ECO – Um Ecossistema para o
Desenvolvimento Ágil de Sistemas Web

André Luís Gouvêa de Figueiredo

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 26.04.2005

ECO – Um Ecossistema para o Desenvolvimento Ágil de Sistemas Web

André Luís Gouvêa de Figueiredo

Orientadora: Profa. Dra. Rosely Sanches

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos

Abril de 2005

AGRADECIMENTOS

Este trabalho não teria sido possível sem ajuda de pessoas maravilhosas que estiveram ao meu lado nesta jornada.

Em primeiro lugar, agradeço à minha orientadora, Profa Dra Rosely Sanches. Este trabalho só foi possível graças à sua compreensão, apoio e confiança irrestrita.

Ao meu grande amor, Osnete Ribeiro de Souza, por ter trabalhado ao meu lado e me motivado a seguir adiante, mesmo nos momentos mais difíceis.

À minha amada mãe, Silene Figueiredo, exemplo de fibra, conduta e consciência. Porto seguro, sempre atenta e pronta a me apoiar, e meu pai, Plauton Figueiredo, pelo exemplo de fé e força de vontade.

A todos os amigos e familiares que compartilharam momentos felizes e outros nem tanto, em especial à Laura Nuruki e Christian Vidigal, promotores de eventos sociais que foram fundamentais para manter minha sanidade.

À equipe da Lúcida, em especial Rodrigo Oliveira e Luciane Lamour, pela compreensão e pelas horas-extras que viabilizaram este trabalho.

Ao ICMC, e em especial à equipe da Secretaria da pós, Beth, Laura e Ana Paula, e à CPG, pela compreensão e ajuda na finalização deste trabalho.

A todos aqueles que contribuíram, direta e indiretamente, para a concretização de mais esta etapa da minha vida.

RESUMO

A expansão vertiginosa do uso de Sistemas Web como ferramenta de negócio colocou grande pressão sobre o desenvolvimento de software, exigindo entrega de resultado tangível cada vez mais rápido, num ambiente altamente instável e dinâmico. Em resposta a essas necessidades, surgiu uma nova classe de metodologias de desenvolvimento de software, conhecidas como Metodologias Ágeis.

Este trabalho apresenta as principais características desta nova classe de metodologias, analisando em detalhes três dos principais Métodos Ágeis existentes. O objetivo primordial deste trabalho é a definição de um Método Ágil especializado para as características dos Sistemas Web, ou usando uma terminologia mais alinhada com a base filosófica que permeia o trabalho, o objetivo é a criação de um Ecossistema de Desenvolvimento Ágil de software, especializado para Sistemas Web.

ABSTRACT

The vertiginous expansion of the use of the Web Systems as business tool imposed great pressure on the software development, demanding delivery of tangible result faster time, in an unstable and highly dynamic environment. In response to these necessities, appeared a new methodology class of software development: Agile Methodologies.

This work presents the main characteristics of this new kind of methodologies, analyzing in details three of the main existing Agile Methods. The primordial objective of this work is the definition of a specialized Agile Method for the characteristics of the Web Systems, or using a terminology aligned with the philosophical base of this work, the objective is the creation of an Ecosystem of Agile Software Development, focused on Web Systems.

ÍNDICE GERAL

CAPÍTULO 1: INTRODUÇÃO.....	1
1.1. CONTEXTO	1
1.2. MOTIVAÇÃO.....	5
1.3. OBJETIVO	7
1.4. ORGANIZAÇÃO.....	8
CAPÍTULO 2: MÉTODOS ÁGEIS.....	9
2.1. CONSIDERAÇÕES INICIAIS.....	9
2.2. DEFINIÇÃO.....	10
2.3. ORIGEM	11
2.4. CONTEXTUALIZAÇÃO.....	13
2.5. ESCOPO.....	15
2.6. CONSIDERAÇÕES FINAIS	16
CAPÍTULO 3: ECOSISTEMAS DE DESENVOLVIMENTO ÁGIL DE SOFTWARE.....	18
3.1. CONSIDERAÇÕES INICIAIS.....	18
3.2. SCRUM.....	19
3.3. XP – EXTREME PROGRAMMING.....	25
3.4. FDD – FEATURE DRIVEN DEVELOPMENT.....	33
3.5. CONSIDERAÇÕES FINAIS	42
CAPÍTULO 4: ECO PARTE I: CONCEITOS.....	44
4.1. CONSIDERAÇÕES INICIAIS.....	44
4.2. ESCOPO.....	45
4.3. PÚBLICOS	46
4.4. FUNCIONALIDADES	47
4.5. AMBIENTE	50
4.6. CICLO DE VIDA	54
4.7. CICLOS DE DESENVOLVIMENTO	58
4.8. PAPÉIS	59
4.9. ARTEFATOS	64
CAPÍTULO 5: ECO PARTE II - PROCESSOS.....	72
5.1. CONSIDERAÇÕES INICIAIS.....	72

5.2. VISÃO GERAL	73
5.3. PROCESSOS	77
5.4. CONSIDERAÇÕES FINAIS	108
CAPÍTULO 6: ESTUDO DE CASO.....	109
6.1. CONSIDERAÇÕES INICIAIS	109
6.2. EMPRESA	109
6.3. ESTUDOS DE CASO	112
6.4. CONSIDERAÇÕES FINAIS	117
CAPÍTULO 7: CONCLUSÃO E TRABALHOS FUTUROS.....	118
7.1 CONCLUSÕES	118
7.3. TRABALHOS FUTUROS	119
REFERÊNCIAS BIBLIOGRÁFICAS.....	122

ÍNDICE DE FIGURAS

<i>FIGURA 1: MAPA EVOLUTIVO DOS MÉTODOS ÁGEIS.</i>	13
<i>FIGURA 2: ESCOPO DOS PRINCIPAIS MÉTODOS ÁGEIS.</i>	16
<i>FIGURA 3: ESCOPO DE ECO EM RELAÇÃO AOS ECOSISTEMAS DESCRITOS NO CAPÍTULO 3</i>	46
<i>FIGURA 4: AMBIENTE LÓGICO DE ECO</i>	51
<i>FIGURA 5: EXEMPLO DE MATRIZ LÓGICO-OPERACIONAL.</i>	54
<i>FIGURA 6: ATIVIDADES COBERTAS POR ECO</i>	73
<i>FIGURA 7: PROCESSOS DA ATIVIDADE DE AQUISIÇÃO.</i>	74
<i>FIGURA 8: PROCESSOS DA ATIVIDADE DE ENTREGA.</i>	76
<i>FIGURA 9: PROCESSOS DA ATIVIDADE DE ENCERRAMENTO.</i>	77
<i>FIGURA 10: MODELO ETVS, UTILIZADO PARA A DESCRIÇÃO DOS PROCESSOS DE ECO.</i>	78
<i>FIGURA 11: ESTRUTURA DA EMPRESA ONDE OS ESTUDOS DE CASO DE ECO FORAM REALIZADOS.</i>	110

CAPÍTULO 1: INTRODUÇÃO

1.1. CONTEXTO

Desde o surgimento dos sistemas computacionais¹, houve uma evolução fantástica em sua variedade e escopo de aplicação. Em pouco mais de meio século de existência, esses sistemas passaram de uma “mera” ferramenta de cálculo utilizada por uma pequena quantidade de cientistas e engenheiros a um elemento de uso cotidiano, presente no dia-a-dia de milhões de pessoas ao redor do mundo.

O impacto é tão amplo e profundo que diversos autores tratam o advento dos sistemas computacionais como a força motriz de uma nova era na história da humanidade [TOFFLER, 2000; OSBORNE, 1979]. Termos como “computação onipresente”

¹ O termo “sistemas computacionais” refere-se a qualquer dispositivo digital, incluindo hardware, software e rede subjacente.

[NORMAN, 1999] e “aldeia global” [MCLUHAN & POWERS, 1992] foram cunhados para tentar descrever o impacto que os sistemas computacionais têm, e terão, na sociedade moderna. De fato, parte central da organização e estabilidade da sociedade atual depende cada vez mais dos sistemas computacionais.

Não obstante aos incríveis resultados alcançados neste curto período de tempo, a criação de sistemas computacionais em geral, e em particular o desenvolvimento de software, é uma disciplina recente e, portanto, ainda em evolução. Nos últimos anos diversas publicações e artigos apresentaram visões sobre as idiossincrasias da disciplina [DEMARCO, 1995; YOURDON, 1997], *bugs* famosos [YOURDON, 1998] e a descrição de alguns projetos de software mal-sucedidos [FLOWERS, 1997]. Apesar do nítido avanço e grandes realizações da indústria de software em sua curta história, uma “crise” [PRESSMAN, 1987] persegue o desenvolvimento de software desde que este passou a representar a maior parcela do investimento necessário para a criação de sistemas computacionais.

Em 1968, auge da “crise do software”, surge o termo Engenharia de Software visando mostrar, para a comunidade em geral, os problemas do desenvolvimento de software da época e, para a comunidade de desenvolvedores, a necessidade de uma abordagem mais disciplinada para o desenvolvimento de software [NAUR & RANDALL, 1969]. Entretanto, mais que uma “mera” designação para uma nova disciplina ou comunidade de profissionais, Engenharia de Software é uma poderosa metáfora² [BRYANT, 2000], que trouxe consigo uma ampla carga filosófica, termos e conceitos advindos de outras disciplinas da engenharia [COPLIEN, 1999]. Com efeito, a base de conhecimento da engenharia - principalmente da engenharia elétrica, pela “proximidade” com o software até os anos 70, e, em menor grau, da engenharia civil - influenciou profundamente o desenvolvimento de software [PERROCHON & MANN, 1999].

Graças a isso, as atividades, envolvendo o software, e o desenvolvimento de software são, normalmente, definidos em termos da “metáfora da engenharia”, com o uso de termos

² Para os objetivos deste trabalho, o termo metáfora é uma “figura de linguagem em que um nome ou termo é associado a um objeto para o qual originalmente não se aplicaria” [BRYANT, 2000].

como construção, componentes, manutenção, protótipo, entre outros. Estes termos e as imagens a que remetem são, simultaneamente, intuitivamente claras, mas acompanhadas de algumas restrições, principalmente de comunicação e abordagem [BRYANT, 2000]. Assim, por exemplo, é comum desenvolvedores de software falarem sobre *manutenção de software* sabendo, porém, que o termo é peculiar a este contexto, não havendo uma relação direta com o termo usado na engenharia tradicional. De fato, uma designação mais clara para a atividade conhecida como *manutenção de software* é *evolução de software* [ARTHUR, 1988].

A influência da engenharia no desenvolvimento de software não se restringiu apenas a termos e conceitos; o processo³ de desenvolvimento de software também foi profundamente influenciado. Os processos de desenvolvimento advindos desta visão são caracterizados pela completa elucidação dos requisitos e posterior elaboração de um plano completo e detalhado que guiará todo o andamento do projeto. Esses processos são conhecidos como desenvolvimento de software direcionado por planejamento⁴ [BOEHM, 2002] (utilizado neste trabalho), privilegiados [TRUEX *et al*, 2000] monumentais [HIGHSMITH, 2000] e de estilo catedral [RAYMOND, 1997].

A metáfora da engenharia foi e é de vital importância para o desenvolvimento de software, ancorando a disciplina em conceitos sólidos e cientificamente embasados [BRYANT, 2000]. Entretanto, esta metáfora é difícil de ser mapeada para as necessidades e pressões impostas ao desenvolvimento de software atual [COPLIEN, 1999].

Os esforços de desenvolvimento atuais, ocorrem, via de regra, em um ambiente altamente instável, tanto em termos de requisitos e necessidades, como em termos de tecnologia e infra-estrutura, e com pressão cada vez maior por entrega rápida, com projetos de poucos meses de duração [PRESSMAN, 2001]. A Engenharia de Software tradicional e, mais especificamente, os processos de desenvolvimento de software direcionados por

³ Neste trabalho, os termos “processo de desenvolvimento”, “método” e “metodologia” serão usados indistintamente, referindo-se, genericamente, a “um conjunto de convenções que um grupo concorda” [COCKBURN, 2002].

⁴ Do inglês *plan-driven*.

planejamento, não são a abordagem mais adequada para este tipo de problema [PERROCHON & MANN, 1999]. A mudança dos termos utilizados por Pressman é elucidativa neste sentido: O que antes era “crise do software” [PRESSMAN, 1986] tornou-se “aflição crônica” [PRESSMAN, 2001]. Ou seja, apesar dos excepcionais resultados alcançados com a Engenharia de Software tradicional, algum problema continua existindo, a ponto de tornar-se crônico.

A evolução nas idéias de Frederick P. Brooks aponta para a mesma direção. Em 1975, Brooks [1995] defendia a idéia de que todo projeto de software deveria ser planejado levando em consideração que a primeira versão do software seria descartada, numa clara menção ao processo de desenvolvimento direcionado por planejamento. Mais tarde, em 1995, ao analisar suas antigas idéias, Brooks [1995] mostra nova visão, constatando que “o modelo cascata está errado!” e recomendando uma abordagem incremental e evolutiva para o desenvolvimento de software. Esta mudança de visão é ilustrada por Brooks através de episódios descrevendo um sistema construído de forma evolutiva.

De fato, já em 1986, em seu clássico artigo *No silver bullet*, Brooks [1995] relata o impacto causado pela “mudança de paradigma metafórico” ocorrida quando deixou-se de “escrever” software e passou-se a “construir” software. Segundo ele, esta mudança na metáfora do desenvolvimento de software elevou a disciplina a um novo patamar de entendimento e produtividade. Brooks conclui que é chegado o momento de mais uma mudança de paradigma metafórico no campo do desenvolvimento de software, passando da “construção” de software para a “criação” ou “cultivo”⁵ de software.

Nos últimos anos, a mudança de paradigma descrita por Brooks parece estar se consolidando. Diversos trabalhos publicados e em andamento utilizam termos alinhados com esta nova metáfora para o desenvolvimento de software. O instituto Forrester Research utiliza o termo “TI Orgânica”⁶ para descrever o novo modelo de infra-estrutura computacional que estará disponível até 2006 [GILLETT *et al*, 2002]. Groot, em seu artigo

⁵ Do inglês *grow software*.

⁶ Do inglês *Organic IT*.

“Towards Organic Software”⁷ [GROOT, 1999], compara a abordagem de desenvolvimento do Linux com o cultivo de um jardim, vivo e em constante evolução. Alguns autores estão substituindo o termo “metodologia” ou “processo de desenvolvimento”, por “ecossistema de desenvolvimento” de software, visando mostrar uma abordagem holística, mais evolutiva e adaptativa do que planejada e repetível [HIGHSMITH, 2002; COCKBURN, 2002]. Beedle [2002] mantém um *website* “dedicado ao estudo e aplicação da metáfora do viver – os padrões da vida – aos sistemas computacionais e organizações humanas”.

1.2. MOTIVAÇÃO

Mais que uma mudança meramente filosófica, a “mudança de paradigma metafórico” em curso, tem motivações e conseqüências práticas, impulsionadas por necessidades de um mercado cada vez mais ávido por novidades e impaciente nos prazos.

No centro desta transformação estão o surgimento e crescimento vertiginoso dos Sistemas Web⁸. O desenvolvimento de Sistemas Web é caracterizado pela constante pressão por prazos cada vez menores, num ambiente onde a competição depende da velocidade no atendimento às necessidades dos usuários e do negócio [CHAUBEY, 2001]. À luz da nova visão do desenvolvimento de software, os Sistemas Web são considerados sistemas vivos [GINIGE & MURUGESAN, 2001a], que crescem e evoluem constantemente de acordo com as necessidades e desafios impostos pelo ambiente. Em determinados casos, esta evolução ocorre a uma velocidade muito maior que a encontrada em qualquer outro tipo de esforço de desenvolvimento, com entregas diárias ou semanais de novas versões do sistema [PRESSMAN, 2001].

As características dos Sistemas Web exigem, portanto, um estilo de desenvolvimento próprio, altamente disciplinado e flexível.

⁷ Rumo ao Software Orgânico, em português.

⁸ Para os objetivos deste trabalho, o termo “Sistemas Web” (SW) engloba todo tipo de sistema computacional adequado aos padrões da Internet. Para uma discussão mais aprofundada do assunto, ver [GINIGE & MURUGESAN, 2001b].

O gerenciamento de um projeto, incluindo-se os de desenvolvimento de software, baseia-se no controle de quatro variáveis principais, quais sejam: custo, prazo, qualidade e escopo [BECK & FOWLER, 2001]. Como essas variáveis não são ortogonais, alterações em uma, implicam em mudanças nas demais e, portanto, o gerenciamento de um projeto consiste em manter o equilíbrio dessas quatro variáveis [HUNT & THOMAS, 2000]. Alguns autores tratam custo e recursos (pessoas, equipamentos, etc.) separadamente, argumentando que recursos estão sob controle do gerente de projetos, sendo elemento chave no controle e planejamento do projeto, enquanto o custo é controlado por um patrocinador ou cliente [WYSOCKI *et al*, 2000]. Esta distinção é irrelevante para os objetivos deste trabalho.

Para que uma equação composta por estas quatro variáveis seja solucionável, uma delas deve ser “fixada”. Como consequência, qualquer mudança nessa variável implicará em mudanças nas demais sendo, portanto, um risco para o sucesso do projeto. Assim, essa variável deve ser objeto de um controle minucioso, com grande parte do esforço de gerenciamento sendo despendido em seu controle e nos impactos que eventuais mudanças causarão nas demais [WYSOCKI *et al*, 2000].

Neste sentido, os Sistemas Web mudaram o foco do gerenciamento de projetos de desenvolvimento de software do “o quê” (escopo) para o “quando” (prazo) [AOYAMA, 1998]. Na Engenharia de Software tradicional, e em particular nas metodologias de desenvolvimento direcionadas por planejamento, o gerenciamento está focado no controle do escopo, com toda sua abordagem sendo baseada na completa elucidação inicial dos requisitos, controle minucioso de mudanças e andamento durante todo o desenvolvimento. Já nos Sistemas Web, o gerenciamento está focado no prazo, com as demais variáveis sendo ajustadas de acordo com as datas e tempo disponíveis para o projeto.

Alinhadas a essa abordagem para o desenvolvimento e gerenciamento de projetos de software, uma corrente crescente de novas metodologias vem tomando corpo nos últimos anos. Estas novas metodologias de desenvolvimento, denominadas de Métodos Ágeis [FOWLER, 2000b], baseiam-se em conceitos e práticas já conhecidas, muitas delas herdadas da Engenharia de Software tradicional, unidas de forma diferenciada, num

processo mais adequado com a visão e necessidades atuais [ABRAHAMSSON et al, 2003].

O advento dos Métodos Ágeis de desenvolvimento de software é uma resposta clara à pressão imposta ao desenvolvimento de software, apresentando um conjunto de processos, atividades e papéis que se propõe a entregar software de qualidade no menor prazo possível. Esses métodos, entretanto, abrangem, parcialmente, as peculiaridades do desenvolvimento de Sistemas Web, deixando lacunas a serem preenchidas por empresas e equipes focadas no desenvolvimento de Sistemas Web.

1.3. OBJETIVO

Apesar da grande repercussão dos Métodos Ágeis – tanto no mercado como, mais recentemente, na academia - e de sua adoção generalizada nos mais diversos setores, esses ainda carecem de exploração. São raros os trabalhos nesse sentido, tanto nos aspectos relacionados à base de conhecimento teórico dos Métodos Ágeis, como nos relacionados à prática dos mesmos.

Outrossim, apesar de representarem parcela significativa dos esforços de desenvolvimento empreendidos na atualidade, a base de conhecimento sobre os Sistemas Web ainda é muito pequena, restringindo-se a alguns poucos livros e artigos. Com efeito, ainda existe muita divergência sobre a melhor forma de desenvolver esse tipo de sistemas.

Não obstante à falta de informação sobre Sistemas Web e Métodos Ágeis, é clara a sinergia entre os conceitos e a necessidade de pesquisas neste campo, seja tratando cada um dos conceitos de forma individual, seja tratando sua aplicação conjunta, seja, mais amplamente, investigando seu papel no contexto da nova visão que surge no campo do desenvolvimento de software.

Assim, o objetivo primordial deste trabalho é a definição de um Método Ágil especializado para o desenvolvimento de Sistemas Web, ou, usando uma terminologia

mais alinhada com a base filosófica que permeará o trabalho: o objetivo é a criação de um Ecossistema de desenvolvimento ágil de software, especializado para Sistemas Web.

O foco será nas sinergias e aplicação conjunta dos conceitos, gerando como produto principal um Método Ágil afinado com os conceitos e tecnologias próprias dos Sistemas Web. Para isso, entretanto, é necessária uma definição mínima de cada conceito, com profundidade suficiente para viabilizar seu entendimento no contexto. Também de forma secundária e superficial, será utilizada a “metáfora da vida” no lugar da “metáfora da engenharia”, embasando o trabalho nesta nova visão do desenvolvimento de software.

1.4. ORGANIZAÇÃO

O restante deste documento encontra-se dividido conforme segue.

No **Capítulo 2 – Métodos Ágeis**, introduz-se este novo tipo de processo de desenvolvimento, definindo o que são Métodos Ágeis, sua origem, contexto e escopo de aplicação. No capítulo seguinte, **Capítulo 3 – Ecossistemas de desenvolvimento ágil de software**, são apresentados os principais Métodos Ágeis existentes hoje, trazendo uma definição mais abrangente de três dos principais métodos existentes, caracterizando-os quanto a suas práticas, papéis e processo.

Nos dois próximos capítulos, a metodologia desenvolvida neste trabalho é apresentada. O primeiro, intitulado **Capítulo 4 - ECO Parte I: Conceitos**, são apresentadas as definições dos conceitos utilizados na metodologia; o segundo, intitulado **Capítulo 5 – ECO Parte II: Processos**, apresenta os processos que compõem a metodologia. E, para analisar a aplicação prática da metodologia proposta, um estudo de caso é apresentado no **Capítulo 6 – Estudo de caso**.

No Capítulo 7, são apresentados **Conclusão e Trabalhos Futuros**, e por fim, são apresentadas as **Referências Bibliográficas** que influenciaram e embasaram este trabalho.

CAPÍTULO 2: MÉTODOS ÁGEIS

2.1. CONSIDERAÇÕES INICIAIS

Recentemente, o assunto Métodos Ágeis tem despertado grande interesse e provocado discussões acaloradas na comunidade de desenvolvimento de software. Interesse por sua simplicidade e promessa de entrega rápida de resultado, e discussões acaloradas graças ao discurso de ruptura utilizado por parte de seus principais proponentes.

Não obstante ao volume crescente de artigos e publicações sobre o assunto, ainda são raros os estudos acadêmicos sobre o tema, visto que a quase totalidade dos trabalhos realizados foi escrita por consultores ou praticantes. Particularmente escassos são os estudos de caso e relatos sobre a efetividade desta nova classe de metodologias de desenvolvimento, tornando sua avaliação e análise bastante limitada e superficial.

Espera-se, entretanto, que uma base de conhecimento mais sólida e abrangente surja nos próximos anos, graças ao impacto que o assunto teve nos meios acadêmicos e ao apelo de sua proposta para o mercado.

Este capítulo tem como objetivo definir e localizar os Métodos Ágeis dentro do contexto do desenvolvimento de software. Nos tópicos seguintes são apresentadas definições sobre o que são os Métodos Ágeis, sua origem, contexto e escopo de aplicação.

2.2. DEFINIÇÃO

Os Métodos Ágeis – com o termo ágil denotando “a qualidade de ser ágil; prontidão para o movimento; rapidez, atividade, destreza no movimento”⁹ – constituem uma nova classe de metodologias de desenvolvimento de software criada para atender à crescente pressão do mercado por processos mais ágeis e leves, com ciclos de desenvolvimento cada vez mais curtos [ABRAHAMSSON, 2003]. Essas características são particularmente marcantes no volátil e crescente mercado de Sistemas Web, assim como no emergente ambiente de aplicações móveis [ABRAHAMSSON, 2003].

Highsmith [2002] define agilidade como “a habilidade para criar e responder à mudança de modo a prosperar em um ambiente de negócios turbulento.” Segundo ele, as organizações ágeis criam mudança, colocando grande pressão sobre os competidores. Assim, mais que responder à mudança de forma rápida, agilidade significa gerar mudança como diferencial competitivo. Sob este ponto de vista, Métodos Ágeis são processos de desenvolvimento de software com capacidade de adaptar-se rapidamente a novas necessidades, sejam elas impostas externamente ou identificadas internamente como oportunidade [HIGHSMITH, 2002].

Uma importante característica presente em todos os Métodos Ágeis é o reconhecimento do desenvolvimento de software como um processo empírico [WILLIAMS & COCKBURN, 2003]. Existem dois tipos básicos de processos: os processos definidos e os processos empíricos [SCHWABER & BEEDLE, 2002]. Em

⁹ Tradução da descrição do verbete *ágil*, obtida em <http://dictionary.oed.com>.

linhas gerais, os processos definidos são aqueles onde é possível a criação de um plano *a priori* que direcionará todo o projeto, produzindo sempre o mesmo resultado. Processos empíricos, por outro lado, são aqueles onde há grande incerteza, pesquisa e descoberta, tornando impraticável a definição completa do projeto num único momento [SCHWABER & BEEDLE, 2002].

2.3. ORIGEM

As principais metodologias que hoje são conhecidas como Métodos Ágeis – *Extreme Programming* (XP) [BECK, 2000], *Crystal Methods* [COCKBURN, 2002], *Lean Development* [HIGHSMITH, 2002], *Feature-driven development* (FDD) [PLAMER & FELSING, 2002], *Adaptive Software Development* (ASD) [HIGHSMITH, 2000], *Scrum* [SCHWABER & BEEDLE, 2002] e *Dynamic System Development Methodology* (DSDM) [HIGHSMITH, 2002] – surgiram por volta da década de 90, período de crescimento do uso comercial da Web.

Graças à crescente pressão por entregas rápidas, esses processos de desenvolvimento ganharam destaque por atender justamente a este tipo de necessidade [COCKBURN, 2002]. Assim, por oferecerem uma alternativa às metodologias mais utilizadas e conhecidas durante a década de 90, chamadas por Highsmith [2000] de metodologias “monumentais” ou “peso-pesado”, essa nova classe de metodologias foi, inicialmente, designada como metodologias leves, ou metodologias peso-leve [COCKBURN, 2002].

Somente a partir de 2001, com a publicação do Manifesto para o desenvolvimento ágil de software [MANIFESTO, 2001], os Métodos Ágeis passaram a ser conhecidos como tal. O Manifesto foi o resultado de uma conferência onde os principais líderes e proponentes das metodologias supracitadas se reuniram para discutir as semelhanças e diferenças de suas abordagens para o desenvolvimento de software [HIGHSMITH, 2001].

