6
Referências Bibliográficas

AGILE ALLIANCE. **Manifesto for agile software development**. Disponível em <http://www.agilealliance.org/>. Acesso em 12 abr.2010.

AGILE ALLIANCE. **"What is Scrum".** Disponível em www.controlchaos.com. Acesso em 10 de maio 2010.

ASTELS, David. **Extreme programming; guia prático**. Rio de Janeiro: Campus, 2002.

BECK, Kent. *Extreme Programming explained*: Embrace change. Reading, Mass,. Addison- Wesley, 1999.

COHEN, David et al. **Agile software development: a DACS satate of art report.** NY, 2003. Disponível em www.thedacs.com/thecs/agile Acesso em maio 2010.

DAFLON, Leandro R. **Um Framework para a Representação e Analise de Processos de Software,** Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, 2004.

Feature-Driven Development Web Site, Disponivel em: http://www.featuredrivendevelopment.com/

FERREIRA, Eduardo. **Um modelo de Gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de software e nos princípios da produção enxuta,** Dissertação de Mestrado, Escola Politécnica, Universidade de São Paulo, 2007

FIORINI,S. **Engenharia de Software com CMM**, Rio de Janeiro: Brasport,1988.

FOWLER Martin. **Continuous Integration**. Disponível em http://<www.martinfowler.com>. Acesso junho 2010.

GOLDAMN, A; Kon. ; Silva, P.J. ; Yoder, J. W. Being Extreme in the Classroom: **Experiences Teaching XP. Journal of the Brazilian Computer Society**, p. 1-18, 2004.

GUINATO, **Produção e Competitividade: Aplicações e Inovações,** Ed: Adiel T. de Almeida & Fernando M.C. Souza, Edit da UFPE, Recife, 2000.

HUMPHREY, Watts S. **A Discipline for Software Engineering. Reading**, MA: Addison-Wesley, 1995.

ISNARD Marshall Junior, Agliberto Alves Cierco, Alexandre Varanda Rocha, Edmarson Bacelar Mota,Sérgio Leusin. **Gestão Da Qualidade** – 9. Ed. – Rio de Janeiro: Editora FGV, 204 p. (Gestão empresarial (Publicações FGV Management)), 2008.

ISOTTON NETO, E. **Scrumming: Ferramenta educacional para ensino de práticas do Scrum. Dissertação de Mestrado em Informática. Faculdade de Informática.** Porto Alegre: Pontifícia Universidade Católica do Río Grande do Sul,2008, 123 p.

PUC-Rio - Certificação Digital Nº 0913426/CA

JOHNSON,J. **Micro Projects Cause Constant Change.** The Standish Group International, 2001.

KERZNER, Harold. **Gestão de projetos: as melhores práticas.** Porto Alegre: Book,am, 2002.

KNIBERG, Henrik. **Scrum and XP from the Trenches**,2007

MANIFESTO AGIL. Disponível em: http://www.manifestoagil.com.br/principios.html.

MARTINS, J. **Técnicas Para Gerenciamento De Projetos De Software**. Rio de Janeiro: Brasport, 2007.

MNKANDLA, E; DWOLATZKY, B. **Defining Agile Software Quality Assurance**. In: International Conference on Software Engineering Advances (ICSEA06), IEE Computer, 2006.

NASCIMENTO, Gustavo. **Um modelo de referencia para o desenvolvimento ágil de software. Dissertação de Mestrado. Instituto de Ciências Matemáticas e de Computação**: USP- São Carlos, 2008.

NOCÊRA, Rosaldo de Jesús. **Gerenciamento de Projetos** – Teoría e Prática. Santo André, SP.: Ed. do Autor, 2009.

PELLEGRINELLI, S.; **BOWMAN, C. Implementing strategy through projects. Long Range Planning,** v.27, n.4, 1994

POPPENDIECK, M e POPPENDIECK, T, Engenharia **Enxuta de Software, 2003.** Disponivel em :http://www.poppendieck.com/

PMI - SP 2010, **Gerenciamento de Projetos**, Disponível em http://www.pmi.org.br.

