Universidade Federal do Rio de Janeiro
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
Programa de Engenharia de Sistemas e Computação

Exame de Qualificação

APOIO À DECISÃO PARA O BALANCEAMENTO DE COLABORAÇÃO E DISCIPLINA NOS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

Andréa Magalhães Magdaleno

Orientadora: Cláudia Maria Lima Werner

Co-orientadora: Renata Mendes de Araujo

Rio de Janeiro, RJ – BRASIL

Julho de 2010

SUMÁRIO

1. INTRODUÇÃO 1

1.1. Motivação 1

1.2. Caracterização do Problema 3

1.3. Enfoque de Solução 5

1.4. Visão Geral da Proposta de Tese 8

1.5. Contexto da Proposta de Tese 11

1.6. Organização do Trabalho 15

2. PROCESSO DE DESENVOLVIMENTO DE SOFTWARE 17

2.1. Modelos de desenvolvimento de software 17

2.1.1. Modelo de desenvolvimento de software orientado ao planejamento 17

2.1.2. Modelo de desenvolvimento de software ágil 18

2.1.3. Modelo de desenvolvimento de software livre 20

2.2. Conciliação de modelos de desenvolvimento de software 21

2.2.1. Revisão quasi-sistemática 25

2.2.1.1. Planejamento da revisão quasi-sistemática 25

2.2.1.2. Execução da revisão quasi-sistemática 27

2.2.1.3. Resultados da revisão quasi-sistemática 29

2.2.1.4. Caracterização da conciliação de processos de desenvolvimento de software 31

2.3. Adaptação de processos de desenvolvimento de software 35

2.4. Considerações finais 40

3. COLABORAÇÃO E DISCIPLINA 41

3.1. Colaboração 41

3.2. Disciplina 42

3.3. Aspectos de colaboração e disciplina no desenvolvimento de software. 43

3.4. Colaboração e disciplina 47

3.5. Instrumentos para Colaboração e Disciplina 48

3.5.1. Modelo de Maturidade em Colaboração (CollabMM) 49

3.5.1.1. Aspectos de Apoio à Colaboração 49

3.5.1.2. Níveis de Maturidade em Colaboração 53

3.5.2. Medição 56

3.6. Considerações finais 59

4. REDES SOCIAIS 61

4.1. Propriedades topológicas 62
4.2. Modelos de evolução de redes 64
4.3. Metodologia para estudo de redes sociais 66
4.4. Visualização de redes sociais 67
4.5. Análise de redes sociais 69
4.5.1. Propriedades de análise de redes sociais 70
4.6. Características de colaboração em redes sociais de desenvolvimento de software 71
4.7. Requisitos para ferramentas de redes sociais 73
4.8. Ferramental de Apoio: EvolTrack-SocialNetwork 80
4.8.1. Ferramental de apoio: EvolTrack 80
4.8.2. Arquitetura do EvolTrack-SocialNetwork 83
4.9. Considerações finais 85

5. LINHA DE PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE 87

5.1. Engenharia de Domínio e Linha de Produtos de Software 88
5.1.1. Modelagem de características 90
5.2. Linha de processos 93
5.3. Propostas para linha de processos 96
5.4. Abordagem de engenharia de linha de processos 98
5.4.1. Meta-Modelo 99
5.4.2. Notação 101
5.4.3. Método 101
5.4.3.1. Estratégias de captura de conhecimento 102
5.4.3.2. Engenharia de Domínio de Processo (EDP) 103
5.4.3.3. Engenharia de Aplicação do Processo (EAP) 108
5.4.4. Ferramental de apoio 110
5.5. Considerações finais 111

6. GESTÃO DE CONTEXTO 112

6.1. Contexto 112
6.2. Gestão de contexto 113
6.3. Abordagem de gestão de contexto 115
6.4. Representação de contexto em processos de desenvolvimento de software 116
6.4.1. Abordagem UbiFEX para representação de contexto 118
6.5. Dimensões de contexto de desenvolvimento de software 119

6.6. Informações de contexto de desenvolvimento de software 121
6.6.1. Dimensão organização 122
6.6.2. Dimensão projeto 123
6.6.3. Dimensão equipe 125
6.7. Situações de contexto de desenvolvimento de software 129
6.8. Regras de contexto de desenvolvimento de software 131
6.9. Linha de processos gerida por contexto 132
6.10. Cenários de uso das informações de contexto 133
6.11. Ferramental de apoio 135
6.12. Considerações finais 138
7. BALANCEAMENTO DE COLABORAÇÃO E DISCIPLINA 140
7.1. Apoio à Decisão 140
7.2. Problemas de Otimização 143
7.3. Modelagem do Problema de Balanceamento na Adaptação do Processo do Projeto 145
7.3.1. Definição do Problema 146
7.3.2. Modelagem do Problema 147
7.3.2.1. Seleção das características do processo 148
7.3.2.2. Combinação de características 151
7.3.2.3. Seleção dos componentes de processos 152
7.3.2.4. Combinação dos componentes de processo 153
7.3.2.5. Sequenciamento dos componentes de processo 155
7.3.3. Funções de custo 157
7.3.4. Análise do espaço de busca 158
7.4. Considerações finais 159
8. CONCLUSÃO 161
8.1. Principais contribuições 161
8.2. Metodologia de pesquisa 163
8.2.1. Atividades realizadas ou em andamento 163
8.2.2. Atividades previstas 166
8.3. Resultados preliminares 169
8.4. Resultados esperados 170
8.5. Cronograma proposto 171
REFERÊNCIAS BIBLIOGRÁFICAS 173

1. Introdução

Este capítulo descreve a motivação e caracteriza o problema estudado. Em busca de solução para o problema apresentado, é discutido o enfoque adotado neste trabalho. Em seguida, é descrita a proposta de tese e o contexto geral de pesquisa onde esta proposta se encaixa. Por fim, é resumida a forma como os capítulos deste documento foram organizados.

1.1. Motivação

As organizações de software são continuamente desafiadas pela necessidade de melhorar a qualidade dos produtos de software gerados e atender às necessidades de customização dos seus clientes. As tecnologias evoluem rapidamente e os sistemas estão se tornando cada vez maiores e mais complexos. Somam-se a isso, os novos desafios trazidos pelo desenvolvimento distribuído de software (HERBSLEB *et al*., 2005).

Com o aumento da exigência por parte dos clientes, as organizações de desenvolvimento de software vêm buscando cada vez mais oferecer serviços e produtos com qualidade. Pesquisas na área (CUGOLA e GHEZZI, 1998, FEI DAI e TONG LI, 2007, FUGGETTA, 2000, OSTERWEIL, 1987, PRESSMAN, 2001, RAMAN, 2000) apontam que a qualidade do produto de software depende do processo de desenvolvimento[1] adotado para construí-lo e que muitos dos problemas associados ao desenvolvimento de produtos de software podem ser resolvidos aperfeiçoando-se este processo.

A busca pela melhoria da qualidade levou as organizações a recorrerem a modelos de maturidade e normas de qualidade, tais como o CMMI-DEV (*Capability Maturity Model Integration*) (CHRISSIS *et al*., 2006), a ISO/IEC (*International Organization for Standardization*) 12207 (ISO/IEC, 1995) e o MPS.BR (Melhoria de Processos do Software Brasileiro) (SOFTEX, 2009), para apoiar a definição de processos de desenvolvimento de software menos caóticos, mais previsíveis e, consequentemente, mais gerenciáveis.

Relatos sobre o uso destes modelos de maturidade (GIBSON *et al.*, 2006, TRAVASSOS e KALINOWSKI, 2008) demonstram resultados alcançados

[1] Um processo de desenvolvimento de software é "um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software" (FUGGETTA, 2000).

relacionados ao aumento de produtividade, redução de custos, melhoria da qualidade, aumento da satisfação do cliente e uma maior capacidade de lidar com grandes projetos. Entretanto, esta melhoria costuma ser de longo prazo e requer grandes investimentos (GOLDENSON e GIBSON, 2003). Além disso, por ser um modelo de desenvolvimento de software fortemente baseado em planejamento e processos bem definidos, ele atinge bem os seus objetivos em ambientes relativamente estáveis, mas quando confrontado com projetos inovadores possui dificuldade em se adaptar rapidamente às mudanças que ocorrem ao longo do projeto, o que ameaça o cumprimento de prazos e custos previstos (BOEHM e TURNER, 2003).

Apesar de ter as suas origens há mais de 75 anos atrás (GLAZER *et al*., 2008), somente a partir da década de 90 algumas abordagens ágeis se destacaram. Os métodos ágeis começaram a ganhar espaço nas organizações devido à sua capacidade de responder e se adaptar às mudanças que surgem durante um projeto de desenvolvimento, através de ciclos de desenvolvimento mais curtos (COCKBURN, 2001). Norteados por uma filosofia de desenvolvimento baseada em um conjunto de valores e princípios, resumidos no Manifesto Ágil (BECK *et al*., 2001), os métodos ágeis se apresentam como um modelo alternativo de desenvolvimento de software, ao “valorizar indivíduos e interações entre eles mais do que processos e ferramentas”.

Também na década de 90, com o crescimento da Internet, o sucesso de alguns projetos de software livre, como Linux, Apache e Mozilla, chamou a atenção da academia, da indústria e dos usuários, por produzir software de alta qualidade, rápida e gratuitamente, sem processos formalmente definidos e altamente baseados em colaboração (CAPILUPPI *et al*., 2003). O software livre não traz em si algum tipo de ruptura tecnológica, mas traz uma nova alternativa viável de desenvolvimento de produtos de software, baseada em características próprias (RAYMOND, 2001, REIS, 2003).

Estes três modelos têm o mesmo objetivo: melhorar o desenvolvimento de software; mas adotam enfoques distintos. Todos tiveram, na última década, um enorme impacto e a sua perspectiva de evolução futura é igualmente promissora (EBERT, 2007, TAURION, 2004, THEUNISSEN *et al*., 2008). Como cada um representa um universo de desenvolvimento com características peculiares, a pesquisa na área tem discutido como conciliar as particularidades

de cada modelo para a definição de processos de desenvolvimento mais eficazes (BARNETT, 2004, BOEHM e TURNER, 2003, GLASS, 2001, GLAZER *et al*., 2008, MAGDALENO *et al*., 2009, PATEL *et al*., 2006, PAULK, 2001, TURK *et al*., 2002, TURNER e JAIN, 2002, WARSTA e ABRAHAMSSON, 2003).

1.2. Caracterização do Problema

Como o universo dos projetos de software é extenso e diversificado, um único modelo de desenvolvimento não consegue atender aos requisitos de todos os projetos ou organizações, pois eles têm objetivos, características e necessidades diferentes (FEI DAI e TONG LI, 2007, MACHADO, 2000). Para enfrentar este desafio, a comunidade de pesquisa tem sugerido abordagens que equilibrem as características de diferentes modelos de desenvolvimento (BOEHM, 2002).

As organizações também perceberam que o uso de um modelo único não satisfazia às variações do desenvolvimento. Tal fato pode ser observado pelo crescimento da adaptação dos processos de desenvolvimento de software por parte das organizações (HANSSON *et al*., 2006), enquanto decresce a quantidade de organizações que seguem um modelo de maturidade de modo totalmente prescritivo (PATEL *et al*., 2006).

A adaptação de processos é o "ato de particularizar um processo geral para derivar uma definição aplicável a um contexto mais específico" (GINSBERG e QUINN, 1995, PEDREIRA *et al*., 2007). Apesar da adaptação de processos ser uma das principais tarefas executadas pelo gerente do projeto, ela não é simples. Esta tarefa exige experiência, envolve o conhecimento de muitos aspectos da Engenharia de Software e requer a harmonização de muitos fatores do contexto da equipe, do projeto ou da organização (BARRETO, 2007).

Devido a esta complexidade e sem orientações claras, geralmente, o gerente de projeto não é capaz de avaliar todas as opções de adaptação do processo e acaba fazendo esta adaptação de forma ***ad-hoc***, baseado na sua experiência e conhecimento. Este conhecimento, geralmente, é **limitado a alguns modelos de desenvolvimento** apenas, pois não é de se esperar que o

gerente esteja familiarizado com cada um deles, dada a sua diversidade. Como resultado, o processo adaptado pode não ser a melhor alternativa para o projeto em questão.

Além disso, as decisões de adaptação do processo costumam ser tomadas pelo gerente no **início do projeto** e registradas no plano de projeto. No entanto, reconhece-se que é impossível prever neste momento um processo totalmente adequado, pois o contexto do projeto muda ao longo da sua execução.

Como o orçamento do projeto, o prazo de desenvolvimento e a qualidade do produto dependem diretamente da qualidade do processo de software, a adaptação de processos precisa ser feita corretamente para evitar que as seguintes possíveis consequências negativas se materializem: um processo de software ruim pode envolver atividades desnecessárias que levam a perda de tempo e dinheiro; e a omissão de atividades necessárias pode prejudicar a qualidade do produto (PEDREIRA *et al*., 2007).

Para enfrentar estes desafios, soluções para a adaptação de processos têm sido exploradas por diversas áreas de pesquisa. A análise das principais propostas existentes permitiu observar que:

(i) Não existe uma preocupação em balancear as características dos diferentes modelos de desenvolvimento;

(ii) As propostas não oferecem uma sistemática para apoiar a adaptação e se limitam a oferecer apenas orientações gerais que não são suficientes para ajudar efetivamente nesta tarefa (ARMBRUST *et al*., 2008, PEDREIRA *et al*., 2007, PENG XU e RAMESH, 2008);

(iii) A adaptação de processos pode ser feita partindo do processo padrão da organização ou através de estruturas de reutilização de processos, como os componentes de processos (CHRISSIS *et al*., 2006);

(iv) Na adaptação do processo devem ser considerados os contextos da equipe, do projeto e da organização, ainda que o entendimento sobre como eles afetam a adaptação de processos seja limitado (PENG XU e RAMESH, 2008) e falte suporte eficiente (JAUFMAN e MUNCH, 2005);

(v) É possível pensar em automatizar pelo menos alguns dos passos da atividade de adaptação de processos (PARK *et al*., 2006).

Com base nestas conclusões, este trabalho propõe que uma abordagem de adaptação de processos deve possuir as seguintes características:

(i) **Balanceamento** entre os aspectos de colaboração e disciplina dos modelos de desenvolvimento de software;

(ii) **Sistemática** para realizar a adaptação de processos baseada em uma estrutura de reutilização de processos;

(iii) **Dinamicidade** para evoluir ao longo do tempo e se adequar ao contexto da organização, do projeto e da equipe.

Assim, chegamos ao problema de pesquisa desta proposta de tese:

Como adaptar processos de desenvolvimento de software
de forma balanceada, sistemática e dinâmica?

1.3. Enfoque de Solução

Com o propósito de facilitar a adaptação de processos, é possível fornecer apoio ao gerente de projeto na execução desta atividade, uma vez que alguns dos passos para a solução do problema podem ser automatizados (PARK *et al.*, 2006), possivelmente diminuindo o esforço necessário para sua execução e melhorando os resultados obtidos.

Este apoio deve fornecer ao gerente um embasamento para a tomada de decisão sobre a melhor forma de adaptar o processo específico para o projeto, uma vez que possibilita que seja considerado um universo maior de fatores e alternativas. O apoio sugerido diz respeito a propor ao gerente de projeto um processo que satisfaça as restrições e que otimize alguns dos fatores envolvidos no problema. Porém, a decisão final sobre o processo que será efetivamente adotado continua sendo dele.

A partir desta ideia, e visando resolver o problema de pesquisa enunciado na seção anterior, a hipótese geral investigada neste trabalho é:

Se a tomada de decisão
sobre a adaptação de processos de desenvolvimento de software
for baseada em uma abordagem sistemática de reutilização de processos
e na gestão das informações de contexto de colaboração e disciplina,
então será possível adaptar dinamicamente os processos
ao contexto do projeto ao longo da sua execução.

Neste cenário, parte do enfoque de solução deste trabalho é que dois fatores importantes de serem considerados nesta adaptação de processos de desenvolvimento de software são a **colaboração** e a **disciplina**. Ainda que a adaptação de processos possa ser feita sob diferentes aspectos (SHARP e MCDERMOTT, 2001), a colaboração e a disciplina são os aspectos que se destacam quando se observa as características dos diferentes modelos de desenvolvimento de software. Estes dois aspectos estão presentes nos três modelos de desenvolvimento, ainda que com variações de ênfase e forma.

A colaboração representa o trabalho em conjunto de uma ou mais pessoas para a realização de objetivos comuns (Adaptado de FERREIRA, 2009, MINICUCCI, 2001). A colaboração é um fator importante para que as organizações de software possam alcançar seus objetivos de produtividade, qualidade e compartilhamento de conhecimento (MAGDALENO *et al*., 2007). Em particular, o desenvolvimento de software é um processo complexo que envolve a colaboração de diversas pessoas durante um período de tempo para atingir um objetivo comum (CUGOLA e GHEZZI, 1998). Portanto, o desenvolvimento de software é um típico exemplo de trabalho colaborativo (CATALDO e HERBSLEB, 2008, DEMARCO e LISTER, 1999).

Por outro lado, a disciplina está relacionada ao nível de planejamento adotado na definição do processo de desenvolvimento de software e ao nível de controle empregue na execução deste processo. Desta forma, a disciplina impõe ordem, sistematizando o trabalho, evitando o caos e os sucessos dependentes dos talentos humanos individuais (BOEHM e TURNER, 2003).

A colaboração e a disciplina são complementares e essenciais para qualquer projeto de desenvolvimento de software, mas em diferentes proporções, dependendo do contexto da organização, do projeto e da equipe. A partir desta ideia, a primeira sub-hipótese (H1.1) deste trabalho é:

É possível balancear os aspectos de colaboração e disciplina na tomada de decisão sobre a adaptação de processos de desenvolvimento de software.

A complexidade da adaptação de processos levou à necessidade de construir uma **sistemática** para guiá-la. Neste trabalho, esta sistemática é estabelecida valendo-se de uma estrutura de **reutilização de processos**. A ideia é que a mesma sistematização alcançada pela reutilização em Linhas de

Produtos de Software[2] possa ser aplicada também a processos (MONTERO *et al*., 2007).

Transferindo o conceito de linha de produtos para o contexto de processos, surge então a ideia de **linha de processos** (BARRETO, 2007, JAUFMAN e MUNCH, 2005, NUNES *et al*., 2010b, ROMBACH, 2006, WASHIZAKI, 2006), que são linhas de produtos cujos produtos são processos.

Como os processos de software – apesar de também serem variáveis entre os projetos de software – ainda não são gerenciados da mesma forma sistemática que os produtos de software (ROMBACH, 2006), a estrutura de reutilização adotada é baseada em linha de processos. Desta forma, os processos de software das organizações também podem ser organizados de acordo com as suas similaridades e diferenças, facilitando e guiando a sua adaptação às necessidades específicas dos projetos.

Desta forma, a segunda sub-hipótese (H1.2) formulada para esta pesquisa é:

É possível definir uma abordagem sistemática para apoiar a tomada de decisão sobre a adaptação de processos de desenvolvimento de software baseada na reutilização de processos.

Uma das premissas para o uso de linhas de processos está associada ao fato de que quando se entende o processo implementado na organização, é possível fazer mudanças durante a execução do processo. Conforme visto na seção anterior, não basta que a adaptação de processos seja feita uma única vez no início do projeto, pois os requisitos e o ambiente do projeto mudam. Portanto, a adaptação de processos deve ser executada de forma contínua, considerando o contexto de execução do processo e mantendo o alinhamento do processo com as necessidades atuais do projeto (BRINKKEMPER, 1996, FITZGERALD *et al*., 2003, MAGDALENO, 2010a, PENG XU e RAMESH, 2008).

Neste sentido, a adaptação dinâmica de processos está associada ao uso de informações de contexto[3] para identificar modificações no processo em

[2] Uma linha de produtos de software é um "conjunto de produtos de software que compartilham um conjunto de características comuns e controladas, que satisfazem necessidades de um segmento de mercado em particular, e são desenvolvidos a partir de artefatos, de forma predefinida" (NORTHROP, 2002).

[3] Contexto é uma descrição complexa do conhecimento compartilhado sobre circunstâncias físicas, sociais, históricas e outras dentro das quais ações ou eventos ocorrem (BREZILLON, 1999).

tempo de execução, visando se adaptar a novas situações. Assim, é papel da gestão de contexto perceber mudanças no contexto da organização, do projeto e da equipe ao longo da execução do projeto para permitir que a tomada de decisão sobre as sugestões de adaptação do processo possa ser revista e continuamente atualizada, de acordo com o novo contexto identificado.

Por fim, a terceira sub-hipótese (H1.3) que será investigada nesta proposta é:

É possível apoiar a tomada de decisão sobre a adaptação de processos de desenvolvimento de software ao longo da execução do projeto através da gestão de contexto.

1.4. Visão Geral da Proposta de Tese

O objetivo principal desta proposta de tese é apoiar o gerente de projeto na tomada de decisão sobre a melhor forma de adaptar o processo específico para o projeto. Para enfrentar este desafio e construir uma solução com todas as características descritas na seção anterior, este trabalho se diferencia ao combinar diferentes técnicas e tecnologias. Apesar de existirem propostas que, de alguma forma, tratam cada um destes elementos isoladamente, não foi encontrada na literatura uma abordagem que trate todas essas questões de uma forma integrada.

A visão geral da proposta é apresentada na Figura 1.1, onde pode ser observado que esta proposta de tese é composta por quatro partes principais, brevemente descritas a seguir.

Tendo em vista o objetivo desta pesquisa e a complexidade de sua resolução, observa-se a necessidade de um ambiente automatizado de apoio à decisão sobre a adaptação de processos de desenvolvimento de software (Figura 1.1a). Este ambiente, que será usado pelo gerente de projeto, deve ser capaz de sugerir um processo específico para o projeto de acordo com o contexto da organização, do projeto, da equipe e de outros contextos similares. Esta adaptação do processo também deve levar em consideração o balanceamento necessário entre colaboração e disciplina.

O ambiente de apoio à decisão será construído utilizando como base métodos de otimização que têm como finalidade buscar soluções que atendam

um conjunto de restrições impostas, para maximizar ou minimizar um determinado fator (PAPADIMITRIOU e STEIGLITZ, 1998). O problema de otimização que se quer resolver neste trabalho é: como adaptar o processo específico do projeto a partir da linha de processos, usando as informações de contexto de forma a balancear a colaboração e disciplina? (MAGDALENO, 2010b).