Cockburn [2002] descreve que durante este evento, houve consenso, num primeiro nível, sobre a necessidade de responder à mudança, desfazendo o senso comum de que mudanças são caras e complexas. Portanto, num nível mais alto, esta é uma característica

comum a todos os Métodos Ágeis. Num nível mais baixo, foram definidos os quatro valores chave de todo Método Ágil, descritos como segue em [MANIFESTO, 2001]:

“Nós estamos descobrindo melhores maneiras de se desenvolver software, desenvolvendo-o e auxiliando outras pessoas a fazê-lo. Através deste trabalho, nós valorizamos:

Indivíduos e interações sobre processos e ferramentas

Software funcional sobre documentação abrangente

Colaboração do cliente sobre negociação de contratos

Responder à mudança sobre seguir um plano

Ou seja, apesar dos itens da direita serem importantes, nós valorizamos mais os da esquerda”.

Num terceiro nível, houve acordo parcial sobre doze outros valores mais detalhados [MANIFESTO, 2001]. Cockburn [2002] descreve que não houve consenso num quarto nível, tratando sobre os detalhes táticos da condução de um projeto. Entretanto, foi consenso de que esta divergência era saudável para a disciplina na medida em que dá margem para novas explorações e descobertas [COCKBURN, 2002].

Cockburn [2002] ressalta que todos os Métodos Ágeis baseiam-se em conceitos amplamente utilizados e conhecidos, adquiridos durante as várias décadas de desenvolvimento da Engenharia de Software. Beck [1999] enumera as referências que influenciaram as práticas de XP, afirmando que, isoladamente, as práticas que constituem XP não são novas. Abrahamsson [2003] criou um mapa evolutivo dos Métodos Ágeis, apresentando os trabalhos que influenciaram os principais Métodos Ágeis existentes e o relacionamento entre eles. Este mapa é mostrado na *Figura 1*.

Larman e Basili [2003] definem, de forma genérica, Métodos Ágeis como versões modernas de processos de desenvolvimento iterativos e incrementais, existentes desde os anos 70. Apesar de publicados e utilizados a décadas, diversas organizações privadas e governamentais apenas agora estão descobrindo os benefícios desta abordagem, advindo daí, grande parte da repercussão dos Métodos Ágeis [LARMAN & BASILI, 2003].

Quadro 1: Diferenças entre processos privilegiados e marginalizados [ABRAHAMSSON *et al* 2002]

Metodologias privilegiadas	Metodologias marginalizadas
Desenvolvimento de sistemas de informação é um processo gerenciado e controlado.	Desenvolvimento de sistemas de informação é um processo aleatório e acomodatório.
Desenvolvimento de sistemas de informação é um processo linear e seqüencial.	Desenvolvimento de sistemas de informação são processos simultâneos, sobrepostos, com lacunas.
Desenvolvimento de sistemas de informação é um processo universal e replicável.	Desenvolvimento de sistemas de informação ocorre de forma única e local.
Desenvolvimento de sistemas de informação é um processo direcionado por objetivo, determinado e reacional.	Desenvolvimento de sistemas de informação é negociado, volúvel.

DeMarco [DEMARCO & BOEHM, 2002] argumenta que “os processos direcionados por planejamento resultam dos esforços para ensinar organizações a criarem software, enquanto que os métodos ágeis são focados no desenvolvimento das pessoas envolvidas na construção de software”. A partir desta visão, essas abordagens passam a ser complementares, ao invés de antagônicas. De fato, diversos autores concordam com esta visão [GLASS, 2001; BOEHM, 2002; LYCETT *et al*, 2003] apesar de não necessariamente partilharem da mesma motivação de DeMarco.

Outros tantos autores relacionam os Métodos Ágeis, em particular *Extreme Programming*, com o CMM¹⁰, considerado como um modelo para processos direcionados por planejamento, concluindo que há compatibilidade entre ambos [PAULK, 2001a; PAULK, 2001b; REIFER, 2003; LYCETT *et al*, 2003]. À luz da visão de DeMarco [DEMARCO & BOEHM, 2002], esses trabalhos ganham nova dimensão, já que, muito mais que compatíveis, esses conceitos são complementares, com o CMM voltado para o nível de maturidade do processo de desenvolvimento da organização, e os Métodos Ágeis focados nas práticas de desenvolvimento em equipes e por indivíduos.

¹⁰ *Capability Maturity Model* (mais detalhes em <http://www.sei.cmu.edu/cmm/>).

PSP e TSP¹¹ são iniciativas do SEI, mesma entidade responsável pelo CMM, voltadas para as práticas diárias de pessoas e equipes, respectivamente. Mais que analisar Métodos Ágeis e o CMM, novos trabalhos paralelizam Métodos Ágeis com PSP e TSP [STARK, CROKER, 2003; DEMARCO & BOEHM, 2002], trazendo um novo nível de entendimento para o assunto e mudando o foco da discussão do “qual é o mais adequado” para “como relacioná-los num modelo coerente”. Nesse sentido, Boehm e Turner [2003] propõem um método para estruturar projetos de modo a incorporar características tanto dos Métodos Ágeis como dos processos direcionados por planejamento, de acordo com os riscos envolvidos no projeto.

2.5. ESCOPO

O escopo de uma metodologia consiste dos papéis, atividades e fases do ciclo de vida que ela cobre [COCKBURN, 2002]. As fases do ciclo de vida abordadas por uma metodologia indicam em quais pontos ela inicia e termina sua cobertura. Também faz parte do escopo de uma metodologia a definição dos papéis executados em cada fase, além da determinação das atividades a serem realizadas pelos papéis em cada fase [COCKBURN, 2002].

Apesar de os Métodos Ágeis “concorrerem” com os processos direcionados por planejamento, nem todos os Métodos Ágeis existentes são adequados para todo o ciclo de vida do desenvolvimento de software [ABRAHAMSSON *et al*, 2002]. De fato, os Métodos Ágeis existentes diferem bastante em termos de seu escopo. A *Figura 2* compara os principais Métodos Ágeis existentes em termos do suporte à gerência de projetos, definição do processo e descrição de práticas, atividades e artefatos [ABRAHAMSSON *et al*, 2002; ABRAHAMSSON *et al*, 2003]. No capítulo seguinte são analisadas as práticas, papéis e processo de três dos principais Métodos Ágeis, a saber: *Scrum*, FDD e XP.

Os Métodos Ágeis não são aplicáveis a qualquer domínio [COCKBURN & HIGHSMITH, 2001]. A utilização de Métodos Ágeis em equipes desmotivadas ou em

¹¹ *Personal Software Process* e *Team Software Process* (mais detalhes em <http://www.sei.cmu.edu/tsp/>).

organizações centradas em processo, não é aconselhável. Igualmente desaconselhável é a adoção de Métodos Ágeis em projetos onde a colaboração do cliente é limitada ou com equipes muito grandes [COCKBURN & HIGHSMITH, 2001]. O tamanho médio das equipes utilizando métodos ágeis é de nove pessoas, embora existam exemplos de projetos bem sucedidos com até 250 pessoas [COCKBURN & HIGHSMITH, 2001].

Figura 2: Escopo dos principais Métodos Ágeis [ABRAHAMSSON et al, 2002; ABRAHAMSSON et al, 2003].

2.6. CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivo apresentar uma visão geral sobre o estado da arte dos Métodos Ágeis, conceituando-os, apresentando suas influências, contextualizando-os em relação à base de conhecimento da engenharia de software tradicional e discutindo seu escopo e aplicabilidade.

Uma questão importante não abordada neste capítulo é a apresentação de dados e estudos de caso sobre os resultados da aplicação de Métodos Ágeis na prática. Apesar de ainda raros, alguns trabalhos já foram publicados relatando a experiência no uso de Métodos Ágeis [RISING & JANOFF, 2000; STARK & CROCKER, 2003; AMBLER, 2002], incluindo casos de empresas com CMM nível 3 [REIFER, 2003]. Dados obtidos através de pesquisas apresentam a visão dos praticantes sobre o uso deste tipo de metodologia [SHINE, 2002; COCKBURN & HIGHSMITH, 2001; REIFER, 2002].

No Capítulo 6 deste trabalho são apresentados resultados práticos do uso de ECO em situações reais. Estes estudos de caso têm o viés de terem sido realizados em apenas uma empresa e, portanto, terem valor limitado no que tange à aplicabilidade geral. Os estudos também foram realizados de maneira informal, sem a coleta de indicadores que apresentem de forma mais clara os resultados atingidos com o seu uso. De qualquer forma, os resultados obtidos têm valor significativo para este trabalho e, sem dúvida, podem ser valiosos para comparações futuras.

CAPÍTULO 3: ECOSSISTEMAS DE DESENVOLVIMENTO ÁGIL DE SOFTWARE

3.1. CONSIDERAÇÕES INICIAIS

O capítulo anterior foi dedicado ao estudo dos Métodos Ágeis em geral sem, no entanto, ater-se aos detalhes de nenhum em particular. Neste capítulo, três dos principais Métodos Ágeis existentes – *Scrum*, *Extreme Programming* e *Feature-driven development* (FDD) - são analisados em maior profundidade.

Com o intuito de facilitar o entendimento desses métodos e de simplificar eventuais comparações entre eles, cada um é discutido em termos de suas práticas, papéis e processo. As práticas representam o conjunto de atividades, convenções ou subprodutos definidos

pelo método. Papéis descrevem as responsabilidades que pessoas ou equipes assumem durante o andamento do projeto. O processo define o ciclo de vida do projeto, contendo as fases pelas quais o produto passa e como as práticas e papéis se relacionam neste contexto.

3.2. SCRUM

Scrum refere-se a uma estratégia do jogo de *Rugby* onde uma bola perdida é colocada novamente em jogo através do trabalho em equipe [SCHWABER & BEEDLE, 2002]. O termo foi utilizado pela primeira vez fora do mundo dos esportes para descrever um modelo de processo japonês utilizado no desenvolvimento de novos produtos [TAKEUSHI & NONAKA, 1986]. Este tipo de desenvolvimento baseia-se em equipes multidisciplinares e auto-organizáveis, onde todos os membros têm visão global do objetivo a ser atingido e não existem atribuições fixas, cabendo à equipe se organizar da melhor forma para atingir o objetivo definido [TAKEUSHI & NONAKA, 1986].

Baseado nestes conceitos, o Método Ágil *Scrum* define um processo de desenvolvimento de software iterativo, incremental e adaptativo. O objetivo de *Scrum* é entregar a maior quantidade possível de software de qualidade através de uma série de intervalos de tempo, fixos e curtos, que tipicamente duram um mês [SCHWABER & BEEDLE, 2002].

O foco do método são o gerenciamento e controle de um projeto de desenvolvimento de software, descrevendo as diretrizes básicas a serem adotadas para garantir o sucesso do projeto. Trata-se de um processo empírico, no qual não existe a descrição de atividades técnicas e processos, apenas a definição de um ciclo de vida básico, alguns poucos papéis e práticas que garantam total controle e visibilidade do projeto, por um lado, e poder e liberdade para a equipe de desenvolvimento, por outro [SCHWABER & BEEDLE, 2002].

3.2.1. PRÁTICAS

As principais práticas de *Scrum* são definidas como uma linguagem padrão¹² [BEEDLE *et al*, 1999a; BEEDLE *et al*, 1999b]. Os tópicos seguintes descrevem estas práticas.

BACKLOG

Um *Backlog* representa o trabalho a ser realizado num projeto, contendo tudo o que é necessário existir na versão final do produto a ser desenvolvido. Trata-se de uma lista de atividades, priorizadas e estimadas, onde a mais prioritária será trabalhada primeiro e a menos prioritária será abordada por último.

Em *Scrum* existe o *Backlog* do produto, contendo o trabalho global contemplando todo o projeto, e *Backlogs* específicos para cada iteração (*Backlog* do *Sprint*), com um subconjunto do *Backlog* do produto a ser realizado durante a iteração.

O *Backlog* é uma estrutura viva e dinâmica, constantemente modificada e atualizada de acordo com a evolução do conhecimento sobre o produto a ser criado e do ambiente em que este será utilizado. O objetivo é garantir que o produto construído seja o mais apropriado, competitivo e utilizável possível, dentro do prazo e orçamento existente.

SPRINT

Um *Sprint* é um período de tempo de aproximadamente 30 dias no qual atividades do *Backlog* são realizadas. Trata-se, portanto, de uma iteração do projeto onde a equipe se auto-organiza para criar uma nova versão do produto.

Cada *Sprint* é iniciado com uma reunião de planeamento, onde uma determinada quantidade de atividades advindas de um *Backlog* do produto é selecionada para execução.

¹² Do inglês *Pattern Language*.

Os itens são selecionados de acordo com o objetivo do *Sprint*, prioridades e estimativa da quantidade de trabalho que pode ser realizado.

Como regra, nenhuma atividade externa pode ser adicionada durante um *Sprint*, apenas novas atividades identificadas a partir das originalmente selecionadas para realização. Novas necessidades externas são adicionadas ao *Backlog* do produto, podendo ser selecionadas para execução no próximo *Sprint*. Assim, durante um *Sprint*, o caos externo é isolado, permitindo que a equipe de desenvolvimento tenha total controle sobre a melhor forma de atingir o objetivo do *Sprint*.

Fechando o *Sprint* ocorre uma reunião de revisão, onde a equipe apresenta o trabalho realizado para avaliação. Quaisquer modificações ou novidades são incluídas no *Backlog* para eventual construção em *Sprints* subsequentes.

REUNIÕES SCRUM

Durante um *Sprint* a equipe deve se auto-organizar para produzir o *Backlog* do *Sprint*. A organização, distribuição de tarefas, sincronismo e controle do projeto são feitos pela própria equipe, através de reuniões diárias de aproximadamente 15 minutos de duração chamadas de Reuniões *Scrum*. Durante essas reuniões, cada membro da equipe deve responder às seguintes perguntas:

- Quais as atividades realizadas desde a última Reunião *Scrum*;
- Quais os impedimentos ou problemas enfrentados;
- Quais atividades serão realizadas até a próxima Reunião *Scrum*;

As Reuniões *Scrum* são rituais que criam uma cultura na equipe, servindo como irradiadores de conhecimento, planos, problemas e andamento do projeto. Durante essas reuniões, todas as atividades realizadas, problemas enfrentados e as próximas atividades a serem atacadas são registrados para fins de controle e planejamento.

Como estas reuniões ocorrem diariamente – e idealmente no mesmo local e hora -, as tarefas executadas são pequenas em termos de tempo, minimizando os riscos inerentes à sua realização. Outrossim, a visibilidade do andamento do projeto é clara, com qualquer impedimento ou problema de produtividade sendo rapidamente detectado. Interessados no andamento do projeto podem assistir a estas reuniões sem, no entanto, poderem participar ou interferir.

3.2.2. PAPÉIS

Scrum propõe um conjunto mínimo de papéis, com foco nas atividades gerenciais necessárias para garantir sua aplicação efetiva. Os tópicos seguintes descrevem estes papéis e equipes [SCHWABER & BEEDLE, 2002].

MESTRE SCRUM

Este é um novo papel gerencial proposto por *Scrum* cuja principal função é a de propagar e garantir o uso efetivo dos valores, práticas e regras de *Scrum*. O Mestre *Scrum* é responsável por iniciar o uso das práticas propostas e garantir o seu uso, ocupando, desta forma, papel central no sucesso da adoção de *Scrum*.

Além de garantir o uso correto e efetivo de *Scrum*, o Mestre *Scrum* é responsável também por garantir que a equipe trabalhe com máxima produtividade, removendo qualquer impedimento ou problema enfrentado, como, por exemplo, solicitações externas de alocação de algum dos membros da equipe de projeto.

RESPONSÁVEL PELO PRODUTO

Em *Scrum*, apenas uma pessoa é responsável pelo controle e gerenciamento do *Backlog* do produto. Esta pessoa é conhecida como Responsável pelo Produto e, como o nome deixa claro, é a pessoa oficialmente responsável pelo resultado do projeto. Cabe ao Responsável pelo Produto manter o *Backlog* do produto e garantir sua visibilidade para

todos os interessados, de modo que qualquer pessoa saiba os itens de maior prioridade e, conseqüentemente, quais os próximos itens a serem trabalhados.

É importante que o Responsável pelo Produto seja apenas uma pessoa, e não um comitê. Embora possa existir um comitê para auxiliar esta pessoa com conselhos e idéias, qualquer alteração de prioridades ou nos itens do *Backlog* deve passar, necessariamente, pelo crivo do Responsável pelo Produto.

EQUIPE SCRUM

Uma Equipe *Scrum* é um time multifuncional contendo 7 pessoas, mais ou menos duas. A equipe deve possuir pessoas com todas as capacidades necessárias para se atingir o objetivo do *Sprint*. Entretanto, não existem papéis definidos, cabendo à equipe se auto-organizar de modo que todos contribuam da melhor forma para o objetivo final. Todos os membros da equipe aplicam suas capacidades e conhecimentos a todos os problemas encontrados de modo a oferecer a melhor solução possível.

Cabe à Equipe *Scrum* analisar o *Backlog* do produto e, a partir dos itens mais prioritários, selecionar todos os que possam ser construídos no próximo *Sprint*, ou seja, estimar as atividades selecionadas e construir, a partir das atividades mais prioritárias e do tempo disponível no *Sprint*, o *Backlog* do *Sprint*. Cabe também à Equipe *Scrum* identificar suas limitações e, eventualmente, solicitar novos membros com as capacidades necessárias para se atingir o objetivo do *Sprint*.

CLIENTE

Em *Scrum*, o cliente é responsável por participar das atividades relacionadas ao *Backlog* com o intuito de garantir que o produto gerado atenda às necessidades vislumbradas. Assim, cabe ao Cliente realizar solicitações de funcionalidades, auxiliar na priorização do *Backlog* e avaliar a qualidade funcional do projeto.

GERÊNCIA

A Gerência representa em *Scrum* o último nível de tomada de decisão e o patrocinador do projeto. Cabe à Gerência, por exemplo, definir o Responsável pelo Produto, participar na definição de objetivos e requisitos, além de definir padrões e convenções a serem seguidas pela equipe.

3.2.3. PROCESSO

O ciclo de vida de um projeto *Scrum* inclui três fases: pré-desenvolvimento, desenvolvimento e pós-desenvolvimento¹³ [SCHWABER & BEEDLE, 2002]. A fase de desenvolvimento inclui a parte ágil de *Scrum*, sendo que as fases de pré-desenvolvimento e pós-desenvolvimento servem, respectivamente, para a definição do projeto e sua entrega final. Normalmente, essas fases são mapeadas como *Sprints* “especiais” que tem como objetivo a elaboração de um plano de projeto, na fase pré-desenvolvimento, e atividades como documentação e treinamento na fase pós-desenvolvimento.

Durante a fase pré-desenvolvimento, duas atividades principais são executadas: planejamento e modelagem de alto nível. O objetivo primordial do planejamento é a criação das condições mínimas para o início do projeto [SCHWABER & BEEDLE, 2002]. Essas condições incluem a definição da equipe, ferramentas e infra-estrutura, análise de riscos, aprovação gerencial e a criação de um *Backlog* com todos os requisitos conhecidos no momento. Esses requisitos podem ser definidos por qualquer pessoa interessada no projeto, como cliente, equipe de marketing, vendas, etc., e, uma vez incluído no *Backlog*, será estimado e priorizado. Durante esta fase é criado também um modelo de alto nível, incluindo a arquitetura do projeto, que atenda aos requisitos definidos no *Backlog* até então.

A fase de desenvolvimento ocorre através de uma série de *Sprints* onde as funcionalidades do projeto são implementadas. Os *Sprints* incluem em si todas as fases tradicionais de desenvolvimento, incluindo levantamento de requisitos, análise, projeto,

¹³ Os nomes em inglês para as fases de Scrum são *pre-game*, *development* (ou *game*) e *post-game*.

implementação, testes e entrega. Durante esta fase, a arquitetura, modelos, documentos e *Backlog* evoluem através de constantes modificações e adaptações.

A fase de desenvolvimento é tratada como uma “caixa-preta”, onde imprevistos são esperados. Todas as variáveis técnicas e de negócio (como tempo, qualidade, requisitos, recursos, ferramentas, tecnologias e influências vindas do mercado) são observadas e controladas através das práticas de *Scrum*. Ao invés de considerar todas estas variáveis apenas no início do projeto, em *Scrum* elas são constantemente observadas e controladas visando manter a flexibilidade na adaptação às mudanças.

A fase de pós-desenvolvimento inclui as atividades de fechamento de uma versão. Esta fase é iniciada tão logo se obtenha um acordo quanto à suficiência das variáveis do projeto, como requisitos e tempo disponível. A partir deste acordo, todas as atividades relacionadas com a entrega da versão são realizadas, como documentação, treinamento e implantação em produção.

3.3. XP – EXTREME PROGRAMMING

XP é uma metodologia voltada para equipes de até 20 pessoas engajadas no desenvolvimento de software cujos requisitos são vagos ou em constante mudança [BECK, 2000]. As raízes de XP remontam à comunidade *Smalltalk*, e em particular à colaboração entre Kent Beck e Ward Cunningham. Durante os anos 80, estes autores refinaram um conjunto de práticas em numerosos projetos, evoluindo suas idéias nos anos 90 para uma abordagem de desenvolvimento de software adaptativa e orientada a pessoas [FOWLER, 2000b].

Segundo Kent Beck [2000], XP foi “teorizada” para seu formato atual após diversos êxitos em projetos reais. A motivação para a criação de XP foi a necessidade de ciclos de desenvolvimento cada vez mais curtos [BECK 2000]. Assim, a principal atividade executada durante um projeto de desenvolvimento de software em XP é a codificação – daí o uso do termo “*programming*” no nome. O outro termo que forma o nome da metodologia – “extreme” – vem do uso extremado de um conjunto de boas práticas de desenvolvimento.

3.3.1. PRÁTICAS

Individualmente, as práticas que constituem XP não são novas. Muitos autores chegaram a conclusões similares sobre a melhor forma de se desenvolver software em ambientes onde há grande volatilidade dos requisitos [BECK, 2000]. A principal contribuição de XP está na aplicação extremada destas práticas dentro de um modelo coeso e sinérgico. Os tópicos seguintes descrevem estas práticas.

O JOGO DO PLANEJAMENTO

O planejamento em XP baseia-se num diálogo constante entre especialistas de negócios (cliente) e tecnologia (programadores.) Nesta interação, tecnólogos estimam o tempo para a implementação das funcionalidades – que em XP são chamadas de histórias – enquanto os especialistas de negócio criam histórias e as priorizam em versões do software.

RELEASES CURTOS

Em linhas gerais, um *release* é uma versão do software apta a ser implantada e utilizada. Em XP, as *releases* devem ocorrer com a maior brevidade possível, em intervalos de no máximo dois ou três meses. Idealmente, versões intermediárias devem ser geradas diariamente com o intuito de manter o software sempre integrado e pronto para um *release*.

METÁFORAS

Todo projeto de desenvolvimento de software utilizando XP baseia-se numa metáfora para descrever a arquitetura da solução. Quaisquer palavras utilizadas para designar entidades técnicas devem ser obtidas a partir da metáfora utilizada. O objetivo desta abordagem é oferecer a todos os envolvidos no projeto um vocabulário único e uma história concreta com a qual trabalhar, viabilizando a comunicação efetiva e clara entre

programadores e clientes. Em outras palavras, uma metáfora representa uma arquitetura que é facilmente comunicada e simples de elaborar.

MODELO SIMPLES

Em XP parte-se do princípio de que o futuro é incerto e que mudanças podem ser feitas de forma simples e com baixo custo. Assim, incluir funcionalidades não necessárias apenas complica o modelo e o torna mais complexo e de difícil manutenção. A regra é incluir o que for desejável apenas quando isso for necessário [BECK, 2000].

TESTES

Testes são elementos na estrutura de XP. A metodologia enfatiza a criação de testes unitários automatizados, codificados antes mesmo da implementação da funcionalidade. Toda funcionalidade deve possuir testes unitários automatizados, e para que uma nova funcionalidade seja colocada em produção, a versão integrada com a nova funcionalidade deve passar em todos os testes. Testes funcionais, escritos pelo cliente, e testes de integração também desempenham importante papel em XP.

REFACTORING

Refactoring é o processo de modificar o código de um software de tal maneira que seu comportamento externo não seja alterado mas sua estrutura interna seja aprimorada. Trata-se de uma abordagem disciplinada para tornar o código de um software mais claro e de fácil manutenção, minimizando a probabilidade de inclusão de erros. Em essência, quando se faz o *refactoring* de um código, aprimora-se seu modelo após sua codificação [FOWLER, 2000a].

PROGRAMAÇÃO EM PARES

Todo código de produção deve ser escrito por duas pessoas trabalhando numa mesma máquina, com um teclado e um mouse [BECK, 2000]. Os pares devem ser dinâmicos, com

constante troca de papéis e dos membros de cada par. Programadores responsáveis pela realização de tarefas em áreas do software com as quais eles não estejam familiarizados devem solicitar a ajuda a programadores mais experientes, o mesmo se aplicando a tecnologias ou técnicas específicas.

POSSE COLETIVA

Em XP, toda a equipe é responsável pelo sistema como um todo. Todos os membros da equipe podem modificar qualquer parte da base de código do software.

INTEGRAÇÃO CONSTANTE

Toda a base de código deve ser integrada e testada em intervalos de poucas horas ou, no máximo, a cada dia. XP orienta que haja uma máquina dedicada a esta atividade e sempre que um par de programadores tenha código para ser integrado e testado, eles usem essa máquina para carregar a versão atual, integrar suas modificações (resolvendo qualquer problema de colisão) e executar todos os testes até que a base de código passe por 100% dos testes.

40 HORAS POR SEMANA

Beck [2000] descreve esta prática da seguinte forma: “Eu quero estar descansado e motivado todas as manhãs, e cansado e satisfeito todas as noites. Na sexta, eu quero estar cansado e satisfeito o suficiente para passar o final de semana pensando em outras coisas além do trabalho. Então, na segunda, quero voltar totalmente renovado e cheio de idéias.”

EQUIPE COMPLETA

Em XP, todos os colaboradores do projeto devem fazer parte de uma equipe única, alocada fisicamente no mesmo local. Esta equipe deve possuir um especialista de negócio trabalhando em tempo integral com a equipe de projeto [JEFFRIES, 2003]. Originalmente, esta prática era denominada “cliente participante da equipe” [BECK, 2000].

PADRÕES DE CODIFICAÇÃO

Todo projeto XP deve basear-se num sólido e estável conjunto de padrões e estilo de codificação e documentação interna, tornando os códigos escritos num documento claro para a troca de informações entre programadores.

