Modelos de processo de evolução e manutenção de software no contexto dos métodos ágeis

Saulo Eduardo Galilleo Souza dos Santos[1], Adicinéia Aparecida de oliveira[2]

[1]Mestrando do Programa de Pós-Graduação em Ciência da Computação – Universidade Federal de Sergipe. Professor do Inst. Federal de Sergipe. e-mail: saulo.galilleo@ifs.edu.br.
[2]Professora do Programa de Pós-Graduação em Ciência da Computação – Universidade Federal de Sergipe. e-mail: adicineia@gmail.com.

Resumo: Atualmente há uma grande tendência na utilização de metodologias ágeis para desenvolvimento de software. Processos bem definidos e equipes estruturadas com metodologias eficientes têm proporcionado a construção de software com qualidade. Mas, ainda existe uma grande questão, boa parte dos esforços em novas metodologias focam o método de desenvolvimento, a preocupação com o processo de manutenção não tem tido o espaço necessário. Nesta lacuna, o presente trabalho apresenta os modelos de processo de manutenção de software, a sistemática destes modelos, bem como aborda os métodos ágeis e suas principais características. As características dos métodos ágeis são destacadas no trabalho para realizar um paralelo com a manutenção de software. E por fim são apresentados os resultados na análise dos modelos de processo de manutenção de software e a compatibilidade destes com as características dos métodos ágeis.

Palavras–chave: manutenção de software, métodos ágeis, modelo, processo

1. INTRODUÇÃO

Ao longo dos anos, os modelos de processo de manutenção de software têm evoluído no sentido de minimizar o custo do produto de software. Atividades de manutenção e evolução do software são inevitáveis, quase todo software estimula pedido de melhoria e mudanças. (BENETTI, 2000, p2).

Manter o software é uma das atividades previstas no processo do software, pois este é definido como o conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para conceber, desenvolver, implantar e manter um produto de software. (FUGGETA, 2000, p2). Desta forma, tecnologias que proporcionem um novo método ou técnica para manutenção e evolução de software podem gerar reduções de custo no processo de software.

No contexto do desenvolvimento de software têm surgido diversas técnicas e metodologias para construção de software de qualidade. Muitas destas técnicas focam nas pessoas envolvidas, nos processos e em modelos de iterações com *stakeholders*.

O Modelo de desenvolvimento incremental e iterativo amadureceu ao longo de pesquisas durante anos. Nos anos 2000, diversos projetos relataram sucessos na implantação do desenvolvimento incremental iterativo.

Neste trabalho será apresentado um estudo de modelos de processo de manutenção de software, aplicando uma visão crítica quanto à compatibilidade destes modelos com as características do processo de desenvolvimento ágil.

O artigo está organizado da seguinte forma, Seção 2 define modelos de processo de manutenção de *sofware*. Características dos métodos ágeis são apresentadas na Seção 3. A seção 4 apresenta a resultados e discussões. A seção 5 apresenta trabalhos relacionados e a última seção apresenta a conclusão do trabalho.

2. MODELOS DE PROCESSOS DE MANUTENÇÃO DE SOFTWARE

A manutenção de software é uma atividade ampla, o processo de manutenção de software inicia quando o produto final de software é entregue, a partir do momento que o software torne-se operacional. Esta atividade abrange correção de erros, melhoria, adição e exclusão de recursos, adaptação as mudanças de requisitos de dados e ambientes operacionais, bem como a melhoria de desempenho, usabilidade ou qualquer outro atributo de qualidade. (CANFORA, 2000, p5).

ISBN 978-85-62830-10-5

O termo manutenção de software é definido pela norma IEEE 1219, como a modificação de um produto de software depois da entrega para corrigir falhas, para melhorar o desempenho ou outros atributos, ou para adaptar o produto a mudanças do ambiente. (IEEE1219, 1998, p9).

O processo do desenvolvimento de software tem necessidade de uma boa documentação, existe uma relação direta entre o nível de maturidade e redução de custos, quanto maior a maturidade no desenvolvimento do software maior a economia. Modelos de processo de software fornecem as operações necessárias e detalhes de entradas e saídas às operações. (IEEE1219, 1998, p14).

