de Teófilo Otoni
[2]Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM
Teófilo Otoni MG.

domingos@unipacto.com.br, cirosantos@gmail.com

Abstract. The software development community has faced great difficulties in delivering quality software, within the agreed period, with all the requirements provided and in accordance with the approved budget. The search for new approaches have characterized the software engineering that seeks to improve the processes and software products that have the classic model, one of the most widespread approaches in the 70's. In the late 90s, there were the so-called agile methods, which have made great advances in its broadcast from 2001 with the publication of the Agile Manifesto. These methods are characterized by values, principles and practices that are guided by a perspective different from the traditional approach. While the traditional approach emphasizes the process and rigorous documentation, the agile approach focuses on people, short iterations that deliver products and lightness of the process. This article proposes a comparative analysis of two fronts of these methodologies, the traditional and agile methodologies.

Resumo. A comunidade de desenvolvimento de software tem enfrentado grandes dificuldades para entregar software de qualidade, no prazo contratado, com todos os requisitos contemplados e de acordo com o orçamento aprovado. A busca de novas abordagens tem caracterizado a engenharia de software que busca a melhora dos processos e produtos de software que tem no modelo clássico, uma das abordagens mais difundidas na década de 70. No final da década de 90, surgem os métodos denominados ágeis, que tiveram grande impulso na sua difusão a partir de 2001, com a publicação do manifesto ágil. Esses métodos caracterizam-se por valores, princípios e práticas que se orientam por uma perspectiva diferente da abordagem tradicional. Enquanto a abordagem tradicional dá ênfase para o processo e documentação rigorosa, a abordagem ágil foca pessoas, iterações curtas com entrega de produtos e leveza do processo. Este artigo propõe uma análise comparativa de duas frentes destas metodologias, as tradicionais, e as metodologias ágil.

1. Trabalhos Relacionados

Os problemas de qualidade, de prazos e de orçamentos aprovados vêm sendo amplamente pesquisados pela comunidade de desenvolvimento de solftware. A seguir

faremos uma breve revisão dos principais trabalhos que originaram análises comparativas que foram consideradas neste artigo.

O trabalho apresentado por Cecilia Keiko – "Impacto do uso de métodos ágeis no processo de ensino-aprendizagem de engenharia de softwa", demonstra claramente a tendência atual rumo a metodologia ágil e seu grande crescimento no mercado. Já o Oscar Nogueira em "Análise comparativa das metodologias de desenvolvimento de software tradicionais e ágeis", demostra a necessidade de se avaliar primeiramente a dimensão do projeto, podendo inclusive usar o que a de melhor entre ambas as metodologias.

2. Introdução

Desde a Crise do Software, que forçou com que as *Software Houses* realizasse projetos de desenvolvimento de Software de uma maneira mais profissional e organizada, gerando uma documentação para acompanhar o produto, muitas metodologias para esse desenvolvimento surgiram. Linguagens foram criadas para modelar e facilitar o entendimento do produto pelo cliente e pela própria empresa desenvolvedora [PRESSMAN 02].

Essa documentação gerada a partir da análise da especificação dos projetos, era acompanhada de um método de desenvolvimento para especificar ao cliente o que seria desenvolvido e como isso seria feito. Foi então que surgiram Métodos para visualizar o processo de desenvolvimento, dividindo-o em etapas, sempre com foco na qualidade final do produto.

Os processos sempre começavam com uma primeira etapa de Análise, onde os requisitos eram estabelecidos de acordo com a necessidade do cliente. Depois da Análise, tem-se a Modelagem que gera os documentos necessários e por último o Desenvolvimento e a fase de Testes [XISPE 04].

Levando-se em consideração que o processo de desenvolvimento é bastante mutável, pois o surgimento de novos requisitos (tanto funcionais como não-funcionais) por parte do cliente é normal, assim como a não conformidade de algumas solicitações, era necessário alterar constantemente a documentação e o produto em si. Isso demandava muito tempo.

Para circular esse problema de constantes alterações, assumindo que essas mudanças fazem parte do processo de desenvolvimento surgiram os Métodos Ágeis, aqueles com foco no código e otimizados para alterações de requisitos [XISPE 04], como a XP-Extreme Programming [WELLS 04], esses métodos também prezam pela qualidade do Software, mas a sua filosofia de desenvolvimento é diferente, dando ênfase principalmente no código, sendo que as alterações necessárias não devem acarretar em tanto tempo gasto.