PROJECT MANAGEMENT INSTITUTE, **Relatório Principal – Perspectiva Geral Estudo de Benchmarking em Gerenciamento de Projetos** – Chapters Brasileiros, 2009.

PROQUALITI, Qualidade Na Produção De **Software: O gerenciamento de Projetos de Software Desenvolvidos a Luz das Metodologías Ageis: Uma visão Comparativa,** 2005.

PROJECT MANAGEMENT INSTITUTE- **PMI. A Guide to the Project Managment Body of Knowledge (PMBOK, GUIDE) .**Fourth Edition, 2008.

SANTOS SOARES, M. **Comparação entre Metodologias Ágeis e Tradicionais para o desenvolvimento de software. Dissertação de Mestrado.** Conselheiro Lafaiete, MG: Universidade Presidente Antonio Carlos, 2004.

ROYCE, W. W. **Managing the development of large software systems: concepts and techniques.** Proc Wescon, 1970, p 1-8

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

SOFTWARE ENGINEERING INSTITUTE (SEI), **CMMI for Development, Software Engineering, University of Carnegie Mellon,** Versão 1.2, 2006. Disponível em http://www.sei.cmu.edu/cmmi. Acesso em julho 2010

TELES, V. **Extreme Programming, Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade.** São Paulo: Novatec Editora, 2004.

THE STANDISH GROUP. **The Chaos Report,** disponível em http:// www.standishgroup.com, 1994.

PUC-Rio - Certificação Digital Nº 0913426/CA

THE STANDISH GROUP. **The Chaos Report**, disponível em http:// www.standishgroup.com, 2006.

VARGAS, RICARDO VIANA, **Gerenciamento de Projetos: Estabelecendo Diferenciais Competitivos,** Rio de Janeiro – Brasport , 2005.

WOMACK, J. P; JONES, D. T. & ROOS, D. **A máquina que mudou o mundo.** Campus. 5a Edição. Rio de Janeiro, 1992.

PUC-Rio - Certificação Digital Nº 0913426/CA

7
Apêndice I : Princípios XP

Tabela 7.1 – Princípios XP

Trabalhar com os clientes	Os usuários devem cooperar fortemente com a equipe durante todo o processo de desenvolvimento de software. Eles definem, juntamente com a equipe de desenvolvimento a ordem da implementação das funcionalidades definidas para o sistema, contribuem na elaboração dos testes necessários para o sistema e também fornecem *feedback* constante sobre o sistema que está sendo construído.
Uso de metáforas para conceitos difíceis	Uma única estória deve guiar todo o desenvolvimento do software, indicando através de uma metáfora como o sistema funciona. Através desta metáfora todos podem entender os elementos básicos e relações existentes no projeto. Para tanto, um conjunto de palavras associadas a tal metáfora deve ser utilizado durante todo o decorrer do projeto.
Planejar	O planejamento serve para fornecer uma compreensão mútua para todas as partes de quanto tempo, aproximadamente, levará o projeto.
Reuniões curtas	Prática das **reuniões em pé**, realizadas diariamente, no início do expediente, nestas reuniões se discute a execução do trabalho no dia anterior, identificando e solucionando os problemas encontrados, e como dar seguimento ao trabalho no dia corrente. Ao final da reunião, um planejamento diário é decidido, indicando que funcionalidades deverão ser implementadas naquele dia e quem será o responsável por cada uma delas.
Testar primeiro	Para cada funcionalidade implementada, um conjunto de testes deve ser escrito com o intuito de verificar a integridade do que foi implementado. Os testes devem ser feitos de maneira exaustiva por funcionalidade antes que ela seja integrada ao restante do sistema. Depois, o sistema por completo também deve ser testado com o intuito de verificar se ele ainda continua funcionando corretamente com a nova funcionalidade que foi acrescentada.
Ser simples	Deve-se manter o projeto o mais simples possível, pensando-se apenas nas soluções necessárias para o momento e, desta maneira, implementando-se a opção mais simples que funcionar. Com o passar do tempo, no decorrer do projeto, pode-se ir simplificando o projeto continuamente.