Os modelos de processo de manutenção de software serão destacados adiante no sentido de identificar processos eficazes. Desta forma, a identificação das principais características permite que seja analisada a utilização quanto à continuidade do desenvolvimento do software baseado em uma metodologia ágil.

Existem muitos modelos de processo de manutenção de software, os modelos apresentados possuem uma grande aceitação na indústria e na comunidade acadêmica. Destarte, são abordadas as principais idéias que norteiam os seguintes modelos de manutenção de software: Boehm, Quick-Fix, Iterativo de Melhoria e Orientado a reuso.

2.1. Modelo Boehm

Em 1983, Boehm propôs um modelo de processo para manutenção do software baseado em princípios e modelos econômicos (AGGARWAL, 2007, p19). Neste modelo o processo de manutenção é representado através de um ciclo fechado conforme a Figura 1.

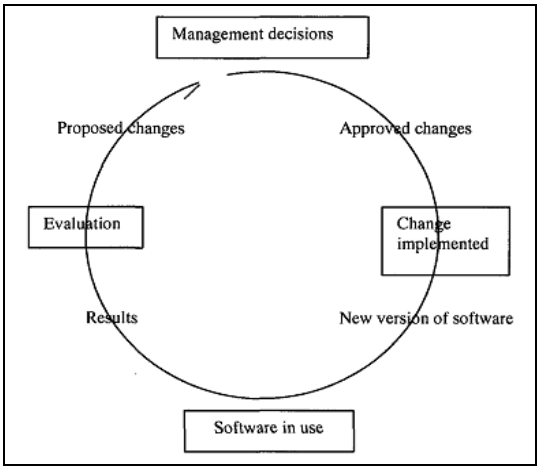

Figura 1 – Modelo de Boehm. (AGGARWAL, 2007).

O Modelo apresentado por Boehm possui uma etapa para o gerenciamento de decisões, nesta etapa são realizadas estratégias particulares e avaliação do custo benefício para um conjunto de alterações propostas. As propostas de alterações aprovados, seguem o ciclo para implementação das mudanças, geração da nova versão, utilização, resultados e avaliação.

Baseado em modelos econômicos, o modelo de processo de Boehm valoriza a decisão econômica para melhorar a produtividade de manutenção. Este modelo reflete a economia de investimentos e a relação entre os ganhos em três fases. A primeira fase, a de investimento, com baixa entrada de recursos e poucos benefícios, correspondente a um produto de software recém lançado que possui alta exigência de correções e melhorias de emergência obrigatórias. A segunda fase, de alto retorno, onde é visível o aumento do benefício do produto de software e os problemas inicias são resolvidos. E por último a fase de retornos decrescentes, onde a taxa de aumento de benefício cumulativo diminui, nesta fase mudanças radicais tornam-se menos custosas e menos eficazes também.

2.2.Modelo *Quick-Fix*

ISBN 978-85-62830-10-5

O *Quick-Fix* é considerado na literatura um modelo *ad-hoc* para a manutenção de software, onde ao acontecer o problema é realizada a tentativa de corrigir o quanto antes, dispensando análise detalhada dos efeitos a longo prazo. A Figura 2 representa o modelo *Quick-Fix.*

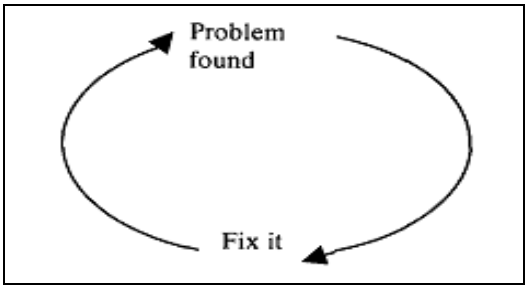

Figura 2 – Modelo *Quick-Fix*. (GRUBB, 2003)

O modelo apresenta uma realização da tarefa de forma rápida e mais barata, utilizado largamente em situações onde existe pressão de prazos e recursos. Porém o custo posterior de manutenção pode não compensar o benefício imediato, pois problemas complexos e caros podem ocorrer devido manutenções mal documentadas e executadas. (GRUBB, 2003, p76).