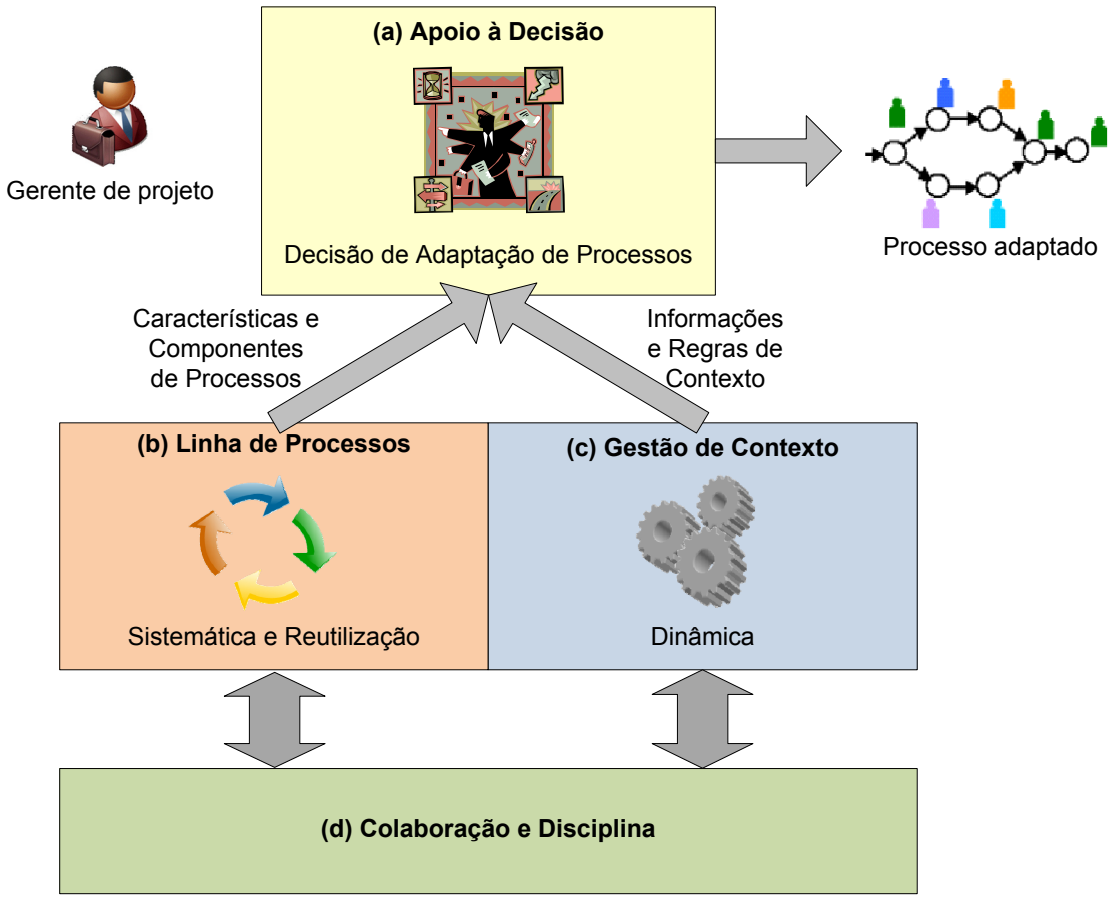


Figura 1.1 – Visão geral da proposta de tese

Apesar do foco deste trabalho estar relacionado ao apoio à decisão, é possível perceber no problema enunciado acima que também são necessários outros elementos para compor a solução proposta.

A estrutura de reutilização de processos adotada é baseada no uso de **linha de processos** (Figura 1.1b). Esta abordagem é composta por duas fases principais: uma para a criação da linha de processos e outra para a instanciação de processos a partir da linha. Na fase de criação da linha são construídos os artefatos, tais como o modelo de características e o modelo de componentes de processo, levando-se em consideração características comuns e variáveis existentes em um domínio específico. Na fase de

instanciação, os processos são derivados aplicando-se regras de composição e recorte aos artefatos criados anteriormente. Assim, o problema de otimização que se pretende resolver está concentrado na segunda fase quando ocorre a adaptação do processo a partir da linha de processos.

Neste trabalho, a estrutura de reutilização de processos foi combinada com uma abordagem de **gestão de contexto** (Figura 1.1c) capaz de: estabelecer relações entre uma determinada situação de contexto e as características e os componentes da linha de processos; compreender o contexto do projeto atual através de um conjunto de informações de contexto; identificar mudanças no contexto da organização, do projeto e da equipe; e gerir informações de contexto de colaboração e disciplina.

Assim, o ambiente de apoio à decisão recebe insumos destas duas abordagens e combina todas estas informações em busca de uma solução para adaptação de processos que balanceie os aspectos de colaboração e disciplina.

Para que os aspectos de colaboração e disciplina possam ser introduzidos no processo de desenvolvimento de software adaptado, é preciso definir como planejar e acompanhar os níveis de colaboração e disciplina (Figura 1.1d) necessários ou desejados. Neste sentido, estão sendo considerados alguns instrumentos, tais como: o modelo de maturidade em colaboração (CollabMM); as redes sociais; e a medição.

O CollabMM (MAGDALENO *et al*., 2007, 2009, 2008a, 2008b) funciona como um *framework* que define os níveis de colaboração e suas principais características. Além disso, também é necessário um mecanismo que ajude a explicitar e medir a colaboração existente entre as pessoas nos projetos de desenvolvimento de software. As **redes sociais**[4] (BARABASI, 2003) foram escolhidas como este instrumento devido ao seu potencial em explicitar como a colaboração acontece dentro de um grupo.

No caso da **disciplina**, é adotada uma abordagem baseada em **medição**[5] (BASILI *et al*., 1994) como meio para apresentar informações

[4] Uma rede social consiste em um "conjunto finito de atores e as relações definidas entre eles" (WASSERMAN e FAUST, 1994).

[5] A medição é o "processo através do qual valores (símbolos ou números) são atribuídos a atributos de entidades do mundo real, de modo a descrevê-los de acordo com regras claramente definidas" (ROBERTS, 2009).

relevantes ao gerente de projeto, contribuindo para a compreensão e controle da situação atual do processo, do projeto e do produto.

1.5. Contexto da Proposta de Tese

A presente proposta de tese está inserida em um contexto de pesquisa mais amplo dentro do Grupo de Reutilização de Software do Programa de Engenharia de Sistemas e Computação da COPPE-UFRJ e do Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec) da UNIRIO. Devido à dimensão do trabalho envolvido e ao conhecimento da impossibilidade de todo o trabalho ser conduzido por um único indivíduo, este tema de pesquisa está sendo tratado por um grupo composto por dois alunos de doutorado, sendo um deles a autora desta proposta de tese, e alguns alunos de mestrado e graduação, alocados em trabalhos mais específicos. Os interesses individuais de pesquisa, as contribuições de cada trabalho e as interfaces com a presente proposta são explicados a seguir:

(a) Colaboração e disciplina

Como instrumentos para planejar e acompanhar o nível de colaboração no processo adaptado, foram adotados o modelo de maturidade em colaboração (CollabMM) (MAGDALENO *et al*., 2009) e as redes sociais (BARABASI, 2003) (Figura 1.2).

A proposta de organização do CollabMM em níveis de colaboração já foi validada, inclusive em contextos reais (MAGDALENO *et al*., 2008a). Entretanto, ainda era preciso relacioná-lo às redes sociais. Um primeiro passo foi dado nessa direção com o trabalho de Santos (2010) que, a partir da interpretação e composição de propriedades de análise de redes sociais (Figura 1.2 – b.3), sugere um conjunto de características de colaboração, especificamente de coordenação, em processos de desenvolvimento de software, organizadas tendo como inspiração os níveis de maturidade em colaboração do CollabMM (MAGDALENO *et al*., 2009).

Para estudar as redes sociais é preciso minerá-las, visualizá-las e analisá-las (CROSS *et al*., 2004). O presente trabalho irá se concentrar na fase de análise (Figura 1.2-b.3), pois esta possui o maior potencial para apoiar a compreensão da colaboração. Entretanto, no contexto do grupo de pesquisa,

outros trabalhos relacionados a este tema foram ou estão sendo conduzidos. Para oferecer mecanismos de visualização de redes sociais (Figura 1.2 – b.2), foi definido um trabalho de iniciação científica que diz respeito à extensão da ferramenta EvolTrack (CEPEDA *et al.*, 2008), originalmente construída para a visualização da evolução de software, criando o EvolTrack-SocialNetwork.

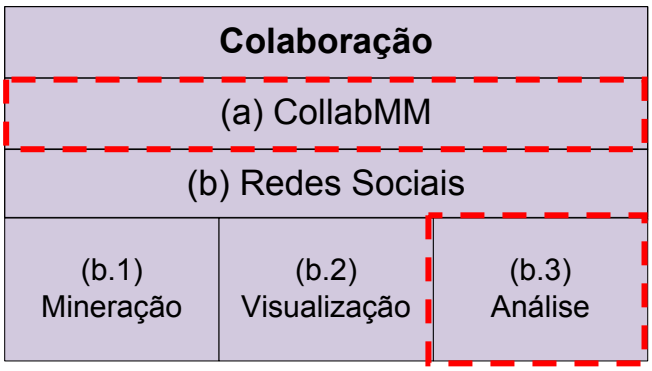


Figura 1.2 – Instrumentos para colaboração

Como instrumentos para planejar e acompanhar o nível de disciplina no processo adaptado, nesta proposta pretende-se adotar técnicas de medição (MCGARRY *et al.*, 2001, PUTNAM e MYERS, 2003, SOLINGEN e BERGHOUT, 1999). Estas técnicas poderão ser exploradas em contextos diferentes, utilizando projetos de desenvolvimento de software ágeis e livre. Porém, no momento, não existem outros trabalhos do grupo de pesquisa relacionados a este tema.

(b) Linha de Processos

Existem dois outros trabalhos, um de mestrado e outro de doutorado (NUNES *et al.*, 2010b), relacionados a este tema. Em conjunto, este grupo construiu uma definição para linha de processos de forma a garantir a integração dos resultados das pesquisas individuais. Para embasar e formalizar esta definição, bem como expandir o conhecimento sobre o uso e gestão de linhas de processos, uma revisão *quasi*-sistemática da literatura foi planejada e sua execução está em andamento, também como uma iniciativa em comum deste grupo de pesquisa.

Por fim, a abordagem sistemática de reutilização baseada em linha de processos também foi concebida como um trabalho colaborativo deste grupo. Em linhas gerais, esta abordagem prevê a modelagem da linha de processos e a adaptação dos processos específicos para os projetos a partir da linha de

processos (NUNES *et al.*, 2010b). Ao final desta abordagem, os processos estão prontos para serem executados (Figura 1.3).

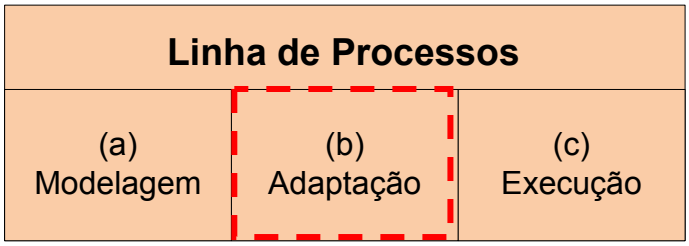


Figura 1.3 – Fases da Linha de processos

Esta abordagem sistemática está sendo utilizada nesta proposta de tese como base para o apoio à decisão na adaptação de processos. Ainda que todas as fases sejam necessárias e importantes para a solução deste trabalho, não será possível tratar todas em profundidade nesta proposta de tese. Assim, este trabalho concentrará esforços: (i) na fase de adaptação do processo (Figura 1.3-b), quando as características e componentes de processos da linha de processos são recortados e adaptados ao contexto do projeto com a ajuda do ambiente de apoio à decisão; (ii) na integração da abordagem de engenharia de linha de processos com a gestão de contexto para que as informações de colaboração e disciplina também possam ser geridas.

Em relação à fase de modelagem (Figura 1.3-a), a definição de como os conceitos e artefatos da linha de processos, principalmente da fase de análise do domínio do processo, podem ser utilizados, adaptados ou implementados no ferramental de apoio já existente, é objeto de uma dissertação de mestrado. No escopo do trabalho de doutorado de Nunes (NUNES *et al.*, 2010b), está incluída a integração com a gestão de contexto na fase de execução (Figura 1.3-c).

(c) Gestão de Contexto

A gestão de contexto é composta por seis elementos: representação, definição de informações, captura, recuperação, inferência e visualização (Figura 1.4). Apesar da sua importância, nem todos estes elementos serão tratados no escopo desta proposta de tese, que irá se limitar a atuar: (i) no estudo de mecanismos de representação de informações de contexto para o domínio de processos de desenvolvimento de software (Figura 1.4-a) (NUNES *et al.*, 2010a); (ii) na definição das dimensões, informações, situações e regras de contexto para o domínio de processos de desenvolvimento de software (Figura

1.4-b); (iii) na integração com as demais partes da proposta de solução.

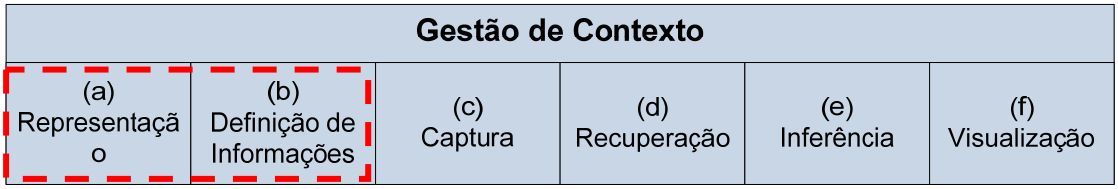


Figura 1.4 – Elementos da gestão de contexto

O detalhamento da abordagem de gestão de contexto completa faz parte da pesquisa de doutorado de Nunes (NUNES *et al*., 2010b). Em relação à definição das informações de contexto, apesar de um conjunto inicial já ter sido identificado nesta proposta de tese através de revisão bibliográfica, planeja-se realizar uma revisão sistemática da literatura para ajudar a identificar outros critérios comumente adotados na adaptação de processos de desenvolvimento de software. Esta revisão sistemática, bem como a validação destes critérios, com os especialistas, está no escopo de uma dissertação de mestrado.

(d) Apoio à Decisão

O apoio à decisão nesta proposta de tese diz respeito a ajudar o Gerente de projeto na adaptação do processo específico para o projeto, levando em consideração o balanceamento de colaboração e disciplina. Para isso, um ambiente automatizado de apoio à decisão poderá contribuir fornecendo informações que seriam difíceis de obter de outra forma e permitindo ao gerente testar diferentes soluções e medir o impacto de cada uma delas no projeto. Porém, a decisão final sobre o processo que será efetivamente adotado continua sendo dele.

O ambiente de apoio à decisão será construído utilizando como base métodos de otimização que têm como finalidade buscar soluções que atendam um conjunto de restrições impostas, para maximizar ou minimizar um determinado fator (PAPADIMITRIOU e STEIGLITZ, 1998).

Neste sentido, tanto a definição e modelagem formal (MAGDALENO, 2010b), quanto a implementação da solução do problema (Figura 1.5), fazem parte do escopo do presente trabalho. Por enquanto, não existem outros alunos do grupo de pesquisa trabalhando neste tema, ainda que se vislumbre a possibilidade de algum trabalho vir a contribuir na implementação da solução do problema (Figura 1.5-a.2).

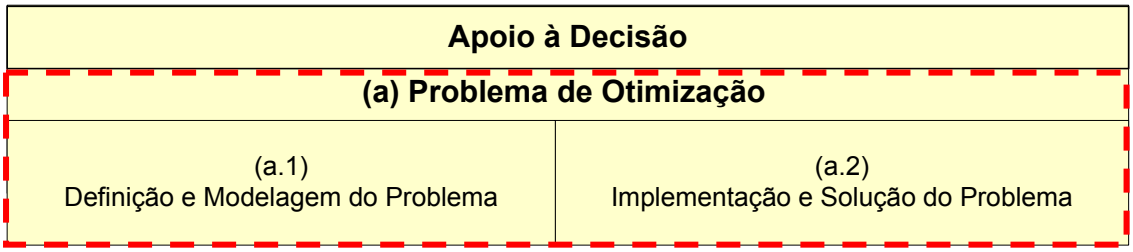


Figura 1.5 – Apoio à Decisão

1.6. Organização do Trabalho

Para apresentar este trabalho, o restante deste documento está organizado da seguinte forma: no **Capítulo 2** foi compilado um resumo das principais características dos três modelos de desenvolvimento de software. Além disso, este capítulo apresenta uma breve revisão bibliográfica sobre as principais propostas existentes para a conciliação dos modelos de desenvolvimento. Em seguida, são relatados o planejamento, a execução e os resultados obtidos com uma revisão *quasi*-sistemática da literatura sobre o tema. Esta revisão mostrou que o caminho para balancear estes modelos de desenvolvimento de software, não é através da conciliação, mas sim da adaptação de processos. Este capítulo também aborda as principais propostas existentes para a adaptação de **processos de desenvolvimento de software**.

Como o enfoque de solução deste trabalho envolve o balanceamento dos aspectos de **colaboração e disciplina** durante a adaptação de processos de desenvolvimento de software, no **Capítulo 3** estes dois aspectos são definidos e investigados dentro do conjunto de características dos modelos de desenvolvimento de software. Além disso, faz parte do escopo deste capítulo apresentar os instrumentos, tais como o CollabMM e a medição, que se pretende adotar para planejar e acompanhar os níveis de colaboração e disciplina no processo adaptado.

O **Capítulo 4** apresenta o conceito e os modelos de formação de **redes sociais**. Seguindo uma metodologia de estudo de redes sociais, são mostrados exemplos de instrumentos para a visualização de redes sociais e exploradas as propriedades de análise de redes sociais. A partir daí, são investigadas algumas ferramentas de redes sociais existentes para identificar os requisitos desejáveis a este tipo de ferramenta. Como nenhuma das ferramentas analisadas atende a todos os requisitos pretendidos, é proposta a criação da

ferramenta EvolTrack-SocialNetwork cuja arquitetura é detalhada ao final deste capítulo.

No **Capítulo 5** é definido o conceito de **linha de processos** junto com uma breve revisão da literatura sobre as principais propostas existentes. Em seguida, é descrita a abordagem sistemática de engenharia de linha de processos, incluindo o meta-modelo, a notação, o método e o ferramental de apoio sugerido.

Os elementos principais da abordagem de **gestão de contexto** são descritos no **Capítulo 6**, onde, seguindo uma abordagem de representação de contexto, também são detalhadas as dimensões, informações, situações e regras de contexto para o domínio de processos de desenvolvimento de software. Ainda neste capítulo é mostrada a integração da linha de processos com a gestão de contexto. Por fim, também é apresentado o ferramental de apoio sugerido para esta representação de contexto.

O **Capítulo 7** é dedicado ao **balanceamento entre colaboração e disciplina**, utilizando problemas de otimização para implementar a solução de apoio à decisão. As características deste tipo de problema, bem como os tipos de métodos existentes também são discutidos. A definição, modelagem e análise de complexidade do problema são detalhadas. Por fim, são feitas algumas considerações sobre as perspectivas futuras para a implementação da solução e sobre as possibilidades de avaliação da solução proposta.

Finalmente, o **Capítulo 8** contém as **considerações finais** sobre a presente proposta de tese, incluindo as principais contribuições, a metodologia aplicada ou prevista na continuidade desta pesquisa, os resultados preliminares obtidos, os resultados esperados e o cronograma de atividades previsto até a conclusão do trabalho.

2. Processo de Desenvolvimento de Software

Este capítulo resume as principais características dos modelos de desenvolvimento de software orientado ao planejamento, ágil e livre. Em seguida, são apresentadas algumas propostas existentes para a conciliação destes modelos de desenvolvimento e os resultados obtidos com a revisão quasi-*sistemática da literatura realizada. Como alternativa à conciliação através da combinação de práticas destes modelos de desenvolvimento, as propostas para adaptação de processos foram estudadas.*

2.1. Modelos de desenvolvimento de software

Um modelo de desenvolvimento de software é um conjunto de práticas recomendadas para o desenvolvimento de software (SOMMERVILLE, 2004). Estas práticas são organizadas em um processo de software que corresponde ao "conjunto de tarefas de Engenharia de Software necessárias para transformar os requisitos dos usuários em software" (HUMPHREY, 1989). Um processo de software foi definido por Fuggetta (2000) como: "um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software".

Os modelos de desenvolvimento de software desenvolvimento de software orientado ao planejamento, ágil e livre têm o mesmo objetivo: melhorar o desenvolvimento de software; mas adotam enfoques distintos. Todos tiveram, na última década, um enorme impacto e a sua perspectiva de evolução futura é igualmente promissora (EBERT, 2007, TAURION, 2004, THEUNISSEN *et al*., 2008). Como cada modelo representa um universo de desenvolvimento singular, foi necessário estudá-los para compreender a sua filosofia e principais características.

2.1.1. Modelo de desenvolvimento de software orientado ao planejamento

O modelo de desenvolvimento orientado ao planejamento (também conhecido na literatura como tradicional ou rigoroso) é tipicamente exemplificado pelos modelos de maturidade e normas de qualidade, tais como o CMMI-DEV (*Capability Maturity Model Integration*) (CHRISSIS *et al*., 2006), a ISO/IEC (*International Organization for Standardization*) 12207 (ISO/IEC, 1995) e o MPS.BR (Melhoria de Processos do Software Brasileiro) (SOFTEX, 2009).

Historicamente, estes modelos de qualidade foram concebidos para lidar com ambientes de alto risco e projetos grandes e complexos com altos custos envolvidos, onde os relacionamentos entre cliente e equipe de desenvolvimento se caracterizam pelo baixo nível de confiança e são governados por definições contratuais. A rigidez de um contrato torna mais difícil as adaptações às mudanças. Além disso, devido ao baixo nível de confiança, são necessárias aprovações frequentes e, ao longo de cada fase do ciclo de vida de desenvolvimento, existe uma preocupação com a completude da documentação e com a rastreabilidade entre a documentação gerada em cada uma das fases (GINSBERG e QUINN, 1995, GLAZER *et al*., 2008).

À luz destes modelos e normas, e visando alcançar os objetivos de previsibilidade, estabilidade e confiabilidade, este modelo de desenvolvimento se tornou fortemente orientado a planejamento e baseado em processos bem definidos e melhorados continuamente pela organização (BOEHM e TURNER, 2003). Estes processos estabelecem uma abordagem sistemática que guia o desenvolvimento do software desde a definição dos requisitos até a implantação do software.

De modo geral, este modelo atinge bem os seus objetivos em ambientes relativamente estáveis, mas quando confrontado com projetos inovadores ou exploratórios, a previsibilidade e a estabilidade degradam e o projeto incorre em gastos significativos para tentar manter a aderência ao processo e o planejamento atualizado (BOEHM e TURNER, 2003).

Este modelo de desenvolvimento utiliza uma estrutura de coordenação baseada em comando e controle e se baseia no registro do conhecimento explícito sobre o produto construído. A comunicação entre os membros do projeto também é formalizada através dessa documentação. O cliente desempenha um papel importante, mas principalmente durante as fases de especificação de requisitos e homologação (NERUR *et al*., 2005).

2.1.2. Modelo de desenvolvimento de software ágil

As características (ABRANTES e TRAVASSOS, 2007, QUMER e HENDERSON-SELLERS, 2008) e premissas do desenvolvimento ágil, observadas em métodos tais como XP (*Extreme Programming*) (BECK, 1999), Scrum (SCHWABER, 2004), FDD (*Feature Driven Development*) (PALMER e

FELSING, 2002), DSDM (*Dynamic Systems Development Methodology*) (STAPLETON e CONSTABLE, 1997) e *Crystal* (COCKBURN, 2004), podem ser resumidas pelos quatro valores fundamentais do *Manifesto for Agile Software Development* (BECK *et al*., 2001): **indivíduos e interações** entre eles mais que processos e ferramentas; **software em funcionamento** mais que documentação abrangente; **colaboração com o cliente** mais que negociação de contratos; **responder a mudanças** mais que seguir um plano.