3.3.2. PAPÉIS

XP baseia-se num diálogo constante e evolutivo entre programadores e clientes, os principais papéis realizados pelas pessoas num projeto XP. Através de seu conjunto de práticas, clientes e programadores estão em contato constante, trabalhando lado-a-lado na construção do software. XP valoriza a “comunicação quente”, ou seja, através de contato real, e não via documentos ou outras formas de comunicação que não o diálogo face-a-face.

Embora programador e cliente desempenhem os papéis centrais no projeto, outros papéis suportam o conjunto de práticas e a eficiência do processo. Mais que um conjunto de atividades, cada papel em XP tem um conjunto de responsabilidades. As minúcias da organização da equipe são deixadas totalmente a cargo da equipe, que deve se auto-organizar para resolver o problema enfrentado da melhor forma. Cada membro da equipe deve assumir a responsabilidade pelo sucesso do projeto, esforçando-se ao máximo e de todas as formas para atingir este objetivo.

Os tópicos seguintes descrevem os papéis que as pessoas ocupam em XP segundo [BECK, 2000].

PROGRAMADOR

O programador é um papel chave em XP, sendo de sua responsabilidade todas as atividades técnicas realizadas no projeto. Cabe ao programador a codificação dos testes unitários e do software em si, as estimativas para a implementação de estórias (que em XP

significam funcionalidades definidas pelo usuário), a definição de riscos e impactos técnicos, entre outras.

CLIENTE

O cliente é o outro lado da dualidade essencial de XP. Os programadores sabem como fazer o software. Os clientes sabem o que deve ser feito.

Cabe ao cliente a definição das histórias e seus testes de aceite, a priorização de histórias, e direcionamento do projeto. Em XP, o cliente faz parte do time e tem grande responsabilidade pelo sucesso do projeto. Todos os requisitos são criados pelo cliente na forma de histórias e respectivos testes funcionais. Desta forma, o cliente influencia decisivamente o projeto, mas não tem controle sobre ele.

TESTADOR

Como os programadores são responsáveis por grande parte dos testes do projeto, o testador tem como atribuição auxiliar o cliente a selecionar e definir os testes funcionais. O testador também deve executar os testes regularmente e divulgar os resultados e a qualidade do conjunto de testes disponível.

CONTROLADOR

O controlador é o historiador de uma equipe XP. Cabe ao controlador manter registros de estimativas, realizações, reporte de erros e correções, quais os testes adicionados ao projeto e notas de entrega. De posse destas informações, o controlador deve saber informar qual o índice de acuidade das estimativas de cada programador, qual o índice de erros no sistema, se este número está aumentando ou diminuindo e qual o impacto da correção de erros nas estimativas e andamento do projeto.

ORIENTADOR

O orientador é responsável pelo processo como um todo, atuando como facilitador junto ao time. Nos momentos de dificuldades ou caos, cabe ao orientador auxiliar o time a se organizar e a oferecer alternativas e boas práticas para a solução de problemas. O orientador deve ter um perfil conciliador, voltado ao trabalho em equipe e, acima de tudo, deve saber manter-se calmo nos momentos de maior pressão.

CONSULTOR

O consultor é um membro externo à equipe que participa do projeto para suprir a falta de conhecimento ou experiência em alguma tecnologia ou área de conhecimento.

CHEFE

O chefe é o responsável pelas decisões finais do projeto e o líder da equipe no que tange às questões políticas do projeto.

3.3.3. PROCESSO

O ciclo de vida de um projeto XP inclui cinco fases, quais sejam: exploração, planejamento, iterações para um *release*, preparação para produção e morte [BECK, 2000].

Durante a fase de exploração, os clientes escrevem a maior quantidade possível de histórias, mas, sobretudo, as histórias a serem implementadas na primeira *release*. Simultaneamente, a equipe de projeto desenvolve alguns protótipos para validar a arquitetura, obter familiaridade com a tecnologia, validar idéias, mitigar riscos e avaliar viabilidade. Dependendo do porte do projeto, a duração desta fase pode variar de semanas a meses.

Na fase de planejamento, que leva apenas alguns dias, a equipe de projeto estima o tempo necessário para o desenvolvimento de cada uma das histórias escritas pelo cliente. De posse destas estimativas e da velocidade da equipe, isto é, da capacidade produtiva por iteração, o cliente organiza as histórias em versões, e essas versões em *releases*.

Normalmente esses planejamentos incluem o cronograma de um *release*, que dura cerca de dois meses.

A fase de iterações para um *release* baseia-se em uma série de iterações que executam o plano elaborado na fase anterior. Essas iterações duram entre uma e quatro semanas, dependendo da atividade a ser realizada. As histórias a serem implementadas, bem como sua ordem de execução, são definidas pelo cliente com auxílio dos programadores, que fazem considerações técnicas sobre a ordem de implementação e riscos envolvidos. A primeira iteração deve ser focada na construção completa da arquitetura do projeto sendo, portanto, a fase onde há maior influência da equipe técnica na seleção das histórias. Ao final de cada iteração, os testes funcionais escritos pelos clientes são avaliados, erros e considerações são escritos como novas histórias e, eventualmente, planejadas para a próxima iteração.

Durante a fase de preparação para produção, o sistema passa por uma nova bateria de testes e avaliações de performance e disponibilidade antes de ser entregue. Apesar de modificações ocorrerem, o foco é na estabilização do sistema. Logo, a duração das iterações durante esta fase costuma ser de apenas uma semana.

Uma vez colocado em produção, o sistema é mantido em fase de manutenção. Nesta fase são comuns atividades de suporte ao cliente, como treinamento e apresentações, paralelamente às atividades de desenvolvimento. Como a carga de trabalho aumenta, durante esta fase são comuns ajustes na estrutura da equipe, com a inclusão de mais pessoas.

Quando o cliente não tiver mais histórias para serem implementadas, inicia-se a fase de morte. Neste momento o projeto entra em sua fase de ajustes finais para satisfazer o cliente, incluindo atividades como elaboração de manuais e documentação. Esta fase também pode ocorrer em condições onde o sistema não atenda às expectativas do cliente ou o orçamento disponível se encerre.

3.4. FDD – FEATURE DRIVEN DEVELOPMENT

Como todos os Métodos Ágeis, FDD é um processo voltado à entrega freqüente de resultados tangíveis [PALMER & FELSING, 2002]. FDD é focado nas atividades de desenvolvimento de software, dando ênfase a mecanismos de controle e divulgação de informação sobre o projeto [PALMER & FELSING, 2002]. O objetivo é oferecer a todos os envolvidos no projeto uma visão clara do software em criação ou evolução, seja através do próprio software, com entrega constante de *releases*, seja através de dados sobre a situação do projeto, como índice de completude e plano de *releases*.

Ao contrário da maioria dos demais Métodos Ágeis, FDD promete ser altamente escalável, tendo sido aplicado em projetos com centenas de pessoas [PALMER & FELSING, 2002]. FDD enfatiza a criação inicial de um modelo global do projeto e de uma lista de funcionalidades que direciona todo o projeto. A organização do processo, seus papéis e práticas garantem a estrutura necessária para o uso de FDD em projetos de grande porte [PALMER & FELSING, 2002].

3.4.1. PRÁTICAS

A maioria das práticas de FDD não são novas e nem inéditas, mas a organização do processo, aliada a algumas características peculiares, o tornam uma abordagem única e diferenciada [PALMER & FELSING, 2002]. Em particular, a prática de modelagem de objetos do domínio, descrita a seguir, baseia-se na utilização de um padrão de projeto¹⁴ denominado DNC¹⁵, ou Componente independente de domínio [COAD *et al*, 1999]. Este padrão é composto por um conjunto de arquétipos – classes que seguem, aproximadamente, a um padrão, compartilhando algumas características semânticas básicas –, oferecendo uma estrutura genérica para qualquer tipo de problema de negócio [COAD *et al*, 1999].

Os tópicos seguintes descrevem as práticas de FDD [PALMER & FELSING, 2002].

¹⁴ Do inglês *design pattern*.

¹⁵ Do inglês *Domain Neutral Component*

MODELO DE OBJETOS DO DOMÍNIO

Um modelo de objetos do domínio é uma representação de alto nível da solução definida para o problema enfrentado. Trata-se de um diagrama de classes que representa o arcabouço conceitual do problema, sobre o qual todas as funcionalidades serão construídas. Em FDD esta prática está intimamente relacionada com o padrão de projeto DNC, que oferece a base conceitual para a execução da atividade.

DESENVOLVIMENTO POR FUNCIONALIDADES

Todas as atividades técnicas e gerenciais em FDD baseiam-se em funcionalidades, ou seja, em requisitos do sistema com valor percebido pelo cliente. O planejamento do projeto, a ocupação técnica de atividades, o controle de andamento e divulgação de resultado do projeto se dá através de funcionalidades.

POSSE DE CÓDIGO (CLASSES)

Em FDD, cada classe tem um único responsável por sua implementação, consistência, performance e integridade. Toda funcionalidade que envolva alterações na porção de código detida por determinada pessoa deve ter, necessariamente, sua participação.

TIMES DE FUNCIONALIDADES

Intimamente relacionada às duas práticas anteriores, esta prática consiste na formação dinâmica de equipes pequenas, cujo objetivo é a implementação de uma funcionalidade. Uma vez implementada, testada e validada a funcionalidade, a equipe pode ser desfeita, com seus membros sendo alocados em outro time de funcionalidade. Uma mesma pessoa pode participar de mais de um time de funcionalidade a cada instante, sendo que estes times são, idealmente, totalmente independentes.

INSPEÇÕES

Em FDD, existem inspeções formais na lista de funcionalidades (realizada pelo cliente), nos modelos gerados e códigos implementados (realizados, na grande maioria dos casos, por membros da equipe de projetos.)

INTEGRAÇÃO CONSTANTE

Trata-se de manter uma versão com qualidade de produção sempre disponível. Esta versão constitui a linha de base sobre a qual novas funcionalidades são adicionadas. Assim, além de minimizar riscos de integração e servir como referência, a última versão funcional é o principal artefato para a demonstração do estado do projeto.

GERÊNCIA DE CONFIGURAÇÃO

Uma gestão de configuração eficiente é vital para o sucesso de FDD. Em virtude dos ciclos curtos, deve-se poder voltar versões de forma simples e rápida. A integração constante exige um mecanismo altamente funcional para a gestão das linhas de base. Soma-se a isso o grande dinamismo da equipe, que exige formas efetivas de controle de versões do software.

VISIBILIDADE DOS RESULTADOS

A última versão funcional do software é o principal artefato para apresentar o estado do projeto. Complementando este indicador, FDD oferece técnicas para a apresentação de progresso e planejamento para todos os níveis da organização, sempre baseando-se em funcionalidades como a unidade de controle central.

3.4.2. PAPÉIS

Os papéis sugeridos por FDD são divididos em três grupos: papéis chaves, papéis de suporte e papéis adicionais [PALMER & FELSING, 2002]. Os seis papéis chaves incluem

o gerente de projeto, arquiteto chefe, gerente de desenvolvimento, programador chefe, dono de classe e especialista no domínio. Os papéis de suporte são cinco, e incluem o gerente de *release*, guru, engenheiro de construção, administrador do sistema e responsável por ferramentas. Testadores, “implantadores” e redatores técnicos são papéis adicionais necessários em todo projeto usando FDD.

Em FDD, um membro da equipe pode desempenhar diversos papéis, da mesma forma que um único papel pode ser assumido por um grupo de pessoas. Os tópicos seguintes descrevem estes papéis [PALMER & FELSING, 2002].

GERENTE DE PROJETOS

Em FDD, o gerente de projetos é o líder administrativo da equipe, responsável por informar o progresso do projeto, gerenciar o orçamento disponível, definir e conseguir recursos, espaço e pessoas de acordo com as necessidades impostas pelo projeto. De maneira geral, o papel do gerente de projetos é criar e manter um ambiente onde os demais membros da equipe de projetos possam trabalhar de forma produtiva.

O gerente de projetos em FDD não força as pessoas a trabalharem, ao contrário, oferece os recursos para que elas trabalhem. Além de disponibilizar o ambiente necessário, cabe ao gerente de projetos isolar a equipe de fatores e pressões externas. É responsabilidade do gerente de projetos resolver todo impasse envolvendo escopo, cronograma e alocação dentro do projeto. Assim, é o gerente de projetos quem direciona o projeto através dos obstáculos administrativos e financeiros enfrentados.

ARQUITETO CHEFE

O arquiteto chefe é o responsável pelo modelo global do projeto. Apesar de ser dele a última palavra no que se refere às questões técnicas do projeto, quando a equipe não chega a um consenso, seu papel é eminentemente de um facilitador e líder, sendo o

responsável por coordenar reuniões e atividades em grupo onde a equipe colabora no desenvolvimento do modelo da solução.

Este é um papel profundamente técnico, exigindo sólidos conhecimentos e experiência da pessoa que o desempenha, além de habilidades de liderança e coordenação. Nas situações onde o problema a ser resolvido ou a arquitetura tecnológica envolvida é muito complexa, este papel pode ser quebrado em *Arquiteto de domínio* e *Arquiteto técnico*. Desta forma, cabe ao arquiteto chefe direcionar o projeto através dos obstáculos técnicos enfrentados, complementando o papel do gerente de projetos no que tange às atividades técnicas.

GERENTE DE DESENVOLVIMENTO

Este é um papel que exige bons conhecimentos técnicos e habilidades de coordenação. A principal função do gerente de desenvolvimento é a coordenação do dia-a-dia das atividades de desenvolvimento, resolvendo conflitos por alocação de recursos quando os programadores chefe não a fazem sozinhos.

Por envolver atividades relacionadas tanto com o universo técnico do projeto como com atividades administrativas, em alguns projetos este papel pode ser executado pela mesma pessoa que atua como gerente de projetos ou arquiteto chefe. A última palavra no que se refere à alocação de recursos técnicos é do gerente de desenvolvimento, cabendo a ele ou ela a resolução de qualquer conflito por alocação de recursos.

PROGRAMADOR CHEFE

Um programador chefe é um desenvolvedor de software experiente, que já tenha passado algumas vezes por todas as etapas do desenvolvimento de software. O programador chefe participa das atividades de análise e modelagem de alto nível e é

responsável por coordenar equipes pequenas, de até 6 pessoas, nas atividades de detalhamento destas atividades e implementação de funcionalidades.

Programadores chefe têm como principal característica serem motivados a produzir resultado de qualidade num prazo determinado. Combinam excelente capacidade técnica com suficiente habilidade de coordenação para liderar suas equipes rumo à produção de resultado a cada poucos dias. Programadores chefe também atuam com outros programadores chefe para resolver questões técnicas e de alocação e, em geral, são pessoas que gozam de grande respeito tanto entre outros desenvolvedores como entre seus gerentes.

Coad [1999] afirma que programadores chefe são uma exceção à regra de que adicionar mais pessoas a projetos atrasados, apenas os atrasa mais [BROOKS, 1995]. Segundo ele, programadores chefe, graças a seu foco em resultado e habilidades técnicas, podem e devem ser adicionados a projetos com problemas de prazo, com resultados altamente positivos.

DONO DE CLASSE

Donos de classe são desenvolvedores que atuam como membros de pequenas equipes de desenvolvimento sob a liderança de um programador chefe. Cada dono de classe participa das atividades de modelagem, codificação, teste e documentação das funcionalidades exigidas pelo sistema em desenvolvimento.

Em geral, os donos de classe são excelentes desenvolvedores de software. Entretanto, existem dois tipos básicos de donos de classe: aqueles que, com mais experiência, podem tornar-se programadores chefe, arquitetos ou até mesmo gerentes, e aqueles que não se interessam por atividades gerenciais e de coordenação, sendo focados em tornarem-se mestres em determinada tecnologia.

ESPECIALISTA NO DOMÍNIO

Especialistas do domínio são usuários, clientes, patrocinadores, analistas de negócio ou qualquer variação destes papéis. Sua atribuição primordial em FDD é explicar à equipe de desenvolvimento, em diversos níveis de detalhe, as tarefas que o sistema em desenvolvimento precisa executar. Assim, os especialistas do domínio fazem às vezes da base de informações a qual a equipe de desenvolvimento recorre para entregar o sistema correto.

As pessoas atuando neste papel devem ter boa capacidade de explanação tanto verbal como escrita, realizando seminários e escrevendo documentos que direcionem a equipe rumo à solução idealizada. O sucesso do projeto é totalmente dependente do conhecimento e participação efetiva dos especialistas do domínio. Desta forma, é importante que as pessoas executando este papel sejam altamente entusiasmadas com as possibilidades do sistema em desenvolvimento e tenham paciência e tolerância com a equipe de desenvolvimento.

GERENTE DE *RELEASE*

Os gerentes de *release* recebem este nome por ser sua responsabilidade reunir-se com a equipe constantemente para determinar as funcionalidades entregues no último *release*. Assim, a principal responsabilidade deste papel é garantir a visibilidade do status do projeto, comparando planejado versus realizado e mantendo todos os interessados informados sobre o andamento do projeto.

GURU

É um membro da equipe de dispõe de profundo conhecimento sobre alguma área específica utilizada amplamente no projeto, como por exemplo, uma linguagem de programação. Este papel ganha importância quando o projeto utiliza alguma nova tecnologia não dominada pela equipe.

ENGENHEIRO DE CONSTRUÇÃO

O engenheiro de construção é o responsável por preparar, manter e executar o processo de construção de *releases*, incluindo atividades envolvendo o controle de versões e o gerenciamento dos documentos de *release*.

RESPONSÁVEL POR FERRAMENTAS

Este papel tem como responsabilidade a definição e manutenção das ferramentas utilizadas pela equipe de projeto, incluindo ferramentas comerciais, bancos de dados e *web sites* utilizados no projeto, e pequenos aplicativos desenvolvidos para automatizar tarefas, como testes e geração de código.

ADMINISTRADOR DO SISTEMA

O administrador do sistema é responsável por manter toda a infra-estrutura necessária para o trabalho da equipe, como rede, servidores, estações de trabalho, ambientes de homologação e produção. O administrador do sistema participa também das atividades relacionadas à implantação do sistema em produção.

TESTADOR

A principal responsabilidade dos testadores é fazer a avaliação funcional do sistema em desenvolvimento, garantindo que este atenda às necessidades do cliente. Este papel pode ser executado por uma equipe separada ou por membros da equipe de projeto.

IMPLANTADOR

Em linhas gerais, o trabalho do “implantador” é colocar novas *releases* em produção, realizando todas as atividades necessárias para este fim, como conversão de dados e preparação de ambientes. Este papel pode ser executado por uma equipe separada ou por membros da equipe de projeto.

REDATOR TÉCNICO

Responsável pela preparação dos documentos a serem entregues ao cliente, como manuais e documentação. Pode ser executado pela equipe de projetos ou por equipe independente.

3.4.3. PROCESSO

FDD é composto por cinco processos: desenvolva um modelo global, construa uma lista de funcionalidades, planeje por funcionalidade, modele for funcionalidade e construa por funcionalidade.

O primeiro processo - Desenvolva um modelo global - tem como objetivo criar um modelo de classes de alto nível que represente a solução para o problema enfrentado. Este modelo é criado conjuntamente entre especialistas no domínio, programadores e arquitetos e funciona como o arcabouço do projeto, sobre o qual todas as funcionalidades são adicionadas. Este processo baseia-se em especificações de requisitos pré-existentes, em sessões de modelagem conjuntas, semelhantes a sessões JAD, e na prática de modelagem de domínio, fazendo amplo uso do padrão de projetos DNC [COAD *et al*, 1999].

O modelo global desenvolvido mais as especificações pré-existentes fornecem a base para o processo seguinte: construa uma lista de funcionalidades. Este processo consiste na criação de uma lista contendo todas as funções com valor para o cliente que o sistema deve oferecer. As funcionalidades listadas são organizadas em grupos, denominadas conjuntos de funcionalidades, e estes são agrupados em grupos maiores, denominados grandes grupos de funcionalidades. A lista de funcionalidades é objeto de revisões constantes, tanto por parte da equipe, como por parte do cliente, com o objetivo de validar sua completude e validade.

Uma vez criados o modelo global e a lista de funcionalidades, o processo seguinte, Planeje por funcionalidades, estabelece um planejamento de alto-nível para o projeto. Este planejamento é realizado de acordo com as prioridades definidas e dependência entre funcionalidades. O plano de projeto criado inclui a atribuição de funcionalidades a

programadores chefe e classes a programadores (donos de classe). Durante este processo, também são definidos marcos e pontos de controle para o projeto.

Os três processos descritos anteriormente são executados de forma eminentemente sequencial, apesar de ocorrerem com menos frequência durante todo o projeto. Os processos seguintes - Modele por funcionalidade e Construa por funcionalidade - são processos iterativos, ocorrendo em períodos de tempo que vão de algumas horas a, no máximo, duas semanas.

O processo modele por funcionalidade visa refinar o modelo global, criado no processo 1. Este refinamento engloba apenas as áreas do modelo que tenham relação com as funcionalidades a serem implementadas, e envolve, basicamente, refinar o modelo de classes com métodos e atributos das funcionalidades a serem construídas e diagramas de sequência mostrando o relacionamento dinâmico do modelo para as funcionalidades a serem construídas. Este processo inclui também atividades de revisão do modelo.

A partir do modelo detalhado criado, o processo seguinte, construa por funcionalidade, consiste na implementação das funcionalidades. Este processo inclui tarefas como codificação, inspeções, testes unitários, revisão de código, integração e implantação. Após uma iteração bem sucedida, as funcionalidades desenvolvidas são promovidas para produção, passando a fazer parte da *release* do projeto.

3.5. CONSIDERAÇÕES FINAIS

Neste capítulo os Métodos Ágeis *Scrum*, XP e FDD foram analisados em termos dos papéis, práticas e processo que os definem. Entretanto, existem diversos outros métodos classificados como ágeis – *Adaptative Sofwtare Development*, *Lean Development*, *Dynamic System Development Method*, *Crystal family of methodologies*, *Internet-speed Development*, software livre, *dx*, *xBreed*, *Grissly* - além de trabalhos que não constituem metodologias propriamente ditas – *Agile Modeling*, *Pragmatic Programming* -, mas são compatíveis com os valores dos Métodos Ágeis.

Nos capítulos seguintes é apresentada ECO, uma metodologia ágil para o desenvolvimento de Sistemas Web. ECO baseia-se nas metodologias ágeis descritas neste capítulo, sobretudo FDD, mas acrescenta papéis, práticas e atividades focadas no desenvolvimento de Sistemas Web. ECO também é mais abrangente que as metodologias apresentadas, cobrindo atividades anteriores ao desenvolvimento propriamente dito, e posteriores a ele.

CAPÍTULO 4: ECO PARTE I: CONCEITOS

4.1. CONSIDERAÇÕES INICIAIS

O capítulo anterior descreveu três dos principais Ecossistemas de Desenvolvimento Ágil de Software existentes, descrevendo-os em termos de suas práticas, papéis e processo. Estes três ecossistemas foram detalhados por serem eles os que mais tiveram influência neste trabalho, cada um deles emprestando a ECO, o Ecossistema de Criação Ágil de Sistemas Web proposto neste trabalho, alguns de seus conceitos. Ao longo dos próximos três capítulos, onde será feita a descrição do estado atual de ECO, serão encontrados vários dos conceitos expostos no capítulo anterior, alguns utilizados de forma idêntica, outros com alterações superficiais e alguns apenas referenciando vagamente sua utilização original.

A descrição de ECO será feita em três partes. Neste capítulo é feita uma introdução e preparação conceitual sobre o Ecossistema, descrevendo seu escopo, conceitos básicos e principais práticas. No capítulo seguinte, descrevem-se os processos que compõem o Ecossistema e as atividades fundamentais que os agrupam. Em seguida, no Capítulo 6, descrevem-se os resultados da aplicação continuada de ECO por cerca de 2 anos em uma empresa desenvolvedora de Soluções Web.

Em tempo: Nos próximos capítulos os termos Sistema Web e Sistemas Web serão abreviados, indistintamente, como SW.

4.2. ESCOPO

Uma visão que vem se difundindo é a de que não existe uma única metodologia adequada a qualquer situação. Em virtude da amplitude de utilização dos Sistemas Computacionais e da variedade de ambientes em que estes são desenvolvidos, pode-se dizer que uma metodologia é uma questão situacional. Nos SW, a situação é semelhante – atualmente existem SW dos mais complexos e críticos, como sistemas financeiros e de intervenções cirúrgicas remotas, até os mais simples e despretensiosos, como *websites* pessoais ou meramente informativos.

O foco de ECO, e dos Ecossistemas Ágeis em geral, é o desenvolvimento de SW onde a velocidade de entrega de resultados tem prioridade sobre a robustez do produto gerado. De forma bastante ampla, ECO é voltado para aplicações de marketing e negócios na web, como ações promocionais, *e-commerce*, publicação de conteúdo em geral e diversas outras aplicações envolvendo ações de relacionamento entre empresas e pessoas.

Em termos de amplitude metodológica, ECO cobre todos os processos de um projeto, desde a aquisição até o seu encerramento, sendo adaptável para direcionar o desenvolvimento de produtos em qualquer etapa de seu Ciclo de Vida, conforme exposto

Figura 3 apresenta o escopo de ECO em comparação com os ecossistemas de desenvolvimento ágil descritos no capítulo anterior, segundo diagrama proposto por [ABRAHAMSSON et al, 2003].

Figura 3: Escopo de ECO em relação aos Ecossistemas descritos no capítulo 3

ECO foi desenvolvido para a seguinte situação geral:

- Equipes pequenas, entre 1 e 10 pessoas.
- Equipes compostas por pessoas atuando de forma compartilhada em vários projetos simultaneamente.

- Planejada para pequenas empresas de desenvolvimento de SW ou departamentos responsáveis por projetos de SW em empresas de maior porte.
- Projetos onde o tempo de entrega é o principal fator de sucesso.

4.3. PÚBLICOS

A definição dos Públicos é uma atividade chave no desenvolvimento de Sistemas Web. Um Público (ou Público-alvo) representa um conjunto de usuários do sistema com objetivos similares, mas não idênticos. Por exemplo, duas pessoas pertencentes ao Público “consumidores” podem receber ofertas diferentes em um Sistema de Web de *e-commerce* de acordo com suas preferências pessoais e histórico de transações junto ao sistema. Neste sentido, um Público determina um modelo geral da camada de Experiência Humana de um Sistema Web, definindo a estrutura da informação, modelo de navegação, linha de comunicação visual e textual, sem, no entanto, definir as especificidades de cada pessoa em particular. Cada membro de um Público tem acesso a Conteúdos e Serviços distintos, idealmente personalizados de acordo com suas características pessoais. Neste sentido, um Público é uma abstração de um determinado grupo de usuários, restringindo o conjunto de informações e recursos a que estes têm acesso, sem, no entanto, detalhar ou restringir a experiência pessoal de cada pessoa. Em outras palavras, um Público define regras gerais de navegação, autenticação e autorização, regras estas que podem ser modificadas para cada pessoa em particular.