PUC-Rio - Certificação Digital Nº 0913426/CA

Programar em pares	Toda a implementação do sistema deve ser realizada por duplas de programadores. Esta prática favorece a comunicação entre os membros da equipe de desenvolvimento e a discussão de ideias, um dos valores do XP. Além disso, os erros são menos frequentes, pois é possível que enquanto um componente do par esteja programando, o outro supervisione o código sendo produzido, apontando os erros.
Codificação dentro de padrões	Como o código é implementado por diferentes pessoas, a utilização de regras de codificação facilita o entendimento e a manutenção do código por todos. Esta prática garante que outras práticas, como o *refactoring* e código coletivo sejam executadas sem maiores dificuldades.
Fazer a propriedade coletiva	Evitar bloqueios exclusivos de gravação das classes para que outros membros de desenvolvimento possam modificar ou utilizar essas classes.
Integrar continuamente	A integração das diferentes unidades do sistema deve ser feita frequentemente, sempre após a execução do seu conjunto de testes. Esta prática garante que os conflitos entre as diferentes partes e o custo de integração sejam mínimos.
Fazer refactoring	Sempre que se percebe que se pode melhorar ou simplificar o projeto em determinada parte, deve-se fazê-lo. No entanto, as alterações podem introduzir erros indesejáveis. Assim, depois que o *refactoring* é realizado, todos os testes devem ser executados novamente para que os possíveis erros possam ser detectados e solucionados.
Fazer releases em incrementos pequenos	Fazer releases antecipados, frequentes, uma versão pequena de releases está entre um mês e menos de seis meses de trabalho, isto ajuda a planejar, e fornece feedback valioso.
Não se desgastar (40 horas de trabalho semanais)	A semana de trabalho de toda a equipe não deve ter uma carga horária maior do que quarenta horas, para evitar que os desenvolvedores se desgastem e permitir que eles estejam sempre aptos a executar as outras práticas.
Adotar as alterações	O principal conceito para um projeto XP, é que alteração é uma norma (nunca sabemos todo desde o inicio).

FONTE: Adaptado de Extreme Programming Guia Pratica (2002)

PUC-Rio - Certificação Digital Nº 0913426/CA

Apêndice II:
Responsabilidades XP

Tabela 0.1 – Responsabilidades XP: A equipe do cliente

Contadores de histórias	Aqueles que têm especialização no domínio, eles experimentaram primeiro o problema que o sistema deve solucionar. A equipe de desenvolvimento deve recorrer a estas pessoas para esclarecer requisitos.
Os aceitantes	Pessoas que agem em seu nome para executar os testes de aceitação
Os proprietários de ouro	Provedores de recursos para o projeto.
Planejadores	São as pessoas que veem as necessidades de distribuição do corpo de usuários pretendidos.
O chefão	Pessoa que lidera o sistema.

FONTE: Adaptado de Extreme Programming Guia Pratica (2002)

Tabela 0.2 – Responsabilidades XP: A equipe do desenvolvimento

O técnico	Em XP é o responsável por garantir o processo se desenrole normalmente, traz habilidades para aconselhar a equipe, a voz da razão nas crises.
O acompanhador	E o mecanismo de *feedback* para as nossas estimativas, avalia quanto tempo foi preciso para concluir o trabalho. È o responsável de comunicar as alterações para toda a equipe de desenvolvimento.
O facilitador	Aquele que na equipe de desenvolvimento tem as melhores habilidades de comunicação, seu papel é facilitar o relacionamento entre as duas equipes.
O arquiteto	Produzem e fazem o *refactoring* à arquitetura conforme a necessidade.