De modo geral, os objetivos principais dos métodos ágeis são fornecer valor rapidamente e responder às mudanças de mercado, tecnologia e ambiente (COCKBURN, 2001, HIGHSMITH e COCKBURN, 2001). Graças a esta orientação à adaptação, ao invés de predição, o projeto é executado de forma incremental, com ciclos de desenvolvimento curtos (MILLER, 2001).

O desenvolvimento ágil é mais aplicável em ambientes turbulentos, que sofrem muitas mudanças e onde existe certo nível de incerteza técnica (BOEHM, 2002, HIGHSMITH e COCKBURN, 2001, LINDVALL *et al*., 2002) que requer a combinação de habilidades através do trabalho colaborativo dos membros da equipe. Ele também opera melhor em uma cultura organizacional centrada em pessoas e colaborativa (COCKBURN e HIGHSMITH, 2001).

Durante o projeto existe um grande envolvimento e participação do cliente no levantamento, priorização e validação dos requisitos. Os especialistas do negócio ficam disponíveis e chegam a fazer parte da equipe de desenvolvimento. Esta disponibilidade reduz o tempo de *feedback* da solução proposta e agiliza o entendimento da equipe sobre as necessidades dos usuários (ABRAHAMSSON *et al*., 2003, COCKBURN, 2001, HIGHSMITH e COCKBURN, 2001, LINDVALL *et al*., 2002, TURK *et al*., 2002).

Como resultado, o desenvolvimento ágil produz software com parcimônia na documentação (NERUR *et al*., 2005, PATEL *et al*, 2006), desenvolve apenas o mínimo realmente necessário para atender ao conjunto de requisitos da iteração atual, integra o código constantemente e usa os testes de regressão (BECK *et al*., 2001, COCKBURN e HIGHSMITH, 2001, COCKBURN, 2001).

Segundo Kent Beck (1999), os métodos ágeis parecem funcionar melhor com equipes pequenas, geograficamente centralizadas, trabalhando em aplicações pequenas, pois este modelo de desenvolvimento se baseia

fortemente no conhecimento tácito (NERUR *et al*., 2005, PATEL *et al*., 2006) e na comunicação face-a-face, o que requer que os desenvolvedores estejam fisicamente próximos (TURK *et al*., 2002).

No desenvolvimento ágil, o papel convencional do gerente de projeto como planejador, organizador e controlador é substituído por um papel de facilitador do trabalho da equipe que se auto-direciona e se auto-organiza para lidar com o trabalho (BECK, 1999, NERUR *et al*., 2005). Além disso, o gerente de projeto vai se dedicar a estabelecer uma relação de parceria com os clientes (COCKBURN e HIGHSMITH, 2001).

2.1.3. Modelo de desenvolvimento de software livre

O software livre é aquele que oferece aos usuários quatro liberdades básicas e irrevogáveis: executar, estudar, melhorar e distribuir o código do programa (FSF, 2008). O acesso ao código-fonte do programa é pré-requisito para esta liberdade. Entretanto, o software livre não é necessariamente gratuito e pode ser usado, desenvolvido e distribuído comercialmente (FSF, 2008).

Apesar de terem algumas características em comum, existem muitas diferenças entres os projetos de software livre (GACEK e ARIEF, 2004) e estas diferenças influenciam também no processo de desenvolvimento adotado. Desta forma, nesta seção, serão abordadas algumas características gerais, presentes na maioria dos projetos de software livre e que podem oferecer uma visão geral deste modelo de desenvolvimento.

O modelo de desenvolvimento de software livre pode ser entendido pela metáfora do bazar (RAYMOND, 2001), onde os projetos são desenvolvidos de forma colaborativa e transparente, sem centralização do seu planejamento e execução. Em geral, os projetos de software livre se caracterizam pelo trabalho voluntário e colaborativo de desenvolvedores, com habilidades e disponibilidades distintas, geograficamente distribuídos, organizados em uma comunidade virtual através da Internet, que contribuem espontaneamente com o desenvolvimento do software.

O engajamento voluntário confere às comunidades um caráter de alto compromisso com seu sucesso, o que pode explicar sua capacidade de produção que se sobrepõe às dificuldades técnicas naturais do trabalho remoto e distribuído (CAPILUPPI *et al*., 2003, CUBRANIC e BOOTH, 1999, FELLER e

FITZGERALD, 2001, GACEK e ARIEF, 2004, HEALY e SCHUSSMAN, 2003, REIS, 2003, WARSTA e ABRAHAMSSON, 2003). Para organizar o trabalho distribuído e simultâneo dos desenvolvedores, os projetos de desenvolvimento de software livre, costumam investir na modularidade do código (TAURION, 2004).

O modelo de desenvolvimento de software livre se destaca pelo seu alto grau de colaboração e interações. Com uma equipe geograficamente dispersa, raramente seus membros se encontram pessoalmente. Devido a essa ausência de comunicação face-a-face, a comunicação acontece predominantemente de forma assíncrona e escrita, utilizando a Internet como canal de comunicação a distância (CUBRANIC e BOOTH, 1999, TAURION, 2004, YAMAUCHI *et al*., 2000). Além disso, as comunidades contam também com a participação atuante dos usuários finais que se comunicam com os desenvolvedores e entre si, comunicando problemas e trocando experiências do uso do software (FELLER e FITZGERALD, 2001, WARSTA e ABRAHAMSSON, 2003).

A reputação e o prestígio na comunidade são a base para o modelo de liderança meritocrático, onde um ou mais indivíduos contam com a confiança e reconhecimento do grupo para coordenar informalmente o trabalho (CUBRANIC e BOOTH, 1999, TAURION, 2004, YAMAUCHI *et al*., 2000). Devido ao caráter de trabalho voluntário, os padrões de gerenciamento típicos não são aplicáveis. Assim, não existe uma divisão formal ou alocação prévia do trabalho, ou seja, os desenvolvedores possuem liberdade para a escolha das tarefas que serão realizadas, com base em suas preferências e habilidades (HAEFLIGER *et al*., 2007, SCACCHI, 2007).

O desenvolvimento de software livre se adapta melhor em ambientes onde predomina a capacidade criativa e de inovação (EBERT, 2007) e existe pouca sistematização do trabalho. Outras características do desenvolvimento de software livre, especificamente voltadas à colaboração e reutilização, foram detalhadas em (MAGDALENO e WERNER, 2008).

2.2. Conciliação de modelos de desenvolvimento de software

De acordo com as características dos modelos de desenvolvimento de software, apresentadas na seção anterior, é possível perceber que eles têm

similaridades e também diferenças significativas. Porém, nenhum deles é efetivo para todos os projetos (BOEHM e TURNER, 2003, CAMERON, 2002, COCKBURN, 2001, GLASS, 2001, MILLER, 2001, PEDREIRA *et al*., 2007, TURNER e JAIN, 2002), pois cada um dos modelos funciona melhor ou enfrenta dificuldades diante de determinadas características dos projetos de desenvolvimento.

Cada um com as suas peculiaridades, seus casos de sucesso e seus desafios, os modelos de desenvolvimento orientado ao planejamento, ágil e livre seguiram caminhos distintos. Devido a diferenças de vocabulário, más interpretações e mau uso das abordagens, os três modelos costumam ser percebidos como opositores (GLAZER *et al*., 2008, REIFER, 2002), mas diversos autores defendem que eles são compatíveis e podem co-existir (BARNETT, 2004, BOEHM e TURNER, 2003, GLASS, 2001, GLAZER *et al*., 2008, PATEL *et al*., 2006, PAULK, 2001, TURK *et al*., 2002, TURNER e JAIN, 2002, VINEKAR *et al*., 2006, WARSTA e ABRAHAMSSON, 2003).

De fato, Taurion (2004) chega a afirmar que “a conciliação de processos (anárquicos?) propostos pelo modelo de software livre com os conceitos de Engenharia de Software atuais deverá ser uma das grandes oportunidades de pesquisa para os próximos anos”. Assim, a pesquisa sobre conciliação tenta identificar sob que condições cada um dos modelos funciona melhor e compreender como o melhor de cada um pode ser combinado (JIANG e EBERLEIN, 2008).

Os trabalhos de pesquisa nesta área se concentram na conciliação dos modelos tradicional e ágil. Alguns apontam oportunidades para as duas abordagens aprenderem uma com a outra, se complementarem e evoluírem (ORR, 2002). Por exemplo, o CMM poderia se tornar menos centrado em documentação, adotar a política minimalista, enfatizar a colaboração e o foco nas pessoas e incentivar a velocidade, juntamente com a qualidade. Por sua vez, o desenvolvimento ágil poderia dar mais atenção às questões de gestão, arquitetura, documentação, integração e ferramentas.

Entretanto, a maioria se limita a compará-los, analisando a equivalência das suas práticas para identificar similaridades e diferenças e explorar as possibilidades de combinação (ARIMOTO *et al*., 2009, FRITZSCHE e KEIL, 2007, KAHKONEN e ABRAHAMSSON, 2004, PAULK, 2001, SANTANA *et al*.,

2006, TURNER e JAIN, 2002). Este tipo de análise, geralmente, avalia o quanto os métodos ágeis estão ou não aderentes aos modelos de maturidade e como podem ser modificados para incorporar as práticas, principalmente organizacionais, que faltam. Assim, pretende-se chegar a um modelo híbrido que incorpore características dos dois modelos, mas que pode colocar em risco os seus potenciais benefícios e filosofias.

Outro tipo de proposta de conciliação identificada tenta resolver o problema no nível organizacional. Através da estrutura organizacional, as equipes que trabalham com cada um dos modelos de desenvolvimento são separadas em unidades organizacionais distintas, subordinadas a uma gerência maior comum (VINEKAR *et al*., 2006). Esta proposta dificulta a colaboração e a comunicação entre as equipes de desenvolvimento, sem lidar de fato com a conciliação, já que pressupõe que um projeto será desenvolvido de forma totalmente ágil ou puramente orientada ao planejamento.

Algumas propostas de conciliação (COHAN e GLAZER, 2009, GLAZER *et al*., 2008, REIFER, 2003) se baseiam na ideia de que os modelos de maturidade trabalham em um alto nível de abstração, definindo apenas *o que* o projeto deve fazer, mas sem indicar o uso de uma metodologia específica. Já os métodos ágeis representam uma metodologia de desenvolvimento de software e, por isso, focam em *como* os projetos devem desenvolver os produtos. Logo, ao definir o seu processo de desenvolvimento de software, as organizações podem adotar um modelo de maturidade como um *framework* de melhoria de processos para acomodar práticas sugeridas por diferentes metodologias de desenvolvimento de software.

Neste sentido, a escolha de uma metodologia ágil, que define como os projetos devem ser devolvidos, é uma possibilidade concreta e há evidências (HANSSON *et al*., 2006, REIFER, 2002, VINEKAR *et al*., 2006) que parecem indicar que as organizações estão realmente tentando utilizar o desenvolvimento orientado ao planejamento e ágil conjuntamente, apesar da ausência de orientações sobre como fazê-lo.

Quanto à conciliação dos modelos ágil e livre, Warsta e Abrahamsson (2003) argumentam que o software livre está no meio do caminho entre os métodos ágeis e o modelo orientado ao planejamento, tendo mais características em comum com os métodos ágeis. Por exemplo, ambos

enfatizam o lançamento frequente de versões, lidam com o mínimo de documentação e têm foco em testes (BARNETT, 2004). Outras similaridades e diferenças entre os dois modelos são apontadas por Tsirakidis *et al*. (2009). Esta visão, de que os valores e práticas ágeis são aplicáveis também ao software livre, é compartilhada pelos próprios desenvolvedores dos projetos de software livre (DIAS e OLIVEIRA, 2009).

Contudo, a natureza complexa da atividade de desenvolvimento de software e a grande variedade de métodos existentes tornam a tarefa de comparação dos modelos de desenvolvimento, adotada pela maioria das propostas, uma tarefa árdua e imprecisa. Esta forma de integração entre os modelos de desenvolvimento de software limita o potencial de sinergia entre eles, pois constrói uma metodologia híbrida e incompleta (BOEHM e TURNER, 2003), onde não se consegue mais garantir que o processo resultante realmente tenha as características desejadas. Assim, destacam-se as propostas de uso de controle estatístico (ANDERSON, 2005) e da análise de riscos (BOEHM e TURNER, 2003).

A proposta apresentada por Anderson (2005), como resultado da experiência da Microsoft, se baseia em mecanismos objetivos para medir e entender a variação natural do processo e utiliza a análise estatística para verificar se o processo está sob controle, ou seja, se o processo está sendo executado de acordo com o previsto. A partir desta ideia, o autor defende que a compatibilidade do CMMI, em todos os seus níveis, com os métodos ágeis pode ser alcançada ao usar métricas ágeis e projetar métodos de planejamento e controle baseados nesta análise estatística. O interessante nesta proposta é que ela percebe a importância de não apenas conciliar os modelos, mas sim de planejar e acompanhar o balanceamento das características desejadas no nível do processo e ao longo da sua execução.

Boehm e Turner (2003) sugerem a análise de riscos das características do projeto como o caminho para alcançar o equilíbrio entre agilidade e disciplina. Esta análise considera três categorias de riscos: os riscos do ambiente, os riscos de se adotar um modelo ágil e os riscos de se adotar um modelo orientado ao planejamento. O método proposto possui cinco passos e usa a análise de riscos para incorporar tanto a agilidade quanto a disciplina nas

proporções necessárias ao projeto, customizando o processo de desenvolvimento de software.

Esta proposta de Boehm e Turner (2003) é a que tem mais similaridade com o presente trabalho de pesquisa, pois leva em consideração a caracterização do projeto. Porém ela foca apenas no desenvolvimento ágil e no orientado ao planejamento, sem levar em consideração o modelo de desenvolvimento livre. Além disso, ao enfatizar a agilidade e a disciplina, no fundo esta proposta também remete à conciliação de modelos de desenvolvimento que não é o objetivo do presente trabalho.

A partir das propostas de conciliação identificadas, observou-se a necessidade de caracterizar melhor a área e identificar outras abordagens mais próximas dos objetivos desta pesquisa. Assim, na próxima seção, são apresentados os resultados de uma revisão sistemática realizada com este objetivo.

2.2.1. Revisão *quasi*-sistemática

Kitchenham (2004) define uma revisão sistemática da literatura como um "meio de identificar, avaliar e interpretar toda pesquisa disponível relevante a uma questão, ou área, ou fenômeno de interesse de uma pesquisa particular". Revisões sistemáticas são baseadas em uma estratégia de pesquisa definida, que visa detectar o máximo possível de literatura relevante.

Apesar de preservar o mesmo formalismo e seguir este mesmo processo da revisão sistemática, a presente revisão caracteriza-se como uma revisão *quasi*-sistemática (TRAVASSOS *et al*., 2008), pois o objetivo deste estudo é caracterizar a área de conciliação de processos de desenvolvimento de software e, portanto, não envolve a comparação de alternativas de solução.

Esta seção descreve os pontos principais desta revisão, assim como seus resultados, que servem como embasamento para a pesquisa desenvolvida. A íntegra desta revisão pode ser encontrada em (MAGDALENO *et al*., 2009).

2.2.1.1. Planejamento da revisão quasi-*sistemática*

A revisão *quasi*-sistemática inicia pela definição de um protocolo de revisão que especifica a questão principal (QP) e as questões secundárias (QS) da pesquisa e os métodos que serão utilizados para executar a revisão. O

protocolo completo foi publicado em (MAGDALENO *et al*., 2009). As próximas subseções descrevem os principais componentes do protocolo utilizado na presente revisão. Este protocolo foi definido com base no *template* proposto por Biolchini *et al*. (2005) e no exemplo apresentado por Abrantes e Travassos (2007).

a) Objetivo

O objetivo desta pesquisa é caracterizar a conciliação de processos de desenvolvimento de software tradicional, livre e ágil.

b) Questões de pesquisa

- **QP** - Como caracterizar a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?
    - **QS1** - Quais são as oportunidades e os desafios para a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?
    - **QS2** - Quais são as propostas para a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?
    - **QS3** - Quais são as estratégias adotadas, pelas organizações ou pelas comunidades, para a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?
    - **QS4** - Quais foram os resultados obtidos pelas organizações ou comunidades que conciliaram processos de desenvolvimento de software tradicional, livre e ágil?

c) Escopo

Para a definição do escopo desta revisão, é adotada uma abordagem PICO que estrutura a questão de pesquisa em quatro elementos básicos: população, intervenção, comparação e resultado (PAI *et al*., 2004).

- **Aplicação:** Servir de base ou apoiar pesquisas sobre a conciliação de processos de desenvolvimento de software tradicional, livre e ágil.
- **População:** Organizações de desenvolvimento de software e Projetos de desenvolvimento de software.
- **Intervenção:** Processo de desenvolvimento de software (tradicional, livre e ágil).
- **Comparação:** Não há.

- **Resultados:** Caracterização da conciliação; identificação das oportunidades e desafios; identificação das propostas; identificação das estratégias adotadas; e identificação dos resultados obtidos.

d) Seleção de fontes

As bibliotecas digitais selecionadas são: Compendex, IEEE, Scopus e Web of Science. Estas bibliotecas foram escolhidas porque são as que se tem acesso para recuperação de referências, bem como maior facilidade para recuperação do texto completo do artigo. Além disso, estas fontes foram consideradas significativas no sentido de oferecerem publicações pertinentes e que podem contribuir para o resultado da pesquisa.

e)* Strings *de busca

De acordo com Pai *et al*. (2004) os quatro elementos básicos que estruturam a questão de pesquisa podem ser relacionados com o operador lógico AND. Como o elemento "comparação" não é aplicável a este estudo, isto foi feito para o conjunto de palavras-chave escolhidas para representar cada um dos elementos "população" (P), "intervenção" (I) e "resultado" (R), resultando na seguinte estrutura: (P) AND (I) AND (R). Para cada um destes três elementos da estrutura, as respectivas palavras-chave foram combinadas com o operador lógico OR.

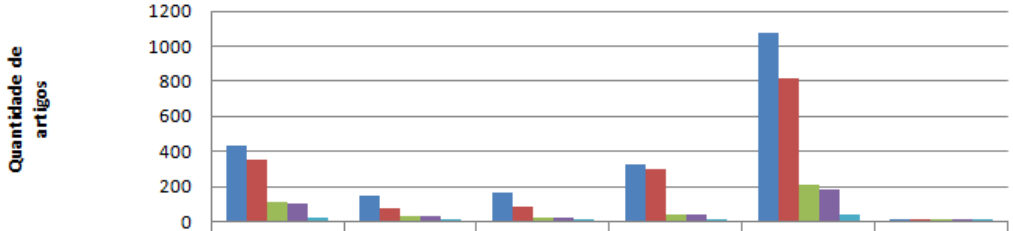
Devido ao grande número de termos, optou-se por dividir a busca em várias *strings,* de acordo com as combinações possíveis entre os processos de desenvolvimento de software tradicional, ágil e livre. Com isso, a complexidade da *string* de busca foi reduzida, o que garante melhor legibilidade e diminui a probabilidade de erro (humano e das máquinas de busca). A mesma lógica foi utilizada na formação das *strings* das QSs. As *strings* de busca utilizadas que podem ser consultadas em Magdaleno *et al*. (2009).

2.2.1.2. Execução da revisão quasi-*sistemática*

Após o término da fase de planejamento da revisão *quasi*-sistemática, o protocolo foi estabelecido e a execução da revisão foi iniciada. Esta seção descreve brevemente como a revisão *quasi*-sistemática foi conduzida. As buscas foram realizadas em cada uma das fontes selecionadas no período de 26/04/2009 a 27/04/2009.

Inicialmente, a execução do protocolo retornou um total de 1.078 artigos. Estes artigos foram submetidos a um processo de filtragem, composto por

quatro passos (Figura 2.1): eliminação de duplicatas, filtro por título e resumo, filtro por disponibilidade de texto completo e filtro pela leitura completa do conteúdo do artigo. O número de artigos retornados de cada biblioteca digital, durante cada passo da filtragem, pode ser visto na Figura 2.1.



	Compendex	IEEE	Scopus	Web of Science	Total	%
Resultados das máquinas de busca	436	150	162	330	1078	100%
Após exclusão das duplicatas	356	75	85	302	818	76%
Após filtro por título e resumo	113	27	25	43	208	19%
Após filtro por texto completo	99	27	20	36	182	17%
Após filtro por conteúdo	22	8	4	10	44	4%

Figura 2.1 – Processo de filtragem de artigos

Como primeiro passo para a análise dos documentos recuperados, as repetições encontradas nas buscas foram eliminadas, mantendo-se o artigo remanescente contabilizado para a biblioteca digital com maior quantidade de itens recuperados. Desta forma foram excluídas 24% das referências. Este percentual de repetições é reflexo das superposições de trabalhos indexados pelas diferentes bibliotecas digitais.

Após a eliminação das duplicatas, foi feita uma primeira avaliação, a partir da leitura de todos os títulos e resumos. Assim, foram excluídos 25% dos trabalhos avaliados, pois nitidamente tratavam de outros assuntos não pertinentes à pesquisa.

Após a execução do primeiro filtro dos artigos, a lista de artigos excluídos foi revista, por amostragem, por 4 outros pesquisadores. O objetivo desta verificação é evitar que artigos sejam excluídos indevidamente e que assim se percam trabalhos relevantes para a pesquisa. Então, em caso de impasse entre os pesquisadores, ou dúvida, a publicação foi incluída na lista de selecionadas. Os resultados apresentados acima já foram consolidados após esta verificação.

Para uma avaliação mais apurada e detalhada, é necessária a leitura do texto completo dos artigos. Então, o próximo passo é a recuperação dos arquivos com os textos completos dos artigos. Entretanto, não foi possível

obter acesso ao texto de referências, mesmo utilizando as máquinas de buscas ou tentando contato por e-mail com os respectivos autores.

Os artigos recuperados foram lidos pelo pesquisador, visando selecionar os candidatos a fazer parte da revisão sistemática. Para todos os artigos lidos foi feita uma nova análise dos critérios de inclusão e exclusão e foram selecionados 44 artigos finais. Para cada artigo selecionado foi preenchido o formulário de extração de informações para responder às questões de pesquisa. A lista completa dos artigos selecionados e as informações detalhadas extraídas de cada um deles, podem ser consultadas em Magdaleno *et al*. (2009).

2.2.1.3. Resultados da revisão quasi-*sistemática*

Como resultado da revisão *quasi*-sistemática, 44 artigos foram efetivamente selecionados. Este total de artigos selecionados demonstra que apenas 4% dos documentos obtidos inicialmente contribuíram efetivamente para esta revisão. A grande maioria dos documentos lidos foi excluída, pois não trazia uma contribuição direta para caracterizar a conciliação de processos de desenvolvimento de software. Observando-se os critérios de exclusão aplicados, é possível perceber que as principais razões para exclusão de trabalhos foram: foco apenas em um dos processos de desenvolvimento de software de forma individual (CE4); falta de relevância para o tema da presente pesquisa (CE3); e foco em ferramentas ou ambientes computacionais e não no processo de desenvolvimento de software (CE5).

A maneira como os artigos se referenciam e a evolução cronológica das publicações, considerando os 44 artigos obtidos na execução da revisão *quasi*-sistemática, é mostrada pela Figura 2.2. Esta representação gráfica auxilia na identificação dos artigos mais influentes e de possíveis tendências de pesquisa.