O conceito de Público distingue-se do de Ator [BOOCH *et al*, 1999] por ser mais genérico que este, definindo entidades que interagem com o Sistema Web de maneira bastante abstrata. Um Público, mais que definir as possíveis interações com o sistema, visa capturar um perfil demográfico e comportamental de um conjunto de pessoas que podem vir a interagir com um determinado Sistema Web, oferecendo a estes um conjunto rico de Conteúdos e Serviços capazes de atender às suas expectativas. Diferentemente do conceito de Ator, onde existe certa preocupação com a estrutura e atomicidade dos papéis de cada Ator, Públicos distintos podem ter, e normalmente tem, sobreposição de papéis. Assim, um Público reflete um segmento (no sentido de marketing da palavra) do universo de pessoas que podem vir a utilizar o SW.

Como veremos a seguir, em termos de Conteúdos, os Públicos de um SW auxiliam na definição da linha de comunicação a ser adotada. Em termos de Serviços, os Públicos auxiliam no mapeamento de alto nível dos mecanismos de segurança, atribuindo a cada membro do Público um conjunto inicial de credenciais de autorização.

4.4. FUNCIONALIDADES

O conceito de Funcionalidade é semelhante ao de Estória utilizado em XP [JEFFRIES *et al*, 2001], ou seja, é uma breve descrição de um comportamento esperado do sistema, sob o ponto de vista dos usuários do mesmo. As Funcionalidades são descritas utilizando-se a linguagem característica de seus Públicos, com termos e referências advindas do dia-a-dia destas pessoas, sem a utilização de termos técnicos ou jargões alheios à sua realidade. Idealmente, as funcionalidades são escritas por representantes dos Públicos, ou por estes em conjunto com a Equipe de Projeto. Em FDD [PALMER & FELSING, 2002], uma Funcionalidade é descrita no formato <ação><resultado><objeto>, com as devidas preposições entre a ação, resultado e objeto. Em ECO, este formato é uma referência para a criação do título de uma Funcionalidade ou para a descrição de um Serviço.

O tempo de implementação de uma Funcionalidade deve ser facilmente estimável pela Equipe de Projeto, levando poucas horas ou dias para ser completada, caso contrário, deve-se dividi-la em Funcionalidades menores. Depois de implementada, uma Funcionalidade deve entregar valor tangível para o cliente, sendo, portanto, um valioso recurso de acompanhamento do andamento do projeto na medida em que permite a visualização do estado atual do SW através de código funcional e utilizável.

Em contraste com Casos de Uso [COCKBURN, 2001], Funcionalidades são descrições um tanto quanto informais dos requisitos do sistema, sendo compostas por um título e um breve texto sobre seu objetivo. Ao contrário dos Casos de Uso, a descrição de uma Funcionalidade é breve e direta, não possuindo uma estrutura definida e, via de regra, não sendo descrita na forma de um diálogo entre um Ator e o sistema. Entretanto, caso a

equipe de projeto ache importante, nada impede uma descrição mais longa e formal para uma ou mais Funcionalidades. O importante é ter em mente que, como veremos a seguir neste capítulo, a descrição de uma Funcionalidade como Artefato tem como objetivo último trazer à tona o contexto da discussão que a gerou, remetendo a equipe a pontos específicos da teoria criada para resolver o problema em questão.

Por serem escritas com poucas palavras, de forma textual e livre, a descrição de uma Funcionalidade é superficial e sujeita a ambigüidades, podendo gerar problemas de comunicação. Entretanto, isto não é um problema visto que uma Funcionalidade, antes de uma especificação, representa, segundo a descrição de Alistair Cockburn para uma Estória em XP, “uma promessa para futuras conversações entre desenvolvedores e clientes” [BECK & FOWLER, 2001]. Ademais, a brevidade e informalidade na descrição de uma Funcionalidade evita alguns dos principais problemas enfrentados pelos Casos de Uso, quais sejam:

- Dificuldade de definição da granularidade ideal de cada Caso de Uso;
- Indução à estruturação pré-matura dos requisitos através do uso de estereótipos `<<include>>` e `<<extend>>` [BOOCH *et al*, 1999];
- Descrições semi-estruturadas trabalhosas de se criar e que, via de regra, geram documentos extensos, com dezenas de páginas, difíceis de manter e atualizar;

De fato, a utilização rigorosa de Casos de Uso em projetos com requisitos em constante mudança pode levá-lo a um estado de *Analysis Paralysis* [BROWN *et al*, 1998], onde a constante (re)criação de artefatos de análise diminui o ritmo de entrega de código e resultado tangível a ponto de colocar em risco o sucesso do projeto.

Em ECO o desenvolvimento de uma Funcionalidade é feito através da criação e implementação dos Conteúdos e Serviços que a compõem. Os tópicos seguintes descrevem esses conceitos.

4.4.1. CONTEÚDOS

Em ECO, um Conteúdo é todo objeto visual de um Sistema Web, como textos, imagens, formulários, botões, animações, etc., ou seja, um Conteúdo é a parte interativa de uma Funcionalidade, a interface entre o mundo exterior e o Sistema Web. Com efeito, o resultado da experiência humana em um SW baseia-se na interação de seus Públicos com os diversos Conteúdos oferecidos. Por serem Sistemas Computacionais de uso generalizado, com potencial de serem utilizados por todos os tipos de pessoas, dos neófitos aos especialistas, de diversas culturas, regiões geográficas e culturas, o tratamento eficiente e eficaz dos Conteúdos em um Sistema Web é fator crítico para seu sucesso.

Do ponto de vista do desenvolvimento de Sistemas Web, a riqueza e complexidade dos Conteúdos a serem criados e tratados fez surgir uma nova gama de perfis profissionais, especializados neste domínio de conhecimento. Apesar de o estudo da Interação Homem-Computador não ser novo, os SW trouxeram novos desafios para este campo, culminando com o surgimento de uma nova classe de profissionais envolvidos no desenvolvimento de SW. Esses novos perfis, descritos a seguir, tem como principal objetivo a criação e organização dos Conteúdos de um SW.

4.4.2. SERVIÇOS

Um SW é um Sistema Computacional eminentemente distribuído, ou seja, seus componentes podem estar fisicamente separados, usando uma rede como meio de comunicação. Existem diversas maneiras de organizar e articular os componentes de um SW, entretanto, como veremos a seguir, ECO utiliza o conceito de Computação Orientada a Serviço [HUHNS & SINGH, 2005], que consiste, basicamente, no acesso remoto à interface de um objeto utilizando, via de regra, a tecnologia de *Web Services*.

Assim, em ECO, a implementação de Funcionalidades que envolvam algum tipo de processamento é realizada através de Serviços. Com efeito, esta característica induz o desenvolvimento orientado a serviço, uma boa prática em termos de separação de conceitos e flexibilidade.

4.5. AMBIENTE

Como qualquer ser vivo, um SW “vive” e se desenvolve em um ambiente com o qual mantém relacionamentos e dependências vitais. Qualquer mudança no ambiente gera impacto nos organismos que nele sobrevivem e, dependendo da abrangência e profundidade das alterações, diversos organismos podem não se adaptar à nova situação e morrer.

Os tópicos a seguir descrevem os principais componentes de um ambiente computacional no contexto de ECO. O ambiente externo, ou mercado, tem grande impacto sobre os SW, entretanto não entraremos em maiores detalhes sobre o ambiente externo. Para o propósito deste trabalho basta considerarmos que o ambiente externo impõe grande pressão para a evolução constante e rápida dos SW.

4.5.1. AMBIENTE LÓGICO

O Ambiente Lógico de um SW define as camadas lógicas que comporão o sistema. A divisão lógica em camadas é uma das técnicas mais utilizadas na Ciência da Computação para separar um sistema complexo em partes mais simples e gerenciáveis. Encontra-se este tipo de divisão lógica na arquitetura de computadores, onde linguagens de programação realizam chamadas ao sistema operacional, que por sua vez se comunica com *drivers* de dispositivo e instruções da CPU, e estes com portas lógicas em chips [FOWLER, 2003]. Outro exemplo é a *World Wide Web*, cujos mecanismos de comunicação baseiam-se no protocolo HTTP, que está sobre a camada TCP, a qual depende da camada IP que, por sua vez, utiliza-se da *ethernet* para trafegar informações.

Cada camada lógica tem um objetivo claro no SW, além de agrupar algumas características distintivas que auxiliam no gerenciamento do projeto. Neste sentido, cada camada do Ambiente Lógica de um SW possui características próprias no que tange a:

- Tecnologia
- Papéis
- Artefatos
- Processos

Existem várias abordagens para a divisão lógica de em camadas, cada qual utilizando diferentes quantidades de camadas, nomenclaturas e objetivos para cada uma [BROWN *et al.*, 2001; KIRTLAND, 1998; MARINESCU, 2002; NILSSON, 2002; FOWLER, 2003; PALMER & FELSING, 2002]. O Ambiente Lógico de ECO baseia-se no proposto por [ALUR *et al*, 2003], conforme ilustrado na Figura 3.

Figura 4: Ambiente Lógico de ECO

O foco de ECO está nas camadas de Experiência Humana, Camada de Aplicação e Camada de Negócios. A Camada de Recursos pode ser considerada como fazendo parte do Ambiente Operacional do SW, como veremos a seguir, enquanto que a Camada de Integração exige pouco esforço na medida em que novas tecnologia automatizam sua criação.

CAMADA DE EXPERIÊNCIA HUMANA

A simplicidade de uso e interatividade é uma característica chave de qualquer SW. A camada de Experiência Humana é responsável por estas características, representando qualquer dispositivo cujo objetivo seja viabilizar a interação humana com o SW. Via de regra, estes dispositivos são *browsers* HTML, mas, mais recentemente, vem proliferando-se outros dispositivos, como telefones celulares e *hand helds*.

CAMADA DE APLICAÇÃO

Esta camada encapsula a lógica de aplicação do SW, sendo responsável por capturar as requisições recebidas e fornecer as respostas solicitadas de forma personalizada, ou seja, receber e enviar informações de acordo com as características e perfil de cada Público do SW. Assim, enquanto a Camada de Experiência Humana tem como principal responsabilidade a interação com pessoas, a Camada de Aplicação é responsável por orquestrar a navegação pelo SW e as chamadas à Camada de Negócio, de acordo com as permissões e características de cada Público e dispositivo de acesso.

CAMADA DE NEGÓCIO

Todas as regras de negócio e lógica de processamento de dados são encapsuladas nesta camada, que oferece um conjunto de serviços relacionados a um determinado domínio de problema. Assim, os serviços necessários para atender às necessidades dos Públicos do SW estão nesta camada que, para este tipo de sistema, é organizada em forma de componentes isolados acessados via uma fachada de serviços, normalmente implementada utilizando-se a tecnologia de *Web Services*.

Os serviços oferecidos por esta camada são normalmente acessados pela Camada de Aplicação, quando o objetivo é atender a alguma requisição de um Público do SW, ou por uma Camada de Integração (descrita a seguir) quando os serviços oferecidos são utilizados como insumo de outras Camadas de Negócio. Por viabilizarem um modelo de computação altamente distribuído e com importantes implicações econômicas, este segundo caso está dando origem a uma nova área de estudo, conhecida como SOC (do inglês *Service-Oriented Computing* [HUHNS & SINGH, 2005]). Neste sentido, diversos SW atuais não incluem as Camadas de Aplicação e de Experiência Humana, visto que seu objetivo é somente a oferta de serviços para outros SW. Este tipo de SW vem ganhando importância crescente, com diversos modelos de negócio sendo baseados unicamente na oferta de serviços.

CAMADA DE INTEGRAÇÃO

Uma importante característica dos SW é a sua natureza componentizada, sendo, via de regra, criados pela integração de diversos sistemas distintos. Esta camada tem como objetivo prover acesso aos recursos e sistemas que fornecem dados e serviços utilizados pelo SW. Assim, a Camada de Integração é a responsável por tarefas que vão desde o acesso a um banco de dados relacional até a integração com *mainframes* ou o acesso à serviços externos ao SW, como descrito anteriormente no caso de soluções baseadas no modelo SOC.

CAMADA DE RECURSOS

Esta camada contém todos os recursos externos e dados utilizados pelo SW para atingir seus objetivos. Estes recursos podem ser desde uma base de dados relacional ou um repositório XML, até um *mainframes* e serviços oferecidos por outros sistemas.

4.5.2. AMBIENTE OPERACIONAL

O Ambiente Operacional define as características físicas e tecnológicas de um SW em termos de hardware e software. No que tange ao hardware, o Ambiente Operacional define o tipo de equipamento utilizado para “hospedar” o SW e as características de conectividade necessárias para seu funcionamento adequado. As definições de software indicam os sistemas operacionais, os componentes e subsistemas, linguagens de programação e ferramentas necessárias para o funcionamento do SW.

O Ambiente Operacional está intimamente relacionado ao Ambiente Lógico e, até certo ponto, dependente deste, na medida em que cada camada do Ambiente Lógico implica num conjunto restrito de opções para a definição do Ambiente Operacional. Em ECO, a dependência entre Ambiente Lógico e Operacional é representada através de uma matriz, denominada Matriz Lógico-Operacional (MLO). As linhas desta matriz trazem as camadas do Ambiente Lógico do SW, as colunas trazem as características que definem o

Ambiente Operacional, enquanto que em cada célula lista-se as opções selecionadas ou, mais comumente, a lista de opções possíveis (*Figura 5*).

Camada	HDW	Conectividade	SO	Comp./Subsis	Ling.	Fer.
EH	PC ou Mac, resolução de 800x600	Conexão banda larga (acima de 128 Kbs)	Windows, Mac, Linux	Flash 6.0, IE 6.0, Netscape 5.0, Firebird	HTML 4.0, DHMTL, Action Script	Macromedia MX
APP	Intel PIV, 1Gb RAM, 80 Gb disco	2 MB	Linux	Struuts, Tomcat 4.0	Java, JSP, Servlets,	Eclipse
NEG	Intel PIV, 1Gb RAM, 80 Gb disco	2 MB	Linux	JVM 2.0	Java	Eclipse
INT	Intel PIV, 1Gb RAM, 80 Gb disco	2 MB	Linux	Hibernate	Java	Eclipse
REC	Intel PIV, 1Gb RAM, 80 Gb disco	2 MB	Windows 2000 Adv Server	SQL Server 2000	SQL	Enterprise Mng

Figura 5: Exemplo de Matriz Lógico-Operacional.

4.6. CICLO DE VIDA

Entende-se por Ciclo de Vida todo o período de existência de um SW, desde a definição inicial de objetivos e propósito até a sua desativação e atividades posteriores. Assim, o termo refere-se ao SW como um produto ou entidade, e não às etapas ou detalhes do seu processo de desenvolvimento.

O Ciclo de Vida em ECO indica o processo de evolução de um SW, refletindo seu tempo de maturação e uso, principais desafios enfrentados em cada fase do Ciclo de Vida e características do SW em cada etapa do mesmo. Durante o Ciclo de Vida de um SW diferentes equipes podem atuar em sua evolução. Dependendo da etapa do Ciclo de Vida em que uma equipe assume o SW, os desafios impostos podem ser significativamente diferentes já que, em última análise, o desenvolvimento de um SW é um processo de aprendizado constante e a criação de uma teoria voltada à resolução do problema proposto [NAUR, 1985]. Neste sentido, evoluir um SW a partir de teorias formuladas por equipes distintas pode ser uma tarefa bastante complexa visto que o racional de muitas decisões é perdido à medida que a equipe original se dispersa. Esta característica faz com que em muitos casos a nova equipe force o abandono da solução atual em detrimento de uma nova.

De fato, em muitos casos esta opção pode ser a mais vantajosa em termos de custo e benefício.

ECO utiliza um Ciclo de Vida composto pelas etapas de Concepção, Gestaço, Nascimento, Infncia, Maturidade, Velhice, Morte e Legado. A seguir descrevem-se estas etapas em maiores detalhes.

4.6.1. CONCEPÇÃO

Na fase de Concepção definida a viso de longo prazo do SW, qual o problema a ser abordado, qual seu objetivo, Pblicos alvo, premissas, fatores crticos para o sucesso e Ambiente, conforme definido anteriormente. Por lanar as bases do SW, balizar todas as atividades subseqentes e definir a viso do projeto, o sucesso de um SW est, em grande medida, vinculado qualidade da etapa de Concepção.

4.6.2. GESTAÇÃO

Uma vez Concebido, inicia-se a criao do SW. O perodo compreendido entre a sua Concepção e lanamento inicial (ou Nascimento) denominado Gestao. Esta etapa caracteriza-se por atuar sobre o SW antes de seu lanamento oficial, ou seja, antes que este esteja em produo. Os principais objetivos so relacionados s definies do Ambiente do SW e codificao de um conjunto mnimo de Funcionalidades, suficientes para o lanamento de uma verso inicial. Muitas decises e abordagens adotadas nesta fase permearo o SW durante todo o restante do seu Ciclo de Vida. Assim, nesta etapa realizam-se importantes provas de conceito e definies. Em linhas gerais, um SW na etapa de Gestao j existe enquanto entidade sem, no entanto, possuir verses em produo.

4.6.3. NASCIMENTO

Mais que um marco ou a data em que o sistema entra em produo pela primeira vez aps a etapa de Gestao, o Nascimento um processo que inclui atividades feitas antes da

entrega oficial inicial (pré-natal) e após a entrega oficial inicial (pós-natal). Esta fase inclui a preparação do Ambiente Operacional de produção, treinamento dos Públicos, implantação e estabilização do SW. Diversas atividades realizadas nesta etapa são repetidas inúmeras vezes durante o Ciclo de Vida do SW na medida em que cada Ciclo de Desenvolvimento, definido a seguir, é terminado com um novo lançamento do SW. Entretanto, nos demais lançamentos, o SW já está “vivo” e em uso, possuindo, portanto, desafios bastante diversos dos enfrentados nesta fase.

4.6.4. INFÂNCIA

Após seu Nascimento, um SW entra na etapa de Infância. Nesta etapa do Ciclo de Vida o SW já está “vivo” (em produção), mas normalmente possui apenas um conjunto mínimo de Funcionalidades e ainda carece de ajustes de performance, estabilidade e disponibilidade. Analogamente a uma criança, o SW na etapa de infância possui todos os recursos de que precisará durante toda sua vida, porém ainda não os usa com total eficiência. Assim, Sistemas Web na etapa de Infância são acompanhados de perto pela Equipe de Projeto que o desenvolveu e sofrem atualizações num ritmo acelerado. SW recém-nascidos podem sofrer atualizações diárias durante vários meses e, em casos extremos, várias atualizações diárias durante longos períodos.

4.6.5. MATURIDADE

Nesta etapa de seu Ciclo de Vida, um SW está mais estável e previsível do ponto de vista de performance, estabilidade e disponibilidade. Apesar de um SW nesta etapa ainda estar em evolução, sofrendo atualizações e acréscimo de novas Funcionalidades, estas acontecem num ritmo mais lento que o visto na etapa de Infância. É relativamente comum um SW maduro sofrer alterações significativas em sua camada de Experiência Humana, visando trazer nova vitalidade e uma imagem de “renovação” para o produto, sem, no entanto, oferecer novas Funcionalidades e recursos interessantes. .

4.6.6. VELHICE

Em virtude de inovações tecnológicas, soluções mais adequadas para os problemas a que um SW se propõe a solucionar ou mesmo o término de alguma oportunidade sazonal, SW deixam de atender às exigências e necessidades de seus Públicos. Nesta etapa do Ciclo de Vida, denominada Velhice, um SW tem reduzida a sua aplicabilidade e volume de utilização. Consequentemente, sua evolução ocorre num ritmo muito lento, podendo inclusive cessar completamente. SW nesta etapa tendem a morrer (serem desativados) ou continuarem funcionando em condições mínimas, apenas o suficiente para atender a demandas históricas e pontuais. Ademais, SW nesta etapa normalmente são de difícil evolução em virtude de sua longa história de mudanças e, acima de tudo, pelos formuladores de sua teoria terem se dispersado.

4.5.7. MORTE

Consiste na desativação de um SW que não mais atende ao propósito para o qual foi criado ou que foi substituído por uma solução mais adequada e moderna. A etapa de Morte, assim como o Nascimento, é um processo que possui atividades realizadas antes de sua desativação e após a sua desativação.

4.5.8. LEGADO

Mesmo depois de sua desativação (morte) algumas soluções continuam tendo importância, sobretudo como referência para as soluções que a substituíram. Usando uma analogia, é como se depois de mortos determinados componentes fossem transplantados para a nova solução. A (re)utilização de componentes de soluções desativadas pode se dar através do uso de partes de código ou, mais comumente, através da utilização de algoritmos implementados em novas tecnologias.

A fase de Legado também é útil como base de conhecimento para a análise de práticas e abordagens úteis, que funcionaram no SW morto e podem ser aplicadas em novos esforços de desenvolvimento. Outra importante utilidade deixada como Legado de um SW fora de uso são os contra-exemplos ou “lições aprendidas” durante seu Ciclo de Vida. Além de:

- Dados que podem ser “importados” para uma nova solução
- Trechos de código que podem ser reutilizados
- Algoritmos e soluções que podem ser implementadas em outras tecnologias
- Registros importantes sobre o uso do sistema, como características de carga, sazonalidades no uso, etc.
- Detalhes sobre usabilidade e características dos Públicos do SW

4.7. CICLOS DE DESENVOLVIMENTO

Em ECO, um Ciclo de Desenvolvimento é o esforço coordenado de uma Equipe de Projeto destinado a alterar de alguma forma um determinado SW, como, por exemplo, através da criação de novos Conteúdos e Serviços. Durante o Ciclo de Vida de um SW, ocorrem vários Ciclos de Desenvolvimento, cada qual com um objetivo específico.

Durante um Ciclo de Desenvolvimento são executadas as tarefas de análise, codificação, testes e implantação. Entretanto, a ordem em que estas atividades ocorrem em ECO não segue, necessariamente, a sequência preconizada pelo modelo cascata. Alguns Métodos Ágeis, incluindo XP, propõe a inversão da sequência tradicional, que começa com a análise e termina com os testes. Em XP - e nos chamados Métodos Direcionados por Teste - começa-se o desenvolvimento com testes unitários automatizados e termina-se na análise, através de uma atividade conhecida como *refactoring*, que consiste em criar um modelo mais genérico a partir de uma solução já implementada. O objetivo desta atividade é tornar a modelo da solução mais flexível e elegante, de modo a simplificar a integração das funcionalidades em desenvolvimento.

ECO não define uma ordem clara para execução destas tarefas, cabendo à Equipe de Projeto definir a forma mais adequada em função do desafio a ser enfrentado. O Capítulo 5 descreve as atividades e processos de ECO, incluindo a atividade de Entrega, onde os principais processos que compõem um Ciclo de Desenvolvimento são executados.

4.8. PAPÉIS

Papéis são funções que pessoas desempenham em projetos, estando relacionados com atividades, responsabilidades e artefatos. Em ECO existem três grupos principais de papéis, quais sejam: Produto, Coordenação e Produção.

4.8.1. PRODUTO

Os papéis de produto são aqueles relacionados à definição e financiamento do produto de software a ser criado. Via de regra são executados pessoas advindas do contratante. De maneira coletiva, estes papéis são referenciados como Equipe Cliente.

GERENTE DE PRODUTO

Responsável por todas as decisões relacionadas ao direcionamento do produto. É o Gerente de Produto quem prioriza as funcionalidades e planeja quais recursos estarão disponíveis em cada versão do SW, sendo o responsável por definir se uma determinada versão está ou não apta a ir para produção. O Gerente de Produto é também um dos principais responsáveis pela validação do Produto e definição dos Públicos aos quais o mesmo deverá atender. Normalmente este papel é desempenhado por um membro da equipe do cliente ou, em caso de desenvolvimentos internos, por membros das equipes de marketing ou negócios.

PATROCINADOR

O Patrocinador é o financiador do projeto. Cabe ao Patrocinador garantir a disponibilidade dos recursos necessários para a criação do SW. Cabe ao Patrocinador a decisão final sobre qualquer aspecto do projeto.

ESPECIALISTA NO DOMÍNIO

Pessoa com profundos conhecimentos do domínio do problema a ser resolvido pelo SW, responsável pela elucidação de questões e com participação efetiva na definição do produto. Alguns métodos ágeis, como XP, exigem que um ou mais Especialistas (Cliente,

na terminologia de XP) sejam parte integrante da equipe, trabalhando em tempo integral junto à equipe de Produção.

4.8.2. COORDENAÇÃO

Cabe aos detentores de papéis de coordenação o planejamento, negociação, alocação de recursos e a verificação das atividades do projeto.

GERENTE DE PROJETO

O Gerente de Projeto é o responsável pelo bom andamento do projeto, definição de prioridades junto ao cliente, avaliação da corretude e validade das funcionalidades implementadas. Atua como facilitador nas reuniões de projeto, como as reuniões de “15 minutos”, reuniões de priorização e apresentação de Funcionalidades. É o principal responsável pela qualidade funcional do projeto, sendo de sua responsabilidade a liderança na integração das tarefas que criarão uma nova funcionalidade.

Em ECO o Gerente de Projetos é também responsável pela manutenção das versões do SW.

GERENTE DE RECURSOS

Uma característica cada vez mais comum nas organizações atuais é a alocação de uma mesma pessoa em diversos projetos simultâneos. Esta característica é a norma em pequenas empresas de desenvolvimento de SW, principal foco de ECO. Neste tipo de ambiente, é fundamental a existência de um mediador para resolver os constantes conflitos de alocação tendo em mente as prioridades de cada projeto, planejamento de alocação e disponibilidade de recursos com as competências e perfis necessários.

Este é a função do Gerente de Recursos. Cabe a ele atuar junto aos Gerentes de Projeto na definição da alocação de recursos de forma a garantir o atendimento às demandas conflitantes existentes num dado momento. Este papel é chave para a resolução

de conflitos e a garantia do bom andamento dos projetos em momentos onde há grande necessidade por algum perfil.altamente compartilhamento. Cabe também ao Gerente de Recursos o planejamento de contratações (definitivas ou temporárias) e o auxílio no direcionamento dos planos de capacitação da equipe.

GERENTE DE NEGÓCIOS

O Gerente de Negócios é o responsável pelas negociações de orçamento e contratos junto ao cliente. É ele o responsável por conduzir os processos voltados para a elaboração de Propostas e acompanhar o andamento das negociações junto ao cliente. Em ECO o Gerente de Negócios executa também um papel importante após o fechamento do negócio, funcionando como um *ombudsman* do cliente junto à Equipe de Projeto. Após o encerramento de um contrato, cabe ao Gerente de Negócios gerenciar os processos de finalização do mesmo e, eventualmente, aproveitar a oportunidade para explorar novas possibilidades.