Existem estratégias de adotar técnicas de Quick-Fix em outros modelos mais sofisticados, no geral o modelo apresenta limitações, mas que por diversas vezes reflete o ambiente do mundo real.

2.3. Modelo Iterativo de Melhoria

A proposta do modelo Iterativo de Melhoria é baseada na premissa de que, alterações em um sistema, ao longo de sua vida é um processo iterativo.(GRUBB, 2003, p84). O modelo iterativo de melhoria é adequado para sistemas que possuem vida longa e evoluem com o passar do tempo. Este modelo apresenta um custo inicial alto, porém os benefícios são sentidos a longo prazo. O modelo é composto por um ciclo de três fases, conforme a Figura 3.

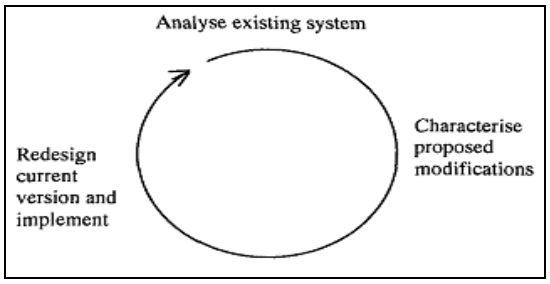

Figura 3 - Modelo Iterativo de Melhoria. (GRUBB, 2003).

O modelo Iterativo de Melhoria é construído baseado no modelo de processo iterativo do ciclo de vida do software. Requer que a cada rodada de iteração exista um documento completo da propagação da mudança e reprojeto do software.

Este modelo é caracterizado por três fases, Análise do sistema existente, caracterização da proposta de modificação, redesenho e implementação. As mudanças nesse modelo são propagadas por um conjunto de documentos e o sistema redesenhado. (GRUBB, 2003, p85).

O modelo iterativo de Melhoria pode conter de certa forma, características de outros modelos, como por exemplo, o *Quick-Fix,* onde uma solução rápida pode ser realizada. Assim sendo percebe-se o quanto adaptável o modelo é de forma que este evolua em uma melhoria constante.

2.4. Modelo Orientado a Reuso

ISBN 978-85-62830-10-5

O modelo orientado a reuso é baseado no princípio de que a manutenção pode ser vista como uma atividade que envolve a reutilização de componentes de programas existentes. (GRUBB, 2003, p85). O modelo de reutilização possui quatro principais etapas:

- Identificação das partes do sistema antigo candidatos a reutilizar.
- Entendimento das partes do sistema.
- Modificação das partes do sistema antigo adequando para os novos requisitos.
- Integração das partes alteradas no novo sistema.

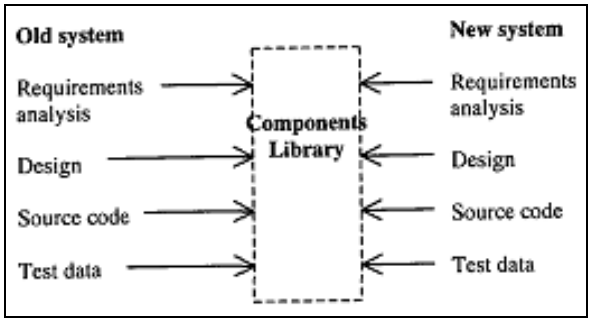

Figura 4 - Modelo orientado a reuso. (GRUBB, 2003)

No modelo orientado a reuso a premissa de reutilização é baseada na utilização de componentes, conforme Figura 4, onde é necessário que estes componentes possuam documentações completas e detalhadas para a classificação de componentes e análise de possíveis mudanças.

A abordagem de componentização permite a reaplicação de conhecimentos de um sistema em um sistema semelhante, no sentido de reduzir os esforços de desenvolvimento e manutenção do sistema.