Os artigos mais citados estão destacados e nota-se claramente a influência dos artigos de Boehm e Turner neste tema de pesquisa. Através da análise deste gráfico (Figura 2.2) também é possível perceber que, apesar de não ter sido imposta nenhuma restrição temporal na busca, os resultados obtidos datam a partir de 2001, o que demonstra ainda uma imaturidade desta área de pesquisa e a necessidade, também reforçada por alguns autores, de mais trabalhos científicos relacionados ao tema. Esta juventude da área se deve também a percepção inicial de que os modelos de desenvolvimento

seriam opositores e que não haveria espaço para nenhuma tentativa de integração entre eles. Porém, é importante destacar o crescimento do número de trabalhos relacionados ao tema nos últimos anos (Figura 2.2).

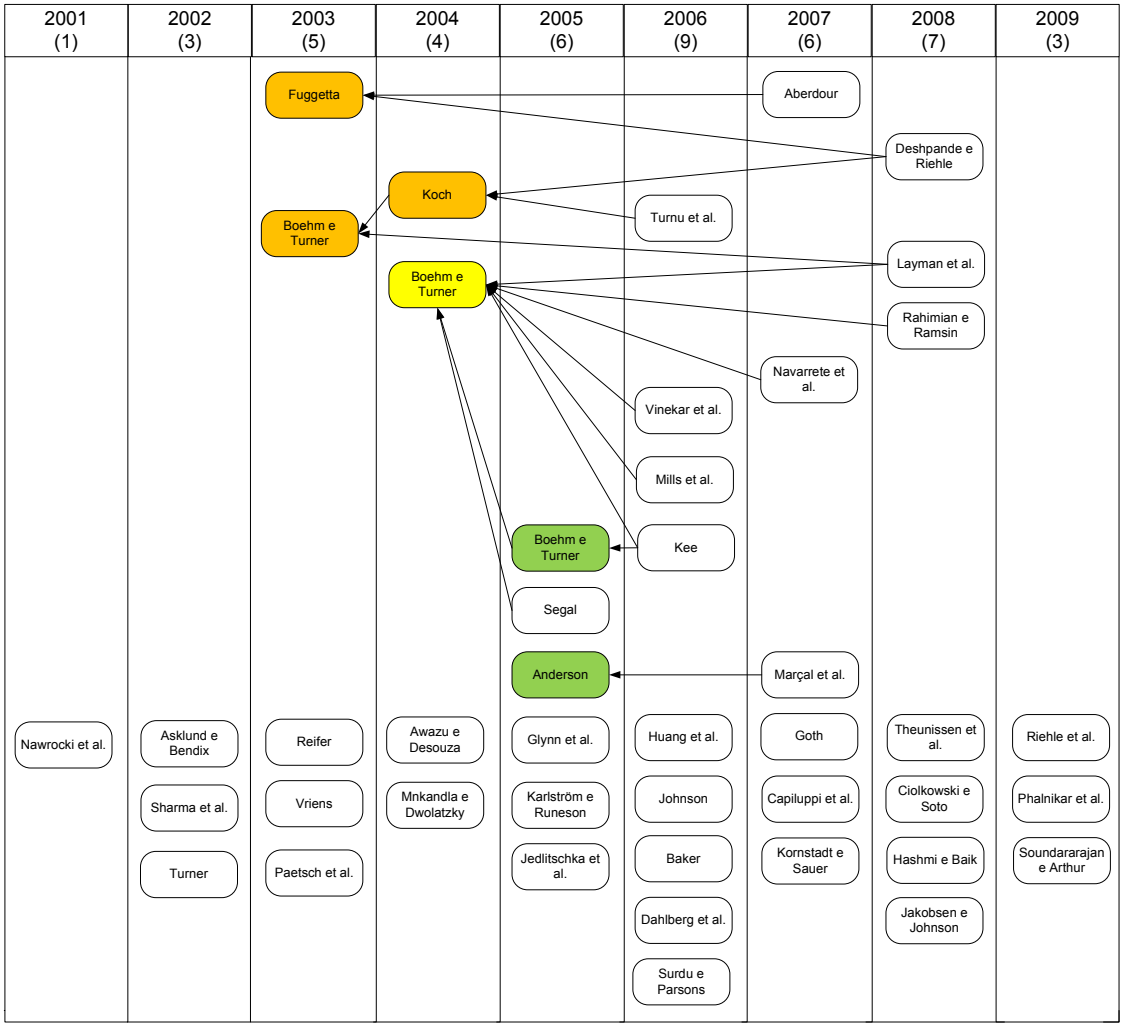


Figura 2.2 – Mapa de citações entre os artigos selecionados[6]

Para uma análise completa dos resultados obtidos com esta revisão *quasi*-sistemática, é importante considerar também as limitações desta revisão. Em primeiro lugar, as buscas foram realizadas em abril/2009, o que significa que os artigos publicados depois desta data não foram incluídos, o que pode dar a falsa impressão de que houve poucas publicações este ano. Assim, como trabalho futuro, pretende-se re-executar, em 2010, este protocolo tentando captar referências mais recentes que estendam o espaço de busca cronologicamente. Também podem ser incluídas outras bibliotecas digitais, em uma tentativa de recuperar documentos indexados somente por estas máquinas, que poderiam alargar o espaço de busca geograficamente. Embora

[6] Este gráfico foi construído com base no exemplo de Kalinowski (2009).

a abordagem sistemática seguida neste estudo garanta a confiabilidade e integridade dos resultados, isto poderá ser ampliado durante uma re-execução.

Além disso, a ausência de bibliotecas digitais, que indexem os trabalhos nacionais, impede que estudos realizados por pesquisadores brasileiros, possivelmente relevantes ao tema de pesquisa, tenham sido incluídos, o que ameaça a completude desta revisão. Para minimizar este risco, estes trabalhos são acompanhados através de pesquisas em conferências e *workshops* nacionais. Alguns trabalhos (ARIMOTO *et al*., 2009, SANTANA *et al*., 2006) nesta situação já foram identificados e analisados e observou-se que eles também se encaixam na caracterização da conciliação de processos apresentada na próxima seção, como uma proposta do tipo combinação de práticas de modelos.

Por fim, outros trabalhos, pertinentes ao tema e publicados internacionalmente, não são indexados pelas máquinas de busca, devido a pouca relevância do veículo de publicação. Estes trabalhos (GLASS, 2001, GLAZER *et al*., 2008) foram identificados na revisão inicial da literatura e utilizados como base para a preparação do protocolo de busca.

2.2.1.4. Caracterização da conciliação de processos de desenvolvimento de software

O objetivo desta seção é caracterizar a conciliação de processos de desenvolvimento de software, de acordo com os resultados obtidos com a revisão *quasi*-sistemática. Como a questão de pesquisa foi subdividida em questão principal e questões secundárias, cada uma delas será respondida.

As informações extraídas desta revisão contribuíram para as seguintes observações: (i) poucos estudos tratam a conciliação dos três modelos de desenvolvimento; (ii) a conciliação do modelo tradicional com o ágil é mais comum; (iii) as grandes organizações estão interessadas na conciliação e estão tentando combinar estes modelos, mesmo sem qualquer orientação sobre como fazer isso; (iv) a conciliação de modelos de desenvolvimento de software, ainda é uma questão em aberto, pois a maioria das propostas ainda está em um estágio inicial de pesquisa; e (v) a maioria dos trabalhos se dedica a uma combinação de práticas, onde o resultado é híbrido, mas não necessariamente atende às necessidades dos projetos e organizações.

QP: Como caracterizar a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?

Em geral, o que pôde ser percebido através da execução do estudo é a escassez de relatos na literatura sobre a conciliação dos três processos de desenvolvimento de software. O único artigo obtido englobando os três processos trata dos desafios para a conciliação dos processos de desenvolvimento ágil e livre dentro de uma cultura corporativa (THEUNISSEN *et al*., 2008). O modelo de conciliação considerado no artigo em questão diz respeito a uma equipe ágil trabalhando dentro de uma organização e colaborando com um projeto de software livre externo. De fato, o enfoque do artigo não é exatamente nos três processos de desenvolvimento de software, visto que do ponto de vista tradicional não é abordado o processo de desenvolvimento, mas sim o ambiente e a cultura organizacional. Além disso, esta conciliação não acontece com os três modelos sendo usados dentro da mesma organização. Contudo, este trabalho foi o mais próximo disso que foi possível encontrar.

Tal resultado nos leva a seguinte indagação: por que será que não existem mais trabalhos relacionando os três processos de desenvolvimento de software? Após algumas reflexões, a primeira possibilidade considerada é que nem todos os autores e pesquisadores enxergam a existência de um processo de desenvolvimento no software livre, por considerar que os projetos de software livre são realizados de forma *ad-hoc*.

Entretanto, a alternativa que nos parece mais razoável para responder a esta questão é que a conciliação de apenas dois processos de desenvolvimento de software já é um grande desafio. Assim, os pesquisadores estariam iniciando os trabalhos nesta área de pesquisa, que ainda é bem recente se considerarmos que os primeiros trabalhos foram publicados em 2001, tentando combiná-los dois a dois, antes de envolver um terceiro elemento complicador nesta equação.

Esta hipótese também é embasada pela observação de que existe uma quantidade mais significativa de trabalhos combinando os modelos tradicionais e ágeis, principalmente XP e Scrum. Começar a conciliação por estes dois modelos parece ser uma opção natural se considerarmos que este poderia ser um caminho de evolução progressiva para se chegar à conciliação dos demais.

A conciliação entre o modelo de desenvolvimento ágil e o livre também é incipiente ainda e, de forma geral, se baseia na conciliação de uma prática específica, como a integração de código (DESHPANDE e RIEHLE, 2008) ou o desenvolvimento orientado a testes (TURNU *et al*., 2006).

QS1 - Quais são as oportunidades e os desafios para a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?

As principais oportunidades para a conciliação dos modelos de desenvolvimento tradicional, ágil e livre são a disponibilidade e transparência no acesso ao código-fonte, o estímulo a colaboração, a melhoria na qualidade do código e a redução de custos (MAGDALENO *et al*., 2009).

Estes resultados sinalizam uma busca pela disponibilidade do código, que é vista como uma oportunidade para a inovação no desenvolvimento de software. Além disso, as outras oportunidades citadas são argumentos comuns para pesquisas relacionadas à Engenharia de Software e aos processos de desenvolvimento de software em geral.

Existe uma diversidade de desafios citados para a conciliação dos três processos de desenvolvimento de software, entre os quais destacam-se: (i) falta de estudos científicos sobre o tema, o que resulta em carência de orientações para as organizações e ausência de resultados experimentais; (ii) resistência a mudança por parte dos clientes e desenvolvedores (MAGDALENO *et al*., 2009). O conhecimento destes desafios pode ajudar a traçar estratégias para superá-los no futuro.

QS2 - Quais são as propostas para a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?

O tipo de proposta mais comum é aquela que sugere a customização de modelos tradicionais para introduzir práticas ágeis de forma que possam ser aplicados a contextos específicos como reutilização de software de prateleira, desenvolvimento distribuído e desenvolvimento de aplicações móveis.

Outro tipo, bastante recorrente, de proposta apresentada na literatura é a combinação de práticas de diferentes modelos. Este tipo de proposta envolve o mapeamento da equivalência das práticas dos modelos de desenvolvimento distintos, tais como XP com CMMI ou Scrum com ISO, e geralmente resulta em

uma análise do quanto os métodos ágeis estão ou não aderentes aos modelos de qualidade tradicionais e como os métodos ágeis podem ser modificados para incorporar as práticas que faltam.

Dentre o universo de propostas analisadas, destacam-se as propostas de uso de controle estatístico (ANDERSON, 2005) e da análise de riscos (BOEHM e TURNER, 2003), apresentadas anteriormente na seção 2.2.

QS3 - Quais são as estratégias adotadas, pelas organizações ou pelas comunidades, para a conciliação de processos de desenvolvimento de software tradicional, livre e ágil?

Existe uma quantidade significativa de grandes empresas (tais como Motorola, Philiphs, Volvo, Exército Americano e etc.) interessadas na conciliação. Em geral, estas empresas estão adotando estratégias de combinação de modelos, usando, principalmente, CMMI, XP e Scrum (MAGDALENO *et al*., 2009).

QS4 - Quais foram os resultados obtidos pelas organizações ou comunidades que conciliaram processos de desenvolvimento de software tradicional, livre e ágil?

A maioria dos relatos das organizações tem dificuldades em mensurar os resultados obtidos com a conciliação, mas enumeram diversas lições aprendidas. Também são citadas a redução de defeitos e a melhoria na produtividade (MAGDALENO *et al*., 2009).

Os resultados obtidos com esta revisão *quasi*-sistemática mostram que a comunidade tem sugerido abordagens que conciliem os diferentes modelos de desenvolvimento através da combinação das suas práticas. Entretanto, neste trabalho de pesquisa defende-se que a solução para a integração dos universos representados por estes modelos de desenvolvimento é o balanceamento das suas características. A visão defendida neste trabalho é que o caminho para a definição de processos mais eficazes envolve a adaptação de processos de desenvolvimento de software aos contextos específicos dos projetos e organizações. Este é, portanto, o tema da próxima seção.

2.3. Adaptação de processos de desenvolvimento de software

Tendo em mente que organizações são diferentes, e ainda que dois projetos dentro da mesma organização também podem ser diferentes (BERGER, 2003, CUGOLA e GHEZZI, 1998), não existe um processo de software que possa ser genericamente aplicado a todos os projetos (MACHADO, 2000). Dependendo das características do projeto, um processo aplicado com sucesso em um projeto pode ser um fracasso em outro (PEDREIRA *et al*., 2007).

Diante deste cenário, surge a necessidade de adaptar os processos de desenvolvimento às necessidades dos projetos e organizações. A adaptação de processos é o "ato de particularizar um processo geral para derivar uma definição aplicável a um contexto mais específico" (GINSBERG e QUINN, 1995, PEDREIRA *et al*., 2007).

Apesar da adaptação de processos ser uma das principais tarefas executadas pelo gerente do projeto, ela não é simples. Esta tarefa envolve o conhecimento de muitos aspectos da Engenharia de Software e requer a harmonização de muitos fatores do contexto da equipe, do projeto ou da organização (BARRETO, 2007). Devido a esta complexidade, geralmente, o gerente de projeto não é capaz de avaliar todas as opções de adaptação do processo e acaba fazendo esta adaptação de forma *ad-hoc*, baseado na sua experiência e no conhecimento que possui de alguns modelos de desenvolvimento apenas. Como resultado, o processo adaptado pode não ser a melhor alternativa para o projeto em questão. Além disso, as decisões de adaptação são tomadas no início do projeto e registradas no plano de projeto, mas é impossível prever, neste momento, um processo totalmente adequado, pois os requisitos e o ambiente do projeto mudam.

Como o orçamento do projeto, o prazo de desenvolvimento e a qualidade do produto dependem diretamente da qualidade do processo de software, a adaptação de processos precisa ser feita corretamente para evitar que as seguintes possíveis consequências negativas se materializem: um processo de software ruim pode envolver atividades desnecessárias que levam a perda de tempo e dinheiro; a omissão de atividades necessárias pode prejudicar a qualidade do produto (PEDREIRA *et al*., 2007).

Para enfrentar estes desafios, soluções para a adaptação de processos têm sido exploradas por diversas áreas de pesquisa. A área de *contingency factors* (BEKKERS *et al*., 2008, KUMAR e WELKE, 1992) diz respeito à adaptabilidade dos métodos de acordo com o ambiente da organização e com as variáveis que descrevem o contexto do projeto. Estas variáveis ajudam a caracterizar os diferentes projetos para determinar a seleção do método apropriado a partir de um portfólio de métodos (DAVIS, 1982). Entretanto, o principal problema desta abordagem é que se espera que os desenvolvedores estejam familiarizados com diferentes métodos para que possam escolher o mais adequado para cada situação (FITZGERALD *et al*., 2000). Uma forte familiaridade com um método não é tão comum na prática entre os desenvolvedores. Assim, esperar que eles atinjam este nível de conhecimento em diversos métodos não é uma expectativa realista.

Outra área relacionada, que surge devido à ausência de diretrizes sobre como relacionar os *contingency factors* aos métodos de desenvolvimento, é a de *method engineering* (BRINKKEMPER, 1996, SLOOTEN e BRINKKEMPER, 1993). Esta área tem por objetivo oferecer flexibilidade às organizações na configuração de um processo específico para o projeto de desenvolvimento, usando métodos ou fragmentos de métodos, armazenados em um repositório (HARMSEN *et al*., 1994) e selecionados para satisfazer o contexto de um projeto em particular. Esta área pretende oferecer orientações sobre o uso e integração das várias alternativas de abordagens, técnicas e ferramentas disponíveis para o desenvolvimento de software dentro de uma organização. Contudo, os seus principais desafios são estabelecer o repositório e definir os fragmentos de processos (FITZGERALD *et al*., 2000). Além disso, não são encontrados relatos de experiências práticas da aplicação dessas abordagens.

Alguns modelos de maturidade, tais como o CMMI (CHRISSIS *et al*., 2006), propõem o uso de guias ou diretrizes de adaptação de processos. Estas diretrizes de adaptação são estabelecidas para guiar os projetos na seleção do modelo de ciclo de vida (dentre as opções aprovadas pela organização) e para adaptar o processo padrão de forma que atenda às especificidades do projeto. Porém, ao contrário do que se poderia esperar, estes modelos não definem efetivamente o que estes guias de adaptação devem conter e fornecem apenas

algumas orientações gerais que não são suficientes para efetivamente ajudar na adaptação (ARMBRUST *et al*., 2008, PENG XU e RAMESH, 2008).

Nesta mesma linha de pesquisa, Machado (2000) sugere um modelo para definição, especialização e instanciação de processos de software. Este trabalho contribui com a definição dos níveis de adaptação e propõe algumas características que devem ser levadas em consideração em cada um destes níveis de adaptação. Segundo este modelo, o processo padrão da organização é definido com base na norma ISO/IEC 12207, nos modelos de maturidade, tais como o CMMI, e nas características específicas da organização. O processo padrão pode ser especializado para o tipo de software e as características específicas da metodologia de desenvolvimento. Por fim, este processo especializado pode ser instanciado de acordo com as particularidades dos projetos. Assim, resultam de tais etapas, processos de software em três diferentes níveis de abstração (Figura 2.3).

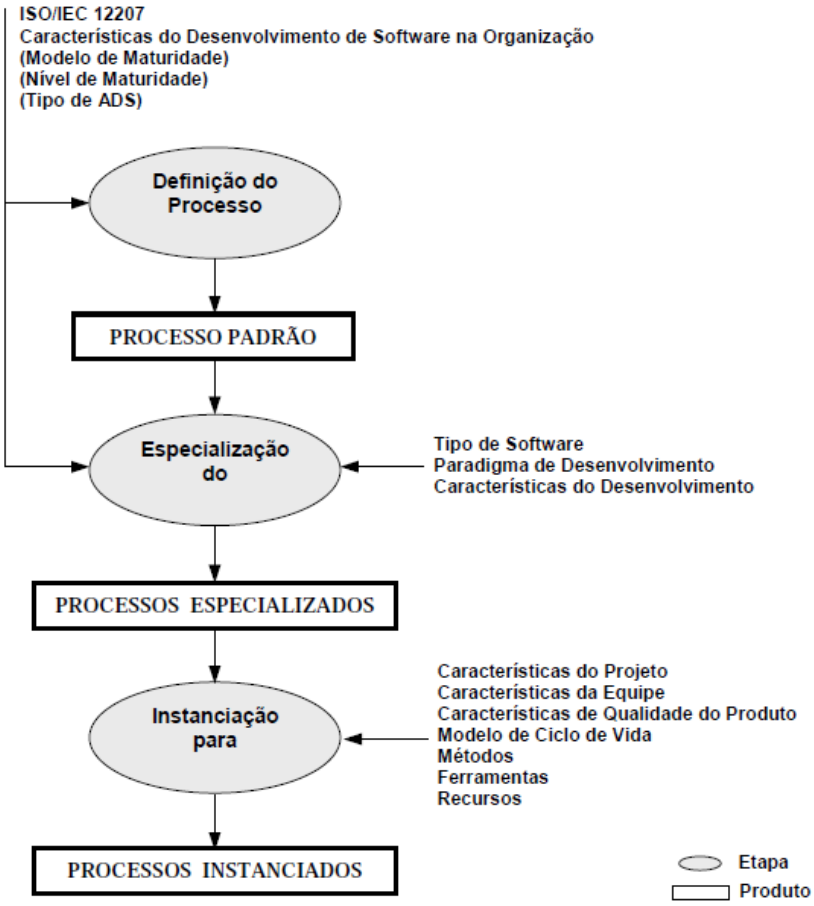


Figura 2.3 – Modelo de definição/especialização/instanciação de processos (MACHADO, 2000)

Porém, a estratégia comumente adotada pelas organizações envolve apenas dois níveis de adaptação (PEDREIRA *et al*., 2007). O primeiro nível é a

definição de um processo padrão com as práticas que se deseja incorporar em todos os projetos para guiar o desenvolvimento dentro da organização. Em geral, esta definição do processo padrão é feita pelo Grupo de Definição de Processos de Software (ou equivalente) da organização. O processo padrão estabelece um conjunto de elementos fundamentais e uma estrutura comum a ser utilizada pela organização nos seus projetos de software. Desta forma, estabelece-se um processo básico que servirá como ponto de partida para a posterior instanciação dos processos adequados às diferentes características de cada projeto (BERGER, 2003, GINSBERG e QUINN, 1995, MACHADO, 2000, ROCHA *et al*., 2001).

Para acomodar todas as iniciativas de desenvolvimento em uma organização, o processo padrão vai inevitavelmente estar em um alto nível de abstração que atenda às necessidades de todos os projetos, mas não vai ser capaz de fornecer apoio específico às atividades individuais dos projetos. Desta forma, ser adaptável e configurável torna-se um importante requisito a ser atingido na definição de um processo padrão (BERGER, 2003).

O processo padrão da organização deve ser adaptado caso a caso, considerando-se as especificidades do projeto. Um processo específico para o projeto é uma coleção de atividades concretas e inter-relacionadas organizadas ao longo da linha do tempo do projeto e que leva em consideração as características do projeto (ROCHA *et al*., 2001). O gerente de projeto é responsável por este nível de adaptação do processo padrão da organização para o processo específico do projeto.

Os modelos de maturidade (CHRISSIS *et al*., 2006, SOFTEX, 2009) também estabelecem que os processos podem ser definidos com base em unidades menores e reutilizáveis de processos. Assim, os processos são decompostos em componentes de processo. Estes componentes são reutilizáveis e podem ser conectados para instanciar processos novos e mais complexos (BARRETO, 2007, FIORINI, 2001, MAGDALENO *et al*., 2007). O fato de estar reutilizando processos permite fazer uso do conhecimento e da experiência adquiridos previamente. Desta forma, é possível agilizar a tarefa de adaptação de processos.