Uma especialização possível do Gerente de Negócios ocorre em projetos onde o contrato é fechado por alocação de horas por mês (sem escopo definido) e, portanto, exige constante esforço de negociação junto ao cliente. Neste caso, a denominação usada em ECO é Gerente de Relacionamento, que reflete melhor o objetivo último do papel, que é manter saudável e duradouro o relacionamento entre a Equipe de Projeto e o cliente.

4.8.3. PRODUÇÃO

Os papéis de produção são aqueles relacionados à criação do produto definido pela equipe de Produto. Inclui as funções relacionadas à criação do produto final bem como de artefatos e produtos intermediários, e todas as atividades correlatas, como modelagem, testes, documentação, treinamento, etc. Uma importante função dos papéis de produção é a realização de estimativas de esforço. Em ECO todas as estimativas são feitas pela equipe de produção, idealmente por membros da equipe de desenvolvimento.

Existem dois tipos principais de papéis de Produção: Desenvolvedores, responsáveis pelas atividades de implementação do Produto, e Líderes, responsáveis pelas atividades de definição da solução mais adequada para a implementação do Produto.

Os papéis de produção estão intimamente relacionados com o Ambiente do SW. O Ambiente Lógico define os tipos de papéis de Desenvolvimento e Líderes a serem utilizados de acordo com as camadas adotadas no SW. O Ambiente Operacional define as especialidades que os Desenvolvedores devem possuir para a implementação da solução em cada camada. A influência do Ambiente Operacional nos papéis de Liderança é menor, na medida em que o modelo da solução é pouco impactado por este ambiente.

LÍDER DE EXPERIÊNCIA HUMANA

A partir do entendimento das necessidades e características de cada Público do SW, o Líder de Experiência Humana deve modelar como será a organização dos conteúdos do SW, sua hierarquia, vínculos, forma de apresentação e estrutura de navegação. Assim, cabe ao Líder de Experiência Humana garantir a qualidade geral da Experiência Humana do SW, propondo soluções para maximizar a simplicidade e a usabilidade do SW.

Nos SW Direcionados por Serviços, o Líder de Experiência Humana é também o responsável pela integração dos Serviços aos Conteúdos.

LÍDER DE SOFTWARE

Cabe ao Líder de Software criar a solução tecnológica para o problema proposto. É o Líder de Software quem faz a maior parte das definições de Ambiente e é o responsável pelo modelo geral do SW. Assim, analogamente ao Líder de Experiência Humana, cabe ao Líder de Software garantir a qualidade geral da solução técnica do SW.

Nos SW Direcionados por Conteúdo, o Líder de Software é também o responsável pela integração dos Conteúdos aos Serviços.

DESENVOLVEDOR DE EXPERIÊNCIA HUMANA

A camada de Experiência Humana de um SW possui diversas tecnologias e características próprias, além de ser uma área em constante evolução. O Desenvolvedor de Experiência Humana deve conhecer a fundo as tecnologias disponíveis, como linguagens de marcação, linguagens de script e padrões de projeto, sendo o responsável pela implementação da solução criada pelo Líder de Experiência Humana.

DESENVOLVEDOR DE SOFTWARE

Responsável pela implementação do SW utilizando as tecnologias definidas no Ambiente do mesmo segundo a solução criada pelo Líder de Software. O Desenvolvedor de Software deve ser um especialista em tecnologias de desenvolvimento de SW, conhecendo a fundo os conceitos envolvidos, padrões de projeto, linguagens de programação, etc.

DIRETOR DE ARTE

Uma importante característica dos SW é a utilização de conteúdos multimídia para enriquecer a Experiência Humana. O Diretor de Arte é o responsável pela definição das diretrizes visuais do SW, criando e desenhando os itens de comunicação visual utilizados, como imagens, figuras, animações, etc.

DIRETOR DE CRIAÇÃO

Em muitos SW, sobretudo os Direcionados por Conteúdo (detalhes no próximo capítulo), é fundamental uma comunicação bem planejada e fundamentada. Neste tipo de SW o Diretor de Arte e o Líder de Experiência Humana são auxiliados por um Diretor de Criação, cujo objetivo é estabelecer a linha criativa e de comunicação do SW, direcionando os esforços de todos os envolvidos na camada de Experiência Humana no sentido de criar o melhor conceito de comunicação com os públicos alvo do SW. Este é um papel opcional.

REDATOR

Responsável pela definição da linha de comunicação do SW e pela criação de todos os conteúdos textuais utilizados no mesmo. Secundariamente, o Redator auxilia na elaboração de outros textos do projeto, como manuais e documentos em geral. Nos SW Direcionados por Conteúdo, tem papel importante na captura do estilo de comunicação adequado e a sua transformação em textos. Também tem papel de destaque em iniciativas de Marketing feitas através da internet.

4.9. ARTEFATOS

Uma característica comum a diversas metodologias é a premissa de que os usuários sejam capazes de oferecer uma descrição clara, completa e explícita das suas necessidades logo no início do desenvolvimento. Assim, a ênfase destas metodologias está em técnicas voltadas à elucidação desta descrição, através de especificações de requisitos e modelos do sistema solicitado [EHN, 1992].

Entretanto, mais importante que a corretude destas descrições, ou quão fidedignamente elas mapeiam a “realidade”, o principal objetivo destas descrições é remeter-nos de volta às discussões que as geraram, às teorias criadas durante sua construção. Nas palavras de [EHN, 1992], artefatos “não são modelos no sentido Cartesiano de refletirem imagens da realidade. No “jogo-da-linguagem” da modelagem, usamos estas ferramentas como lembretes de nossas reflexões sobre o futuro da aplicação computacional e seu uso. Utilizando-se estes artefatos, trazemos experiências anteriores de volta à nossa mente, moldando (ou direcionando) nossa forma de pensar sobre o passado e o futuro. Eu penso que este é o motivo para os encarmos como representações. E esta é a maneira como eles informam nossa prática. Se estes artefatos são bons, eles auxiliarão boas decisões de modelagem”. Assim, “o sentido de um artefato é o seu uso em um “jogo-de-linguagem” de modelagem, e não como ele reflete a realidade. Sua capacidade de auxílio neste sentido depende do tipo de experiência que ele remete”.

A partir desta visão dos artefatos de um sistema, como referências para o processo de criação de teorias e paradigmas feitos em conjunto entre os usuários e a Equipe de Projeto, fica claro o problema da divisão rígida das equipes de análise/modelagem e programação. O time de modelagem cria, em conjunto com Especialistas no Domínio e membros dos Públicos do sistema, uma teoria e tenta mapeá-la através de diagramas, modelos e textos. Entretanto, como visto, antes de imagens da realidade, artefatos são lembretes ou representações da teoria criada, tendo pouco ou nenhum significado para quem não participou do processo de sua criação. A partir de modelos criados por outra equipe, o máximo que um time de programação consegue é derivar sua própria teoria do problema e da solução, criando, como resultado, o objeto mais importante do esforço de desenvolvimento de um sistema: seu código fonte sem que este, no entanto, reflita a teoria criada pelo time de análise e especificação.

Neste sentido, ECO – assim como os demais Métodos Ágeis aqui estudados - dá ênfase total à geração do código executável do SW, relegando os Artefatos a um papel secundário. ECO propõe um conjunto mínimo de artefatos, sempre construídos, mantidos e evoluídos em conjunto por todos os envolvidos no projeto. Apesar de os artefatos terem como “donos” os Líderes de cada camada do Ambiente lógico, esta posse não deve ser encarada como atitude individualista ou restritiva, mas sim como o de responsabilidade pela manutenção e evolução dos mesmos. O objetivo é oferecer à Equipe de Projeto uma base sólida de comunicação e trabalho.

Cada conjunto de papéis de ECO, descritos anteriormente, possui seu conjunto de artefatos. Neste sentido, os artefatos de ECO voltados para os papéis operacionais estão intimamente relacionados ao Ambiente Lógico do SW. Existe um conjunto de artefatos para cada camada do Ambiente Lógico, sendo que cada artefato é de responsabilidade de um papel específico.

4.9.1. ARTEFATOS DA CAMADA DE EXPERIÊNCIA HUMANA

Em ECO, a camada de Experiência Humana é criada e validada através de um Protótipo Navegável que evolui para a solução final. Trata-se, portanto, de um protótipo evolutivo [PRESSMAN, 2001].

A Camada de Experiência Humana lida com conceitos abstratos e subjetivos, envolvendo gosto pessoal, cultura, modismos, entre outras características. Assim, a aprovação do trabalho desenvolvido para a Camada de Experiência Humana depende, além da qualidade do trabalho e da capacidade de argumentação da Equipe de Criação, de decisões baseadas em conceitos relativos por parte da Equipe Cliente. Este fato gera, normalmente, grande quantidade de “retrabalho”, com impactos no moral da equipe, no resultado financeiro do projeto e nos prazos estimados.

Para minimizar esta situação, ECO propõe a criação conjunta pela Equipe de Criação e pela Equipe Cliente de alguns artefatos que resultam, naturalmente, no Protótipo Navegável, garantindo um processo de aprovação mais rápido e tranquilo para a camada de Experiência Humana do SW. Em linhas gerais, os artefatos são criados na seguinte ordem:

1. Mapa de Navegação
2. Modelos de Grade
3. Diretrizes visuais
4. *Layouts* de base
5. Protótipo Navegável

A seguir descrevem-se estes artefatos.

MAPA DE NAVEGAÇÃO

O Mapa de Navegação representa a estrutura “estática” de um SW, apresentado suas áreas e seções e forma hierárquica, em formato de árvore. Os Mapas de Navegação também são conhecidos como “Mapa do Site” ou “Mapa Estrutural”. O uso de cores em cada nó do Mapa de Navegação adiciona uma nova dimensão no diagrama bastante útil para refletir diferentes tipos de “páginas”, como por exemplo páginas em HTML puro, páginas geradas dinamicamente via *scripts* executados no servidor web ou páginas com animações em *flash*, por exemplo.

MODELOS DE GRADE

Os Modelos de Grade, também conhecidos com *Wireframes*, descrevem a estrutura de Conteúdos de uma página web sem levar em consideração os detalhes visuais dos mesmos. Seu objetivo é apresentar através de um modelo simples, a organização geral dos Conteúdos da página, suas estruturas de navegação e principais áreas. Os Modelos de Grade representam, portanto, o modelo estrutural e navegacional do SW, mostrando onde cada Conteúdo se encaixa e os principais padrões de usabilidade utilizados [GRAHAM, 2003].

DIRETRIZES VISUAIS

O documento de Diretrizes Visuais, também chamado de *Guidelines*, descreve as regras de aplicação de marcas e logotipos, estilos para os textos, pantone de cores e todos os detalhes relacionados às características visuais e padrões de estilo utilizados num SW. Este documento pode ser criado pela Equipe de Criação durante o desenvolvimento mas, em geral, sobretudo para projetos desenvolvidos para grandes empresas, são fornecidos pela Equipe Cliente para que o resultado esteja de acordo com padrões utilizados em outros veículos além da internet.

LAYOUTS DE BASE

De forma bastante simplista, os *Layouts* de Base são o resultado da aplicação das Diretrizes Visuais nos Modelos de Grade. São desenhos feitos em alguma ferramenta de desenho digital como, por exemplo, o *Macromedia PhotShop*, apresentando uma determinada página com seu visual final, incluindo aplicação de cores, logotipos, ícones, textos, menus, etc. Em ECO, esses *Layouts* serão posteriormente “recortados” e “montados”, gerando as páginas HTML que comporão do Protótipo Navegável e, posteriormente, evoluirão para o produto final.

A partir do Mapa de Navegação, Modelos de Grade e Diretrizes Visuais pré-aprovados, a aprovação de um *Layout* por parte da Equipe Cliente torna-se mais

determinística do que seria sem a utilização destes recursos. Entretanto, para projetos mais simples, é válido e recomendável lançar mão apenas deste artefato sem a utilização dos demais.

PROTÓTIPO NAVEGÁVEL

A partir dos artefatos descritos anteriormente, cria-se o Protótipo Navegável, que é um *website* em HTML completo, sem a utilização de Serviços. Obviamente, para SW que não exigem Serviços, o Protótipo Navegável é o produto final do projeto.

Assim como os Serviços em ECO, o Protótipo Navegável é construído de forma evolutiva, com a inclusão do formato final de seus Conteúdos de acordo com as prioridades estabelecidas pela Equipe Cliente. Em SW Direcionados por Conteúdo, detalhados posteriormente neste trabalho, o Protótipo Navegável direciona o desenvolvimento de Serviços, que são desenvolvidos e integrados ao Protótipo após a criação dos Conteúdos que os apresentam. Analogamente, para os SE Direcionados por Serviço, o Protótipo Navegável é feito posteriormente ao desenvolvimento dos Serviços, provavelmente utilizando com base para sua criação alguma tela rudimentar que apenas apresenta os resultados do Serviço, sem nenhuma preocupação estética e visual.

4.9.2. ARTEFATOS DA CAMADA DE NEGÓCIO

A Camada de Negócios em ECO é modelada, quando necessário, através de três modelos representados utilizando-se UML [BOOCH *et al*, 1999]. Esses artefatos são descritos a seguir.

DIAGRAMA DE COMPONENTES

Representa a decomposição funcional de mais alto nível do SW. Além de oferecer importantes informações a cerca de reuso e componentização, este diagrama é a base para a realização das primeiras estimativas de esforço em ECO, utilizadas para a elaboração da Proposta Comercial, e para a organização do *Backlog* de Serviços.

MODELO DE DOMÍNIO

A criação de Modelos de Domínio é uma prática reconhecidamente útil para o desenvolvimento de soluções orientadas a objeto. O Modelo de Domínio captura o modelo criado pela Equipe de Projeto para a solução do problema proposta, independentemente dos detalhes do Ambiente do SW. Existem diversas técnicas importantes de apoio à criação de Modelos de Domínio, com destaque para a vasta literatura de padrões de projetos (*Design Patterns*) [GAMMA *et al*, 1995; FOWLER, 2003] e, em particular, para a técnica utilizada amplamente em FDD denominada DNC (*Domain Neutral Component*)[COAD *et al*, 1999].

FACHADA DE SERVIÇOS

A Fachada de Serviços representa um conjunto de *interfaces* de mais alto nível para acesso ao Modelo de Domínio. A importância da Fachada de Serviços em ECO está na representação dos Serviços oferecidos pelo SW, sobretudo nos casos onde não há Camada de Experiência Humana.

4.9.3. ARTEFATOS DE APOIO

Além dos artefatos focados na criação do SW em si, ECO propõe outros artefatos que tem importância enquanto suporte ao produto e seus processos geradores. Esses artefatos são descritos brevemente a seguir.

DOCUMENTO DE VISÃO

Também conhecido como *Project Charter*, o Documento de Visão é o primeiro artefato gerado num processo completo usando ECO. Em linhas gerais, o Documento de Visão descreve, poucas páginas, os objetivos do projeto, contexto de negócio, fatores críticos para o sucesso, breve descrição textual do mesmo e demais tópicos importantes para descrever a visão do projeto.

MLO – MATRIZ LÓGICO-OPERACIONAL

A Matriz Lógico-Operacional descreve de forma sucinta as decisões tomadas sobre o Ambiente do SW (ver Figura 5).

LISTA DE PERFIS

Trata-se de uma listagem simples dos Papéis envolvidos no desenvolvimento do SW, trazendo o nome do papel e a quantidade de pessoas que executarão tal papel ou o esforço em horas por período necessário para o mesmo.

PROPOSTA DE TRABALHO

A Proposta de Trabalho é o documento entregue para a Equipe Cliente descrevendo o entendimento da necessidade exposta no Documento de Visão, a solução proposta, um Plano de Desenvolvimento para sua criação e uma proposta comercial, descrevendo o investimento necessário e o prazo para a execução do trabalho.

NOTAS DE ENTREGA

Breve texto descrevendo, para cada versão lançada de um SW, quais as mudanças ocorridas em relação à versão anterior em termos de Serviços e Conteúdos. Assim, o conjunto das Notas de Entrega de um SW apresenta sua história de evolução e alterações.

BACKLOG

O *Backlog* representa o trabalho a ser realizado em um projeto (ver descrição no tópico 3.2.1.). Trata-se de uma lista de tarefas estimadas e priorizadas. Em ECO existem vários *Baklogs*, como o *backlogs* do Produto (ou Lista de Funcionalidades), com todas as Funcionalidades desejadas, o *Backlog* de Serviços e o *Backlog* de Conteúdos, com todos os

Serviços e Conteúdos a serem criados num Ciclo de Desenvolvimento do SW. Cada membro da Equipe de Projeto tem também ser *Backlog* individual, com todas as suas tarefas pendentes de execução.

Os *Backlogs* são a principal ferramenta de gerenciamento de ECO. Utiliza-se os *Backlogs* para fazer o controle do progresso do projeto, reportar o andamento para os diversos interessados, organizar a equipe, avaliar o desempenho de seus membros, entre outras funções.

CAPÍTULO 5: ECO PARTE II - PROCESSOS

5.1. CONSIDERAÇÕES INICIAIS

O capítulo anterior apresentou as principais características de ECO e os conceitos fundamentais utilizados em sua elaboração. Estes conceitos e características são o fundamento para o entendimento dos processos que definem ECO enquanto um Ecossistema de Criação Ágil de Sistema Web.

Neste capítulo são expostos os 15 processos que compõem ECO utilizando-se o modelo ETVS (apresentado a seguir), e as 3 atividades fundamentais que agrupam tais processos.

5.2. VISÃO GERAL

O objetivo de ECO é oferecer às empresas de pequeno focadas no desenvolvimento de Sistemas Web uma estrutura metodológica na qual se basear para todas as atividades envolvidas deste a aquisição de projetos até seu encerramento. Assim, diferentemente das metodologias vistas no capítulo 3, ECO engloba, além das atividades de desenvolvimento, atividades para aquisição e encerramento de projetos. A Figura 6 ilustra as atividades cobertas por ECO.

Figura 6: Atividades cobertas por ECO

Cada uma destas atividades é composta por um conjunto de processos, cada qual com um objetivo específico e seu respectivo conjunto de tarefas. ECO possui 15 processos distribuídos entre as três Atividades acima.

A não ser pela ordem cronológica das Atividades, que ocorrem de forma eminentemente seqüencial, ECO não propõe uma ordem clara para a execução de seus processos, cabendo à Equipe de Projeto definir a melhor ordem de execução destes processos. Eventualmente podem ocorrer sobreposições entre processos de atividades distintas, mas esta não é a regra.

Conforme representado na Figura 6, os processos das atividades de Aquisição e Encerramento ocorrem, via de regra, uma única vez por contrato, enquanto que os processos da atividade de Entrega ocorrem várias vezes por projeto. Na verdade, os Ciclos de Desenvolvimento, descritos no Tópico 4.7, ocorrem quase que totalmente na atividade de Entrega, com os processos desta atividade detalhando como tais Ciclos de Desenvolvimento ocorrem em ECO.

5.2.1. AQUISIÇÃO

A atividade de Aquisição tem como principal objetivo a formalização de um contrato. Em ECO existem dois modelos de contrato: por Escopo, onde define-se o objetivo do projeto, suas funcionalidades e estima-se o esforço necessário para criar o escopo definido, ou por Esforço/alocação, onde não há definição claro de um escopo, apenas um determinado volume de horas alocadas por período.

O tipo de contrato e a forma como a proposta é elaborada dependem totalmente da etapa do ciclo de vida do projeto. Para projetos nas etapas de Concepção ou Geração, os contratos por Escopo são mais comuns. Para as etapas seguintes, os contratos por Esforço são mais comuns. A etapa do Ciclo de Vida também tem grande influência nos processos da atividade de Aquisição. Para projetos nas etapas de Concepção e Geração, os processos de definição de objetivos e especificações iniciais são baseados na exploração do escopo e objetivos junto ao cliente, visando uma definição inicial do produto e seus objetivos. Para projetos que já estão nas etapas subsequentes do ciclo de vida, os processos de Aquisição envolvem o estudo da solução existente, seus pontos positivos e negativos.

Esta atividade é encerrada com a formalização de um contrato de trabalho. A Figura 7 apresenta os processos da atividade de Aquisição de ECO.

Figura 7: Processos da atividade de Aquisição.

5.2.2. ENTREGA

Uma vez fechado o contrato, inicia-se a atividade de Entrega, cujo objetivo primordial é a entrega do Sistema Web definido e contratado na atividade anterior. É nesta atividade onde concentra-se o foco dos métodos ágeis e os Ciclos de Desenvolvimento.

Esta é a atividade comumente coberta pelas metodologias em geral, e pela maioria dos métodos ágeis em particular, havendo, portanto, vasta literatura sobre esta atividade. Entretanto, a base de conhecimento existente sobre as nuances e especificidades da entrega de Sistemas Web é bastante restrita, tanto no que tange às atividades de criação do produto, quanto no que tange ao gerenciamento de projetos. A contribuição de ECO para esta atividade está, portanto, nos processos de gerenciamento e nas tarefas específicas de criação de Sistemas Web.

Uma importante característica da atividade de Entrega de ECO é o “direcionamento do desenvolvimento”. Como visto anteriormente, as funcionalidades de um Sistema Web são compostas por Conteúdos e Serviços. Sistemas Web onde os Conteúdos tem maior complexidade e/ou importância determinante em seu sucesso são chamados, em ECO, “Sistemas Web Direcionados por Conteúdos”. Analogamente, nos casos onde os Serviços são mais complexos e/ou tem importância central para o alcance dos objetivos traçados para o sistema, diz-se que estes são “Sistemas Web direcionados por Serviços”.

A definição do direcionamento de um SW é uma importante tarefa do projeto, na medida em que influencia decisivamente no estabelecimento de prioridades e na definição do perfil ideal para a realização de determinadas tarefas, como a liderança conceitual do projeto e a integração de Conteúdos e Serviços. Por exemplo, num SW direcionado por Conteúdos, as tarefas de codificação são feitas com base nos Conteúdos definidos através de um Protótipo Navegável já validado pelo cliente. Analogamente, num SW direcionado por Serviços, inicialmente codifica-se os serviços necessários e depois, quando necessário, projeta-se a camada de Experiência Humana.

No caso de um SW direcionado por Conteúdos, normalmente a integração entre as camadas de Experiência Humana (EH) e de Negócio é feita por um Líder de Software, que

parte dos Conteúdos gerados por um Desenvolvedor de EH. No caso de um SW direcionado por Serviços acontece o contrário: a “cola” normalmente é feita por um Líder de EH, que parte do trabalho realizado por um Desenvolvedor de Software, normalmente páginas simples sem a aplicação das Diretrizes Visuais e da linha de comunicação definida para determinado Público.

A definição do direcionamento de um SW também tem influência sobre as características de qualidade mais cuidadosamente avaliadas. Num SW direcionado por Conteúdos, dá-se importância vital à qualidade da experiência humana, ou seja, às características visuais, navegacionais e textuais do SW.

O objetivo principal da atividade de Entrega é a implantação de novas versões do SW, promovendo sua evolução contínua dentro de seu Ciclo de Vida. A Figura 8 apresenta os processos da atividade de Entrega de ECO.

Figura 8: Processos da atividade de Entrega.

5.2.3. ENCERRAMENTO

Após o término do período estipulado no contrato estabelecido como resultado da atividade de Aquisição, executa-se a atividade de Encerramento. Esta atividade tem dois objetivos primordiais, quais sejam: a consolidação do conhecimento adquirido com o projeto e a finalização formal do mesmo.

A consolidação do conhecimento se dá através da realização de *workshops* de encerramento entre a equipe, clientes, parceiros e demais envolvidos no projeto. O intuito é avaliar os pontos positivos e os pontos a desenvolver do projeto, gerando importante *feedback* para futuras empreitadas. Outra tarefa importante é a consolidação de documentos e componentes criados durante o projeto e que tem potencial de reutilização. Este tipo de tarefa deve ser realizado durante todo o projeto, sobretudo ao término de cada ciclo de desenvolvimento e entrega de versão. Entretanto, o término do contrato é um importante marco e ponto para a realização deste tipo de tarefa.

Do ponto de vista de negócio, a tarefa mais importante da atividade de Encerramento é a formalização do término do contrato. Esta tarefa, comumente relegada e não executada, pode gerar grandes prejuízos financeiros visto que a não formalização da execução de determinadas cláusulas contratuais pode gerar muito trabalho não remunerado e, em casos extremos, processos judiciais. Via de regra, estas cláusulas incluem a entrega formal de código-fonte e artefatos, formalização da data de entrega do produto e encerramento do contrato para fins de contagem do tempo de garantia, confidencialidade, não-aliciamento, etc. Esta tarefa é realizada através de uma minuciosa análise das cláusulas contratuais e devidas documentações e assinaturas das partes envolvidas.

A Figura 9 apresenta os processos da atividade de Entrega de ECO.

Figura 9: Processos da atividade de Encerramento.

5.3. PROCESSOS

Da mesma forma que os processos da metodologia FDD, os processos de ECO são descritos utilizando o modelo ETVS (Entrada – Tarefa – Verificação - Saída). Este modelo oferece um padrão simples e objetivo para a descrição dos processos da metodologia e tem

como restrição a utilização de apenas uma página frente e verso para cada processo. Assim, ao invés de descrever a metodologia em um documento formal, com centenas de páginas de leitura difícil e cansativa, a metodologia completa é descrita em apenas 15 páginas que podem ser impressas e plastificadas para fácil manuseio acesso e leitura.

ECO – PROCESSO #		
<<NOME DO PROCESSO>>		
<<Breve descrição do processo.>>		
CRITÉRIO DE ENTRADA		
<ul style="list-style-type: none"> Entrada 1. Entrada 2. 		
TAREFAS		
<<Nome tarefa 1>>	<<Papéis>>	<<Obrigatória ou Opcional>>
<<Descrição tarefa 1>>		
<<Nome tarefa 2>>	<<Papéis>>	<<Obrigatória ou Opcional>>
<<Descrição tarefa 2>>		
VERIFICAÇÃO		
<<Nome verificação 1>>	<<Papéis>>	<<Obrigatória ou Opcional>>
<<Descrição verificação 1>>		
<<Nome verificação 2>>	<<Papéis>>	<<Obrigatória ou Opcional>>
<<Descrição verificação 2>>		
CRITÉRIO DE SAÍDA		
<ul style="list-style-type: none"> Saída 1. Saída 2. 		

Figura 10: Modelo ETVS, utilizado para a descrição dos processos de ECO.

A Figura 10 apresenta o modelo ETVS.