FONTE: Adaptado de Extreme Programming Guia Pratica (2002)

PUC-Rio - Certificação Digital Nº 0913426/CA

Apêndice III:
Etapas XP

Nas tabelas 7.4, 7.5, e 7.6 são descritas as etapas da XP explicadas por Astels (2002):

Tabela 0.1 – Etapas XP: Conceitualização do sistema

Criar visão do sistema	O cliente tem o direito de não ter que conceituar todo o sistema desde o inicio. • O cliente pode declarar uma visão sobre a finalidade da criação do sistema. • Podem-se usar metáforas para superar problemas de comunicação na descrição do que o cliente deseja que faça o software. • Podem-se usar metáforas para o desenvolvimento, para descrever a condição da organização, projeto ou código.
Escrever user *stories*	Trata-se de criar os requisitos do sistema pedindo que os usuários contem historias sobre como eles usarão o sistema. Quando se tiver todas as historias de todos os objetivos se terá a compreensão atual do cliente daquilo que deve ser realizado para criar o sistema desejado, isto é conhecido como ***deck do projeto.***
Escrever os testes de aceitação	È uma situação concreta que o sistema pode encontrar e que exibe aquele comportamento, então para cada user story deve haver pelo menos um teste de aceitação. Um teste de aceitação tem 3 partes: • Cenário: número mínimo de coisas que devemos fazer. • Operação: teste em si • Verificação
Solução simples	Precisa-se criar uma base de estimativas que seja comum entre os membros da equipe de desenvolvimento, assim desde que uma solução mantenha essas estimativas uniformes, não importa como ela deve ser implementada.
Prestar atenção as nossas palavras	O padrão de nomes de sistema alinha à equipe de desenvolvimento. A propriedade coletiva do código requer uma compreensão coletiva do domínio.

FONTE: Adaptado de Extreme Programming Guia Pratica (2002)

Tabela 0.2 – Etapas XP: Planejamento

Fornecer estimativas	Uma estimativa é a quantidade de tempo requerida para um único desenvolvedor implementar a história no software. O XP trabalha em equipe de duas pessoas, mais para fins de criação de uma boa estimativa finge-se que se escreve

PUC-Rio - Certificação Digital Nº 0913426/CA

	sozinho.
Planejar releases	Permite que a equipe do cliente tenha uma compreensão inicial do custo do projeto (se faz analise custo/ beneficio).
Planejar as iterações	A equipe do cliente deve selecionar as user stories mais importantes e as programe no seu orçamento. Assim uma primeira iteração vai servir de base para próximas iterações.
Planejamento tático	Este planejamento envolve situações do dia após dia. Está composto por atividades como reuniões de ponto de partida da iteração, reuniões em pé, e atividades de controle para acompanhar o projeto

FONTE: Adaptado de Extreme Programming Guia Pratica (2002)

Tabela 0.3 – Etapas XP: Desenvolvimento

Programação em pares	A XP propõe as seguintes atividades para este ponto: ✓ Desenvolver como uma conversação ✓ Pressão entre parceiros
Teste primeiro	Testar para saber si o sistema funciona, para saber si se trabalha em uma base de código estável, com o objetivo de saber si o sistema funciona, ficar longe do depurador. Deve-se testar: ✓ Antes de fazer o refactoring. ✓ Após de fazer o refactoring. ✓ Quando estiver implementando uma tarefa nova.
Projeto	Projetar em forma ágil, o que precisa no momento que precisa.
Código com intenção	Tornar o código do jeito mais claro possível. O desenvolvimento orientado aos testes é uma ótima maneira de fazer isso, escrever só o código necessário para passar aos testes.
Refactoring	Depois que o *refactoring* é realizado, todos os testes devem ser executados novamente para que os possíveis erros possam ser detectados e solucionados.
Integração implacável	Ter um build bem sucedido disponível: ✓ Acompanhamento de releases de todas as fontes é verificado. ✓ Tudo é compilado desde o inicio. ✓ O sistema está vinculado e distribuído. ✓ O sistema está inicializado, e o conjunto completo de testes é executado sem intervenções ou falhas.