Em uma linha de pesquisa mais voltada a oferecer apoio computacional a esta atividade, Park *et al*. (2006) propõem um modelo de rede neural artificial

para automatizar a fase de seleção das atividades que vão compor um processo específico, a partir do processo genérico, com base nos parâmetros e nas regras de adaptação do processo. Assim, são oferecidos mecanismos capazes de especializar um processo genérico para um determinado projeto, reutilizando conhecimento e aprendendo com as experiências anteriores de adaptação.

Resumindo, a análise de todos estes trabalhos permitiu observar que:

(i) As propostas de adaptação de processos não se preocupam em balancear as características dos diferentes modelos de desenvolvimento de software;

(ii) Em geral, as propostas para adaptação de processos não oferecem uma sistemática para apoiar a realização desta tarefa e se limitam a oferecer apenas orientações gerais que não são suficientes para ajudar efetivamente na adaptação;

(iii) A adaptação de processos pode ser feita partindo do processo padrão da organização ou através de estruturas de reutilização de processos, como os componentes de processos (CHRISSIS *et al*., 2006);

(iv) Na adaptação do processo devem ser considerados os contextos da equipe, do projeto e da organização, ainda que o entendimento sobre como eles afetam a adaptação de processos seja limitado (PENG XU e RAMESH, 2008) e falte suporte eficiente (JAUFMAN e MUNCH, 2005);

(v) É possível pensar em automatizar pelo menos alguns dos passos da atividade de adaptação de processos (PARK *et al*., 2006).

Com base nestas conclusões, este trabalho propõe que uma abordagem de adaptação de processos deve possuir as seguintes características:

(i) **Balanceamento** entre os aspectos de colaboração e disciplina dos modelos de desenvolvimento de software;

(ii) **Sistemática** para realizar a adaptação de processos baseada em uma estrutura de reutilização de processos;

(iii) **Dinamicidade** para evoluir ao longo do tempo e se adequar ao contexto da organização, do projeto e da equipe.

Assim, retomamos o problema de pesquisa deste trabalho:

Como adaptar processos de desenvolvimento de software de forma balanceada, sistemática e dinâmica?

Como resultado, espera-se que esta abordagem possa contribuir para: a redução do esforço requerido para a adaptação de processos; e o aumento na adequação dos processos gerados.

2.4. Considerações finais

Este capítulo apresentou as principais características dos modelos de desenvolvimento de software orientado ao planejamento, ágil e livre. Estes três modelos têm o mesmo objetivo: melhorar o desenvolvimento de software; mas adotam enfoques distintos. Enquanto no desenvolvimento orientado ao planejamento busca-se previsibilidade, estabilidade e confiabilidade (CHRISSIS *et al*., 2006), o desenvolvimento ágil tenta agregar valor ao negócio rapidamente e se adaptar às mudanças de mercado, tecnologia e ambiente (COCKBURN, 2001). Por outro lado, no desenvolvimento de software livre, o objetivo principal é garantir as liberdades básicas dos usuários e desenvolvedores para executar, estudar, adaptar, melhorar e distribuir o código do programa (FSF, 2008).

Em busca de flexibilidade, qualidade e produtividade, as organizações vêm buscando a conciliação destes modelos de desenvolvimento. Porém, esta conciliação envolve mais do que a combinação das práticas dos diferentes modelos. Para que as organizações possam alcançar seus objetivos, elas devem ser capazes de perceber as características de cada projeto para adaptar o processo de desenvolvimento de software a essas especificidades e ao seu contexto organizacional (MAGDALENO *et al*., 2009).

De fato, os resultados da pesquisa realizada, por Hansson *et al*. (2006), em cinco organizações, mostram que elas já adotam processos de desenvolvimento adaptados. Porém, esta adaptação de processos não é uma tarefa simples. Assim, este capítulo também discutiu as principais propostas existentes para entender as suas contribuições e limitações e definir as características desejadas para uma abordagem de adaptação.

A primeira destas características diz respeito ao balanceamento dos aspectos de colaboração e disciplina dos modelos de desenvolvimento de software. Estes dois aspectos serão explicados e justificados no próximo capítulo e o modo como eles serão balanceados é o tema do capítulo 7.

3. Colaboração e Disciplina

Este capítulo define os conceitos de colaboração e disciplina. A partir do entendimento destes conceitos, são identificados os principais aspectos de colaboração e disciplina existentes nos modelos de desenvolvimento de software para justificar a primeira sub-hipótese deste trabalho sobre a possibilidade de balanceá-los. Por fim, são apresentados os instrumentos para o planejamento e acompanhamento dos níveis de colaboração e disciplina no processo adaptado para o projeto de desenvolvimento de software.

3.1. Colaboração

O trabalho em grupo se tornou uma importante estratégia de negócios e vem sendo demandado como instrumento para vencer os desafios do novo mundo globalizado (SARMENTO, 2002, SCHOLTES *et al*., 2003, TELLERIA *et al*., 2002). A complexidade crescente destes desafios passa a requerer habilidades multidisciplinares. Assim, as organizações vêm recorrendo à colaboração para fins de produtividade e compartilhamento de conhecimento (MAGDALENO, 2006).

Durante o tempo no qual as pessoas estão trabalhando em conjunto, elas estão organizadas em grupos. Os grupos tendem a superar o desempenho quantitativo e qualitativo de indivíduos agindo sozinhos. Os grupos podem alcançar uma compreensão que nenhum dos seus membros possuía previamente e que não poderia ter sido obtida caso seus membros tivessem trabalhado de forma isolada, pois representa mais do que a soma individual das partes (ALEIXO, 2003, DEMARCO e LISTER, 1999, FUKS *et al*., 2003b, MINICUCCI, 2001, NUNAMAKER *et al*., 2001, SARMENTO, 2002). Assim, a colaboração pode ser definida como o *trabalho em conjunto de uma ou mais pessoas para a realização de objetivos comuns* (Adaptado de FERREIRA, 2009, MINICUCCI, 2001).

As principais vantagens da colaboração podem ser resumidas da seguinte forma (ALEIXO, 2003, DEMARCO e LISTER, 1999, HARDINGHAM, 2000): redução no tempo necessário para a execução de tarefas; melhoria da capacidade de resolver problemas complexos; aumento da capacidade criativa para gerar alternativas; discussão das vantagens e desvantagens de cada alternativa para selecionar as viáveis e tomar decisões; melhoria na comunicação; aprendizagem; e satisfação pessoal.

Apesar dos seus potenciais benefícios, é importante determinar quando a colaboração é realmente necessária em um projeto e com que intensidade. A colaboração só deve ser adotada quando tem potencial para produzir melhores resultados do que os indivíduos trabalhando isoladamente (HANSEN, 2009). Caso contrário, a colaboração também poderá enfrentar alguns desafios (SARMENTO, 2002): lentidão na execução das tarefas de forma colaborativa; falta de coordenação do trabalho; custo elevado da tomada de decisão; e resistência no compartilhamento de conhecimento.

Em particular, o desenvolvimento de software é um processo complexo que envolve a colaboração de diversas pessoas durante um período de tempo para atingir um objetivo comum (CUGOLA e GHEZZI, 1998). Portanto, o desenvolvimento de software é um típico exemplo de trabalho colaborativo (DEMARCO e LISTER, 1999). Tal fato é reforçado pelas atuais pesquisas sobre desenvolvimento distribuído de software (CATALDO e HERBSLEB, 2008, HERBSLEB *et al*., 2005), onde a preocupação com a colaboração, em particular a coordenação e a comunicação, aparece como um fator recorrente.

3.2. Disciplina

Na literatura técnica específica de Engenharia de Software (BOEHM e TURNER, 2003), não foi encontrada nenhuma definição precisa para disciplina, ainda que se defenda a sua importância para o sucesso de um projeto. Em busca desta definição, o termo disciplina foi consultado nos dicionários. No dicionário de inglês (MERRIAM-WEBSTER, 2004), o verbete “disciplina” tem diversas definições, entre as quais nos interessam:

> “controle adquirido por forçar a obediência ou ordem”
>
> “padrão de comportamento”
>
> “autocontrole”
>
> “regra ou sistema de regras que governam a condução de uma atividade”

Já no dicionário de português (FERREIRA, 2009), destacam-se as seguintes definições:

> “Regime de ordem imposta ou livremente consentida”
>
> “Observância de preceitos ou normas”

Todas estas definições têm em comum algumas palavras-chave: controle, ordem e padrão, mas nenhuma delas expressa exatamente o que se

pretende neste trabalho. Na ausência de uma definição clara, nota-se que este termo é interpretado de forma distinta pelos modelos de desenvolvimento. A disciplina é interpretada como “conformidade com os processos” pelo desenvolvimento orientado ao planejamento e como “autocontrole” pelo desenvolvimento ágil (BOEHM e TURNER, 2003).

Assim, convencidos da necessidade de definir claramente o conceito de disciplina que será adotado como base deste trabalho e buscando alternativas para expressar o que se pretendia, foi investigado também o termo “cerimônia”, citado em alguns trabalhos relacionados (BOEHM e TURNER, 2003).

Em inglês (MERRIAM-WEBSTER, 2004), a cerimônia remete ao “ato ou séries de atos formais prescritos por um ritual, protocolo ou convenção”. Aqui novamente está presente a ideia de padrões ou protocolos. Porém, em português (FERREIRA, 2009) a palavra não tem exatamente o mesmo significado: “Reunião festiva, ou até fúnebre, de caráter solene, por ocasião de um acontecimento; solenidade”. Esta última definição acrescentou uma palavra-chave ao conjunto anterior: solenidade.

A cerimônia é definida por Booch (1993) como a quantidade de controle ou tolerância empregue no processo de desenvolvimento. Uma maior cerimônia corresponde a controles mais rígidos. Por exemplo, uma equipe pode escrever casos de uso em guardanapos e revisá-los durante o almoço. Outra equipe pode preferir preencher um documento de três páginas e investir um dia na sua revisão. Ambos os grupos escrevem e revisam os casos de uso, mas o primeiro grupo trabalha com baixa e o último com alta cerimônia.

A partir de todas estas definições, chegamos à seguinte conclusão: *a disciplina está relacionada ao nível de planejamento embutido na definição do processo de desenvolvimento de software e à rigidez do controle empregue na execução deste processo*.

3.3. Aspectos de colaboração e disciplina no desenvolvimento de software

Para justificar a escolha dos aspectos de colaboração e disciplina é necessário compreender como os aspectos de colaboração (*colab*) e disciplina (*disc*) se apresentam e variam entre os modelos de desenvolvimento de software:

- **Ambiente:** O modelo de desenvolvimento ágil é mais aplicável em ambientes turbulentos, que sofrem muitas mudanças e onde existe certo nível de incerteza técnica que requer a combinação de habilidades através do trabalho colaborativo (*colab*) dos membros da equipe. O desenvolvimento orientado ao planejamento atinge bem os seus objetivos de previsibilidade, estabilidade e confiabilidade em ambientes relativamente estáveis e com alto risco, onde a disciplina busca garantir a padronização do trabalho (*disc*) (BOEHM e TURNER, 2003). Por outro lado, o desenvolvimento de software livre se adapta melhor em ambientes onde predomina a capacidade criativa e de inovação (*colab*) (EBERT, 2007) e existe pouca sistematização do trabalho.
- **Processos:** O modelo de desenvolvimento de software orientado ao planejamento se caracteriza pela ênfase em processos bem definidos e melhorados continuamente pela organização (*disc*). Tanto no desenvolvimento ágil quanto no livre não existe uma descrição ou modelagem explícita do processo adotado. Ao invés disso, ambos os modelos lidam com um conjunto de princípios gerais para nortear o desenvolvimento de software (KOCH, 2004) (*disc*).
- **Pessoas:** No desenvolvimento ágil (COCKBURN e HIGHSMITH, 2001, COCKBURN, 2001, MILLER, 2001) e livre (CUBRANIC e BOOTH, 1999), o foco está nas pessoas, nos seus talentos, habilidades e conhecimentos, e na colaboração entre elas (*colab*). Os desenvolvedores de software livre são pessoas altamente motivadas e comprometidas, que se voluntariam espontaneamente. No desenvolvimento ágil, os membros da equipe também precisam ser motivados e comprometidos para lidar com a quantidade de conhecimento tácito e com as necessidades de comunicação e interação freqüentes (*colab*). Assim, os modelos de desenvolvimento ágil e livre são mais dependentes do talento humano do que o orientado ao planejamento, onde a explicitação do conhecimento e a padronização do trabalho diminuem estes riscos (*colab e disc*).
- **Comunicação:** O modelo de desenvolvimento orientado ao planejamento, a comunicação, entre os membros da equipe de desenvolvimento e entre a equipe e o cliente, é formal e baseada em registros (*colab e disc*). O desenvolvimento ágil e o livre se baseiam na comunicação informal entre

os participantes do projeto. No desenvolvimento do software livre, que opera com uma equipe geograficamente distribuída, essa comunicação informal é feita através da Internet (*colab*) (YAMAUCHI *et al*., 2000). Já no desenvolvimento ágil, orientado a equipes de desenvolvimento pequenas e fisicamente próximas, a comunicação é face-a-face (COCKBURN, 2001) (*colab*).

- **Coordenação:** O modelo orientado ao planejamento utiliza uma estrutura de coordenação baseada no comando e controle (*colab* e *disc*). Por outro lado, o modelo de desenvolvimento de software livre adota um modelo de liderança meritocrático, baseado no reconhecimento e reputação na comunidade, onde um ou mais indivíduos contam com a confiança e respeito do grupo para coordenar informalmente o trabalho, principalmente em relação à seleção e integração de contribuições no código. A participação dos desenvolvedores é voluntária e, portanto, não existe uma divisão formal ou alocação prévia do trabalho (CUBRANIC e BOOTH, 1999, TAURION, 2004) (*colab* e *disc*). No desenvolvimento ágil, o papel convencional do gerente de projeto como planejador, organizador e controlador é substituído por um papel de facilitador do trabalho de uma equipe que se auto-direciona e se auto-organiza (BECK *et al*., 2001) para lidar com o trabalho baseado em um objetivo comum (*colab*).
- **Memória:** O desenvolvimento orientado ao planejamento utiliza a documentação como artefato de comunicação formal entre os membros do projeto, como instrumento para garantir a rastreabilidade dos produtos gerados entre as fases do desenvolvimento e como mecanismo de explicitação do conhecimento do projeto (BOEHM e TURNER, 2003, CHAU *et al*., 2003) (*colab e disc*). Já os métodos ágeis encorajam a parcimônia na documentação (WARSTA e ABRAHAMSSON, 2003). No modelo de software livre a disponibilização livre das informações é muito importante para que os desenvolvedores possam contribuir mais eficientemente (TAURION, 2004). Assim, grande parte do conhecimento sobre o produto é tácito e compartilhado entre os membros do projeto através da comunicação e da interação (CHAU *et al*., 2003) (*colab*).
- **Relacionamento com usuários e clientes:** Os usuários e clientes desempenham um papel fundamental nos três modelos de

desenvolvimento, mas com atuações diferentes. No desenvolvimento de software livre, os usuários são os próprios desenvolvedores ou participam ativamente informando problemas e discutindo funcionalidades (*colab*) (RAYMOND, 2001). No desenvolvimento ágil, os usuários e clientes priorizam os requisitos em conjunto com a equipe (*colab*), tiram dúvidas, analisam as soluções propostas e fornecem *feedback* rapidamente sobre o software entregue em cada iteração (COCKBURN, 2001), ajudando assim na disseminação do conhecimento sobre o domínio de negócio dos usuários e clientes para a equipe de desenvolvimento (CHAU *et al*., 2003). No modelo orientado ao planejamento, o cliente também desempenha um papel importante, mas somente durante as fases de especificação de requisitos e homologação (*colab*). Além disso, o relacionamento se caracteriza por um baixo nível de confiança e é governado por definições contratuais (*disc*) (GLAZER *et al*., 2008).

A análise destes fatores, resumidos pela Tabela 3.1, permite observar que o desenvolvimento de software, nos seus diversos modelos, possui características colaborativas. Em relação à disciplina, nota-se uma variação na ênfase, na forma e no formalismo dos processos, ainda que todos se preocupem com algum nível de controle.

Tabela 3.1 – Resumo dos aspectos de colaboração e disciplina nos modelos de desenvolvimento de software

Aspecto	Orientado ao planejamento	Ágil	Livre
Ambiente	Estável e com alto risco (*disc*)	Mudanças frequentes e incerteza técnica (*colab*)	Voltado à criatividade e inovação (*colab*)
Processos	Definidos e melhorados continuamente pela organização (*disc*)	Sem descrição ou modelagem explícita. Adota um conjunto de princípios gerais (*disc*)	Sem descrição ou modelagem explícita. Adota um conjunto de princípios gerais (*disc*)
Pessoas	Conhecimento explícito (*colab e disc*)	Foco nas pessoas e dependência do conhecimento tácito (*colab*)	Foco nas pessoas e dependência do conhecimento tácito (*colab*)
Comunicação	Formal e baseada em registro (*colab e disc*)	Informal e face-a-face (*colab*)	Informal e distribuída (*colab*)
Coordenação	Comando e controle (*colab e disc*)	Facilitação do trabalho e auto-direcionamento (*colab*)	Meritocracia e voluntariado (*colab e disc*)
Memória	Conhecimento explícito (*colab e disc*)	Parcimônia e conhecimento tácito (*colab*)	Livre acesso e conhecimento tácito (*colab*)
Relacionamento com cliente	Participação limitada (*colab*) e	Clientes chegam a ser parte da equipe de	Usuários são os próprios desenvolvedores ou têm

Aspecto	Orientado ao planejamento	Ágil	Livre
	relacionamento baseado em restrições contratuais (*disc*)	desenvolvimento ou têm participação ativa (*colab*)	participação ativa (*colab*)

3.4. Colaboração e disciplina

Ainda que a adaptação de processos possa ser feita sob diferentes aspectos (SHARP e MCDERMOTT, 2001), conforme visto na seção anterior, a colaboração e a disciplina são os aspectos que se destacam quando se observa as características dos diferentes modelos de desenvolvimento de software.

A **colaboração** é um fator importante para que as organizações de software possam alcançar seus objetivos de produtividade, qualidade e compartilhamento de conhecimento (MAGDALENO *et al*., 2007). Por outro lado, a **disciplina** impõe ordem, sistematizando o trabalho, evitando o caos e os sucessos dependentes dos talentos humanos individuais.

Assim, tanto a colaboração quanto a disciplina são necessárias nos projetos de desenvolvimento de software. Porém, estes projetos têm necessidades diferentes de colaboração e disciplina. Como um pêndulo, os projetos podem variar entre estes dois pólos. Alguns projetos se situam no extremo da colaboração, outros se encontram no extremo oposto da disciplina. Existem ainda os projetos que se encaixam exatamente no meio, ou seja, têm uma combinação igual de colaboração e disciplina. Por fim, existem os projetos que precisam de misturas variadas de colaboração e disciplina.

Em suma, a colaboração e disciplina são complementares e essenciais para qualquer projeto de desenvolvimento de software, mas em diferentes proporções, dependendo do contexto do projeto, da organização e da equipe. A partir desta ideia, a primeira sub-hipótese (H1.1) investigada neste trabalho é:

É possível balancear os aspectos de colaboração e disciplina na tomada de decisão sobre a adaptação de processos de desenvolvimento de software.

Esta ideia do balanceamento será aprofundada ao longo dos próximos capítulos, mas para que ela possa ser progressivamente construída, é necessário que primeiro sejam definidos os instrumentos que serão utilizados

para explicitar a colaboração e a disciplina, conforme apresentado na próxima seção.

3.5. Instrumentos para Colaboração e Disciplina

Conforme visto no capítulo 2, a adaptação inicial do processo é feita pelo gerente como parte das atividades de planejamento do projeto. Neste momento, portanto, devem ser planejados os níveis de colaboração e disciplina necessários ou desejados para o projeto.

Além disso, a adaptação do processo deve ser revista, durante a execução do projeto de desenvolvimento de software devido às mudanças que vão acontecendo no contexto. Neste momento, é preciso avaliar se os níveis planejados de colaboração e disciplina ainda se mantêm os mesmos e, se for o caso, rever o planejamento feito anteriormente. Também é necessário verificar quais são os níveis reais de colaboração e disciplina no projeto. Para apoiar este planejamento e acompanhamento dos níveis de colaboração e disciplina, alguns instrumentos são necessários.

O nível de colaboração pode ser definido e acompanhado com base no modelo de maturidade em colaboração (**CollabMM**) (MAGDALENO *et al*., 2009), que descreve um caminho de evolução progressiva através de níveis de maturidade em colaboração. Cada nível de colaboração é composto por práticas para explicitar e ampliar a colaboração, organizadas de acordo com os aspectos de apoio à colaboração (comunicação, coordenação, memória e percepção). O papel do CollabMM nesta proposta de tese é funcionar como um *framework* que define os estágios da colaboração e resume suas principais características.

Além do CollabMM, também é necessário um mecanismo que ajude a explicitar e medir a colaboração existente entre as pessoas nos projetos de desenvolvimento de software. Para oferecer esta compreensão, a análise de **redes sociais** se apresenta como um caminho promissor, pois existem diversos trabalhos (GAO *et al*., 2003, GOTO *et al*., 2008, JIN XU *et al*., 2005, LOPEZ-FERNANDEZ *et al*., 2004, MADEY *et al*., 2002) que apontam para o potencial das redes sociais em explicitar como a colaboração acontece dentro de um grupo.

Araujo e Borges (2007) defendem que ao explicitar a colaboração, aumenta-se a sua visibilidade, de forma que os membros da organização atinjam maior compreensão e se motivem. Desta forma, o entendimento das redes sociais envolvidas nos projetos de desenvolvimento pode ajudar a compreender e acompanhar o nível de colaboração do projeto.

Por sua vez, o nível de disciplina é estabelecido através de uma abordagem de **medição**. A necessidade da medição é decorrente da premissa de que "não se pode controlar o que não se pode medir" (DEMARCO, 1986). A medição é um importante mecanismo de visibilidade em um projeto e permite aumentar o conhecimento sobre os processos em andamento.

Cada um dos instrumentos mencionados é detalhado nas seções subsequentes, com exceção das redes sociais que, também são objeto de pesquisa deste trabalho e, portanto, tema do próximo capítulo.

3.5.1. Modelo de Maturidade em Colaboração (CollabMM)

O modelo de maturidade em colaboração (CollabMM) (MAGDALENO *et al*., 2007, 2009, 2008a, 2008b) tem como objetivo organizar e apresentar as principais práticas, existentes na literatura e adotadas nas organizações, que podem ser aplicadas à modelagem de processos para explicitar e ampliar a colaboração. O CollabMM organizou o corpo de conhecimento sobre colaboração em quatro aspectos de apoio (comunicação, coordenação, memória e percepção), detalhados na próxima seção.

3.5.1.1. Aspectos de Apoio à Colaboração

A colaboração pode ser entendida sob quatro aspectos de apoio (comunicação, coordenação, memória e percepção) (MAGDALENO, 2006) que são resumidos pelo esquema apresentado na Figura 3.1. Estes aspectos não podem ser considerados isoladamente, pois se encontram intimamente dependentes e relacionados entre si. Para colaborar, os indivíduos têm que trocar informações (*comunicação*), organizar-se (*coordenação*) e operar em conjunto em um espaço de trabalho coletivo (*memória*). Através da *percepção*, o indivíduo se informa sobre o que está acontecendo e adquire as informações necessárias (ARAUJO, 2000, DIAS, 1998, FUKS *et al*., 2003b). Os aspectos de apoio à colaboração são detalhados nas próximas seções.