O topo da página traz o número do processo, seu nome e uma breve descrição de seu objetivo. A cor de fundo utilizada faz referência à atividade da qual o processo faz parte. A seção seguinte traz uma relação dos Critérios de Entrada, condições que devem ser satisfeitas para que o processo possa ser iniciado. Em seguida são listadas as Tarefas que compõe o processo em questão. O cabeçalho das tarefas traz seu nome, os papéis

envolvidos em sua execução e se a tarefa é obrigatória ou opcional. Logo abaixo do cabeçalho vem uma breve descrição da tarefa. Após as tarefas são elencadas as Verificações a serem feitas para garantir a correta execução do processo. As Verificações são descritas de forma análoga às tarefas. Por fim são listados os Critérios de Saída, que são as principais saídas esperadas.

As páginas seguintes trazem os 15 processos que compõem ECO.

ECO – PROCESSO 1

ESTABELEÇA A VISÃO DO PROJETO

Processo inicial de definição dos objetivos do projeto envolvendo Equipe Cliente e Equipe de Aquisição que devem trabalhar em conjunto para estabelecer as bases do projeto, gerando como resultado um Documento de Visão do projeto.

CRITÉRIO DE ENTRADA

- Definido o Gerente de Negócios responsável pela atividade de Aquisição.

TAREFAS

Defina a Equipe de Aquisição	Gerente de Recursos, Gerente de Negócios	Obrigatória
O Gerente de Negócios e o Gerente de Recursos definem os membros da Equipe de Aquisição. Para projetos simples, todos os papéis podem ser realizados pelo Gerente de Negócios com auxílio pontual de outros membros da equipe ou pelo Gerente de Projetos alocado.		

Defina a Equipe Cliente	Gerente de Negócios	Obrigatória
O Gerente de Negócios deve elencar os contatos da Equipe Cliente, composta pelo Patrocinador do projeto, Gerente de Produto e Especialistas no Domínio.		

Conduza uma reunião de apresentação do projeto	Equipe Cliente, Equipe de Aquisição	Obrigatória
A Equipe Cliente deve conduzir uma reunião estilo <i>workshop</i> para apresentação das pessoas envolvidas, visão geral do projeto, objetivos e fatores críticos para o sucesso, contexto de negócio do projeto, expectativas e valor estratégico, todo tipo de documentação disponível sobre o projeto (normalmente pelo menos uma RFP – <i>Request For Proposa</i>) e seu domínio, expectativas de prazos, datas importantes, orçamento disponível, etc. Esta reunião pode ser bastante curta para projetos simples, ou incluir várias reuniões distintas, eventualmente no estilo JAD.		

Para produtos em etapas do Ciclo de Vida posteriores à de Concepção, parte

importante de tarefa é a apresentação da versão atual do Produto, ressaltando seus pontos positivos e pontos a desenvolver, e da Equipe de Projeto responsável pela sua criação. Nestes casos, todas as atividades posteriores são realizadas tendo o Produto atual como base.

Revise os documentos e reunião de apresentação do projeto	Equipe de Aquisição	Opcional
--	----------------------------	-----------------

Caso necessário, a Equipe de Aquisição realiza reuniões para alinhar o conhecimento adquirido, revisar a documentação entregue pela Equipe Cliente, esclarecer quaisquer questões em aberto e, caso exista uma solução já criada, avaliá-la em profundidade à luz dos pontos levantados na tarefa anterior.

Elabore o documento de Visão	Equipe de Aquisição	Obrigatória
-------------------------------------	----------------------------	--------------------

A Equipe de Aquisição, liderada pelo Gerente de Projetos, consolida todas as informações sobre o projeto num Documento de Visão, com foco nos objetivos do projeto, fatores críticos para o sucesso, metas estratégicas, principais riscos identificados e uma breve descrição do projeto. O Documento de Visão deve ser curto e objetivo, possuindo, no máximo 5 páginas.

VERIFICAÇÃO

Avaliação Interna e Externa	Equipe de Negócios e Equipe Cliente	Opcional
------------------------------------	--	-----------------

Após elaborado o Documento de Visão, este deve ser validado internamente pela Equipe de Aquisição e externamente pela Equipe Cliente. Esta verificação é opcional já que o Documento de Visão fará parte da Proposta de Trabalho e, conseqüentemente, será verificado posteriormente.

CRITÉRIO DE SAÍDA

- Documento de Visão redigido e opcionalmente verificado pela Equipe de Aquisição e/ou Equipe Cliente.

ECO – PROCESSO 2

DESENVOLVA MODELOS GERAIS

Atividade de especificação “técnica” do projeto, consistindo na criação das primeiras abstrações da solução no que tange à Experiência Humana e Software. Via de regra este processo é iniciado após a tarefa “Reunião de apresentação do projeto”, ocorrendo paralelamente ao Processo 1.

CRITÉRIO DE ENTRADA

- Tarefa “Reunião de apresentação do projeto” realizada.

TAREFAS

Crie o Mapa de Navegação

Líder de EH

Obrigatória

O Líder de Experiência Humana cria uma primeira versão do Mapa de Navegação cujo objetivo é visualizar a estrutura básica de Conteúdos do projeto.

Crie o Modelo de Componentes

Líder de Software

Obrigatória

O Líder de Software cria uma primeira versão do Diagrama de Componentes do projeto tendo como objetivo visualizar os principais módulos do SW.

VERIFICAÇÃO

**Avaliação Interna e Externa do
Modelos**

**Equipe de Aquisição e
Equipe Cliente**

Opcional

Os Líderes de EH e Software verificam entre si o Mapa de Navegação e Modelo de Componentes, anotando decisões e alternativas para cada modelo. Eventualmente esta verificação é feita também junto à Equipe de Aquisição e a Equipe Cliente. Esta verificação é mais comum para o Mapa de Navegação;

CRITÉRIO DE SAÍDA

- Mapa de Navegação e Modelo de Componentes criados.
- Notas com decisões e alternativas incluídas nos modelos criados.

ECO – PROCESSO 3

CRIE UMA LISTA DE PÚBLICOS E FUNCIONALIDADES

Atividade de especificação “técnica” do projeto consistindo na identificação de todos os Públicos a serem atendidos pelo Sistema Web e das Funcionalidades que o mesmo deve entregar a estes Públicos.

A lista de Públicos é criada em conjunto pelo Líder de EH e o Cliente usando o Mapa de Navegação como insumo. A descrição dos Públicos deve ser detalhada o suficiente para viabilizar a definição das diretrizes de comunicação com cada Público. Caso necessário, o Mapa de Navegação é dividido em Mapas específicos para cada Público-alvo ou conjunto de Públicos.

Para a criação da lista de Funcionalidades, parte-se do(s) Mapa(s) criado(s) e Públicos definidos, listando todas as Funcionalidades esperadas. Posteriormente, o Líder de Software agrupa as Funcionalidades segundo os Componentes que as implementam.

CRITÉRIO DE ENTRADA

- Processos 1 e 2 suficientemente adiantados (versões iniciais do Documento de Visão, Mapa de Navegação e Modelo de Componentes já criados).

TAREFAS

Crie uma lista de Públicos	Equipe de Aquisição, Equipe Cliente	Obrigatória
-----------------------------------	--	--------------------

Equipe de Aquisição e Equipe Cliente criam, através de reunião facilitada pelo Gerente de Projetos, uma lista dos Públicos a serem atendidos pelo SW. Cada Público recebe um breve descrição de seu perfil, com atenção especial para questões culturais a serem consideradas (língua, opções de regionalização, como moeda, sistema de medidas, etc.).

Crie uma Lista de Funcionalidades	Equipe de Aquisição, Equipe Cliente	Obrigatória
--	--	--------------------

A partir da lista de Públicos criada na tarefa anterior (e, provavelmente, concomitantemente à sua criação), Equipe de Aquisição e Equipe Cliente criam, através de reunião facilitada pelo Gerente de Projetos, uma lista das

Funcionalidades a serem entregues pelo SW. Estas descrições refletem objetivos que os diferentes Públicos do SW tem atendidos direta ou indiretamente. Uma funcionalidade tem um título descrevendo o objetivo que ela atende e um breve detalhamento textual de seu funcionamento. Esta descrição não deve ter mais que 10 linhas de texto. Caso a descrição esteja muito longa, deve-se dividir a funcionalidade. Cabe ao Gerente de Projetos garantir que as Funcionalidades listadas possuam a granularidade e adequação necessárias à realização de tarefas posteriores, como estimativas e planejamento dos ciclos de desenvolvimento. As Funcionalidades são agrupadas segundo as áreas do Mapa de Navegação e/ou de acordo com os Componentes definidos.

VERIFICAÇÃO

Avaliação interna e externa	Equipe de Aquisição, Equipe Cliente	Opcional
------------------------------------	--	-----------------

As listas de Públicos e Funcionalidades são verificadas pela Equipe de Aquisição e pela Equipe Cliente com o intuito de checar se o escopo do projeto atende às expectativas do projeto.

CRITÉRIO DE SAÍDA

- Lista de Públicos, com respectivas descrições, criada.
- Lista de Funcionalidades, com respectivas descrições, criada e hierarquizada segundo Mapa de Navegação e/ou Componentes da solução.

ECO – PROCESSO 4

DEFINA O AMBIENTE DO SISTEMA WEB

Atividade inicial de grande abrangência em todo o projeto. Diferentemente da Lista de Funcionalidades do SW, cuja mudança é esperada e até induzida durante todo o projeto, a definição do Ambiente do SW deve ser o mais estável possível na medida em que alterações no Ambiente têm impacto significativo em todas as demais atividades e processos do projeto. A definição do SW deve ponderar necessidades de curto e longo prazo, sendo a tarefa do projeto onde deve-se adotar a postura mais conservadora e “pessimista” possível.

CRITÉRIO DE ENTRADA

- Documento de Visão criado e validado.
- Mapa de Navegação e Modelo de Componentes criados e validados.
- Lista de Públicos e Funcionalidades criadas e validadas.

TAREFAS

Revise os artefatos de decisões de projeto	Gerente de Projetos, Líder de EH e Software	Opcional
---	--	-----------------

Numa reunião facilitada pelo Gerente de Projetos, os Líderes de Software e Experiência Humana revisam os artefatos gerados, decisões de projeto e diretrizes tecnológicas do cliente com o intuito de abalizar a tarefa seguinte “Defina o Ambiente do SW”. Eventualmente pode-se convidar para essa reunião especialistas em determinadas tecnologias com o intuito de enriquecer a avaliação, analisar alternativas e novas opiniões. Para projetos simples ou de tipos com os quais a empresa tenha boa experiência prévia, esta tarefa pode não ser realizada.

Defina o Ambiente do Sistema Web	Líder de Software e Líder de EH	Obrigatória
---	--	--------------------

A partir do conhecimento sobre o projeto adquirido nos demais processos e tarefas, os Líderes de Experiência Humana e Software devem definir o Ambiente do SW. O Ambiente do SW tem impacto significativo em todo o projeto, definindo o perfil da Equipe de Projeto, suas capacidades e

conhecimentos, ferramentas e tecnologias a serem utilizadas no desenvolvimento, características da infra-estrutura de “hospedagem” do SW, questões de compatibilidade, escalabilidade, entre outras. Assim, a definição do Ambiente do SW deve ser feita com extrema cautela e cuidado, ponderando entre a maior simplicidade possível para reduzir os custos no curto prazo, sem prejudicar a evolução do SW no longo prazo. Apesar de possível, mudanças posteriores no Ambiente do SW têm impacto diretamente proporcional ao tempo de existência do produto. Assim, quanto mais adiantado no Ciclo de Vida estiver o produto, maior o custo de se fazer uma mudança no Ambiente do SW.

Crie uma Matriz MLO Líder de Software e Líder de EH Obrigatória

A partir do resultado a tarefa anterior, os Líderes de Software e Experiência Humana devem formalizar suas decisões em uma Matriz Lógico-Operacional, incluindo notas com observações sobre premissas, decisões, alternativas e riscos.

VERIFICAÇÃO

Avaliação interna e externa	Equipe de Aquisição, Consultores externos, Equipe Cliente e Especialistas	Obrigatória
------------------------------------	--	--------------------

Em virtude da importância deste processo, deve-se proceder uma verificação cuidadosa das decisões tomadas. Além da Equipe de Aquisição, deve-se submeter a Matriz MLO à avaliação da Equipe Cliente, Especialistas da equipe e, em casos extremos, a Consultores Externos.

CRITÉRIO DE SAÍDA

- Matriz MLO criada com notas e observações.
- Verificação da Matriz MLO, eventualmente formalizada.

ECO – PROCESSO 5

VENDA POR FUNCIONALIDADE

Este processo visa a criação de uma Proposta de Trabalho baseada no resultado dos processos anteriores, tendo como objetivo último a formalização de um contrato.

CRITÉRIO DE ENTRADA

- Processos 1, 2, 3 e 4 finalizados de forma bem sucedida.

TAREFAS

Defina equipe de Estimativa	Gerente de Projetos, Gerente de Recursos	Obrigatória
------------------------------------	---	--------------------

Gerente de Projetos e Gerente de Recursos definem os membros da Equipe de estimativas. Esta equipe deve ser composta pelo Gerente de Projetos e Líderes de Experiência Humana e Software que compuseram a Equipe de Aquisição, além de Desenvolvedores capacitados para esta atividade. Idealmente, os Desenvolvedores selecionados devem ser os mesmos que comporão a Equipe de Projetos.

Colete estimativas por Funcionalidade	Equipe de Estimativa	Obrigatória
--	-----------------------------	--------------------

Em uma ou mais reuniões facilitadas pelo Gerente de Projetos, a Equipe de Estimativa define uma estimativa de esforço em horas para cada Funcionalidade existente da Lista de Funcionalidades. Durante o processo, pode-se dividir ou agrupar Funcionalidades de acordo com sua complexidade, tamanho (Funcionalidades devem ser implementadas em no máximo duas semanas), interdependência e redundância. As estimativas são feitas utilizando-se o Método Delphi de estimativas e, eventualmente, recorrendo-se à base histórica de estimativas.

Defina o <i>time-box</i> do projeto	Gerente de Projetos	Obrigatória
--	----------------------------	--------------------

A partir do conhecimento adquirido nas tarefas anteriores, resultado das estimativas, restrições de prazo impostas pelo cliente e datas importantes definidas, o Gerente de Projeto define o tamanho do *time-box* do projeto, isto é,

a duração de cada iteração, que deve ser entre 2 semanas e 1 mês, com preferência para os períodos mais curtos.

Defina a Lista de Perfis do Projeto	Gerente de Projetos	Obrigatória
--	----------------------------	--------------------

De posse da estimativa de esforço total, *time-box* do projeto, Matriz MLO, restrições e necessidades do cliente, o Gerente de Projeto deve fazer uma estimativa dos perfis e respectivas quantidade necessárias na Equipe de Projeto, gerando como resultado uma Lista de Perfis.

Estabeleça um cronograma macro	Gerente de Projetos	Opcional
---------------------------------------	----------------------------	-----------------

Em projetos de Escopo fechado, o Gerente de Projetos deve gerar um cronograma macro contendo os principais marcos do projeto, como término esperado de ciclos, datas de entrega de versões, prazo total do projeto, etc..

Elabore a Proposta de Trabalho	Gerente de Negócios	Obrigatória
---------------------------------------	----------------------------	--------------------

O Gerente de Negócios deve reunir todos os artefatos gerados anteriormente, padronizá-los e reuni-los de forma didática e organizada numa Proposta Comercial. Deve conter também a proposta comercial do projeto, ressaltando o prazo estimado, investimento necessário (que pode ser apresentado como valor único ou com custo por Funcionalidade), validade da proposta, Lista de Perfis e observações gerais.

VERIFICAÇÃO

Verificação interna da Proposta de Trabalho	Equipe de Aquisição	Obrigatória
--	----------------------------	--------------------

A Proposta de Trabalho deve passar pelo crivo da Equipe de Aquisição, sobretudo para validação da proposta comercial.

CRITÉRIO DE SAÍDA

- Proposta de Trabalho enviada para o Cliente.

ECO – PROCESSO 6

DETALHE OS MODELOS GERAIS

Primeiro processo realizado após a formalização do contrato, tem como objetivo principal o envolvimento da Equipe de Projeto nos detalhes do projeto e o detalhamento dos Mapa de Navegação e do Modelo de Componentes criados durante a atividade de Aquisição. A Partir de Mapa de Navegação cria-se as Grades para cada Modelo de Página e os *Layouts* base para cada modelo. Com base no Diagrama de Componentes, cria-se a Fachada de Serviços e o Modelo de Domínio do Sistema Web.

CRITÉRIO DE ENTRADA

- Proposta de Trabalho aceita pelo Cliente.
- Contrato formalizado.

TAREFAS

Defina a Equipe de Projeto	Gerente de Projetos, Gerente de Recursos	Obrigatória
-----------------------------------	---	--------------------

A partir de Lista de Perfis criada no Processo 5, Gerente de Projetos e Gerente de Recursos definem as pessoas que executarão cada papel. Deve-se ponderar a experiência necessária para cada papel e disponibilidade das pessoas ao longo do projeto. A Equipe de Projeto é composta pela Equipe de Experiência Humana, pela Equipe de Software e pelo Gerente de Projetos.

Conduza uma reunião de apresentação do Projeto	Equipe de Projeto	Obrigatória
---	--------------------------	--------------------

Reunião conduzida pelo Gerente de Projetos, tem como objetivo iniciar “oficialmente” o projeto junto à Equipe de Projeto, apresentando aos novos membros da equipe o projeto, sua visão e objetivos através dos artefatos e documentos criados nos processos anteriores. Opcionalmente o Gerente de Projetos pode convidar representantes da Equipe Cliente para participar da reunião, ajudando na apresentação do projeto e dirimindo eventuais questões que possam surgir. Normalmente durante esta reunião a Lista de Funcionalidades é revisada e atualizada, com a inclusão de novas

Funcionalidades e, eventualmente, a remoção de outras.

Crie o Modelo de Grade	Equipe de EH	Opcional
------------------------	--------------	----------

Após da reunião da tarefa anterior, a Equipe de Experiência Humana pode optar por refinar o Mapa de Navegação, criando Modelos de Grade a partir dos perfis de cada Público do SW e do conhecimento adquirido nos processos anteriores. Os Modelos de Grade são criados para todas as páginas-base do SW ou para um subconjunto das mesmas. Via de regra, são criados Modelos de Grade para a *homepage* de cada Público (ou conjunto de Públicos) e para os principais tipos de páginas internas.

Crie um Modelo de Domínio	Equipe de Software	Opcional
---------------------------	--------------------	----------

Após do resultado da tarefa “Conduza uma reunião de Apresentação do Porjeto”, a Equipe de Software pode optar por refinar o Modelo de Componentes, criando um Modelo de Domínio e/ou uma especificação da Fachada de Serviços do SW. A Fachada de Serviços é particularmente importante em Sistemas Web Direcionados por Serviço, onde o foco do projeto é a publicação de Serviços na rede.

VERIFICAÇÃO

Avaliação dos Modelos Gerados	Equipe de Projeto e Equipe Cliente	Opcional
-------------------------------	------------------------------------	----------

Aprovação opcional dos artefatos criados pela Equipe Cliente. Esta aprovação é particularmente importante para o Modelo de Grade, visto que este artefato balizará a criação dos *layouts* de base do SW.

CRITÉRIO DE SAÍDA

- Modelos revisados e anotados.
- Equipe de Projetos definida e alocada.

ECO – PROCESSO 7

PLANEJE POR FUNCIONALIDADE

Todos os processos anteriores podem ser considerados como etapas de preparação para o início do projeto. A partir deste processo, e até o Processo 13 - “Entregue/Treine por Funcionalidade”, é feito o desenvolvimento Ágil do produto. As tarefas deste processo visam refinar o cronograma macro feito no Processo 5 – “Venda por Funcionalidade”, estabelecendo os objetivos e missão de cada ciclo de desenvolvimento, ritmo de entrega de versões e prioridades de desenvolvimento. Cumpre ressaltar que este processo, assim como os demais processos compreendidos entre o 7 e o 13 ocorrem de forma totalmente não linear e com diversas intersecções e influências cruzadas. Cabe à Equipe de Projeto auto organizar-se para atingir o objetivo definido para o ciclo da melhor forma.

CRITÉRIO DE ENTRADA

- Lista de Funcionalidades revisada e atualizada.

TAREFAS

Priorize a Lista de Funcionalidades	Gerente de Projetos, Gerente de Produto	Obrigatória
--	--	--------------------

Gerente de Projetos e Gerente de Produtos se reúnem para priorizar a Lista de Funcionalidades de acordo com as prioridades de negócio existentes. A Lista resultante trará a Funcionalidade mais prioritária em primeiro lugar e a menos prioritária em último. A Lista de Funcionalidades resultante será constantemente atualizada com novas Funcionalidades e mudanças de prioridade sendo, portanto, objeto de constante análise e atualização durante todo o projeto.

Colete estimativas por Funcionalidade	Equipe de Projeto	Obrigatória
--	--------------------------	--------------------

Sempre que a Lista de Funcionalidades sofrer alguma alteração, a Equipe de Projeto deve fazer estimativas de esforço em horas para a implementação das Funcionalidades seguindo as mesmas tarefas descritas no Processo 5 – Venda por Funcionalidade.

Defina o Plano de Desenvolvimento	Gerente de Projetos, Gerente de Produto	Obrigatória
--	--	--------------------

A partir da Lista de Funcionalidades prioriza, Gerente de Projetos e Gerente de Produtos planejam o objetivo de negócio de cada ciclo de desenvolvimento, alocam as Funcionalidades a serem desenvolvidas em cada ciclo, de acordo com os objetivos estabelecidos, e definem as entregas de versão. Normalmente uma versão é entregue a cada 3 ou 4 ciclos de desenvolvimento - nos três primeiros ciclos desenvolve-se Funcionalidades de acordo com os objetivos definidos para cada ciclo. No quarto ciclo faz-se ajustes finos, correções e uma avaliação de qualidade mais aprofundada. O resultado desta tarefa é um Plano de Desenvolvimento em ciclos, com a documentação de seus objetivos e respectivas Funcionalidades em ordem de prioridade. Esta tarefa, assim como a anterior, será objeto de constante análise e atualização durante todo o andamento do projeto. Os planos dos ciclos não iniciados podem mudar livremente. Para um ciclo em desenvolvimento, mudanças só são permitidas em casos extremos e mediante aprovação da Equipe Cliente à luz dos impactos apresentados pela Equipe de Projeto.

VERIFICAÇÃO

Validação interna do Plano de Desenvolvimento	Equipe de Projeto	Obrigatória
--	--------------------------	--------------------

A Equipe de Projetos valida o Plano de Desenvolvimento estabelecido, discutindo os objetivos de cada ciclo, opinando com relação a dependências e riscos.

CRITÉRIO DE SAÍDA

- Lista de Funcionalidades validada e priorizada.
- Plano de Desenvolvimento estabelecido e validado.

ECO – PROCESSO 8
CRIE BACKLOGS A PARTIR DA LISTA DE
FUNCIONALIDADES

Uma vez definido o Plano de Desenvolvimento, este processo o detalha criando o *Backlog* do próximo ciclo de desenvolvimento e, caso a Equipe de Projeto ache útil, criando os *Backlogs* de ciclos subsequentes. De acordo com o Direcionamento dado ao projeto (por Conteúdo ou por Serviço), estabelece-se a responsabilidade pela integração das tarefas, compondo as Funcionalidades a serem entregues ao final do ciclo. A evolução do desenvolvimento é feito diariamente através das Reuniões de 15 minutos. Cumpre ressaltar que este processo, assim como os demais processos compreendidos entre o 7 e o 13 ocorrem de forma totalmente não linear e com diversas intersecções e influências cruzadas. Cabe à Equipe de Projeto auto organizar-se para atingir o objetivo definido para o ciclo da melhor forma.

CRITÉRIO DE ENTRADA

- Plano de Desenvolvimento estabelecido.
- Objetivo do ciclo definido e comunicado à equipe de projeto.

TAREFAS

Defina o Direcionamento do Projeto	Gerente de Projetos, Líder de EH e Líder de Software	Obrigatória
------------------------------------	--	-------------

Gerente de Projetos em conjunto com Líderes de Experiência Humana e Software definem se o projeto será Direcionado por Conteúdo ou Direcionado por Serviço. Esta decisão implica na definição da prioridade de desenvolvimento entre Conteúdos e Serviços, e na responsabilidade pela integração e verificação das tarefas das Equipes de EH e Software.

Crie os <i>backlogs</i> de Serviços e Conteúdos	Equipe de Projeto	Obrigatória
---	-------------------	-------------

Em reunião facilitada pelo Gerente de Projetos, a Equipe de Projeto deve gerar os *backlogs* de Serviços e Conteúdos de modo a atingir o objetivo definido para o ciclo de desenvolvimento. Estes Backlogs contém as tarefas funcionais e não funcionais relacionadas a cada subgrupo da Equipe de Projetos. O *backlog* de

Conteúdos possui as tarefas de criação ou adequação da identidade visual, *layout* de páginas, objetos multimídia, textos, compatibilidade entre *browsers*, etc. O *backlog* de Serviços possui a lista de serviços no formato <ação><resultado><objeto>, restrições de desempenho, necessidade de integração, etc.

À medida que os *Backlogs* são criados, a Equipe de Projetos estima o esforço de cada tarefa. Caso o esforço total seja diferente do estimado inicialmente, cabe ao Gerente de Projetos negociar as alterações junto ao Gerente de Produto, revisando o Plano de Desenvolvimento.

Distribua as tarefas entre os membros da Equipe de Projeto

Gerente de Projetos

Obrigatória

O Gerente de Projetos aloca pessoas para a execução de cada entrada existente nos *Backlogs* de Conteúdos e Serviços. Adicionalmente, de acordo com o Direcionamento do Projeto, o respectivo Líder recebe a lista de Funcionalidades a serem entregues. Cabe ao Líder designado organizar a equipe de modo a garantir o sucesso na integração das diferentes tarefas.

VERIFICAÇÃO

Verificação de completude

**Gerente de Projetos,
Líder de EH e Software**

Obrigatória

Gerente de Projetos e Líderes verificam se os *backlogs* criados tem todas as tarefas necessárias para que o objetivo do ciclo seja atingido.

CRITÉRIO DE SAÍDA

- Direcionamento do Projeto Definido.
- *Backlog* de Conteúdos criado, validado e com todas as tarefas atribuídas aos membros da Equipe de Experiência Humana.
- *Backlog* de Serviços criado, validado e com todas as tarefas atribuídas aos membros da Equipe de Software.

ECO – PROCESSO 9

MODELE POR CONTEÚDOS E SERVIÇOS

Este processo ocorre por Funcionalidade a ser criada.