Com o surgimento recente de novas Metodologias de Desenvolvimento de Software – MDS, que dividiram o processo de desenvolvimento de Software, para organizá-lo, planejá-lo e facilitar o entendimento do mesmo, existem agora duas principais áreas de atuação, o Desenvolvimento chamado Tradicional, com embasamento na Análise e Projeto, mantendo tudo em documentos, porém se tornando lento para mudanças, e o Desenvolvimento Ágil, baseado em código, totalmente adaptável a mudanças nos requisitos, entretanto, fraco na parte contratual e de documentos.

O artigo faz um comparativo entre as Metodologias de Desenvolvimento Tradicional e o Ágil, analisando pontos fortes e fracos de cada uma e os pontos em comum.

O foco principal do trabalho é um comparativo entre as metodologias, analisando cada uma de acordo com características comuns a um processo de software, a fim de concluir onde e por que uma é superior a outra conforme diferentes visões.
No entanto, para termos informações suficientes sobre ambas as metodologias foi necessário estudar cada uma separadamente, analisando o processo de cada uma, além de estudarmos o principal expoente de cada, sendo o RUP – Rational Unified Process – das metodologias tradicionais e o XP – eXtreme Programming – como referencial das metodologias ágeis.

3. Desenvolvimento de Software Visão Geral

A engenharia de software surgiu num contexto onde a crise de software se apresentava e, decorrente disso, sistemática de trabalho mais consistentes e formais, inspiradas na engenharia, foram concebidas para solucionar os problemas que tendiam ser, cada dia, maiores e mais complexos e que acompanhavam a comunidade de desenvolvedores ao longo dos anos, de forma crônica (Pressman, 1995).

A partir deste cenário, surgiu a necessidade de tornar o desenvolvimento de Software como um processo estruturado, planejado e padronizado, para que as necessidades fossem atendidas e os gastos com informatização de processos de informações se tornassem compensadores.

Entretanto, a Crise do Software perdura até hoje, onde, mesmo com técnicas avançadas de desenvolvimento e padrões consolidados na área de criação de softwares, ainda existem características da época da crise, como projetos atrasados, erros de estimativa de custos e de tempo, que tornam o processo, ainda que sistematizado, passível de muitos erros.

Metodologia de desenvolvimento é um conjunto de práticas recomendadas para o desenvolvimento de softwares, sendo que essas práticas, geralmente, passam por fases ou passos, que são subdivisões do processo para ordená-lo e melhor gerenciá-lo (Sommerville, 2000).

4. Modelos e Metodologias de Desenvolvimento Tradicionais

O modelo de ciclo de vida clássico ou cascata, descrito pela primeira vez por Winston W. Royce em 1970 (Royce, 1970; Gustafson, 2002), que denominaremos a partir deste ponto como tradicional, caracteriza-se pelo seu caráter preditivo, prescritivo, seqüencial, burocrático, rigoroso, orientado a processos e dados, formais e controlado, que tem o sucesso alcançado desde que esteja em conformidade com o que foi planejado (Pressman, 2006; Paula Filho, 2003).

As metodologias consideradas tradicionais, também chamadas de “pesadas”, têm como característica marcante serem divididas em etapas e/ou fases. Essas fases são muito bem definidas e englobam atividades como Análise, Modelagem, Desenvolvimento e Testes.

Cada fase concluída gera um marco, que geralmente é algum documento, protótipo do software ou mesmo uma versão do sistema.

Muitas “metodologias” pesadas são desenvolvidas no modelo em cascata o que dificulta o controle do projeto, pois a cada alteração em determinado ponto do projeto,

como os requisitos, será necessário uma volta ao início do mesmo para alteração de documentação ou outro marco [PRESSMAN 02].

O foco principal das metodologias tradicionais é a previsibilidade dos requisitos do sistema, que traz a grande vantagem de tornar os projetos completamente planejados, facilitando a gerência do mesmo, mantendo sempre uma linha, caracterizando o processo como bastante rigoroso [OLIVEIRA].

4.1 Modelos de Processo de Software

4.1.1 Modelo Seqüencial Linear

Esse modelo, proposto em 1970, é também conhecido como Modelo em Cascata onde as fases definidas são sistematicamente seguidas de maneira linear [PRESSMAN 02].

4.1.2 Modelo de Prototipagem

A característica principal desse modelo é gerar protótipos do sistema com definições de requisitos dadas pelo cliente. Essas definições geram documentos que, por sua vez resultam no protótipo. Esse protótipo é então testado pelo cliente para validar suas funcionalidades [PRESSMAN 02].

4.1.3 Modelo RAD – Rapid Application Development

Sua principal característica é que o produto de software seja desenvolvido em componentes, pois a reutilização de código permite que a equipe de desenvolvimento possa desenvolver um sistema completamente funcional em pouco tempo.