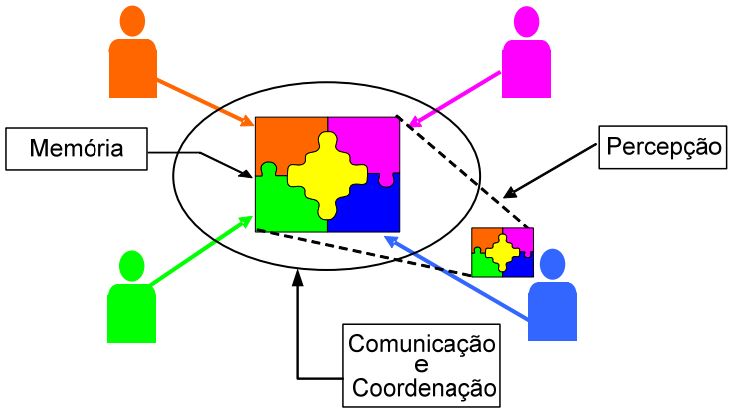


Figura 3.1 – Esquema geral dos aspectos de apoio à colaboração (ARAUJO, 2000)

a) Comunicação

O primeiro obstáculo à colaboração é vencer a distância entre os membros do grupo, ou seja, estabelecer a comunicação entre as partes envolvidas para que elas possam trabalhar em grupo (ARAUJO, 2000). A comunicação é um fator essencial para o bom desempenho dos grupos e para a satisfação dos membros que os integram. As pessoas precisam se comunicar para organizar o trabalho, designar tarefas, tomar decisões e resolver problemas. Por isso, os membros de um grupo precisam interagir regularmente.

Dentro de um grupo, a comunicação pode ser realizada em tempo real (síncrona) ou pode ser realizada em momentos diferentes (assíncrona). Na forma síncrona, os interlocutores estão presentes simultaneamente e disponíveis e a mensagem enviada é recebida imediatamente. Na comunicação assíncrona, o tempo é mais flexível: a mensagem enviada pode ser recebida em um momento posterior indeterminado (ELLIS *et al.*, 1991, FUKS *et al.*, 2003a).

A comunicação assíncrona normalmente é utilizada quando se deseja valorizar a reflexão dos participantes, pois estes terão mais tempo antes de agir. Já na comunicação síncrona, valoriza-se a velocidade da interação, visto que o tempo de resposta entre a ação de um participante e a reação de seus companheiros é curto (FUKS *et al.*, 2003a).

b) Coordenação

O trabalho em grupo demanda um esforço adicional para a coordenação de seus membros, ou seja, é necessário um trabalho de articulação para que colaboração possa ser obtida a partir da soma dos trabalhos individuais. A coordenação organiza o grupo para evitar que esforços de comunicação sejam

perdidos e para que as tarefas sejam realizadas na ordem correta, no tempo exato e cumprindo as restrições e objetivos, a fim de evitar que os participantes se envolvam em tarefas conflitantes ou repetitivas (FUKS *et al*., 2003b, GEROSA *et al*., 2003).

A coordenação envolve a pré-articulação das tarefas, o acompanhamento e a pós-articulação. A pré-articulação são as ações necessárias para preparar a colaboração: identificação dos objetivos, mapeamento destes objetivos em tarefas, seleção dos participantes e a distribuição das tarefas entre eles. O acompanhamento significa controlar a execução das atividades durante o processo, para garantir a produtividade e o sucesso dos objetivos do grupo. A pós-articulação envolve a avaliação das tarefas realizadas e a documentação da colaboração realizada (ARAUJO, 2000, DIAS, 1998, FUKS *et al*., 2003b).

Por último, a coordenação visa manter a "vida do grupo", através de estímulos às contribuições de cada participante, agendamento de eventos e estabelecimento de um ritmo aos trabalhos e aos encontros. Para isso, deve-se estar atento à interrupção brusca da participação de um determinado membro, à falta de referências ou contribuições ao trabalho de um membro do grupo e à argumentação muito longa entre dois ou mais participantes sobre um mesmo tópico (SANTORO, 2001).

c) Memória de grupo

Comunicação e coordenação, apesar de vitais, não são suficientes para a colaboração. Os grupos de trabalho também precisam organizar e compartilhar diferentes tipos de informações relacionadas às atividades sendo realizadas (ELLIS *et al*., 1991). A memória de grupo é o armazenamento dos dados relativos ao desenvolvimento da atividade colaborativa.

Em geral, a gestão de conhecimento nas organizações se concentra na preservação do histórico e organização dos artefatos (conhecimento formal). Entretanto, os participantes compartilham não só artefatos durante as interações, como também ideias e pensamentos sobre o trabalho sendo produzido, mas este conhecimento informal é difícil de ser capturado (NONAKA e TAKEUCHI, 1995).

O conhecimento informal sobre o processo de criação dos produtos,

compreendendo o registro das ideias, fatos, questões, pontos de vista, conversas, discussões e decisões que aconteceram no decorrer do trabalho - deve estar intimamente relacionado aos artefatos produzidos. Ele permite recuperar o histórico da discussão e o contexto em que as decisões foram tomadas (ARAUJO, 2000, DIAS, 1998, FUKS *et al*., 2003b).

d) Percepção

A percepção é o entendimento das atividades dos outros para fornecer contexto a sua própria atividade (DOURISH e BELLOTTI, 1992). Através da percepção, os indivíduos podem tomar ciência do objetivo comum, do papel de cada um dentro do grupo, do que fazer, como proceder, qual o resultado das suas ações, até onde atuar, quem está por perto, ou seja, podem adquirir as informações necessárias para o seu trabalho (GUTWIN e GREENBERG, 1999).

O fenômeno de criação em grupo se dá progressivamente através da geração de novas ideias baseadas nas contribuições já elaboradas pelo grupo. A percepção das contribuições já trazidas ao grupo ajuda a compreender como os resultados gerados pelas atividades alheias podem ser conjugados aos seus, para que os objetivos esperados sejam alcançados mais rapidamente. Desta forma, cada membro do grupo pode oferecer suas contribuições com maior segurança quanto à necessidade e relevância para o produto comum (ARAUJO, 2000, DOURISH e BELLOTTI, 1992).

A percepção também é essencial para a coordenação do grupo (DOURISH e BELLOTTI, 1992). Cada membro deve conhecer o progresso do trabalho dos companheiros: o que foi feito, como foi feito, o que falta para o término, quais são os resultados preliminares, etc. Tendo percepção das atividades dos companheiros, as pessoas terão informações para auxiliar na sincronização do trabalho, de forma que seus esforços individuais agreguem valor ao trabalho do grupo (GEROSA *et al*., 2003). Quando os membros não têm conhecimento sobre o que está sendo desenvolvido pelos outros, o trabalho resultante pode não apresentar coesão e não representar as ideias do grupo como um todo (SANTORO, 2001).

3.5.1.2. Níveis de Maturidade em Colaboração

O CollabMM descreve um caminho de evolução progressiva através de quatro níveis de maturidade em colaboração: casual, planejado, perceptivo e reflexivo (Figura 3.2). A organização em níveis de maturidade é uma forma de ajudar as organizações a priorizarem as ações. Para se alcançar um determinado nível, é necessário satisfazer este nível e os seus níveis inferiores. Conforme visto anteriormente, a proposta de organização do CollabMM em níveis de colaboração é particularmente útil para o enfoque de solução deste trabalho.

Ao todo o modelo descreve 12 (doze) práticas de colaboração distribuídas em três níveis de maturidade: planejado (4), perceptivo (4) e reflexivo (4) conforme a Figura 3.2. Não existem práticas para o nível casual, uma vez que neste nível não se espera que a colaboração esteja formalmente estabelecida na organização.

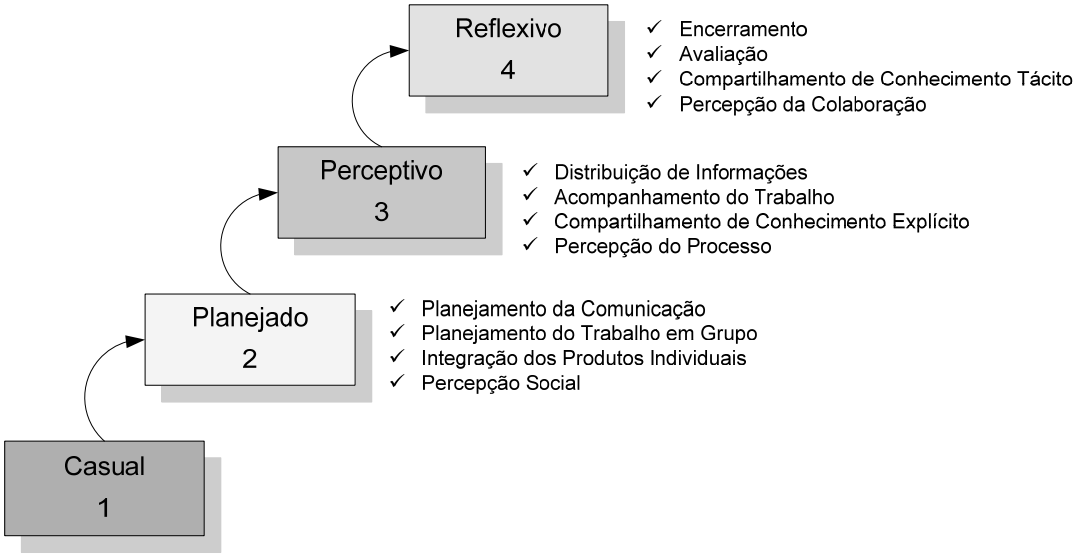


Figura 3.2 – Modelo de Maturidade em Colaboração (CollabMM)

a) Nível 1 – Casual

No nível casual a colaboração ainda não está explícita. Contudo, as organizações no nível 1 não se caracterizam pela total ausência de colaboração. Como a tendência natural do ser humano é trabalhar de forma cooperativa, a colaboração acontece, mas como uma prática isolada e dependente do relacionamento ou da afinidade existentes entre as pessoas. Assim, no nível casual a colaboração ainda é resultado do esforço individual (Figura 3.3). Os aspectos de comunicação, coordenação, memória e percepção estão presentes, mas são tratados de maneira *ad-hoc*.

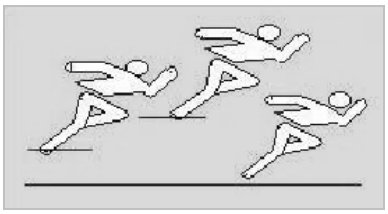


Figura 3.3 – Metáfora do esforço individual (NUNAMAKER *et al*., 2001)

b) Nível 2 – Planejado

A partir deste nível, os processos da organização começam a sofrer modificações de forma a incluir atividades básicas de colaboração. Estas atividades incluem principalmente aspectos de planejamento da colaboração. Este planejamento inclui formalizar os grupos, papéis e responsabilidades e definir os canais de comunicação apropriados aos membros do grupo.

A coordenação é um aspecto forte deste nível e acontece de forma centralizada em um membro que assume o papel de líder (Figura 3.4) e age de acordo com a sua experiência. O líder estimula o comprometimento e a moral do grupo encorajando os membros e comunicando claramente os papéis e responsabilidades. Este líder é o responsável pelo planejamento do trabalho e distribuição de tarefas entre os participantes do grupo. Cabe ao líder balancear a carga de trabalho dos participantes do grupo ao mesmo tempo em que procura aproveitar as habilidades e talentos específicos de cada um.

Os indivíduos já são capazes de se reconhecerem como parte integrante do grupo, conseguem compreender a formação deste grupo e conhecer os seus parceiros de trabalho. Com as informações obtidas sobre os participantes, o indivíduo já consegue estabelecer as conexões sociais ajudando a criar solidariedade, entendimento, confiança, respeito e compromisso no âmbito de cada grupo do qual participe (ARAUJO, 2000, SANTORO, 2001).

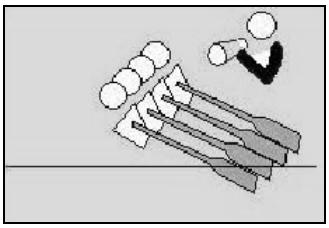


Figura 3.4 – Metáfora do esforço coletivo coordenado (NUNAMAKER *et al*., 2001)

Também é importante garantir que os produtos do grupo não sejam apenas resultados de esforços individuais isolados e nem uma simples composição de trabalhos distintos. Mesmo que exista a divisão de tarefas entre os colaboradores visando gerar maior rapidez na execução de uma atividade,

os artefatos devem ser integrados, de forma a se complementarem como parte de um todo consistente, entendido por todo o grupo (SANTORO, 2001).

c) Nível 3 – Perceptivo

No nível perceptivo, os membros do grupo conhecem as suas responsabilidades e sabem quais atividades executar de forma que o grupo consiga alcançar os seus objetivos. Assim, não é mais necessária uma coordenação centralizada do líder. Por outro lado, é necessário que se garanta ao grupo os recursos necessários para acessar as informações e entender a dependência e a articulação das suas atividades (Figura 3.5).

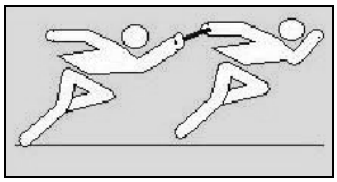


Figura 3.5 – Metáfora do esforço coletivo auto-direcionado (NUNAMAKER *et al*., 2001)

Em relação à comunicação, o que se deseja é garantir que os membros do grupo terão acesso às informações necessárias de forma adequada, respeitando-se o planejamento das comunicações realizado.

Os membros do grupo compreendem o processo do trabalho que irão realizar, entendem seus objetivos, estão conscientes dos passos necessários para alcançar estes objetivos e possuem os conhecimentos necessários para executar as tarefas.

O conhecimento explícito (NONAKA e TAKEUCHI, 1995) é compartilhado, na forma de artefatos, entre os membros do grupo. Para apoiar este compartilhamento, são claramente definidos os artefatos manipulados e o repositório para armazenamento desta documentação.

d) Nível 4 – Reflexivo

As organizações que atingem o nível reflexivo de maturidade nos seus processos são aquelas que percebem o valor do conhecimento que está sendo gerado no trabalho dos grupos e se preocupam em geri-lo e disseminá-lo dentro da própria organização (Figura 3.6). Para isso, os processos já incluem atividades de avaliação e divulgação dos resultados dos trabalhos dos grupos.

Neste nível, o *feedback* sobre a realização dos objetivos é importante tanto para a análise dos resultados individuais quanto para os resultados do grupo como um todo. Nos resultados do grupo, deve-se levar em consideração

tanto a qualidade dos produtos gerados quanto o processo colaborativo.

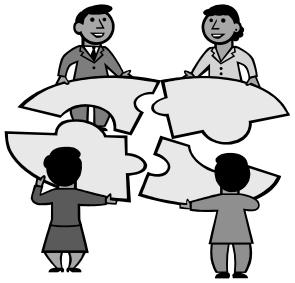


Figura 3.6 – Metáfora do esforço coletivo disseminado

Os participantes devem ter clareza sobre como a colaboração acontece na execução do processo, ou seja, como suas atividades interagem entre si, de forma que cada um possa pautar suas próprias contribuições.

Os membros do grupo participam do encerramento oficial do trabalho, celebrando os resultados alcançados e divulgando-os para o restante da organização. Neste momento, aproveita-se para: capturar as lições aprendidas; analisar as forças e fraquezas do trabalho realizado; compartilhar sucessos e problemas; e extrair ideias para melhorias futuras.

Neste nível, o conhecimento tácito (NONAKA e TAKEUCHI, 1995) também passa a ser compartilhado, na forma de ideias, opiniões e experiências, entre os membros do grupo.

3.5.2. Medição

A necessidade da medição é decorrente da premissa de que “não se pode controlar o que não se pode medir” (DEMARCO, 1986). Dentre as principais motivações para se medir software, estão (BRIAND *et al*., 1996, ESTOLANO, 2005):

- **Compreender** e **caracterizar** processos e produtos de software, com o objetivo de definir o estado ou desempenho atual do processo ou produto de software;
- **Monitorar** para observar tendências e a evolução do desempenho ou do estado dos produtos e processos de software através dos projetos de uma organização;
- **Avaliar** o alcance de objetivos de qualidade, o impacto de tecnologias em produtos e processos e os aspectos onde as tecnologias precisam ser melhoradas ou adaptadas a um contexto específico;
- **Predizer** atributos externos relevantes, utilizando os relacionamentos

existentes entre características dos processos, produtos e recursos;

- **Controlar** a execução de um projeto de software de modo a minimizar riscos, utilizando as relações causais entre características de processos, produtos e recursos;
- **Melhorar** o processo de modo a melhorar a qualidade e a produtividade nos projetos futuros, com base nos relacionamentos causais que influenciem o estado ou desempenho dos processos e/ou produtos.

Para alcançar esses objetivos, aspectos relevantes dos processos, produtos e projetos devem ser observados e medidos. A medição é o "processo através do qual valores (símbolos ou números) são atribuídos a atributos de entidades do mundo real, de modo a descrevê-los de acordo com regras claramente definidas" (ROBERTS, 2009). Ou, colocando de outra forma, a medição corresponde a um "conjunto de operações que tem como objetivo determinar o valor de uma medida" (ISO/IEC, 2007).

O principal objetivo da medição é apoiar as necessidades de informações gerenciais (CHRISSIS *et al*., 2006). Neste sentido, pode-se considerar que a medição favorece a tomada de decisão, pois esta pode ser feita com base em informações reais do projeto e históricas da própria organização.

A medição é um mecanismo para criar memória organizacional e um instrumento que ajuda a responder uma variedade de questões associadas à execução de um processo (BASILI *et al*., 1994). Ela apóia o planejamento do projeto (Quanto o novo projeto irá custar?); permite determinar as forças e fraquezas dos processos e produtos atuais (Qual é a frequência de certos tipos de erros?); fornece o *rationale* para refinar ou adotar alguma técnica (Qual o impacto desta técnica na produtividade dos projetos?); permite avaliar a qualidade de processos e produtos específicos (Qual é a densidade de defeitos em um sistema após a sua implantação?); ajuda a avaliar o progresso durante a sua execução para que ações corretivas possam ser tomadas com base nesta avaliação e ainda ajuda a avaliar o impacto de cada ação.

A importância da medição de produtos e processos é reconhecida também pela comunidade de pesquisa em desenvolvimento ágil (HARTMANN e DYMOND, 2006). Porém, neste ambiente existe uma preocupação em manter o alinhamento com os princípios e valores dos métodos ágeis e,

portanto, existem alguns cuidados que devem ser observados ao definir métricas para um processo de desenvolvimento ágil.

No nível organizacional, os objetivos de medição são estabelecidos e mantidos a partir dos objetivos de negócio e necessidades de informação organizacionais. Os objetivos de medição documentam os propósitos para os quais as medições e análises são feitas.

A partir dos objetivos de medição, devem ser definidas as métricas[7] capazes de satisfazê-los. As métricas selecionadas pela organização devem ser documentadas pelo nome, unidade de medida, descrição e relação com os objetivos de medição. A documentação de cada métrica também deve incluir a definição dos procedimentos de coleta, armazenamento e análise de dados (SOFTEX, 2009). Também é importante que se adote um modelo de medição, por exemplo, baseado no PSM (*Practical Software Measurement*) (MCGARRY *et al*., 2001) ou no GQM (*Goal-Question-Metric*) (BASILI *et al*., 1994, SOLINGEN e BERGHOUT, 1999).

Assim como o processo padrão da organização pode ser adaptado para o processo específico do projeto, o planejamento da medição da organização, também pode ser adaptado no nível do projeto, desde que seja preservada a consistência entre os dois níveis. O planejamento da medição no projeto compreende a definição de: (i) objetivos de medição específicos do projeto alinhados aos objetivos de medição da organização; (ii) um conjunto de métricas adequado ao projeto e orientado pelos objetivos de medição; e (iii) nível de controle que se deseja estabelecer sobre essas métricas.

A partir dos objetivos de medição da organização, um projeto pode definir objetivos específicos. Os objetivos do projeto inicialmente são identificados dentre os objetivos de medição da organização buscando aqueles que são aplicáveis ao projeto. Em princípio, todos os objetivos de medição da organização podem ser considerados também como objetivos do projeto, mas existem situações específicas que fazem com que algum objetivo não seja aplicável. Cabe ao gerente de projeto avaliar a pertinência de cada objetivo da organização para identificar se existem objetivos que podem ou devem ser excluídos sem prejuízo para o projeto (BARRETO e ROCHA, 2009).

[7] Não existe um consenso na literatura para esta nomenclatura e algumas referências adotam o termo medida e métrica como sinônimos. Neste trabalho optou-se por adotar a nomenclatura de métricas.

A definição dos objetivos do projeto é importante, pois também considera as necessidades específicas dos projetos e seus requisitos podendo originar novos objetivos do projeto, independentemente dos objetivos da organização. Desta forma, a implementação dos objetivos da organização é assegurada nos projetos, mas esta não restringe os projetos, permitindo que as necessidades específicas sejam atendidas (BARRETO e ROCHA, 2009).

Ainda no planejamento da medição no nível do projeto, é preciso avaliar, dentre os objetivos selecionados, se todas as métricas são relevantes para o projeto em questão. Neste momento, também podem ser excluídas métricas que não sejam pertinentes ao projeto em questão ou incluídas novas métricas que se façam necessárias.

Posteriormente, o planejamento elaborado precisará ser acompanhado. Assim, é importante planejar também como os objetivos serão monitorados. Para isso, a cada objetivo definido são associados indicadores, que tratam informações quantitativas e qualitativas sobre o desempenho do projeto. Estes indicadores vão permitir a detecção de desvios e a tomada de decisão para a execução de ações corretivas (BARRETO e ROCHA, 2009).

Para estabelecer o nível de controle desejado e permitir a análise de cada indicador de forma a detectar se a execução do processo está em conformidade com o comportamento esperado, também é importante a definição de metas para cada indicador (BARRETO e ROCHA, 2009).

3.6. Considerações finais

Este capítulo definiu os conceitos de colaboração e disciplina e mostrou como ambos estão presentes, ainda que com diferentes ênfases, e são importantes para o desenvolvimento de software. Aliás, talvez esta seja uma das poucas unanimidades entre os modelos de desenvolvimento de software. Nenhum dos trabalhos estudados na revisão da literatura apresentou argumentos contrários à colaboração. Ao mesmo tempo, ainda que se discuta a forma e o formalismo dos processos, todos os modelos de desenvolvimento também reconhecem a necessidade de algum nível de disciplina e controle.

Este capítulo também apresentou a sub-hipótese para apoio à decisão na adaptação do processo de desenvolvimento de software através do

balanceamento entre colaboração e disciplina. Para investigar esta hipótese é preciso primeiro explicitar e medir a colaboração e a disciplina. Neste sentido, estão sendo considerados alguns instrumentos, tais como: o modelo de maturidade em colaboração (CollabMM); as redes sociais; e a medição.