3. MÉTODOS ÁGEIS

Os métodos ágeis surgiram como uma reação a forma tradicional de desenvolvimento de software, como também da necessidade de uma alternativa orientada a documentação do processo de desenvolvimento do software.

Os métodos ágeis são atualmente uma coleção de diferentes técnicas e práticas, que compartilham alguns valores e princípios básicos. Muitos destes métodos são baseados na técnica de melhoria iterativa uma técnica que foi introduzida em 1975 (COHEN, 2004, p3).

O movimento ágil foi estabelecido contendo conceitos e princípios comuns, desenvolvido e publicado por profissionais de software e consultores em 2001. O desenvolvimento ágil descreveu abordagens de desenvolvimento, onde implica nos valores centrais a serem honrados conforme a seguir (ABRAHAMSSON, 2002, p11):

- Indivíduos e interações mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos; e,
- Responder a mudanças mais que seguir um plano.

O primeiro valor destacado afirma que o movimento ágil enfatiza o relacionamento da comunidade de desenvolvedores de software e pessoas com regras que refletem no contrato, em oposto a processos institucionalizados e ferramentas de desenvolvimento. O segundo valor destaca o objetivo do software em funcionamento, desenvolvedores insistem em manter códigos simples e técnicas que permitam avançadas mudanças, em oposto a documentação abrangente, contudo, é mais simples o entendimento do sistema com este em funcionamento do que apenas com documentação. O terceiro valor destaca o relacionamento e cooperação entre os desenvolvedores e clientes é preferencial sobre restrições de contratos. O quarto valor destaca que o grupo de desenvolvimento, compreendendo

ISBN 978-85-62830-10-5

ambos, desenvolvedores do software e representantes do cliente, deve ser bem informado, competente e autorizado para considerar possíveis ajustes que surgem durante o processo do ciclo de vida de desenvolvimento do software.

2.2.1. Caractarísticas

Algumas características em conjunto traduzem a real proposta do processo de desenvolvimento ágil, o amadurecimento de processos existentes com foco no cliente e em agilidade alinham estas características como necessárias em um processo de desenvolvimento de software ágil. Em (MILLER, 2001, p1-3) são relacionadas características do processo de desenvolvimento ágil conforme a seguir:

Modularidade: É um elemento fundamental de qualquer bom processo. A modularidade permite um processo ser dividido em componentes chamados de atividades. Um processo de desenvolvimento de software prescreve um conjunto de atividades capazes de transformar a visão do sistema de software em realidade.

Iterativo: Processos de software ágil percebem coisas erradas antes que seja levada a frente, concentram-se em ciclos curtos e dentro de cada ciclo um certo conjunto de atividades é concluída, os ciclos são iniciados e concluídos em questão de semanas. Portanto, o ciclo curto é repetido muitas vezes para refinar o processo.

Prazos: Iterações tornam a unidade perfeita para o planejamento do projeto de desenvolvimento do software, podemos definir limites de tempo em cada iteração e programá-los em conformidade.

Parcimônia: Processos de software ágeis focam a parcimônia, exigem um número mínimo de atividades necessárias para mitigiar os riscos e alcançar seus objetivos.

Adaptativo: Durante uma iteração, novos riscos podem aparecer, o que requer algumas atividades que não foram planejadas. O processo ágil é adaptado para atacar esses novos riscos encontrados. Se não for possível alcançar a meta com as atividades previstas durante a iteração, novas atividades podem ser adicionadas para permitir atingir o objetivo.

Incremental: Um processo ágil não tenta construir todo um sistema de uma só vez, ao contrário disso, ele cria partições não triviais do sistema gerando incrementos eu podem ser desenvolvidas em paralelo. Cada unidade de incremento é testada de forma independente, quando um incremento é completado e testado, este é integrado ao sistema.

Convergente: Convergência, pois afirma que métodos ágeis atacam todos os riscos, como resultado, o sistema torna-se mais próximo da realidade que se busca a cada iteração. Riscos são atacados de forma proativa, o sistema é entregue em incrementos, tudo é feito para garantir o sucesso o mais rápido possível.