A Equipe de Projeto se auto-organiza para avaliar se a solução atual está apta para receber os Conteúdos e Serviços definidos. Em caso negativo, a Equipe de Projeto deve fazer um *refactoring* da solução de modo que os Conteúdos e Serviços sejam facilmente incluídos e, futuramente, sejam de fácil manutenção e evolução. As tarefas deste processo (e dos demais) não ocorrem de forma linear, havendo muita “comunicação” e influências mútuas entre as diversas tarefas e processos.

CRITÉRIO DE ENTRADA

- *Backlogs* de Serviços e Conteúdos criados e alocados.

TAREFAS

Discuta os detalhes do ciclo e seu objetivo	Equipe de Projeto, Gerente de Produto	Opcional
--	--	-----------------

Caso os Líderes de EH e/ou Software julguem necessário, pode-se agendar reuniões com o Gerente de Produto para discutir os detalhes do objetivo do ciclo, validar das decisões da equipe e esclarecer dúvidas gerais.

Analise os artefatos e documentos disponíveis	Equipe de Projeto	Opcional
--	--------------------------	-----------------

Caso a Equipe de Experiência Humana e/ou a Equipe de Software ache conveniente, pode-se estudar os artefatos e documentação entregue pelo cliente. Esta tarefa normalmente é executada pela Equipe de para Experiência Humana na análise de normas e padrões de aplicação de marcas e diretrizes de identidade visual.

Realize o “briefing de criação”	Equipe de Experiência Humana, Especialista no Domínio	Obrigatória
--	--	--------------------

A Equipe de Experiência Humana reúne-se com o Especialista no Domínio (este especialista é algum representante do cliente capaz de determinar a linha de comunicação desejada, as diretrizes e padrões a serem seguidos, etc.) e

define as diretrizes de comunicação a serem aplicadas nos Conteúdos do SW. Durante esta reunião pode-se gerar um esboço de um documento de Padrões de Comunicação ou receber documentos e informações a este respeito. Esta tarefa é bastante interativa, com a Equipe de EH criando, de forma interativa, os *layouts* e Diretrizes Visuais do SW.

Refine os Modelos Gerais

Equipe de Projeto

Obrigatória

Com base no conhecimento adquirido através das outras tarefas e processos, a Equipe de Projeto refina os Modelos Gerais da solução de modo a viabilizar do desenvolvimento dos Conteúdos e Serviços do Ciclo.

Crie os *layouts* base

**Líder e EH e
Diretor de arte**

Obrigatória

A partir dos resultados do “*briefing* de criação” e dos Modelos de Grade validados pelo cliente (caso estes tenham sido construídos), o Líder e EH e Diretor(es) de arte criam os *layouts* de base para as páginas-base do SW. Comumente esta atividade é bastante interativa, com várias versões de *layout* sendo criados antes da aprovação final. É importante que o Líder de EH e/ou o Diretor de arte tenham boas habilidades de argumentação e apresentação para que os *layouts* sejam aprovados rapidamente (esta argumentação é chamada de “defesa da criação”).

VERIFICAÇÃO

Verificação Interna e Externa

Equipe de projeto

Obrigatória

As Equipes de EH e Software validam seus modelos internamente e externamente. Esta verificação é fundamental para os *layouts* criados.

CRITÉRIO DE SAÍDA

- Diretrizes de comunicação definidas e aprovadas pelo cliente
- *Layouts* base criados e aprovados pelo cliente.
- Modelos atualizados e com anotações de alternativas, decisões e riscos.

ECO – PROCESSO 10

IMPLEMENTE POR CONTEÚDO E SERVIÇO

Este processo ocorre por Conteúdo ou Serviço visando criar Funcionalidades completas com valor para o cliente. A partir dos resultados do processo anterior, a Equipe de Experiência Humana cria um protótipo navegável do SW. Este protótipo será evoluído a cada ciclo e utilizado na versão final do SW. A Equipe de Software, por sua vez, implementa os Serviços necessários adicionando o código gerado de forma incremental ao repositório da solução.

CRITÉRIO DE ENTRADA

- A Equipe de Projeto concluiu o Processo 9 a contento.

TAREFAS

Implemente os Serviços Desenvolvedores de Software Obrigatória

Os Desenvolvedores de Software implementam os Serviços de seu *backlog*, incluindo a implementação de testes unitários automatizados. Fica a cargo da equipe da utilização de práticas como programação em pares e implementação de testes a priori.

Redija os Conteúdos Redatores Obrigatória

Para cada item presente no *Backlog* de Conteúdos o(s) Redator(es) redige(m) os textos necessários de acordo com as diretrizes de comunicação definidas.

Crie os objetos visuais necessários para cada Conteúdo Diretores de arte Obrigatória

Para cada item presente no *Backlog* de Conteúdos o(s) Diretor(es) de arte criam os objetos visuais necessários, como botões, animações em *flash*, etc. aplicando a estes as diretrizes visuais definidas.

Monte os Conteúdos Desenvolvedores de EH Obrigatória

A partir dos *layouts* base, objetos visuais e dos textos redigidos, os Desenvolvedores de EH montam as páginas HTML, incluem os textos no estilo definido pelas Diretrizes Visuais e implementam códigos do lado cliente

necessários (javascript, action script, etc.), adicionando o resultado final ao Protótipo Navegável do SW.

Realize testes unitários	Desenvolvedores de Software	Obrigatória
---------------------------------	------------------------------------	--------------------

Os Desenvolvedores de Software realizam o teste unitário de seus códigos fonte através de alguma ferramenta de teste automatizado.

Teste o Protótipo navegável	Equipe de Experiência Humana	Obrigatória
------------------------------------	-------------------------------------	--------------------

A Equipe de Experiência Humana testa o Protótipo navegável criado avaliando a correteude dos textos, adequação às Diretrizes Visuais, navegabilidade, etc.

Inclua os Conteúdos e Serviços no controle a versão	Líderes de EH e Software	Obrigatória
--	---------------------------------	--------------------

Uma vez testado o Protótipo navegável, este é colocado no controle de versão e rotulado pelo Líder de Experiência Humana. Eventualmente o protótipo é implantado num servidor web para a realização de apresentações e validações. Analogamente, o Líder de Software coloca os códigos implementados e testados sob controle de versão para posterior integração de Conteúdos e Serviços.

VERIFICAÇÃO

Teste do Protótipo navegável realizado	Equipe de EH	Obrigatória
---	---------------------	--------------------

Testes do Protótipo Navegável realizado com sucesso.

Testes Unitários realizados	Equipe de Software	Obrigatória
------------------------------------	---------------------------	--------------------

Todos os testes unitários automatizados foram executados com sucesso.

CRITÉRIO DE SAÍDA

- Todos os itens dos *Backlog* de Conteúdos e Serviços desenvolvidos e testados.

ECO – PROCESSO 11

INTEGRE POR FUNCIONALIDADE

À Medida que os Conteúdos e Serviços são desenvolvidos, os mesmos são integrados para a entrega de Funcionalidades utilizáveis para o cliente. A Integração deve ser feita constantemente, num pior caso uma vez ao dia.

Caso o SW seja Direcionado por Serviços, a integração é feita pelo Líder de Software. Caso contrário, a integração é feita pelo Líder de Experiência Humana. Na prática, a integração pode ser feita por um Desenvolvedor cabendo ao Líder a validação do trabalho feito.

CRITÉRIO DE ENTRADA

- Conteúdos e Serviços que compõem uma Funcionalidade criados, testados e sob controle de versões.

TAREFAS

Realize reuniões de diárias de 15 minutos diárias	Equipe de Projeto	Obrigatória
--	--------------------------	--------------------

Diariamente a Equipe de Projeto se reúne para uma reunião de “15 minutos” onde o progresso do projeto é avaliado. A reunião é facilitada pelo Gerente de Projetos e cada membro da Equipe de Projeto deve responder às seguintes perguntas:

- Quais atividades realizei desde a última reunião de 15 minutos.
- Quais atividades realizarei até a próxima reunião de 15 minutos.
- O que evitou que eu trabalhasse com máxima produtividade.

O Gerente de Projetos anota o andamento do projeto e fica encarregado de resolver qualquer problema citado que esteja afetando a máxima produtividade da equipe.

As reuniões de 15 minutos devem ser breves e objetivas, não havendo espaço para tomadas de decisões ou realização de discussões. Caso seja necessário, reuniões subseqüentes podem ser agendadas para discussões e decisões.

Integre os Conteúdos e Serviços necessários para a Funcionalidade	Líder e Desenvolvedor	Obrigatória
--	------------------------------	--------------------

À medida que os Conteúdos e Serviços que compõem cada Funcionalidade planejada para o ciclo corrente forem concluídos, o Líder e Desenvolvedor de Experiência Humana, no caso de SW Direcionado por Conteúdo, ou o Líder e Desenvolvedor de Software, no caso de Direcionamento por Serviços, realizam a integração dos Conteúdos e Serviços que compõem cada Funcionalidade. Esta integração deve ser feita rotineiramente, tão logo os Conteúdos e Serviços de uma Funcionalidade estejam prontos para tanto.

Promova as Funcionalidades integradas	Líder	Obrigatória
--	--------------	--------------------

As funcionalidades integradas são promovidas no controle de versão. Deve-se manter uma versão completa do SW integrada e pronta para avaliação com atualização no mínimo diária. Cabe à Equipe Cliente decidir se esta versão deve ou não ser implantada em produção.

VERIFICAÇÃO

Funcionalidades do Ciclo integradas	Líder e Gerente de Projetos	Obrigatória
--	------------------------------------	--------------------

Gerente de Projetos e Líder (de EH ou Software) devem verificar se todas as Funcionalidades do ciclo foram integradas e promovidas no controle de versões.

CRITÉRIO DE SAÍDA

- Todas as Funcionalidades planejadas para o ciclo integradas.

ECO – PROCESSO 12

TESTE POR FUNCIONALIDADE

Uma vez integrada, cada Funcionalidade é submetida a testes funcionais conduzidos pelo Gerente de Projetos e conjunto com a Equipe Cliente.

CRITÉRIO DE ENTRADA

- Versão integrada e implantada em ambiente de testes.

TAREFAS

Definir casos de teste	Gerente de Produto, Gerente de Projetos	Opcional
-------------------------------	--	-----------------

Gerente de Produto e Gerente de Projetos documentam os casos de teste funcional a serem executados para cada funcionalidade. Esta tarefa é executada em algum momento entre o término do Plano de Desenvolvimento e o início da tarefa seguinte. O formato como os testes são planejados e conduzidos fica à cargo da equipe, podendo, inclusive, não haver formalização do plano de testes (testes *ad hoc*), situação bastante comum em projetos mais simples, notadamente os Direcionados por Conteúdo.

Conduzir testes funcionais por Funcionalidade	Gerente de Produto, Gerente de Projetos	Obrigatória
--	--	--------------------

Gerentes de Produto e/ou Gerentes de Projetos conduzem testes funcionais para cada Funcionalidade. Qualquer não conformidade é documentada e incluída na Lista de Funcionalidades como uma nova Funcionalidade. Cabe ao Gerente de Produto priorizar as não conformidades em detrimento de outras Funcionalidades.

Rotular Funcionalidades aprovadas	Líder, Gerente de Produto	Obrigatória
--	----------------------------------	--------------------

As Funcionalidades testadas e aprovadas são promovidas para a versão “estável” do SW e rotuladas segundo o padrão de numeração em uso.

VERIFICAÇÃO

**Testes funcionais
executados e aceitos**

**Gerente de Projetos e
Gerente de Produto**

Obrigatória

Testes de aceite conduzidos e aprovados pelo Gerente de Produto.

CRITÉRIO DE SAÍDA

- Funcionalidades testadas e aprovadas incluídas na versão estável do SW.
- Versão estável rotulada.
- Incluída descrição da Funcionalidade integrada no documento de Notas de Entrega.

ECO – PROCESSO 13

ENTREGUE/TREINE POR FUNCIONALIDADE

Nas datas definidas no Plano de Desenvolvimento para a entrega de versões, é feita um seminário de entrega e apresentação da última versão estável do SW para a Equipe Cliente e, eventualmente, para representantes dos Públicos do sistema. De acordo com o contratado, O seminário pode ter o formato de um treinamento ou de uma apresentação “informal”. Via de regra, o seminário é informal e marca a entrega de um manual de uso ou ajuda *on-line* das Funcionalidades desenvolvidas. A entrega de versões marca também a realização de um *workshop* de auto-avaliação por parte da equipe. Eventualmente, caso o Gerente de Projetos julgue necessário, a Equipe Cliente pode participar deste *workshop*.

CRITÉRIO DE ENTRADA

- Chegada a data de entrega de versão definida no Cronograma.
- Última versão estável do SW implantada em ambiente de homologação.

TAREFAS

Desenvolva um manual de uso ou ajuda on-line	Gerente de Projetos, Redator, Equipe de Projeto	Opcional
---	--	-----------------

O Gerente de Projetos, auxiliado por um Redator e, caso necessário, por membros da Equipe de Projeto, redigem um manual de uso para cada Funcionalidade entregue. Este manual pode ser entregue on-line, impresso em formato de manual ou, no caso de SW Direcionados por Serviços, em formato de descritivo de uma API.

Conduza um seminário de apresentação por Funcionalidade	Gerente de Projetos, Equipe Cliente	Obrigatória
--	--	--------------------

Na data definida no Plano de Desenvolvimento para a entrega de versões, o Gerente de Projetos conduz um seminário de apresentação das Funcionalidades presentes na última versão estável do Sistema Web. Este seminário pode ser um treinamento formal para os Públicos do sistema ou uma apresentação informal para a Equipe Cliente. Como resultado do seminário, o Gerente de Projeto coleta uma aprovação “formal” da entrega da versão junto à Equipe Cliente.

Eventuais observações e sugestões são incluídas no *Backlog* do Produto como novas Funcionalidades.

Realize um <i>workshop</i> de avaliação do processo	Equipe de Projeto	Obrigatória
--	--------------------------	--------------------

Após cada entrega de versão (ou, idealmente, após cada ciclo de desenvolvimento), a equipe se reúne para avaliar suas práticas e processos. Nesta reunião, facilitada pelo Gerente de Projeto, a equipe deve definir uma lista de ações para os seguintes pontos:

- O que deve ser mantido?
- Quais os problemas enfrentados?
- O que podemos adotar para minimizar os problemas enfrentados?

O Gerente de Projetos anota o resultado e trabalha junto à equipe para aplicar o resultado alcançado.

VERIFICAÇÃO

Aprovação da versão entregue	Equipe Cliente	Obrigatória
-------------------------------------	-----------------------	--------------------

Equipe Cliente aprova “formalmente” a versão entregue e apresentada no seminário.

Pontos para melhoria contínua	Equipe de Projeto	Obrigatória
--------------------------------------	--------------------------	--------------------

Equipe de Projeto elenca os pontos positivos, problemas e ações a serem tomadas visando a melhoria contínua de suas práticas e processos.

CRITÉRIO DE SAÍDA

- Aceite “formal” de versão por parte do cliente.
- Elaborado, quando contratado, manual de uso das Funcionalidades entregues.
- Ministrado seminários de apresentação de versão estável.
- Definidos pontos para melhoria contínua das práticas e processos da equipe.

ECO – PROCESSO 14

REALIZE WORKSHOPS DE ENCERRAMENTO

Ao término do contrato, o Gerente de Projeto e o Gerente de Negócios realizam um *workshop* de encerramento do contrato junto à Equipe Cliente e à Equipe de Projeto visando coletar informações úteis para a melhoria contínua da empresa, por um lado, e por outro, quando houver possibilidade, visando a formalização de um novo contrato.

A Equipe de Projetos realiza um *workshop* voltado à identificação de artefatos e componentes potencialmente reutilizáveis. É comum a identificação de componentes potencialmente reutilizáveis gerar um novo projeto focado na preparação deste componente para reuso. Os artefatos e componentes reutilizáveis são versionados e publicados.

CRITÉRIO DE ENTRADA

- Contrato formalizado no Processo 5 em vias de encerramento.

TAREFAS

Conduza um <i>workshop</i> de encerramento	Gerente de Projeto, Gerente de Negócios, Equipe Cliente	Opcional
---	--	-----------------

Neste *workshop*, conduzido pelo Gerente de Negócios, é feita uma avaliação por parte da Equipe Cliente sobre o andamento do projeto, qualidade do atendimento prestado, efetividade do resultado produzido, qualidade do produto final, etc. Como resultado do *workshop*, é feita uma lista de pontos positivos e pontos a desenvolver da empresa em geral e da Equipe de Projeto em particular. Caso haja possibilidade, o Gerente de Negócios inicia as negociações de um novo contrato.

Realize um <i>workshop</i> de avaliação do processo	Equipe de Projeto	Obrigatória
--	--------------------------	--------------------

Após o encerramento do contrato, a equipe se reúne para avaliar suas práticas e processos num *workshop* semelhante ao descrito na tarefa homônima do Processo 13.

Realize um <i>workshop</i> de Reuso	Equipe de Projeto	Obrigatória
--	--------------------------	--------------------

Após o encerramento do contrato, a equipe se reúne para avaliar quais artefatos e componentes criados tem potencial de reuso em outros projetos. A Equipe de Projeto cria uma lista de artefatos e componentes potencialmente reutilizáveis.

Controlar e Publicar artefatos e componentes reutilizáveis	Gerente de Projeto, Líder de EH e Software	Obrigatória
---	---	--------------------

Com base da lista de artefatos e componentes potencialmente reutilizáveis gerada na tarefa anterior, o Gerente de Projetos, auxiliado de acordo com a necessidade, pelos Líderes de EH e Software, preparam os artefatos e componentes listados para versionamento no repositório de componentes e publicação no meio de divulgação de componentes utilizado pela empresa, normalmente uma intranet. Estes artefatos e componentes ficam então à disposição para reuso em novos projetos.

VERIFICAÇÃO

Avaliação Interna e Externa da Empresa e do Projeto	Equipe Cliente e Equipe de Projeto	Obrigatória
--	---	--------------------

Avaliação da empresa em geral e da Equipe de Projeto em particular através das tarefas deste processo.

CRITÉRIO DE SAÍDA

- Plano de ação para melhoria contínua.
- Versionamento e publicação de artefatos e componentes reutilizáveis.
- Iniciadas negociações de novo contrato, quando possível.

ECO – PROCESSO 15
FORMALIZE O TÉRMINO DO CONTRATO

Este processo marca o encerramento de um contrato sendo, portanto, executado uma única vez. O Gerente de Projetos e o Gerente de Negócios revisam as cláusulas contratuais visando garantir que todas o correto encerramento do contrato.

CRITÉRIO DE ENTRADA

- Contrato finalizado “informalmente”.

TAREFAS

Revise as cláusulas contratuais	Gerente de Projeto e Gerente de Negócios	Obrigatória
---------------------------------	--	-------------

Com o intuito de evitar perdas financeiras e desgastes desnecessários, o Gerente de Projetos e o Gerente de Negócios analisam as cláusulas contratuais com o objetivo de executar todos os formalismos necessários para o encerramento formal do contrato. Estes pontos incluem, por exemplo, aceite formal de entregas para fins de contagem do prazo de garantia, entrega formal de artefatos e código fonte, acertos em acordos de confidencialidade, não aliciamento e propriedade intelectual, entre outros.

Formalize o encerramento	Gerente de Negócios	Obrigatória
--------------------------	---------------------	-------------

A partir dos resultados da tarefa anterior, o Gerente de Projetos formaliza o encerramento do contrato através da coleta de assinaturas e formalizações necessárias junto à Equipe Cliente e Equipe de Projetos.

VERIFICAÇÃO

Formalização do encerramento	Gerente de Negócios	Obrigatória
------------------------------	---------------------	-------------

Cliente e empresa de acordo com encerramento formal, não havendo pendências de ambas as partes.

CRITÉRIO DE SAÍDA

- Contrato formalmente finalizado.

5.4. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas as atividades fundamentais cobertas por ECO. Estas atividades, comuns ao processo de negócios de qualquer empresa ou equipe fornecedora de Soluções Web oferecem o arcabouço geral para a estruturação dos processos que compõem ECO.

Os processos de ECO são descritos através do modelo ETVS, que permite a descrição completa de cada processo em apenas uma página frente e verso. Assim, ao invés da descrição através de longos documentos de leitura difícil e cansativa, o Ecossistema completo é descrito em apenas 15 páginas, que podem ser impressas e plastificadas para fácil manuseio e referência, ou disponibilizadas numa Intranet corporativa para acesso *online*.

Em última análise, este modelo de apresentação representa muito bem a filosofia utilizada como base para os Ecossistemas de Criação Ágil de Sistemas Web - soluções simples, diretas, fáceis de usar e despojadas de acessórios pouco úteis.

CAPÍTULO 6: ESTUDO DE CASO

6.1. CONSIDERAÇÕES INICIAIS

Os dois capítulos anteriores descreveram o **estado atual** do principal produto deste trabalho, o Ecossistema de Desenvolvimento Ágil de Sistemas Web denominado ECO. O destaque para “estado atual” reflete o processo de evolução pela qual ECO passou desde a sua concepção inicial já que ECO vem sendo utilizado no dia a dia de uma empresa focada no desenvolvimento de Sistemas Web a cerca de dois anos.

Este capítulo apresenta a empresa onde ECO é utilizado, descreve a organização desta empresa e faz uma breve avaliação do tipo de projeto que ela realiza. É feito também um estudo mais detalhado de três estudos de caso envolvendo projetos com características distintas e que resultaram na avaliação de ECO em diferentes situações.

6.2. EMPRESA

Este tópico descreve a Lúcida, empresa na qual ECO é utilizado. As descrições abaixo foram fornecidas pela própria empresa.

6.2.1. VISÃO GERAL DA EMPRESA

A Lúcida posiciona-se como uma fornecedora de soluções completas em Internet, oferecendo experiência e conhecimento no uso operacional, gerencial e estratégico da Internet como ferramenta de negócios e relacionamento. A empresa tem como missão auxiliar seus clientes a obterem o maior valor possível da Internet como ferramenta de negócios e relacionamento.

Para tanto, a Lúcida atua como sendo uma equipe de especialistas em Internet de seus clientes, oferecendo soluções conectadas com os objetivos estratégicos de seus clientes, visando trazer valor tangível e resultado constante.

6.2.2. ESTRUTURA DA EMPRESA

A Lúcida é dividida em quatro unidades de negócio independentes e especializadas, cada qual com um conjunto específico de conhecimentos e habilidades necessárias para a entrega de soluções completas em Internet. A utilização de ECO neste contexto visa viabilizar o trabalho destas unidades de forma coordenada e sinérgica, com o intuito de entregar valor de negócio rápida e constantemente aos clientes da empresa.

Figura 11: Estrutura da empresa onde os estudos de caso de ECO foram realizados.

Os tópicos seguintes descrevem brevemente as unidades de negócio da Lúcida.

GRUPO LÚCIDA

Possui a capacidade gerencial e estratégica para entregar projetos de Internet. Esta unidade é composta por uma equipe de gerentes experiente e capacitada nas melhores práticas metodológicas e de gerenciamento. Durante o período de utilização de ECO, esta unidade de negócio contava, em média, com 8 pessoas, entre Gerentes de Projeto (4), Gerente de Recursos (1), Gerente de Negócios (3).

LÚCIDA STUDIO

Trata-se de uma Agência Interativa, com capacidades de comunicação digital, marketing de relacionamento, layout e identidade visual. Na terminologia de ECO, possui os papéis relacionados à camada de Experiência Humana. Durante o período de utilização de ECO, esta unidade de negócio contava, em média, com 10 pessoas, entre Líderes de EH (2), Diretores de Arte (4), Desenvolvedores de EH (3) e Redatores (1).

LÚCIDA TECNOLOGIA

Unidade focada no conhecimento tecnológico dos SW. Possui especialistas em Web Services, Java, .Net, bancos de dados, infra-estrutura e integração de sistemas. Inclui os papéis de software de ECO. Durante o período de utilização de ECO, esta unidade de negócio contava com 9 pessoas, entre Líderes de Software (3) e Desenvolvedores de Software (6).

LÚCIDA SOFTWARE

Esta unidade de negócios é a responsável por gerir e comercializar os componentes de software desenvolvidos pela empresa. Durante o período de utilização de ECO não havia pessoas dedicadas em tempo integral a esta unidade.

6.3. ESTUDOS DE CASO

Como citado anteriormente, ECO vem sendo utilizado no dia a dia da Lúcida a cerca de 2 anos. A adoção de ECO pela empresa ocorreu naturalmente já que sua organização é adequada para os papéis e práticas utilizados por ECO, não havendo a necessidade de grandes mudanças culturais na empresa para seu uso.

Neste período, diversos projetos foram desenvolvidos utilizando-se ECO. Em linhas gerais, estes projetos possuem Equipes de Projeto com cerca de 7 pessoas, normalmente compartilhadas entre vários projetos, duração média de 6 meses, exceção feita aos projetos denominados pela Lúcida de *Web Outsourcing*, cujos contratos são do tipo esforço por período e estendem-se por diversos meses e, em alguns casos, anos. A variedade de empresas para os quais os projetos utilizando ECO foram desenvolvidos é bastante grande, incluindo empresas nacionais e multinacionais de diversos ramos de atividade, com projetos das mais diversas naturezas e com os mais variados objetivos. Se por um lado a avaliação de ECO feita até o momento tem o viés de ter ocorrido em apenas uma empresa, por outro a diversidade de projetos nos quais esta avaliação ocorreu a torna válida e bastante útil para comparações futuras.

Os tópicos seguintes descrevem três estudos de caso onde ECO foi aplicado. Estes casos serão descritos brevemente seguindo o seguinte formato:

Empresa: Breve descrição da empresa cliente

Projeto: Visão geral do projeto

Equipe: Perfis e tamanho da equipe

Resultados: Uma breve descrição dos resultados atingidos

Lições: Quais as lições tiradas deste projeto e sua influência para ECO

6.3.1. CASO 1: PORTAL DE RELACIONAMENTO

EMPRESA

Empresa multinacional do setor de alimentos, líder em diversos segmentos do mercado. O objetivo principal do projeto era diminuir o volume de ligações recebidas pelo *call center* da empresa com solicitações de receitas. De forma secundária, o projeto deveria oferecer recursos para entender melhor o comportamento e anseios do Público-alvo, composto basicamente por mulheres. Um ponto importante é que todos os membros da Equipe Cliente eram profissionais de marketing, sem nenhuma experiência prévia em gerencia de projetos e Sistemas Web. Também havia grande incerteza com relação ao escopo do projeto.

PROJETO

O projeto foi contratado com modelo de escopo fechado, teve duração estimada em 5 meses com o desenvolvimento realizado em 8 ciclos de 2 semanas cada. A cada 2 ciclos uma versão era entregue para o cliente.