4.1.4 Modelo Incremental

Esse modelo é a "versão" evolucionária do Modelo Sequencial Linear. Apenas assume que o software desenvolvido pode sempre crescer e agregar novas funcionalidades, sendo que cada uma dessas funcionalidades, ou o conjunto delas, será desenvolvido por um incremento e esse incremento segue todas as etapas descritas no modelo linear [PRESSMAN 02].

4.1.5 Modelo Espiral

O Modelo Espiral é um modelo iterativo, como o modelo de prototipagem, e sistemático como o Modelo Linear. É muito utilizado no desenvolvimento de softwares em conjunto com o paradigma da orientação a objetos, onde o desenvolvimento em módulos, somado ao processo de integração, se encaixa nos conceitos do paradigma [PRESSMAN 02].

4.2 Metodologias de Desenvolvimento Tradicionais ou "Pesadas"

4.2.1 RUP – Rational Unified Process

Para melhor analisarmos as características das metodologias pesadas, veremos o principal framework utilizado comercialmente no mercado, o RUP. O maior expoente das metodologias "pesadas" é o RUP, um Processo de Engenharia de Software criado pela Rational Software Corporation e oriundo do Processo Unificado – UP – para descrever como desenvolver um software usando técnicas comerciais com o objetivo de aumentar a qualidade dos softwares gerados pela empresa desenvolvedora.

O RUP é aplicável em classes de projetos variadas, sendo considerado um framework genérico para os processos de desenvolvimento, a contar que este deve ser

configurado de acordo com o tamanho e a necessidade de cada projeto e/ou empresa desenvolvedora.

O RUP é uma metodologia iterativa, ou seja, trabalha em ciclos de desenvolvimento.

Isso provê vários benefícios como [KRUCHTEN 00]: Gerenciamento de Requisitos, Integração dos Elementos, Gerenciamento de Riscos, Testes:

4.2.1.1 Arquitetura do RUP

O RUP é dividido, primariamente, em fases, sendo que essas fases podem conter várias iterações, que são ciclos onde cada etapa acontece. As 4 (quatro) fases do RUP são KRUCHTEN 00]: Concepção, Elaboração, Construção, Transição.

O RUP também se utiliza de Artefatos, que são produtos utilizados durante o desenvolvimento do projeto. Usualmente os artefatos são documentos (relatórios de riscos), modelos (Casos de uso) e modelo de elementos (Diagrama de Classes).
Esses artefatos são agrupados em disciplinas (workflows), já que cada uma produz um conjunto de artefatos diferentes.

Como o RUP é um processo interativo, as atividades repetidas podem voltar a ocorrer, principalmente se tratando de alterações em artefatos de diagramas e modelagem, pois estes devem ser alterados a cada mudança na especificação de requisitos.

Por fim pode-se dizer que o Ciclo de vida do RUP é baseado e suas 4 (quatro) fases. O ciclo é, em sua visão menor, analisando as cada fase, interativo, já que implementam perfeitamente o conceito de interação, porém numa visão total do processo, quando leva-se em consideração todos os passos e a seqüência exata do processo, é considerado em série. Novas versões e *releases* (artefatos) podem ser desenvolvidos e agregados ao projeto original, encaixando-se no modelo incremental.

5. Metodologias de Desenvolvimento Ágeis

Os métodos ágeis caracterizam-se pelo seu caráter adaptativo e orientado para pessoas. São várias as metodologias que são classificadas como ágeis, em que se destacam o XP (Extreme Programming) e Scrum.

Segundo Teles (2004), os Processos Ágeis de Desenvolvimento com partilham a premissa de que o cliente aprende sobre suas necessidades, na medida em que é capaz de manipular o sistema que está sendo produzido e, com base no feedback do sistema, ele reavalia as suas necessidades e prioridades, gerando mudanças que devem ser incorporadas ao software. O aprendizado é importante, porque permite que o cliente direcione o desenvolvimento de modo que a equipe produza sempre aquilo que tem o maior valor para o seu negócio.

As abordagens ágeis compartilham, na sua essência, o processo de desenvolvimento centrado nas pessoas, orientado para a obtenção de artefatos a partir de iterações, o que, conseqüentemente, impõe o caráter adaptativo durante todo o ciclo de desenvolvimento.