FONTE: Adaptado de Extreme Programming Guia Pratica (2002)

PUC-Rio - Certificação Digital Nº 0913426/CA

Apêndice IV:
Níveis de Maturidade CMM

Tabela 0.1 – Níveis de Maturidade CMM

Nivel	Resumo	Caracterização	Tipo de Capacitação
INICIAL	O processo de software, não está documentado e, usualmente, sequer existe. Poucos processos estaveis existem ou são usados, e o sucesso depende de esforços individuais.	Programadores consideram-se artistas. Padrões não existem, ou tendem a ser ignorados. Ferramentas são usadas ao acaso, muitas vezes por iniciativa pessoal. Metodologias são praticadas informalmente. Resultados não são previsiveis. A coleta e análise de dados são ad hoc.	Processo Ad Hoc
REPETITIVO	Estão estabelecidos processos básicos de gerencia de projeto para planejar e acompanhar custos, prazos e funcionalidades. Compromisos são firmados e gerenciados e successos podem ser repetidos	Gerência de projetos estabelecida. Processo organizado. Alguns procedimentos tecnicos escritos. Acompanhamento da qualidade. Gerência de configuração inicial. O sucesso depende do gerente do projeto. Este nível apresenta as seguintes KPAs (Key Process Areas ou áreas chave de processo): Gerenciamento de Requisitos Planejamento de Projetos Acompanhamento e Supervisão de Projetos Gerenciamento de Subcontratação Garantia de Qualidade de Software Gerenciamento de Configuração	Processo Disciplinado
DEFINIDO	Tanto para as atividades de gerência básica como para as de engenharia de software, o processo de software é documentado, padronizado e integrado num processo único, chamado Processo de Software Padrão da Organização. Todos os projetos usam uma versão deste processo, adaptada às	Processos gerenciais e técnicos básicos bem definidos. Possibilidade de avaliação do processo. Ferramentas e metodologías padronizadas. Medições iniciais de desempenho. Inspeções e auditorias rotineiras. Testes padronizados. Gerência de configuração generalizada. Evolução controlada dos processos técnicos e gerencias básicos	Processo Padronizado e Consiste

PUC-Rio - Certificação Digital Nº 0913426/CA

	características específicas do projeto, contemplando o desenvolvimento e manuntanção de software.	KPAs deste nível: Revisões (peer review) Coordenação de Intergrupos Engenharia de Produto de Software Gerenciamento de Software Integrado Programa de Treinamento Definição do Processo da Organização Foco no Processo da Organização	
GERENCIADO	São coletadas medições detalhadas do processo de software são avaliados quantitativamente e são, também, controlados	Está estabelecido e em uso rotineiro um programa de medições. Está estabelecido um grupo de garantía da qualidade. A qualidade é planejada. A qualidade é rotineiramente avaliada e aprimorada KPAs deste nível: Gerenciamento da Qualidade do *Software* Gerenciamento Quantitativo do Processo	Processo Previsível
EMOTIMIZAÇÃO	E realizada rotineiramente a melhoria do processo como um todo. São realizados projetos piloto para a absorção e internalização de novas tecnologías.	Alto nivel de qualidade é alcançado rotineiramente. Melhoría continua. Alto nivel de satisfação dos clientes. KPAs deste nível: Gerenciamento da Mudança no Processo Gerenciamento da Mudança Tecnológica Prevenção de Defeitos	Processo como Melhoría Continua

Fonte: Fiorini pag 27 Engenharia de Software com CMM

PUC-Rio - Certificação Digital Nº 0913426/CA

Apêndice V:
Fator de Ajuste da Equipe de Trabalho Testada

FA

Mês	Dias Disponíveis	Dias Programados	Dias Trabalhados	Fator de Ajuste
Abril	18,0	19,3	23,4	0,19
Maio	21,0	20,1	26,0	0,21
Junho	21,0	16,5	20,1	0,16
Julho	21,0	24,3	21,5	0,17
Agosto	22,0	22,0	23,3	0,19
Setembro	20,0	20,0	23,0	0,19
Total	123,0	122,1	**Media**	**0,19**