EQUIPE

A Equipe alocada para o projeto era composta por um Desenvolvedor de Software em tempo integral e outro em tempo parcial, um Desenvolvedor de EH em tempo parcial, Um Diretor de arte em tempo parcial, um Gerente de Projetos em tempo parcial e dois redatores em tempo parcial, alocados apenas nas últimas interações do projeto.

RESULTADOS

O projeto foi terminado no prazo e dentro do orçamento. Graças às entregas formais intermediárias (4 ao todo), a indecisão da Equipe Cliente com relação ao escopo não teve impacto negativo no andamento do projeto. O desenvolvimento em ciclos com entregas formais foi ressaltado pela Equipe Cliente como uma prática positiva. Os objetivos traçados para o projeto foram atingidos e, após seu Nascimento, o projeto sofreu algumas poucas evoluções. Em virtude de problemas financeiros enfrentados pela empresa cliente, o projeto morreu cerca de 4 meses após seu nascimento.

LIÇÕES

- As reuniões de 15 minutos mostraram-se fundamentais para o bom andamento do projeto em função do compartilhamento de equipe e das constantes mudanças nas funcionalidades planejadas para os ciclos seguintes. Após este projeto tornaram-se prática comum na empresa.
- Duas importantes atividades do projeto ficaram sob responsabilidade da Equipe Cliente e foram concluídas muito depois do previsto. A partir desta experiência, duas ações foram tomadas: Alterou-se o contrato padrão utilizado, eximindo a empresa de responsabilidades sob atrasos no cronograma causados pelo cliente. Também passou-se a incluir no backlog do produto as tarefas passadas para o cliente. Os processos de ECO poderiam ser atualizados para refletir tarefas normalmente realizadas pelo cliente ou para promover o acompanhamento de atividades específicas de responsabilidade da Equipe Cliente.

6.3.2. CASO 2: EVOLUÇÃO CONTÍNUA

EMPRESA

Empresa multinacional do setor de cosméticos, líder em diversos segmentos do mercado. O objetivo principal do projeto era manter o portal de empresa constantemente atualizado e realizar ações de marketing online regularmente, ao ritmo de pelo menos uma ação por mês.

PROJETO

O projeto foi contratado com modelo de esforço por período. Este contrato foi firmado em 2002 e continua vigente até hoje. A utilização de ECO teve início a partir de 2004, quando o volume de horas mensais passou de cerca de 150 horas para mais de 500. Como consequência, o volume de trabalho e a complexidade de organização da equipe aumentaram sobremaneira. Via de regra, este contrato mantinha cerca de 5 projetos em andamento simultaneamente.

EQUIPE

Estes projetos variavam de trabalhos pontuais feitos por uma única pessoa em menos de 1 semana, até projetos com Equipes de Projetos compostas por 5 perfis e com duração de 3 a 4 meses. Por se tratar de um típico projeto Direcionado por Conteúdo, as Equipes de Projeto eram compostas quase que totalmente por perfis de Experiência Humana.

RESULTADOS

As demandas deste cliente sempre exigiram grande agilidade e rapidez na entrega do resultado. Algumas experiências interessantes foram a vivência do Ciclo de Vida de um produto completo ocorrendo em cerca de 6 meses, com a percepção clara de passagem por todas as etapas do Ciclo de Vida. Em virtude do porte e do nível de exigência desta empresa, o fato de o contrato ter sido renovado por três vezes atesta a satisfação com os resultados obtidos. Neste sentido, a experiência com este cliente foi importante já que todos os 15 processos de ECO foram executados completamente por 3 vezes.

LIÇÕES

- A utilização de *backlogs* distintos para cada Projeto em andamento e a separação destes em *Backlogs* de Conteúdos e Serviços mostrou-se bastante útil e funcional. Além de manter a organização e visibilidade do andamento de todas as iniciativas, os diversos *Backlogs* simplificavam o processo de reportagem do consumo do orçamento (leia-se consumo de horas) para a Equipe Cliente.
- Em projetos com carga constante de trabalho e prazos apertados o desgaste da Equipe é bastante acentuado. Assim, é fundamental que haja uma rotação constante na alocação da equipe.

6.3.3. CASO 3: DIRECIONADO POR PLANEJAMENTO X ÁGIL

EMPRESA

Empresa nacional do ramo de telefonia celular. O objetivo principal do projeto era o desenvolvimento de ser novo portal de relacionamento, incluindo diversos serviços, recursos de administração e integração com outros sistemas.

PROJETO

O projeto foi contratado no modelo escopo fechado, mas com abertura para esforço adicional. O projeto vinha sendo desenvolvido por outros dois fornecedores durante cerca de 18 meses através de uma metodologia direcionada por planejamento, sem a entrega de nenhum resultado tangível. Graças a isto, este caso foi interessante por dois motivos. Em primeiro lugar ECO foi utilizado em um produto na etapa de Geração de seu Ciclo de Vida. Em segundo lugar, permitiu a comparação de um Método Ágil com uma Metodologia Direcionada por Planejamento em termos de resultado e satisfação do cliente.

EQUIPE

Este foi o principal caso de ECO em termos de tamanho da Equipe de Projeto. Nos primeiros 4 meses de desenvolvimento, a equipe foi composta por mais de 20 pessoas, divididas em 3 equipes trabalhando em paralelo.

RESULTADOS

Após os 18 meses de desenvolvimento utilizando uma metodologia direcionada por planejamento sem a entrega de nenhuma versão em produção, a adoção de ECO gerou uma versão apta para produção em 3 meses, com entregas de 2 versões intermediárias para avaliação da Equipe Cliente, atendendo ao objetivo de lançar o novo portal antes do natal daquele ano (2004). A versão lançada possuía apenas os Serviços e Conteúdos vitais para o

lançamento do produto e o restante do *backlog* do produto foi entregue ao ritmo de uma nova versão por mês. Após 6 meses de projeto, todas as Funcionalidades prioritárias haviam sido entregues e o restante do *backlog* passou a ser desenvolvido através de um contrato de esforço por mês, com uma equipe de 5 pessoas.

LIÇÕES

- Este projeto foi o responsável pela inclusão do Processo 15 – Formalize o término do contrato - em virtude de problemas ocorridos em função da não formalização de alguns detalhes contratuais após a entrega da primeira versão do produto, dentro do prazo fixado pelo cliente. Esta situação gerou grande desgaste do Gerente de Projetos e do Gerente de Negócios junto à Equipe Cliente, colocando em risco a continuidade do relacionamento. Estes problemas refletiram-se na Equipe de Projeto, gerando o atraso na entrega de uma das versões seguintes.
- É possível aplicar ECO em projetos com equipes maiores que as definidas, como foi o caso neste projeto. Entretanto, são necessárias modificações nos processos para atender a este tipo de necessidade.

6.4. CONSIDERAÇÕES FINAIS

Este capítulo fez uma breve apresentação de uma empresa que utiliza ECO em seu dia-a-dia. Além das características e organização desta empresa, foram apresentados estudos de caso onde ECO foi aplicado em diversas situações distintas, mostrando os pontos fortes deste Ecossistema para o Desenvolvimento Ágil de Sistemas Web e trazendo à tona algumas de suas limitações e situações limite de uso.

CAPÍTULO 7: CONCLUSÃO E TRABALHOS FUTUROS

7.1 CONCLUSÕES

A crescente relevância econômica dos Sistemas Computacionais em geral, e dos Sistemas Web em particular, aliada à sua crescente amplitude de aplicação, sobretudo como suporte a negócios e transações que ocorrem num mercado instável, que exige movimentos rápidos, colocou grande pressão sobre o desenvolvimento de software. O *Manifesto for Agile Software Development* veio como uma resposta a esta demanda, prometendo oferecer resposta rápida à mudança e maior controle sobre os prazos e orçamentos dos projetos.

Este trabalho apresentou uma análise dos Métodos Ágeis, apresentou uma visão histórica desta classe de metodologias, seu escopo e nicho de aplicação, além da descrição

de três dos principais Métodos Ágeis existentes em termos de suas práticas, papéis e processo.

Um processo de mudança que vinha se formando há algum tempo e que tomou corpo com o advento dos Métodos Ágeis, é o da contestação das bases filosóficas que permeiam o desenvolvimento de software. Esta onda de mudança encara o desenvolvimento de software como um processo de aprendizado, necessariamente empírico e complexo. Com efeito, uma corrente cada vez maior de autores e praticantes vem propondo a mudança da metáfora predominante no desenvolvimento de software, abandonando a metáfora da engenharia em detrimento da metáfora do viver.

Neste sentido, este trabalho apresentou como principal resultado ECO, um Ecossistema para o Desenvolvimento Ágil de Sistemas Web que utiliza os conceitos dos Métodos Ágeis e utiliza a metáfora do viver.

A partir dos resultados apresentados pela utilização de ECO, pode-se notar que os Métodos Ágeis em geral, e ECO em particular, representam um avanço em relação às ditas metodologias direcionadas pelo planejamento para esforços de desenvolvimento caracterizados pela instabilidade de requisitos e pressão por entregas constantes. Entretanto, a despeito dos discursos inflamados de muitos “agilistas”, fica claro que estas abordagens não são uma bala de prata no que tange à produtividade da equipe, atestando que o desenvolvimento de software continua sendo uma das atividades humanas mais complexas. Entretanto, a índice de satisfação demonstrada pelo cliente de um dos estudos de caso apresentados indica que os Métodos Ágeis podem ser uma bala de prata no que tange à qualidade percebida pelo cliente, oferecendo uma evolução de uma ordem de grandeza em relação aos processos anteriores no que tange à satisfação do cliente com o produto gerado.

7.3. TRABALHOS FUTUROS

Uma das bases deste trabalho é a crença de que a evolução contínua é a premissa para se atingir resultados de qualidade. A submissão de idéias e propostas à avaliação da

experiência prática é a melhor maneira de revelar problemas não vislumbrados e de enxergar os domínios enfrentados a partir de outros pontos de vista.

Neste sentido, o resultado da experiência no uso de ECO, apesar de ainda restrito, foi de grande valia para a determinação de possíveis melhorias e desdobramentos do trabalho aqui apresentado. Dentre as possibilidades de trabalhos futuros, destacam-se:

- **Evoluir o uso e a aplicação da metáfora do viver:** Uma metáfora é muito mais que um recurso didático, é um poderoso aliado na exploração de novas fronteiras. O uso da metáfora do viver no desenvolvimento de software, sutilmente utilizado neste trabalho, abre novas possibilidades de pesquisa e aplicação que podem ser objeto de trabalhos futuros.
- **Aplicação da Teoria da Complexidade ao desenvolvimento de software:** A aplicação da metáfora do viver nos leva a considerar um software como um sistema vivo, abrindo caminho para pesquisas da aplicação da Teoria da Complexidade ao desenvolvimento de software. Já existem diversos trabalhos neste sentido em outras áreas de conhecimento e alguns estudos começam a ser publicados aplicando a Teoria da Complexidade e Sistemas não lineares na área de desenvolvimento de software. O estudo de um ecossistema de desenvolvimento baseado nestes conceitos é um campo fértil para pesquisas futuras.
- **Evoluir os vínculos entre o Ambiente de um Sistema Web e o Ecossistema que o desenvolve:** ECO utiliza uma definição de Ambiente Lógico bastante simples em seu modelo atual. Entretanto, existem várias possibilidades de organização do Ambiente Lógico que podem ser melhor exploradas para aumentar a amplitude de utilização de ECO.
- **Desenvolvimento de ferramentas para suporte ao processo:** O desenvolvimento de ferramentas de suporte à ECO é um rico campo de trabalhos futuros, sobretudo à medida que ECO tornar-se mais adaptável a diferentes situações além das cobertas atualmente. As possibilidades vão desde ferramentas simples, como por exemplo para o gerenciamento de backlogs, ciclos de desenvolvimento e alocação de recursos,

até possibilidades mais complexas, como sistemas especialistas para auxílio à customização do Ecossistema de acordo com as especificidades do Sistema Web a ser criado.

- **Descrever ECO como uma *Pattern Language*:** A amplitude de aplicação de ECO bem como sua flexibilidade para adaptação a diferentes situações pode ser bastante melhorada através de sua descrição como uma *Pattern Language*. Recentemente, alguns trabalhos neste sentido vem sendo realizados, mas este campo de estudo ainda é bastante inexplorado.
- **Processos de processos de apoio à ECO:** Possibilidades interessantes de trabalhos futuros envolvem a definição de processos de apoio à ECO em particular, e ao Métodos Ágeis em geral, como por exemplo Gestão de Reuso, Controle de Qualidade, entre outros.
- **Adaptação do Ecossistema para projetos de maior porte:** Como visto no estudo de caso 3 apresentado anteriormente, a utilização de ECO em projetos de maior porte é possível, mas inadequada. Aliás, este é um problema já clássico entre os Métodos Ágeis.

REFERÊNCIAS BIBLIOGRÁFICAS

- [ABRAHAMSSON *et al*, 2002] ABRAHAMSSON, Pekka; Salo, Outi; Ronkainen, Jussi; Warsta, Juhani **Agile software development methods: review and analisis**. VTT publications, 2002.
- [ABRAHAMSSON *et al*, 2003] ABRAHAMSSON, Pekka; Salo, Outi; Ronkainen, Jussi; Warsta, Juhani **New directions on agile methods: A comparative analisys**. IEEE Proceedings of the 25th International Conference on Software Engineering (ICSE'03) 2003.
- [ALUR *et al*, 2003] ALUR, Deepak; MALKS, Dan; CRUPI, John. **Core J2EE Patterns: Best Practices and Design Strategies – Second Edition**. Prentice Hall, 2003.
- [AMBLER, 2002] AMBLER, Scott W. **Lessons in agility from internet-based development**. IEEE Software, mar./apr. 2002, p. 66-73.
- [AOYAMA, 1998] AOYAMA, Mikio **Web-Based agile software development**. IEEE Software, nov./dez. 1998, p. 56-65.
- [ARTHUR, 1988] ARTHUR, Lowell Jay **Software evolution – the software maintenance challenge**. John Wiley & Sons, 1988.

- [BECK, 1999] BECK, Kent **Embracing change with extreme programming.** IEEE Computer, oct. 1999, p.70-77.
- [BECK, 2000] BECK, Kent **Extreme programming explained: Embrace change.** Addison-Wesley, 2000.
- [BECK & FOWLER, 2001] BECK, Kent; FOWLER, Martin **Planning extreme programming.** Addison-Wesley, 2001.
- [BEEDLE *et al*, 1999a] BEEDLE, Mike; DEVOS, Martine; SHARON, Yonat; SCHWABER, Ken; SUTHERLAND Jeff **SCRUM: A pattern language for hyperproductive software development.** Pattern languages of program design (PLoPD4), Addison-Wesley, 2000, p; 637-651.
- [BEEDLE *et al*, 1999b] BEEDLE, Mike; DEVOS, Martine; SHARON, Yonat; SCHWABER, Ken; SUTHERLAND Jeff **SCRUM: An extension pattern language for hyperproductive software development.**
Disponível em: <http://www.controlchaos.com/>
Acessado em: 17 abr. 2003.
- [BEEDLE, 2001] BEEDLE, Mike; **Living Metaphor.** 2001.
Disponível em: <http://www.livingmetaphor.org/>
Acessado em: 01 jun. 2003.
- [BOOCH *et al*, 1999] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar **The Unified Modeling Language User Guide.** Addison-Wesley Longman, 1999.
- [BOHEM, 2001] BOHEM, Barry **Get ready for Agile Methods, with care.** IEEE Computer, Jan. 2001, p. 64-69
- [BOHEM & TURNER, 2003] BOHEM, Barry; TURNER, Richard **Using Risk to balance agile and plan-driven methods.** IEEE Computer, Jun. 2003, p. 57-66.
- [BROOKS, 1995] BROOKS, Frederick P., Jr. **The mythical man-month: Essays on software engineering – anniversary ed.** Addison-Wesley Longman, 1995.
- [BROWN *et al*, 1998] BROWN, William H.; MALVEAU, Raphael C.; McCORMICK

- III, Hays W.; MOWBRAY, Thomas **Anti-Patterns – Refactoring Software, Architectures, and Projects in Crisis**. John Wiley & Sons, 1998.
- [BROWN *et al*, 2001] BROWN, Kyle; CRAIG, Gary; HESTER, Greg; NISWONGER, Jaime; PITT, David; STINEHOUR, Russell. **Enterprise Java Programming with IBM WebSphere**. Addison-Wesley, 2001.
- [BRYANT, 2000] BRYANT, Antony ‘It’s engineering Jim ... but not as we know it’ - **Software Engineering: Solution for the software crisis, or part of the problem?** Proceedings of the 2000 International Conference, June 2000
- [CHAUBEY & SURESH, 2001] CHAUBEY, Rahul; SURESH, J.K. **Integration vs. development: an engineering approach to building web applications**. IEEE Software, feb. 2001, p. 171-181.
- [COAD *et al*, 1999] COAD, Peter; Lefebvre, Eric; De Luca, Jeff **Java Modeling in color with UML**. Prentice Hall PTR, 1999.
- [COCKBURN, HIGHSMITH, 2001] COCKBURN, Alistair; HIGHSMITH, Jim **Agile software development: The human factor**. IEEE Computer, nov. 2001, p. 131-133.
- [COCKBURN, 2001] COCKBURN, Alistair **Writing Effective Use Cases**. Addison-Wesley, The Crystal Collection For Software Professional, 2001.
- [COCKBURN, 2002] COCKBURN, Alistair **Agile software development**. Addison-Wesley, The agile software development series, 2002.
- [COPLIEN, 1999] COPLIEN, James O. **Reevaluating the architecture metaphor: Toward piecemeal growth**. IEEE Software, set./out. 1999, p. 40-44.
- [DEMARCO, 1995] DEMARCO, Tom **Why does software cost so much?** Dorset House, 1995.
- [DEMARCO, BOEHM, 2002] DEMARCO, Tom; BOEHM, Barry **The agile Methods fray**. IEEE Computer, jun. 2002, p. 90-92.
- [EHN, 1992] EHN, Pelle **Scandinavian Design: On Participation and Skill**.

- em Usability: Turning technologies into tools, Oxford University Press, 1992, p. 96-132.
- Disponível em:
<http://www.ilt.columbia.edu/Publications/papers/Ehn.html>
 Acessado em: 21 abr. 2005.
- [FLOWERS, 1997] FLOWERS, Stephen **Software failure: Management failure: Amazing stories and cautionary tales**. John Wiley & Son Ltd, 1996.
- [FOWLER, 2000a] FOWLER, Martin **Refactoring: Improving the design of existing code**. The Addison-Wesley object Technology series, 2000.
- [FOWLER, 2000b] FOWLER, Martin **The new Methodology**.
 Disponível em
<http://martinfowler.com/articles/newMethodology.html>.
 Acessado em 17 abr. 2003.
- [FOWLER, 2003] FOWLER, Martin **Patterns of Enterprise Application Architecture**. The Addison-Wesley Signature Series, 2003.
- [GAMMA et al, 1995] GAMMA, Erich; RICHARD, Helm; JOHNSON, Ralph; VLISSIDES, John **Design Patterns – Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.
- [GILLETT et al, 2002] GILLETT, Frank E.; Rutstain, Charles; Schreck, Galen; Buss, Christian; Liddell, Heather; **Organic IT**. Forrester Research WholeView TechStrategy Research, apr. 2002.
 Disponível em:
<http://www.forrester.com/ER/Research/Report/Summary/0,1338,14136,00.html>
 Acessado em: 20 mar. 2003.
- [GINIGE & MURUGESAN, 2001a] GINIGE, A.; MURUGESAN, S. **Web engineering: An introduction**. IEEE Multimedia. jan./mar. 2001. p. 14-18.
- [GINIGE & MURUGESAN, 2001b] GINIGE, A.; MURUGESAN, S. **The essence of web engineering**. IEEE Multimedia. apr./jun. 2001. p. 22-25.

2001b]

- [GLASS, 2001] GLASS, Robert L. **Agile versus traditional: Mark love, not war!** Cutter IT Journal, vol. 14, no. 12., dec. 2001. p. 12-18.
- [GRAHAM, 2003] GRAHAM, Ian A **Pattern Language for Web Usability**. Addison Wesley, 2003.
- [GROOT, 1999] GROOT, Cees de **Towards organic software**. nov. 2001.
Disponível em: <http://www.cdegroot.com/articles/1999-organic-software/>
Acessado em: 21 mar. 2003.
- [HUHNS & SINGH, 2005] HUHNS, Michael; SINGH, Munindar P. **Service-Oriented Computing: Key concepts and Principles**. IEEE Internet Computing, jan./fev. 2005. p. 75-81.
- [HUNT & THOMAS, 2000] HUNT, Andrew; THOMAS, David **The pragmatic programmer**. Addison Wesley Longman, Inc., 2000.
- [HIGHSMITH, 2000] HIGHSMITH, James A. **Adaptive Software development: a collaborative approach to managing complex systems**. Dorset House, 2000.
- [HIGHSMITH, 2001] HIGHSMITH, James A. **History: The agile manifesto**.
Disponível em: <http://www.agilemanifesto.org/history.html>.
Acessado em 22 out. 2002.
- [HIGHSMITH, 2002] HIGHSMITH, James A. **Agile software development ecosystems**. Addison-Wesley, The agile software development series, 2002.
- [HIGHSMITH & COCKBURN, 2001] HIGHSMITH, Jim; COCKBURN, Alistair **Agile software development: The business of innovation**. IEEE Computer, sep. 2001, p. 120-122.
- [JEFFRIES *et al*, 2001] JEFFRIES, Ron; ANDERSON, Ann; HENDRICKSON, Chet **Extreme Programming Installed**. Addison-Wesley, The XP series, 2001.
- [JEFFRIES, 2003] JEFFRIES, Ron **website xprogramming**.
Disponível em: <http://www.xprogramming.com/>.
Acessado em 02 abr. 2003.

- [KIRTLAND, 1998] KIRTLAND, Mary. **Designing Component-based Applications**. Microsoft Press, 1998.
- [LARMAN & BASILI, 2003] LARMAN, Craig; BASILI, Victor R. **Iterative and incremental development: a brief history**. IEEE Computer, jun. 2003, p. 47-56.
- [LYCETT *et al*, 2003] LYCETT, Mark; Macredie, Robert D.; Patel, Chaitali; Paul, Ray J. **Migrating agile methods to standardized development practices**. IEEE Computer, jun. 2003, p. 79-85.
- [MANIFESTO, 2001] **MANIFESTO for agile software development**. feb. 2001, Disponível em: <http://www.agilemanifesto.org/>. Acessado em 5 mar. 2003.
- [MARINESCU, 2002] MARINESCU, Floyd. **EJB Design Patterns**. John Wiley & Sons Inc., 2002
- [MILLER, 1956] MILLER, G. **The magical number seven, plus or minus two**. Psychology Review, 1956.
- [MCLUHAN & POWERS, 1992] MCLUHAN, Marshall; POWERS, Bruce R. **The global village: Transformations in world life and media in the 21st Century**. Oxford University Press, Reprint Edition, 1992.
- [NAUR & RANDELL, 1969] NAUR, Peter; RANDELL, Brian **Report on a conference sponsored by NATO Science Committee**. Germany, 1969. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/>. Acessado em 15 jun. 2003
- [NAUR, 1985] NAUR, Peter **Programming as Theory Building**. Microprocessing and Microprogramming, Vol. 15, 1985, p. 253-261.
- [NILSSON, 2002] NILSSON, Jimmy. **.NET Enterprise Design with Visual Basic.NET and SQL Server 2000**. Sams Publishing, 2002.
- [NORMAN, 1999] NORMAN, David A. **The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution**. MIT Press reprint edition, 1999.

- [ORBORNE, 1979] OSBORNE, Adam **Running wild: The next industrial revolution**. McGraw-Hill Osborne Media , 1979.
- [PALMER & FELSING, 2002] PALMER, Stephen; FELSING, John A **A practical guide to Feature Driven Development**. Prentice-Hall Inc, 2002.
- [PAULK, 2001a] PAULK, Mark C. **Extreme programming from a CMM perspective**. Paper for XP Universe, Raleigh, Jul., 2001.
- [PAULK, 2001b] PAULK, Mark C. **Extreme programming from a CMM perspective**. IEEE Software, nov./dec. 2001, p. 19-26.
- [PERROCHON & MANN, 1999] PERROCHON, Louis; MANN, Walter **Inferred Design**. IEEE Software, sep./oct. 1999, p. 46-51.
- [PRESSMAN, 1987] PRESSMAN, Roger S. **Software engineering: a practioner's approach**. 2nd ed. McGraw-Hill series in software engineering and technology, 1987.
- [PRESSMAN, 2001] PRESSMAN, Roger S. **Software engineering: a practioner's approach**. 5th ed. McGraw-Hill series in computer science, 2001.
- [REIFER, 2002] REIFER, Donald J. **How good are agile methods?** IEEE Software, jul./aug. 2002, p. 16-18.
- [REIFER, 2003] REIFER, Donald J. **XP and the CMM**. IEEE Software, may./jun. 2003, p. 14-15.
- [RISING & JANOFF, 2000] RISING, Linda; JANOFF, Norman S. **The Scrum software development process for small teams**. IEEE Software, jul./aug. 2000, p. 26-32.
- [RAYMOND, 1997] RAYMOND, Eric S. **The cathedral and the bazaar**. may 1997. Disponível em: <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>. Acessado em 02 abr. 2003.
- [SCHWABER & BEEDLE, 2002] SCHWABER, Ken; BEEDLE, Mike **Agile software development with Scrum**. Prentice-Hall, Inc., 2002.
- [SHINE, 2002] SHINE Technologies **Agile methodologies survey results**. 2002. Disponível em: <http://www.shine.com/> Acessado em: 07 jun. 2003.

- [STARK & CROKER, 2003] STARK, John A.; CROKER, Ron **Trends in doftware process: the PSP and agile methods.** IEEE Software, may./jun. 2003, p. 89-91.
- [TOFLLER, 2000] TOFFLER, Alvin **A terceira onda.** Record, 2000.
- [TRUEX et al, 2000] TRUEX, Duane; Baskerville, Richard; Travis, Julie
Amethodical systems development: the deferred meaning of systems development methods. Accounting, Management and Information Technology 10, 2000, p. 53-79.
- [WILLIAMS & COCKBURN, 2003] WILLIAMS, Laurie; COCKBURN. Alistair **Agile software development: It's about feedback and change.** IEEE Computer, Jun. 2003, p. 39-43.
- [WYSOCKI et al, 2000] WYSOCKI, Robert K.; Beck, Robert, Jr.; Crane, David B.
Effective project management. Willey & son, Inc., 2000.
- [YOURDON, 1997] YOURDON, Edward **Death March.** Prentice Hall PTR, 1997.
- [YOURDON, 1998] YOURDON, Edward; J. Yourdon **Time Bomb 2000.** Prentice Hall PTR, 1998.