Para melhor compreensão do que seja o desenvolvimento ágil, os membros da Agile Alliance (http://www.agilemanifesto.org/principles.html) apresentaram à comunidade os seguintes princípios:

- A maior prioridade é satisfazer o cliente, a partir de entregas de produtos de software de efetivo valor em tempo hábil e continuamente;

· Acatar as necessidades de mudanças em qualquer estágio do processo de desenvolvimento, pois o software deve prover efetiva vantagem competitiva ao cliente;
· Os artefatos ou produtos de software devem ser entregues no menor tempo possível e funcionando;
· Todos os envolvidos devem trabalhar efetivamente juntos, diariamente, durante o projeto;
· A confiança deve nortear o processo de desenvolvimento e a construção dos produtos de software deve estar alinhada aos indivíduos mais motivados;
· Privilegiar a comunicação direta, cara a cara;
· A percepção da evolução do projeto se dá através de produtos de software entregues e funcionando;
· Os stakeholders devem ser capazes de manter o ritmo necessário durante o processo de desenvolvimento;
· A agilidade é conquistada pelo constante aprimoramento técnico e a conseqüente qualidade do projeto;
· A simplicidade deve ser a característica marcante dos projetos;
· Decorrem de equipes organizadas as melhores práticas;
· Avaliações regulares devem ser empreendidas para tomadas de decisões para o ajustamento ou adaptações necessárias.

5.1 XP – eXtreme Programming

eXtreme Programming – XP – é uma abordagem deliberada e disciplinada para o desenvolvimento de software. A XP vem sendo bastante utilizada e vem tomando espaço que antes pertencia a metodologias tradicionais, como RUP – Rational Unified Process [BECK 99].

Essa metodologia, por se enquadrar nas metodologias ágeis, carrega consigo todas as características citadas anteriormente. Desde o foco principal na satisfação do cliente até assumir que as mudanças nos requisitos sempre vão ocorrer, levando esse ponto em consideração.

Para delinear o processo de desenvolvimento, a XP foi estruturada segundo algumas ráticas que permitem aumentar a qualidade do software gerado, aumentando assim a satisfação do cliente.

Essas regras e práticas se dividem em 4 (quatro) categorias que são: Planejamento, Designing, Codificação e Testes [BECK 99], [WELLS 04], [XISPE 04].

A XP tem como um dos valores principais a comunicação, portanto durante a todo o decorrer do projeto, é muito importante que haja uma interação entre a equipe de desenvolvimento e o cliente, sempre que possível e necessário.

5.2 SCRUM

O Scrum foi criado por Jeff Sutherland e Ken Schwaber, na década de 90 (Rabello, 2006, p.68); o termo foi inspirado no jogo de rugby.

O Scrum orienta-se por três princípios: a visibilidade, a inspeção e a adaptabilidade (Schwaber, 2004, p. 3). As coisas devem estar visíveis a todos os envolvidos no desenvolvimento, a inspeção deve ser uma ação corrente e, conseqüentemente, as ações para adaptação do produto de software devem ser realizadas.

Vale salientar que essa abordagem é semelhante ao ciclo PDCA (Plan, Do, Check, Action), evidenciando que esse método fundamenta-se em valores já conhecidos e considerados válidos pela comunidade.

São consideradas iniciativas essenciais do Scrum a comunicação, o trabalho em equipe, a flexibilidade, as entregas de produtos de software funcionando, sendo que essas entregas caracterizam-se por serem entregas que, a cada versão, possuem novas funcionalidades ou alguma melhoria que tenha sido introduzida pela equipe (Pressman, 2006, p. 69).

6. Analise Comparativa Entre Metodologia Tradicional e Ágil

Depois de analisar individualmente cada um dos principais expoentes (características) das metodologias tradicionais, com o RUP (Rational Unified Process), e das metodologias ágeis, com o XP (eXtreme Programming), analisaremos, comparativamente, ambas as abordagens de processos.

Esses dois processos que serão analisados têm pontos em comum que permitem que a comparação seja possível. Esses processos são uma seqüência de atividades, realizadas por papéis, que geralmente são indivíduos ou a equipe, a fim de gerar artefatos ou produtos para serem entregues aos clientes.

O RUP, como dito antes, é divido em fases, essas fases são divididas em iterações e essas iterações podem conter vários ciclos. Uma interação do RUP leva entre duas semanas e 6 seis meses. Considerando que em um projeto o número de iterações recomendadas fica entre 3 (três) e 9 (nove), pode-se concluir que a faixa de tempo que um projeto orientado pelo processo dessa metodologia pesada leva de seis semanas a cinqüenta e quatro meses, para sua conclusão (Smith, 2001).