FB

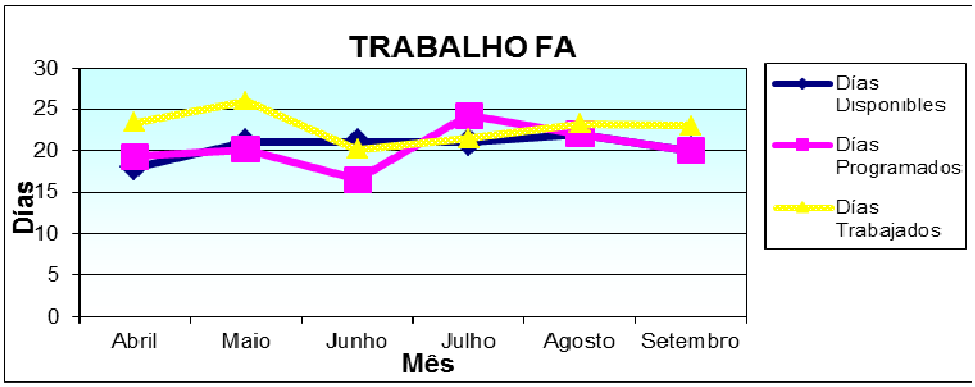
Mês	Dias Disponíveis	Dias Programados	Dias Trabalhados	Fator de Ajuste
Abril	18,0	18,3	20,3	0,16
Maio	21,0	21,3	25,1	0,20
Junho	21,0	19,0	22,6	0,18
Julho	21,0	17,0	18,0	0,15
Agosto	22,0	14,0	14,0	0,11
Setembro	20,0	20,0	22,3	0,18
Total	123,0	109,5	**Media**	**0,17**