Orientado a Pessoas: Processos ágeis favorecem pessoas acima de processos e tecnologias, evoluem através de adaptações de maneira orgânica. Desenvolvedores estão habilitados a aumentar sua produtividade, qualidade e desempenho.

Colaborativo: Processos ágeis promovem comunicação entre os membros da equipe. Comunicação é ponto fundamental em qualquer projeto de desenvolvimento de software. Ao desenvolver produtos particionados, entender como as peças se encaixam é vital para a criação do produto acabado, este modelo exige integração com comunicação simples enquanto incrementos estão sendo desenvolvidos em paralelo.

O desenvolvimento incremental iterativo é característica chave dos processos ágeis, neste modelo os requisitos podem ser introduzidos, modificados ou removidos em sucessivas iterações. A flexibilidade introduzida com o modelo proporcionou ganhos significativos nos prazos de entrega do software como também no custo.

4. RESULTADOS E DISCUSSÃO

A análise dos modelos foi realizada em levantamento bibliográfico, objetivando validar a compatibilidade dos modelos de processo de manutenção de software quanto às características dos métodos ágeis indicadas na seção 2.2.1.

As seguintes características dos métodos ágeis foram analisadas: modularidade, iterativo, prazo, parcimônia, adaptativo, incremental, convergente, orientada a pessoas e colaborativo.

ISBN 978-85-62830-10-5

Baseado nas características do processo de desenvolvimento ágil foi realizado uma análise, a presença de uma determinada característica em um modelo de processo indica que o modelo atende a uma característica do processo. Desta forma, o Quadro 1 representa a análise dos modelos apresentados na Seção 2.1 deste trabalho *versus* as características apresentadas na Seção 2.2.

Quadro 1 – Análise dos modelos de processo de manutenção de software *versus* características dos métodos ágeis.

Característica	Modelo de Boehm	Modelo Quick-Fix	Modelo Iterativo de Melhoria	Modelo Orientado a Reuso
Modularidade	NA	NA	AT	AT
Iterativo	NA	NA	AT	NA
Prazo	NA	AT	AT	NA
Parcimônia	AT	NA	AT	AT
Adaptativo	NA	NA	AT	NA
Incremental	NA	NA	AT	NA
Convergente	SC	SC	AT	SC
Orientado a Pessoas	NA	NA	NA	NA
Colaborativo	SC	SC	SC	SC

NA= Não atende; AT = Atende; SC= Sem classificação.

O quadro apresentado traduz o quanto os modelos de manutenção de software atendem as características dos métodos ágeis. Nesta análise destacam-se níveis de compatibilidade bastante distintos ente os modelos de manutenção de software apresentados. Temos o modelo iterativo de melhoria com uma maior presença de sigla 'AT', referindo-se ao atendimento a característica.

Com o resultado apresentado foi possível criar a tabela 1, que representa o nível de atendimento as característica dos métodos ágeis por cada modelo de processo de manutenção de software.

Tabela 1 – Modelos de processo de manutenção de software *versus* características dos métodos ágeis.

Modelos	Percentual de atendimento as características
Boehm	11,11 %
Quick-Fix	11,11 %
Iterativo de Melhoria	77,78 %
Orientado a Reuso	22,22 %

O estabelecimento de nível de atendimento possibilita identificar a importância de modelos que preservem maior iteratividade e comunicação com *stakeholders.* Diversos projetos no decorrer dos anos falharam ou tiveram seus custos substancialmente aumentados devido ausência destas características.

A necessidade de planejamento é fundamental em um processo de manutenção de software, impactos e nível de manutenabilidade devem ser sempre considerados, modelos de processos que representem essas preocupações terão maior compatibilidade no contexto dos métodos ágeis.

5. TRABALHOS RELACIONADOS

ISBN 978-85-62830-10-5

Em (Stafford, 2003, p18) é destacada as regras da manutenção de software nos métodos de desenvolvimento, neste trabalho foram apresentadas as metodologias Rup e Scrum, no âmbito do desenvolvimento iterativo.