Analisando os valores padrões para a eXtreme Programming, considera-se que as iterações levem 2 (duas) semanas para serem concluídas. Os *releases* na XP são usualmente programados de 2 (dois) em 2 (dois) meses, visto que são determinados em reuniões com os clientes e a equipe. Esses *releases* da XP equivalem as iterações do RUP (Smith, 2001).

Na XP antes dos ciclos e *releases* começarem é necessária uma análise prévia do escopo do projeto para definir se ele é viável ou não, determinar custo, esforço e tempo, esse momento na eXtreme Programming tem a mesma razão que a fase de Concepção no RUP (Smith, 2001).

Comparando o ciclo de vida de ambas as metodologias é possível afirmar que a XP entrega os *releases* mais rapidamente e com uma freqüência maior, exatamente como é citado nas práticas da metodologia, pois não perde muito tempo nas fases de exploração e planejamento, como perde o RUP nas duas primeiras fases. Isto se deve ao fato de que este faz uma análise de risco, que consome algumas iterações na fase de Elaboração, garantindo que a arquitetura desenvolvida não seja falha, impedindo que no futuro seja necessária alguma alteração que impacte em retrabalho e perda de tempo muito grande.

Vendo as comparações conclui-se que ambas as abordagens têm pontos fortes e fracos, sendo que alguns pontos antes exclusivos das metodologias ágeis passam a ser englobados nas metodologias tradicionais e o caminho inverso também é válido, fazendo com que haja uma espécie de metodologia híbrida entre elas.

Enquanto o principal foco do RUP são os grandes projetos, onde se necessitam de especializações e especificações maior do projeto, a eXtreme Programming tem seu foco nos pequenos projetos, onde o excesso de documentos e burocracia recomendada pelo RUP é considerado perda de tempo (Smith, 2001).

7. Conclusão

A engenharia de software surgiu com o intuito de solucionar problemas que, pela sua magnitude, foram denominados de crise de software. Independentemente de ser uma crise ou problema crônico, o fato é que as dificuldades persistem (Pressman, 2002), e o desenvolvimento de software constitui-se numa tarefa muitas vezes árdua e que tem levado a muitos ao insucesso. Esses fatos justificam o grande interesse de todos os stakeholders, particularmente a comunidade de desenvolvedores em buscar abordagens metodológicas que possam ajudar a eliminar ou minimizar significativamente as dificuldades enfrentadas a cada projeto.

Assim, características como formalidade, ênfase na documentação, cumprimento rigoroso das fases estabelecidas e produtos definidos, características da abordagem tradicional, não concorrem com a abordagem ágil, cujo conjunto de valores e princípios serve para resolver problemas cujas especificidades exigem uma ou outra abordagem.

Como qualquer projeto de desenvolvimento de aplicações, independente de qual metodologia utilize, o foco sempre será a qualidade final do produto e a satisfação do cliente.

Para chegar a tal ponto, os processos foram desenvolvidos, porém nunca um deles trará satisfação total somada com qualidade inquestionável, seja em qualquer parte do projeto ou iteração de um processo. Invariavelmente, em algum ponto pecar-se-á, ou por causa do processo ou não.

O comparativo do trabalho ajudará uma melhor escolha quanto a qual metodologia deva ser utilizada em determinados projetos, levando sempre em consideração o foco do tipo de projeto que será desenvolvido. Tudo isso com o objetivo de reduzir a possibilidade de erros que venham comprometer a qualidade do projeto e a satisfação do cliente.

8. Referência

Oliveira, S. R. B, Rocha, T. A.,(2004) Vasconcelos, A. M. L., "Adequação de Processos para Fábricas de Software", Anais do Simpósio Internacional de Melhoria de Processos de Software – SIMPROS, São Paulo.

Pressman, R. S., (2002) "Engenharia de Software", 5ª Ed.,Makron Books.

Sommerville, I.,(2000) "Software Engineering", 6th edition, Addison-Wesley.

Wells, D., (2010) Disponível em http://www.extremeprogramming.org, Visitado em 25/08/2010.

XISPE, X., (2010) Disponível em http://www.xispe.com.br, Visitado em. 25/08/2010.

Neto, Oscar N. S.(2010) "Análise Comparativa das Metodologias de Desenvolvimento de Softwares Tradicionais e Ágeis" Disponivel em http://www.cci.unama.br/margalho/portaltcc/tcc2004, Visitado em 25/08/2010.

Beck, K.,(1999) "Extreme Programming Explained: Embrace Change", 1st Edition, Addison-Wesley.

Smith, J.,(2001) "A Comparison of RUP and XP", Rational Software White Paper.

Kruchten P.,(2003) "The Rational Unified Process" Editora Ciência Moderna. Rio de Janeiro.