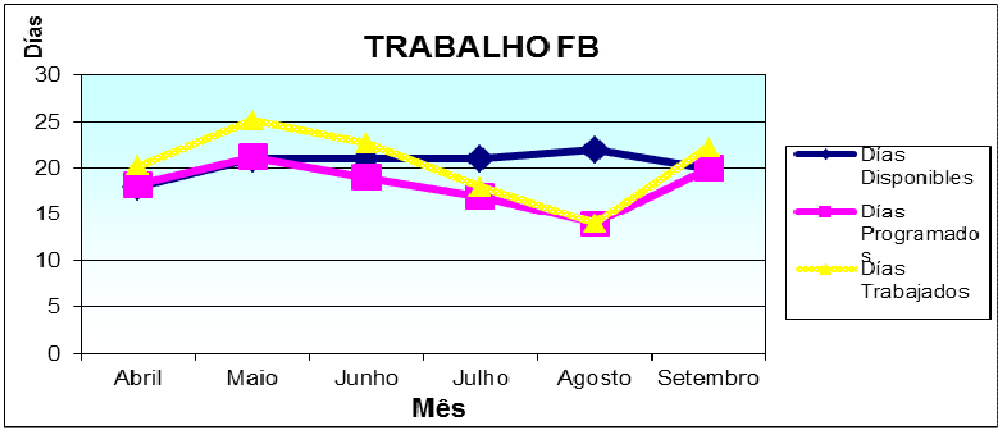
FC

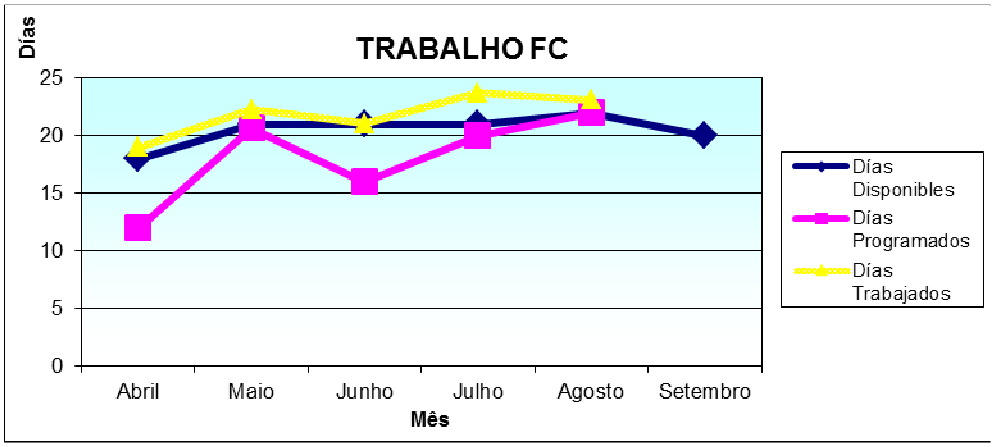
Mês	Dias Disponíveis	Dias Programados	Dias Trabalhados	Fator de Ajuste
Abril	18,0	12,0	20,0	0,16
Maio	21,0	20,6	22,2	0,18
Junho	21,0	16,0	21,0	0,17
Julho	21,0	20,0	23,7	0,19
Agosto	22,0	22,0	23,1	0,19
Setembro	20,0	20,0	20,0	0,16
Total	123,0	110,6	**Media**	**0,18**

FONTE: Elaboração Própria

PUC-Rio - Certificação Digital Nº 0913426/CA







FONTE: Elaboração Própria

PUC-Rio - Certificação Digital Nº 0913426/CA

Apêndice VI:
Características comuns das metodologias usadas no Modelo de Referencia

						Principio de Desenvolvimento Enxuto						
	Práticas	XP	SCRUM	CMM	PMBOK	Eliminar Perdas	Amplificar o aprendizado	Tomar decisões o mais tarde possível	Fazer entregas o mais rápido possivel	Tornar a equipe responsavel	Construir integridade	Visualizar o todo
Inicio	Estimar e declarar escopo do projeto	x	x	x	x					x		
	Trabalhar com os clientes	x	x							x		
	Criar visão do sistema	x	x	x						x		x
	Uso de metáforas para conceitos difíceis	x								x		
	Fornecer estimativas de produtos de trabalho e atributos de tarefas	x		x	x							
	Estimativa de recursos da atividade				x							
	Definição dos papeis da equipe	x	x									
	Planejar recursos para o projeto			x	x							
	Estabelecer orçamento e cronograma			x	x							
Planejamento	Planejar	x	x	x	x							
	Planejar envolvimento dos stakeholders			x								
	Planejar releases	x	x							x		
	Planejar as iterações	x	x							x		
	Planejamento tático	x	x							x		
	Escrever user stories	x									x	
	Escrever os testes de aceitação	x										
	Sequenciamento das Atividades			x	x							
Execução	Reuniões curtas e de progresso	x	x	x			x					x
	Testar primeiro	x	x			x	x	x	x			
	Ser simples	x				x		x				
	Programar em pares	x					x			x	x	x
	Codificação dentro de padrões	x									x	
	Fazer a propriedade coletiva	x								x		x
	Integrar continuamente	x									x	
	Fazer refactoring	x	x							x	x	x
	Fazer releases em incrementos pequenos	x	x							x	x	x
	Não se desgastar (40 horas de trabalho semanais)	x	x									
	Adotar as alterações	x				x	x					
Monitoramento e Controle	Analizar problemas			x		x		x				
	Tomar ações corretivas			x		x		x				
	Gerenciar ações corretivas			x		x		x				
	Gerenciar mudanças em requisitos		x	x	x			x				
	Incremento do Produto	x	x								x	x
	Avaliar objetivamente o processo			x								
	Comunicar e assegurar a resolução de não-conformidades			x								
	Estabelecer registros			x			x					
	Planejamento da qualidade				x							
	Realizar a garantia da qualidade				x							
	Realizar o controle da qualidade				x							
	Grafico do Progresso (burndown Chart)		x									
	Estabelecer critérios e procedimentos de verificação			x								

FONTE: Elaboração Própria

PUC-Rio - Certificação Digital Nº 0913426/CA