Em (Mello, 2009, p1) sua publicação apresenta que boa parte dos esforços, 75% a 85% de volume de software desenvolvido por organizações que não tem como objetivo o desenvolvimento de software, é a manutenção. Afirma também que as mesmas técnicas e métodos aplicados a projetos novos podem ser aplicadas também na manutenção, mas há peculiaridades que poderão ser mais bem exploradas se entendermos a questão da natureza da manutenção, e que adoção de métodos ágeis tem-se revelado mais vantajoso.

Segundo Chapin (2004, p1) existem melhorias existentes no processo de evolução de software com os métodos ágeis. A grande maioria dos métodos ágeis realiza manutenção de software que se qualifica como evolução do software. A utilização de métodos ágeis por desenvolvedores fazem com que o trabalho seja mais estimulante e gratificante.

Em (Rico, 2008, 3p), é demonstrado que métodos ágeis podem ser aplicados a manutenção de software, como também demonstrou a vantagem dos métodos ágeis em comparação aos ciclos de vida tradicionais como o modelo cascata. Destaca também que o uso de métodos ágeis podem diminuir os custos de manutenção de software por mais de três vezes, melhorar a produtividade em mais de três vezes, e aumentar a qualidade do software em até 67%. E demonstrou ainda que utilização dos Métodos Ágeis resultam em um tipo de documentação do software mais utilizado e preferido pelos mantenedores de software, e que consequentemente, melhora a qualidade do software.

6. CONCLUSÕES

O presente trabalho permitiu conhecermos a dimensão da manutenção do software em paralelo aos métodos ágeis. Os modelos de processo de manutenção de software avaliados, de certa forma, não possuem adequação a todas as características dos métodos ágeis.

No levantamento foi possível destacar uma maior compatibilidade as características dos métodos ágeis ao modelo de processo de manutenção iterativo de melhoria. A natureza de um modelo incremental e iterativo representa melhor a comunicação e evolução nos métodos ágeis e proporcionam minimizar problemas históricos de desenvolvimento de software. Comunicação é ponto fundamental em qualquer projeto de desenvolvimento, durante a manutenção do software também. Métodos ágeis também vêm sendo utilizados como processo de manutenção de software, métodos com maior foco no processo tem se destacado nesse sentido, como é o caso do *Scrum*.

REFERÊNCIAS

ABRAHAMSSON, P., SALO, O., RONKAINEN, J., WARSTA, J. **Agile software development methos – Review ans analysis.** *In* ESPOO 2002. VTT Publications 478.

AGGARWAL, K. K., SINGH, Yogesh. **Software Engineering**. 2. ed. New Age international Publishers, 2007.

CANFORA, G., CIMITILE, A., **Software Maintenance**. Universaty of Sannio, Faculty of Engineering at Benevento, Italy, 2000.

CHAPIN, N. **Agile Methods Contributions in Software Evolution**. *In* 20th IEEE International Conference on Software Maintenance, ICSM'04. 2004.

COHEN, D., LINDVALL, M., COSTA, P. **An Introduction to Agile Methods**. Advances in Computer, Vol 62. 2004, Elsevier.

FUGGETA, A. **Software Process: A Roadmap**. *In* ICSE '00 Proceedings of the Conference on The Future of Software Engineering, 2000.

ISBN 978-85-62830-10-5

GRUBB, P., TAKANG, A. A., **Software Maintenance – Concepts ans Pratice**, 2. ed. World Scientific, 2003.

IEEE1219. IEEE STD 1219: ***Standard for Software Maintenance***, 1998.

MELLO, M. C. F., **Para o ciclo de Manutenção, Agilidade já**. developerWorks, IBM, 2009, p1.

MILLER, G. G.,**The Characterístics of Agile Software Processes**. *In* 39th Conf. and Exibition on Technology of object-Oriented languages and Systems. IEEE, TOOLS'01, 2001.

RICO, D. F., ***Agile Methods and Software Maintenance***, 2008, 3p.

STAFFORD, J. A., **Software Maintenance – As Part of the Software Life Cycle**. Department of Computer Science , Tufts University, 2003, p18.

ISBN 978-85-62830-10-5