Ainda que todos estes instrumentos sejam usados na solução proposta por este trabalho, a contribuição desta pesquisa de tese refere-se, principalmente, ao aspecto da colaboração. As técnicas de medição serão apenas aplicadas como meio para possibilitar o uso das informações de disciplina. Assim, como o CollabMM já foi construído em um trabalho de pesquisa anterior e validado até mesmo em contextos reais (MAGDALENO *et al*., 2008a, 2008b), este trabalho se dedicou a um estudo mais aprofundado das redes sociais que são tratadas no próximo capítulo.

4. Redes Sociais

Este capítulo apresenta o conceito de redes sociais, descrevendo uma metodologia para trabalhar com essas redes e aprofundando em mais detalhes as fases de visualização e análise. Em seguida, a partir de um estudo realizado sobre ferramentas de redes sociais, foram identificados os principais requisitos desejáveis a este tipo de ferramenta. Estes requisitos motivaram a criação da ferramenta EvolTrack-SocialNetwork cuja arquitetura é apresentada ao final deste capítulo.

De forma simplificada, uma **rede** é "um conjunto de pontos interligados" (WWF, 2003). Formalizando esta definição, pode-se dizer que uma rede é um "conjunto de vértices ou nós com conexões entre eles, denominadas arestas" (NEWMAN, 2003). Para ilustrar estas definições, dois exemplos de redes, com representações distintas, são apresentados na Figura 4.1.

Em um diagrama de rede, como os exemplos apresentados na Figura 4.1, há necessariamente pontos e linhas. Os pontos representam as unidades que compõem a rede: pessoas, organizações, equipamentos, locais e etc. e também são denominados de nós ou vértices. As linhas representam as relações entre esses elementos e podem ser canais de comunicação, estradas, dutos, fios e etc. Estas linhas recebem o nome de relacionamentos, arestas, ligações ou conexões (WWF, 2003).

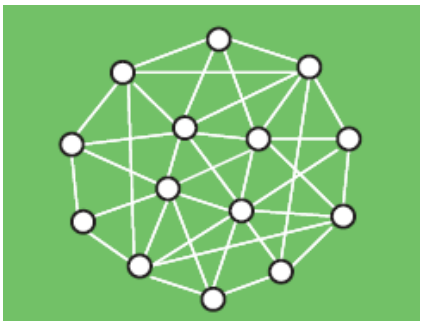
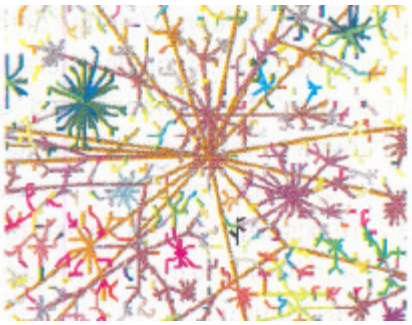
(a)  (b) 

Figura 4.1 – Exemplos de redes (BARABASI e BONABEAU, 2003, WWF, 2003)

Uma **rede social** consiste de um "conjunto finito de atores e as relações definidas entre eles" (WASSERMAN e FAUST, 1994). Em uma rede social, os nós representam os atores e as arestas correspondem aos possíveis relacionamentos entre eles. A semântica do relacionamento depende da análise que se deseja conduzir nesta rede.

Alguns exemplos de redes sociais, com semânticas distintas, são (ALBERT e BARABASI, 2002, BARABASI e BONABEAU, 2003): rede de relações de amizade; casamentos entre famílias; comunidades de negócio; rede de contatos sexuais, usada para estudar a propagação de doenças

sexualmente transmissíveis entre indivíduos; rede de colaboração entre atores que contracenaram juntos em filmes; rede de colaboração científica representada através da co-autoria em artigos acadêmicos; e a rede de comunicação (ligações telefônicas, e-mail ou *instant messaging*), com arestas direcionadas indicando o autor e o receptor da mensagem.

Analisando todos estes exemplos apresentados, é possível perceber que, embora a forma seja um fator decisivo, o desenho da rede não é suficiente para explicá-la ou caracterizá-la em relação as suas propriedades e dinâmicas (WWF, 2003). Assim, é necessário entender também as propriedades topológicas, originadas da teoria dos grafos, que governam estas redes.

4.1. Propriedades topológicas

As redes sociais são baseadas na construção teórica da sociologia (DEGENNE e FORSE, 1999) e na fundamentação matemática da teoria de grafos (SZWARCFITER, 1986). Grafos ou redes têm propriedades topológicas que explicam a sua estrutura. Nesta seção serão apresentadas algumas propriedades topológicas que foram selecionadas devido a sua relevância para este trabalho. O entendimento destas propriedades topológicas, e das leis e fenômenos que as caracterizam, é fundamental para a tarefa de análise das redes sociais.

Formalmente, uma rede é representada através de um grafo G composto por dois conjuntos: um conjunto (V) de objetos (vértices) e um conjunto (E) de pares relacionados (arestas). Assim, dado o grafo G apresentado na Figura 4.2, temos que V = {a, b, c, d, e, f}; e E = {(a,c), (b,c), (b,d), (b,f), (c,d), (d,e), (e,f)}. A partir daí, podem ser observadas as seguintes propriedades:

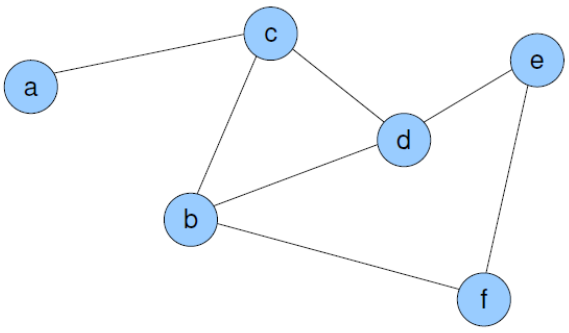


Figura 4.2 – Exemplo de rede para estudo das propriedades topológicas

- **Vértices adjacentes ou vizinhos:** dois vértices são adjacentes ou vizinhos se existe uma aresta entre eles. Assim, considerando o exemplo da Figura 4.2, é possível afirmar que o vértice *a* é vizinho de *c*, pois existe uma aresta ligando-os. Porém, o vértice *d* não é vizinho do vértice *f*, pois eles não estão diretamente relacionados.
- **Grau:** o grau de um vértice (g(v)) corresponde ao número total de arestas que este vértice possui. Nem todos os vértices de uma rede têm o mesmo número de arestas. Por exemplo, na Figura 4.2, alguns vértices têm um grau pequeno, enquanto outros têm um grau mais alto. Assim, g(a) = 1, pois o vértice *a* tem apenas 1 aresta com *c* e g(b) = 3, pois o vértice *b* tem arestas com outros três vértices (c, d, f).
- **Caminho:** um caminho é uma sequência de vértices de um ponto ao outro do grafo. Pode existir mais de um caminho ligando dois vértices. Então, para percorrer o grafo da Figura 4.2 de *a* até *f* é possível escolher um caminho passando por a-c-d-e-f ou passando por a-c-b-f.
- **Comprimento de caminho:** o comprimento de um caminho é calculado pelo número de arestas percorridas ao longo do caminho. Assim, no exemplo citado acima, vale ressaltar que os dois caminhos indicados têm comprimentos distintos. No primeiro caminho são percorridas 4 arestas e no segundo caminho são percorridas apenas 3 arestas para se chegar ao vértice destino. Logo, o comprimento do segundo caminho é menor.
- **Distância:** a distância corresponde ao comprimento do *menor caminho* entre dois vértices. Logo, o cálculo da distância deve levar em consideração todos os caminhos existentes para que possa ser verificado qual deles é o menor. Ainda usando o mesmo exemplo anterior, a distância entre *a* e *f* é igual a 3.
- **Subgrafo:** um subgrafo é parte de outro grafo maior. Então, na Figura 4.2, pode-se considerar que os vértices {b, c, d} formam um subgrafo.
- **Grafo completo:** um grafo é completo quando existem arestas entre todos os seus vértices, atingindo assim o número máximo de arestas possíveis. Desta forma, o grafo utilizado como exemplo na Figura 4.2 não é completo, pois faltam arestas entre alguns vértices, tais como (a, b) e (d, f).

- **Dígrafo:** um dígrafo é um grafo direcionado, ou seja, um grafo onde as arestas possuem uma direção de um vértice para o outro. Assim, o grafo apresentado na Figura 4.2 não é um dígrafo.

Apesar da importância do entendimento destas propriedades topológicas, elas sozinhas não ajudam a responder a seguinte questão: como as redes são formadas? A resposta a esta indagação vem através da compreensão dos modelos de redes, apresentados na próxima seção.

4.2. Modelos de evolução de redes

Esta seção apresenta os três principais modelos que podem ajudar a compreender a formação das redes.

O primeiro modelo é o de **redes randômicas** (ERDOS e RENYI, 1959), onde as arestas são colocadas randomicamente. Logo, cada par de vértices tem a mesma probabilidade de conexão e essa conexão ocorre de forma independente das demais. De acordo com este modelo, é muito raro encontrar um nó que tenha uma diferença significativa no número de relacionamentos em relação à média e a maioria dos nós possui aproximadamente o mesmo número de arestas.

No modelo de **redes livres de escala** (ALBERT e BARABASI, 2002, ALBERT *et al*., 1999, BARABASI, 2003, BARABASI e BONABEAU, 2003), a maioria dos nós tem poucas conexões e alguns poucos nós, denominados de *hubs*, possuem uma grande quantidade de relacionamentos.

Os nós de uma rede variam em termos de sua visibilidade e poder de atração. Os nós mais visíveis e capazes de atrair conexões são os *hubs*. A visibilidade de um nó se dá pelo número de ligações que possui. A identificação de *hubs* é um dos motivadores para a atividade de análise de redes sociais, pois os *hubs* são capazes de proliferar mais rapidamente ideias, conceitos, informações, tendências, doenças e etc. Os *hubs* dominam a estrutura das redes nas quais estão presentes, criando caminhos mais curtos entre dois nós, e estão presentes em uma diversidade de redes.

A evolução das redes livres de escala é governada pela conexão preferencial (BARABASI e BONABEAU, 2003). Cada rede inicia com um determinado conjunto de nós. A partir daí, os outros nós vão sendo adicionados

e progressivamente e a rede vai se expandindo. Pelas leis da conexão preferencial, os novos nós preferem se conectar aos nós que já têm mais conexões, ou seja, a probabilidade de escolha de um determinado nó é proporcional ao número de relações que o nó já tem.

Desta forma, um *hub* mais popular passa a ter cada vez mais relacionamentos ao longo do tempo. Quanto mais conectado é um nó, maior probabilidade ele tem, pela extensão de sua rede, de fazer novas conexões. Este fenômeno, também denominado de "*rich gets richer*" (BARABASI, 2003), favorece os nós mais antigos da rede que têm mais tempo para adquirir novos relacionamentos e, portanto, mais chances de virem a se tornar *hubs*. Por outro lado, se um nó é o último a entrar na rede, nenhum outro nó ainda teve a oportunidade de se ligar a ele. Logo, este fenômeno privilegia a senioridade.

Entretanto, como nada é permanente na rede, os *hubs* mudam. Um nó simples, pela dinâmica da conectividade, pode transformar-se em *hub,* enquanto outros *hubs* morrem. Deste modo, a formação da rede vai se alterando (WWF, 2003). Neste ambiente competitivo, além da senioridade na rede, cada nó também tem o seu *fitness*, que é a medida quantitativa da sua habilidade de se manter na frente da competição pelos relacionamentos na rede. Os nós sempre competem pelas conexões, pois elas representam a sobrevivência em um mundo interconectado.

No modelo de **redes de mundo pequeno**, a distância média entre os vértices é muito pequena, mesmo para redes com milhões de vértices. A maior manifestação popular deste fenômeno é o conceito dos "seis graus de separação", descoberto por Milgram durante um experimento (ALBERT e BARABASI, 2002, NEWMAN, 2003).

Diversos autores (GAO *et al*., 2003, JIN XU *et al*., 2005, LOPEZ-FERNANDEZ *et al*., 2004, MADEY *et al*., 2002) se preocuparam em analisar a topologia das redes sociais nos projetos de desenvolvimento de software livre. Todos estes trabalhos têm em comum o fato de terem reconhecido a propriedade do mundo pequeno. Os resultados dessas análises também mostraram que a rede dos desenvolvedores de software livre não é uma rede randômica, mas sim uma rede livre de escalas com conexão preferencial entre os nós. Tal fato se explica pela inclinação dos desenvolvedores de software livre em se engajar em projetos mais populares (KRISHNAMURTHY, 2002).

Como os desenvolvedores são livres para escolher os projetos dos quais querem participar, é razoável que alguns projetos exerçam mais poder de atração do que outros.

4.3. Metodologia para estudo de redes sociais

Adaptando a abordagem proposta em (CROSS *et al*., 2004), é possível estabelecer quatro passos para a metodologia para análise de redes sociais: definição da rede que se deseja analisar, coleta dos dados necessários, visualização gráfica da rede social e análise quantitativa da rede social.

O primeiro passo é definir o objetivo da análise e estabelecer a semântica dos nós e arestas da rede que se deseja analisar. A análise da rede social será potencialmente mais benéfica em redes que tenham uma necessidade ou potencial de colaboração.

O próximo passo é a coleta de dados para a construção da rede social. Esta coleta pode ser feita através de questionários (papel, e-mail ou Internet) ou facilitada pela mineração de dados (AALST *et al*., 2005) que busca os dados dos repositórios para que eles possam ser manipulados para visualização ou análise. Por exemplo, nos projetos de desenvolvimento de software livre, é comum o uso de *parsing* nas páginas dos projetos e nas suas listas de e-mail e fóruns de discussão.

Em seguida, no passo de visualização de redes sociais, é adotada a representação visual das informações, através de diferentes formas, para diminuir a sobrecarga cognitiva do usuário e facilitar a compreensão e exploração dos dados através de imagens (VIEGAS e DONATH, 2004).

Por fim, a análise de redes sociais é uma forma de compreender a interação e a organização social de um grupo. No desenvolvimento de software, pretende-se utilizar a análise de redes sociais para entender a colaboração entre os membros da equipe de desenvolvimento.

Devido a sua complexidade e importância, os dois últimos passos da metodologia serão aprofundados nas próximas seções.

4.4. Visualização de redes sociais

A visualização transfere do sistema cognitivo para o sistema perceptivo a carga despendida por uma determinada tarefa, aproveitando assim a habilidade do ser humano no reconhecimento de padrões e estruturas em informações visuais. Portanto, o emprego desta técnica torna mais fácil e natural a tarefa de compreensão da informação apresentada.

A visualização de redes sociais permite a observação de fatos e a extração de conhecimento a partir de grafos (FREITAS *et al*., 2008). Geralmente, as redes sociais são representadas por matrizes ou grafos (WASSERMAN e FAUST, 1994). As matrizes são adotadas para o cálculo de métricas e análise das redes sociais, enquanto os grafos são utilizados para facilitar a visualização e apresentação das redes.

Os mecanismos de visualização se diferenciam pelos recursos que oferecem e pelas perspectivas que exploram. Uma perspectiva possível é a temporal. O interesse na visualização temporal de redes sociais gira em torno do entendimento da dinamicidade da rede, ou seja, como as redes se desenvolvem ou mudam no decorrer do tempo, desta forma permitindo a análise longitudinal de redes sociais.

Para isso, os atores e os relacionamentos recebem um ou mais rótulos temporais (r1 ou r2 na Figura 4.3a). Um rótulo temporal representa um intervalo de tempo em que os elementos participaram de uma rede social. Assim, os rótulos temporais podem ser usados na visualização da rede social de um determinado intervalo de tempo desejado. Por exemplo, na Figura 4.3b, optou-se por visualizar apenas a rede do instante r1, enquanto na Figura 4.3c o que está sendo observado é a rede no instante r2. O número de intervalos de tempo usados na análise da rede social e a duração destes intervalos podem ser configurados, pois depende da análise que se deseja realizar (SANTOS e SOUZA, 2008).

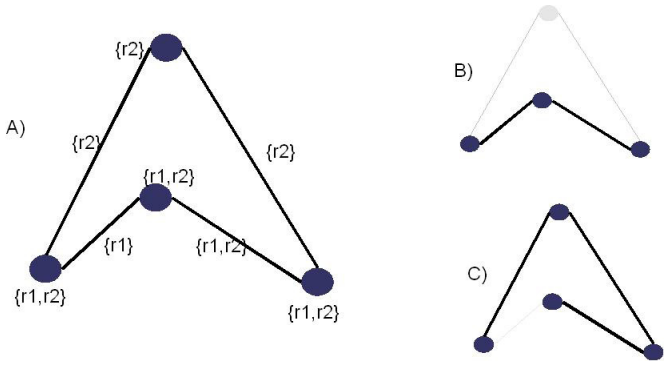


Figura 4.3 – Uso de rótulos temporais (SANTOS e SOUZA, 2008)

Na visualização temporal, os dados podem ser apresentados com histórico ou sem histórico. Na visualização sem histórico (Figura 4.4a) é utilizado um intervalo de tempo com duração constante e um incremento de tempo nos instantes inicial e final para deslocar o intervalo de visualização. Já na visualização com histórico (Figura 4.4b), o intervalo de tempo não tem duração constante, pois é feito um incremento de tempo no instante final, aumentando assim o período total de tempo exibido.

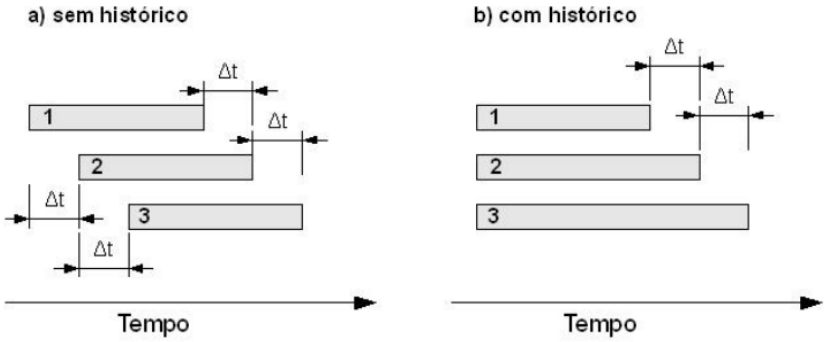


Figura 4.4 – Visualização temporal com ou sem histórico (SANTOS e SOUZA, 2008)

Além disso, a visualização temporal da rede pode usar dois tipos diferentes de layouts (SANTOS e SOUZA, 2008). A visualização por *flip books* utiliza um layout único, onde os nós permanecem em posições constantes e somente as relações mudam de um instante para outro. Este tipo de layout mantém a orientação do usuário, mas para isso deve exibir todos os nós em todos os momentos, mesmo que naquele momento alguns nós ainda não façam parte da rede.

Já na visualização *movies,* os nós movem-se de acordo com as relações que surgem com outros nós, ou seja, existe um rearranjo do layout do grafo. Isso faz com que o usuário perca a orientação visual, mas em compensação permite observar como a estrutura da rede muda no decorrer do tempo.

Outra perspectiva de visualização é a combinação de diferentes informações em uma visualização integrada. Por exemplo, Ogawa e Ma (2008) buscam entender as interações entre os desenvolvedores e os repositórios de software, através de uma visualização em formato de um anel galáctico.

A maioria das análises de redes sociais é baseada em uma única rede. Entretanto, a análise combinada de múltiplas redes sociais poderia ajudar a obter um entendimento mais amplo e descobrir como essas diferentes redes se influenciam ou estão relacionadas entre si. Tendo este objetivo em mente, outra perspectiva de visualização diz respeito à visualização de múltiplas redes sociais (SOUSA JUNIOR *et al*., 2008, SOUSA JUNIOR e DE SOUZA, 2008).

De acordo com os exemplos apresentados, é possível perceber que a visualização da rede é uma informação importante, pois oferece percepção sobre o que está acontecendo com as interações ou colaboração do grupo. Entretanto, sozinha, a visualização não permite a total compreensão da colaboração existente entre os atores da rede e precisa ser complementada pela análise das redes sociais, explorada na próxima seção.

4.5. Análise de redes sociais

A análise de redes sociais é uma abordagem metodológica, originada das ciências sociais, que utiliza os conceitos da teoria de grafos para descrever, entender e explicar as estruturas sociais (BRANDES e WAGNER, 2003). A análise de redes sociais busca entender os relacionamentos entre pessoas, grupos e organizações. A unidade na análise de redes sociais não é o indivíduo, mas sim o grupo de indivíduos e os relacionamentos entre eles (WASSERMAN e FAUST, 1994).

As análises de redes sociais já foram exploradas em contextos sociais, como a defesa do meio ambiente (WWF, 2003) e organizacionais (CROSS *et al*., 2004). Como o desenvolvimento de software é um clássico exemplo de rede social colaborativa, diversas análises de redes sociais também já foram feitas neste contexto. Em Magdaleno *et al.* (2010) é apresentado um breve levantamento sobre as análises de redes sociais existentes para desenvolvimento de software.

4.5.1. Propriedades de análise de redes sociais

As redes sociais podem ser analisadas através das suas propriedades. Algumas destas propriedades possuem maior potencial para explicitar a colaboração e foram detalhadas em Santos (2010) e em Magdaleno *et al*. (2010). A seguir, são apresentados apenas alguns exemplos destas propriedades.

Um dos principais usos da análise de redes sociais é medir a importância de um vértice dentro da rede. Um nó da rede é considerado proeminente se os seus relacionamentos o tornam particularmente visível aos outros nós da rede. Um ator central é aquele que está intensivamente envolvido em relacionamentos com outros atores, seja como transmissor ou como receptor. Este envolvimento o torna mais visível aos outros.

Existem três tipos de propriedades de centralidade do ator (WASSERMAN e FAUST, 1994): centralidade de grau *(degree centrality),* centralidade de proximidade *(closeness centrality)* e centralidade de intermediação *(betweenness centrality).*

- **Centralidade de grau** *(degree centrality)***:** a centralidade de grau do nó está relacionada ao número de relações que este nó mantém na rede. O nó mais central ou *hub* (BARABASI, 2003) é aquele que possui o maior grau. Como ele está em contato direto com muitos outros nós, ele acaba ocupando um lugar central na rede. Em contrapartida, nós com grau pequeno ocupam uma posição periférica na rede. Usando o grafo apresentado na Figura 4.5 como exemplo, o nó mais central é o nó *c* que possui a maior centralidade de grau, que é igual a 3, pois possui 3 arestas. Os outros nós não atingem centralidade de grau igual a 3. Os nós *a* e *b* possuem centralidade de grau igual a 2 e o nó *d* possui a menor centralidade de grau desta rede que é igual a 1.

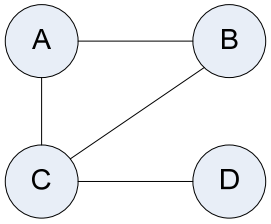


Figura 4.5 – Centralidade do grau

Apesar de essas propriedades de centralidade focarem nos nós, elas também podem ser combinadas entre os nós para obter a medida da rede

como um todo ou de determinados subgrupos. Desta forma, podem ser úteis para comparar diferentes redes sociais, desde que sejam normalizadas. Por outro lado, algumas outras propriedades, como a densidade da rede, exploram diretamente as propriedades da rede como um todo.

- **Densidade da rede:** a densidade da rede está relacionada à quantidade de arestas que mantém interligado um conjunto de vértices. Quanto mais arestas existir numa rede, mais densa ela será (WWF, 2003).

Por exemplo, na Figura 4.6, a mesma rede, composta por 14 vértices, é apresentada com número de arestas e, consequentemente, densidades diferentes. No diagrama (a), os vértices estão interligados por apenas 14 arestas, o que resulta em uma densidade baixa de 0,15. No diagrama (b), 37 arestas relacionam os vértices, representando uma densidade de 0,4. Por fim, no diagrama (c), os mesmos 14 vértices ligam-se por meio de 91 arestas, onde a densidade resultante é máxima e igual a 1, pois todos os vértices estabelecem ligações com os demais formando um grafo completo.

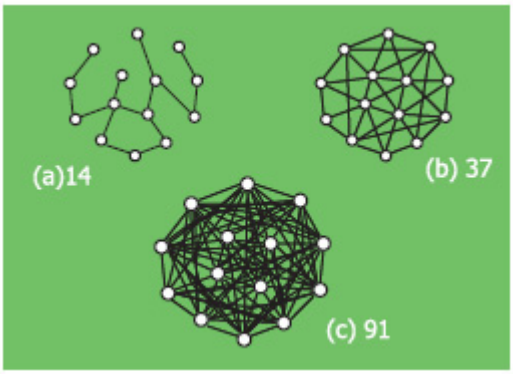


Figura 4.6 – Exemplo de densidade da rede (WWF, 2003)

4.6. Características de colaboração em redes sociais de desenvolvimento de software

A partir da interpretação e composição das propriedades de análise de redes sociais apresentadas na seção anterior, Santos *et al*. (2010, 2009) sugerem um conjunto de características de colaboração, organizadas tendo como inspiração os níveis de maturidade em colaboração do CollabMM (MAGDALENO *et al*., 2009), e focadas no aspecto da coordenação.

A Tabela 4.1 resume as características de colaboração em redes sociais de desenvolvimento de software. Essas características devem ser analisadas

em conjunto para que seja possível a percepção da colaboração em meio ao trabalho de desenvolvedores de software.

Tabela 4.1 – Características de colaboração em redes sociais de desenvolvimento de software (SANTOS *et al*., 2010)

	Coordenação Absoluta Nível Planejado	**Coordenação Múltipla Nível Perceptivo**	**Coordenação Distribuída Nível Reflexivo**
Densidade da rede	Baixa (0,01% a 30,00%)	Baixo-média a médio-alta (30,01% a 70,00%)	Alta (70,01% a 100,00%)
Centralidade de grau da rede	Alta (70,01% a 100,00)	Médio-alta a baixo-média (30,01% a 70,00%)	Baixa (0,00% a 30,00%)
Centralidade de intermediação da rede	Alta a médio-alta (50,01% a 100,00%)	Baixo a baixo-média (0,00% a 50,00%)	Baixa (0,00% a 30,00%)
Incidência de nós centrais	Poucos nós com alta centralidade em relação à centralidade de grau da rede	Maioria dos nós da rede com alta centralidade em relação à centralidade de grau da rede	Maioria dos nós da rede com alta centralidade em relação à centralidade de grau da rede
Incidência de nós intermediários	Poucos nós com alta intermediação em relação à centralidade de intermediação da rede	Poucos nós com alta intermediação em relação à centralidade de intermediação da rede	Maioria dos nós da rede com alta intermediação em relação à centralidade de intermediação da rede

A característica de coordenação absoluta está relacionada ao nível planejado do CollabMM. As redes que possuem esta característica apresentam têm um único nó tão forte que pode dominar toda a rede. Como se trata de uma rede centralizada, os líderes atuam como pontes entre os demais nós da rede. Entretanto, densidade da rede é baixa, pois a quantidade de interações ainda é reduzida (SANTOS *et al*., 2010).

Redes caracterizadas com coordenação múltipla apresentam os primeiros sinais de descentralização e, portanto, existem alguns nós centrais e intermediários estabelecendo ligações entre pequenos subgrupos dentro da mesma rede (SANTOS *et al*., 2010).

Nas redes de coordenação distribuída ocorre a ausência de nós centrais e intermediários na rede, pois os relacionamentos entre os nós tendem a ser distribuídos igualmente na rede. A densidade da rede é considerada alta,

podendo atingir a chamada densidade máxima da rede que representa o grau máximo de colaboração (SANTOS *et al*., 2010).

Estudos exploratórios iniciais em projetos de desenvolvimento de software livre foram realizados com o objetivo principal de verificar a existência destas características de colaboração. Os resultados destes estudos, apresentados em Santos *et al.* (2010, 2009), apontam para a importância destas características como instrumento para oferecer percepção da colaboração.

Desta forma, este trabalho fornece um importante insumo para a presente proposta de tese que pretende se valer das características de colaboração identificadas para acompanhar a colaboração no projeto atual e planejar o nível de colaboração necessário em outros projetos similares.

Por exemplo, ao detectar que um projeto, que precisaria de um nível alto de colaboração devido ao seu contexto específico, está com uma rede de coordenação muito centralizada, é possível sinalizar esta situação ao ambiente de apoio à decisão de forma que o processo possa ser readaptado visando tratar este problema. Neste caso, é possível incluir, alterar ou excluir um ou mais componentes de processo de forma que o processo específico do projeto ganhe novas características necessárias para estimular a colaboração.

4.7. Requisitos para ferramentas de redes sociais

Existem diversas ferramentas propostas para lidar com as redes sociais. Em um estudo realizado foram analisadas um total de dez ferramentas, sendo oito acadêmicas, uma ferramenta *shareware*[8] e uma ferramenta livre (MAGDALENO *et al*., 2010). Este estudo permitiu observar que as ferramentas de análise identificadas já oferecem um conjunto extenso de algoritmos genéricos que podem ser prontamente utilizados para os cálculos das propriedades de análise das redes sociais. Porém, não se dedicam a análises especificamente voltadas à colaboração. Por sua vez, a maioria das ferramentas de visualização não está de fato disponível ou possui limitações significativas.

[8] Uma ferramenta *shareware* permite a redistribuição do software, mas não a modificação do código fonte. Além disso, limita o uso de acordo com uma condição específica como, por exemplo, um período de tempo limite.

A partir da análise das contribuições e limitações das ferramentas, da observação das propostas existentes para análise das redes sociais no desenvolvimento de software, e dos objetivos do presente trabalho de pesquisa, chegamos à lista de requisitos que uma ferramenta de redes sociais deve atender (MAGDALENO *et al*., 2010). Os requisitos foram separados em três categorias: mineração (REQM), visualização (REQV) e análise (REQA):

- **REQM1** - O sistema deve possuir um meta-modelo para a mineração e armazenamento das informações das redes sociais.

  Neste sentido, pode-se considerar o uso do meta-modelo de redes sociais proposto por Balieiro *et al*. (2007) e apresentado na Figura 4.7, mas a definição das informações deve considerar a semântica das análises que se deseja realizar com as redes sociais.

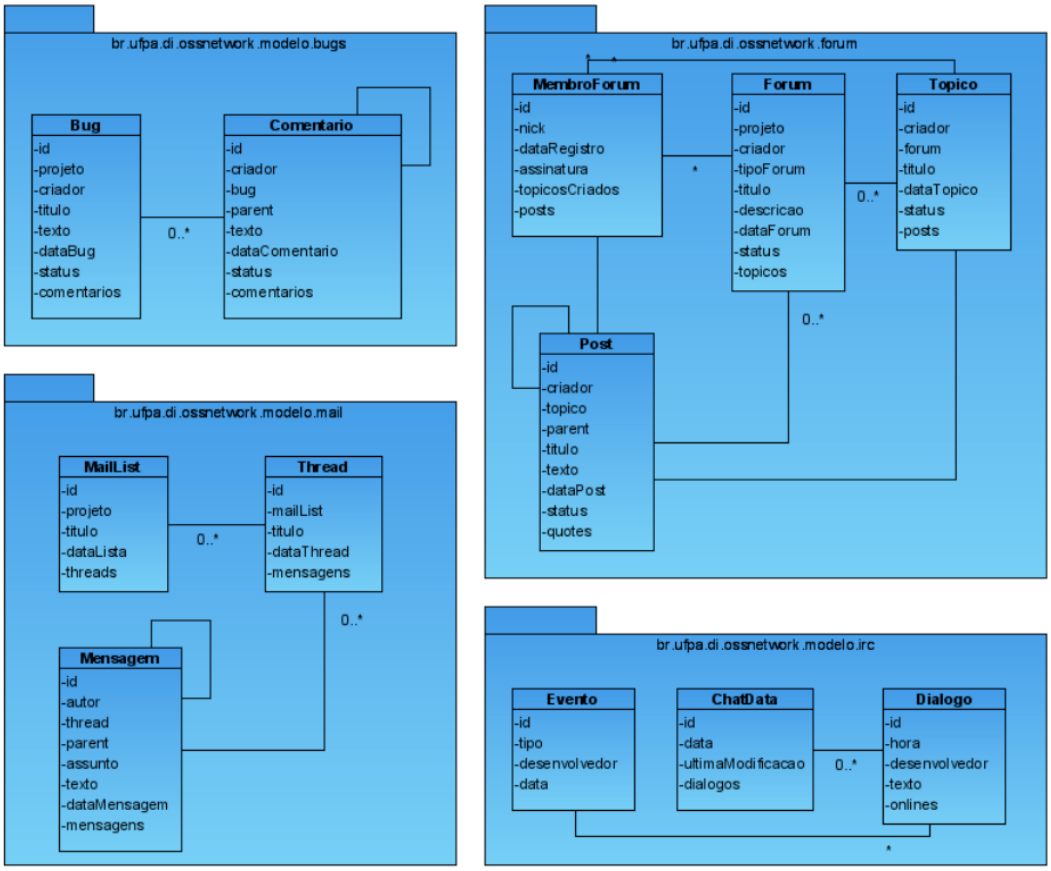


Figura 4.7 – Meta-modelo de redes sociais (BALIEIRO *et al*., 2007)

- **REQM2** - O sistema deve atualizar os dados das redes sociais de forma constante.

Desta forma, evita-se uma das limitações identificada no OSSNetwork (MAGDALENO *et al*., 2010), pois os dados não se limitarão a representar apenas alguns *snapshots* das interações.

- **REQM3** - O sistema deve ser capaz de minerar dados de diferentes fontes de informações de projetos de desenvolvimento de software: repositório de gerência de configuração, código fonte, fórum de discussão e lista de e-mail.

  Considerando as diferentes fontes de informações tratadas pelas ferramentas estudadas, foram selecionadas neste requisito as fontes de informações mais interessantes para o presente trabalho. Estas fontes de informações elencadas compõem um conjunto inicial, mas ao longo desta pesquisa, outras fontes de informações importantes podem ser identificadas e agregadas.

- **REQV4** - O sistema deve oferecer a visualização da rede técnica, através de um grafo, onde os nós são os componentes de software e as arestas representam as dependências existentes entre eles.

  A rede de dependência técnica é puramente baseada no nível do código do software e pode ser extraída através das dependências existentes entre as classes ou pacotes (SOUZA *et al*., 2007, TRAINER *et al*., 2005). Um exemplo de rede técnica é apresentado pela Figura 4.8.

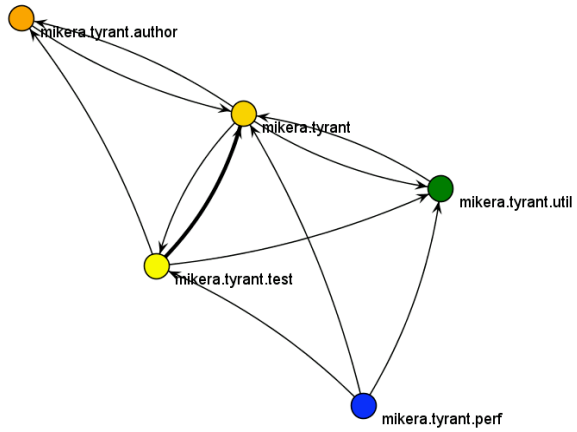


Figura 4.8 – Rede de dependências técnicas no código (TRAINER *et al*., 2005)

- **REQV5** - O sistema deve oferecer a visualização da rede sócio-técnica, através de um grafo, onde os nós maiores são os componentes de software

e as arestas são as dependências existentes entre eles. Os desenvolvedores são representados pelos nós menores e a aresta entre desenvolvedor e um componente indica a criação ou atualização do componente.

A partir da rede técnica dos componentes do software e da identificação do desenvolvedor responsável por um componente, é desenvolvida a rede sócio-técnica (SOUZA *et al.*, 2007, TRAINER *et al.*, 2005), conforme o exemplo apresentado pela Figura 4.9.

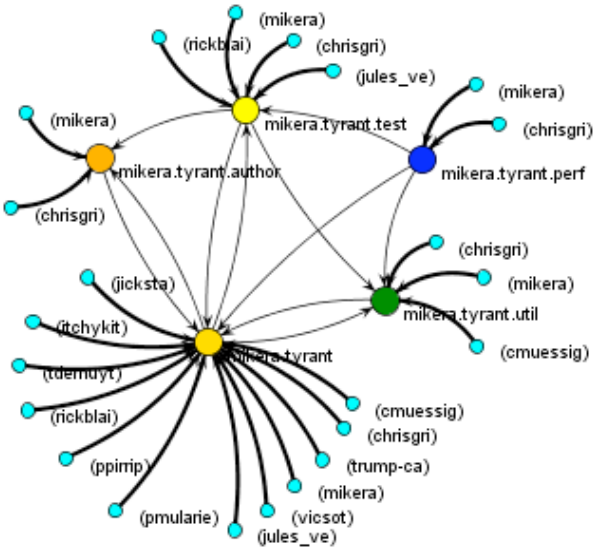


Figura 4.9 – Rede de dependências sócio-técnicas (TRAINER *et al.*, 2005)

- **REQV6** - O sistema deve oferecer a visualização da rede técnica e sócio-técnica com diferentes granularidades: por classe ou por pacote.

- **REQV7** - O sistema deve oferecer a visualização da rede social, através de um grafo, com as relações de dependência entre os atores.

Combinando a rede sócio-técnica com a informação de autoria, é possível extrair indiretamente uma rede social (Figura 4.10) com as dependências sociais entre os desenvolvedores.

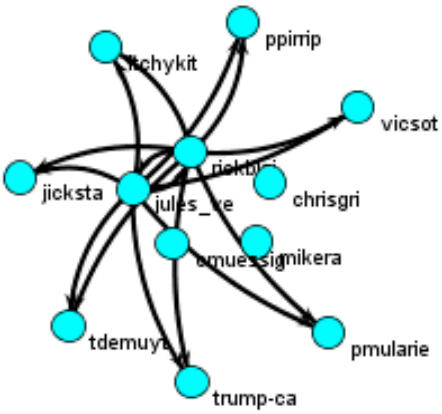


Figura 4.10 – Rede de dependências sociais (TRAINER *et al*., 2005)

- **REQV8** - O sistema deve destacar, na visualização da rede sócio-técnica, o tamanho da contribuição (em número de linhas de código) de acordo com o tamanho do nó (conforme o exemplo da ferramenta Augur apresentado na Figura 4.11).

Figura 4.11 – Recursos de percepção em redes sociais (SOUZA *et al*., 2005)

- **REQV9** - O sistema deve destacar, na visualização da rede sócio-técnica, a informação de temporalidade, mudando a cor da linha de acordo com o quão recente é uma determinada contribuição (conforme o exemplo da ferramenta Augur apresentado na Figura 4.11).

- **REQV10** - O sistema deve oferecer a visualização da evolução da rede ao longo do tempo (de acordo com os mecanismos temporais apresentados na seção 4.4).

- **REQA11** - O sistema deve calcular as propriedades de centralidade de grau, centralidade de intermediação e centralidade proximidade de cada nó

e da rede como um todo (de acordo com o detalhamento das propriedades de análise de redes sociais apresentado na seção 4.5.1).

- **REQA12** - O sistema deve informar a propriedade de densidade da rede (de acordo com o detalhamento das propriedades de análise de redes sociais apresentado na seção 4.5.1).

- **REQA13** - O sistema deve identificar o modelo (randômica, livre de escala e mundo pequeno) de formação da rede analisada (de acordo com o detalhamento dos modelos de redes apresentado na seção 4.2).

- **REQA14** – O sistema deve permitir a integração com outras ferramentas de análise, como o UCINET e o Pajek, para que possam ser utilizados os seus recursos mais avançados para análise das redes sociais.

- **REQA15** - O sistema deve identificar a característica de colaboração aplicável a uma determinada rede social (de acordo com as características de colaboração propostas por Santos *et al*. (2010) e apresentadas na seção 4.6.

A Tabela 4.2 classifica as ferramentas de redes sociais analisadas de acordo com os requisitos apresentados. Os campos preenchidos com ✔ indicam que o requisito em questão é totalmente coberto pela ferramenta. Os campos preenchidos com ✘ indicam que o requisito não é atendido pela ferramenta. Os campos preenchidos com ? indicam que não foi possível inferir se o requisito é atendido pela ferramenta através da documentação ou da versão disponível.

O fato de um requisito não ser atendido por uma determinada ferramenta não significa que essa ferramenta é pior que as demais, pois para o seu objetivo esse requisito pode ser irrelevante. Esses requisitos, apesar de não formarem um conjunto completo nem necessariamente suficiente, servem como um guia para a elaboração de uma ferramenta que se proponha a ser abrangente e que, se possível, traga novas contribuições em relação às

ferramentas atualmente disponíveis na literatura, que é o caso da proposta deste trabalho. Estes requisitos também servirão para uma posterior comparação com a abordagem proposta neste trabalho.

Tabela 4.2 – Tabela ferramentas x requisitos

Requisito	Ariadne	Augur	MiSoN	OssNetwork	Pajek	RaisAware	Sargas	SVNNAT	UCINET	Visone
REQM1	?	?	?	✔	?	?	?	?	?	?
REQM2	?	?	?	✘	✘	✘	✘	✘	✘	✘
REQM3	✘	✘	✘	✔	✘	✘	✔	✘	✘	✘
REQV4	✔	✘	✘	✘	✘	✔	✘	✘	✘	✘
REQV5	✔	✔	✘	✘	✘	✔	✘	✘	✘	✘
REQV6	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘
REQV7	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔
REQV8	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘
REQV9	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘
REQV10	✔	✘	✘	✔	✘	✘	✘	✘	✘	✘
REQA11	✘	✘	✘	✔	✔	✔	✔	✘	✔	✔
REQA12	✘	✘	✘	✔	✔	✔	✘	✘	✔	✔
REQA13	✘	✘	✘	✘	✔	✘	✘	✘	✔	✘
REQA14	✔	?	✔	✔	✔	?	✔	✔	✔	✘
REQA15	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘

Após a análise da Tabela 4.2 podemos concluir que nenhuma das ferramentas de redes sociais analisadas atende a todos os requisitos. Assim, ainda existe espaço para proposta de outras ferramentas que apóiem de forma mais adequada à necessidade de explicitação da colaboração através da implementação destes requisitos. Esta motivação levou a criação de uma nova ferramenta – o EvolTrack-SocialNetwork – apresentada na próxima seção.

4.8. Ferramental de Apoio: EvolTrack-SocialNetwork

A ferramenta EvolTrack-SocialNetwork, proposta como parte da solução deste trabalho, se propõe a atender aos requisitos, identificados na seção anterior, para ferramentas de redes sociais. O EvolTrack-SocialNetwork é uma extensão da ferramenta EvolTrack (CEPEDA *et al*., 2008), que constitui um dos projetos do Grupo de Reutilização de Software.

4.8.1. Ferramental de apoio: EvolTrack

O EvolTrack (CEPEDA *et al*., 2008) é uma abordagem para capturar e visualizar o ciclo de evoluções de um projeto de software. Basicamente, ele extrai, periodicamente, as informações do projeto de uma fonte de dados e, depois de realizar processamentos e transformações, apresenta o design correspondente ao projeto.

A ferramenta foi criada como um *plug-in* do ambiente de desenvolvimento Eclipse e sua arquitetura, projetada para ser flexível, é composta por quatro tipos de componentes:

- **Conectores de fontes de dados:** capturam as informações referentes ao histórico da evolução de um projeto de software (concluído ou em andamento) a partir de determinadas fontes de dados. Estes conectores fazem uma engenharia reversa para criar a representação dos modelos. Exemplos destas fontes de dados podem ser sistemas de controle de versão, como SubVersion, CVS e Odyssey-VCS 2 (MURTA *et al*., 2008), ou até mesmo ambientes de desenvolvimento de software como o Eclipse ou o NetBeans.

Esta coleta de informações pode ser feita desde o início do projeto e permite que o gerente de projeto monitore permanentemente o desenvolvimento, para que possa intervir em algum momento, se necessário. Por outro lado, também é possível realizar uma análise do histórico de um projeto.

- ***Kernel*:** responsável por orquestrar todo o fluxo de informações e por manter a persistência dos modelos criados na ferramenta.
- **Transformador de modelos:** este elemento opcional pode adicionar informação ou agregar valor aos modelos inicialmente criados com os

conectores de fonte de dados, geralmente enriquecendo os modelos com detalhes sobre o projeto como, por exemplo, os dados para o cálculo das métricas.

- **Conectores de visualização:** exibem, utilizando alguma forma de representação, as informações de evolução do software obtidas através do(s) conector(es) de fontes de dados. A partir da mesma fonte de dados, é possível ter diversos tipos de visualização das informações de projeto. Para cada tipo de visualização, existe um conector diferente.

No que tange aos recursos de visualização, a ferramenta utiliza o projeto UML2Tools[9] para apresentar seus diagramas UML (*Unified Modeling Language*). A evolução do software é visualizada em uma sequência temporal, facilitando a compreensão do projeto como um todo e a percepção sobre o que mudou de uma determinada evolução para a outra. Além disso, a representação visual sofre atualização automática sempre que novos dados são gerados e extraídos da fonte de dados.

O EvolTrack possui ainda mecanismos de percepção, utilizando esquemas de cores, para representar o comportamento das métricas ao longo do tempo (Figura 4.12). Estas métricas são configuráveis e já foram utilizadas, por exemplo, no projeto Orion (PRUDENCIO *et al*., 2009) para explorar métricas de avaliação da concorrência no acesso ao código-fonte do projeto. Com base em todo o histórico de evolução do projeto, o EvolTrack é capaz de gerar uma animação que mostra a evolução da estrutura do software através de mudanças nas cores das métricas. Esta funcionalidade será particularmente útil na análise das propriedades das redes sociais.

Assim, o EvolTrack foi escolhido como ponto de partida para a construção da ferramenta de redes sociais, pois já oferece uma infraestrutura inicial para a extração e atualização de dados, alguns recursos de visualização e a funcionalidade para análise de métricas, além de ter sido desenvolvido pelo Grupo de Pesquisa em Reutilização de Software no qual este trabalho de pesquisa também está inserido.

Após esta escolha, foi planejado e realizado um estudo para analisar a sua viabilidade de uso em cenários reais, utilizando sete projetos de software

[9] Site UML2Tools: http://www.eclipse.org/modeling/mdt/?project=uml2tools

livre (CEPEDA *et al.*, 2010). Projetos de software livre foram escolhidos, pois eles publicam os seus artefatos de desenvolvimento, incluindo código fonte, livremente através da Internet. Assim, representam uma oportunidade de pesquisa, devido a sua diversidade, complexidade, representatividade e facilidade de acesso aos dados.

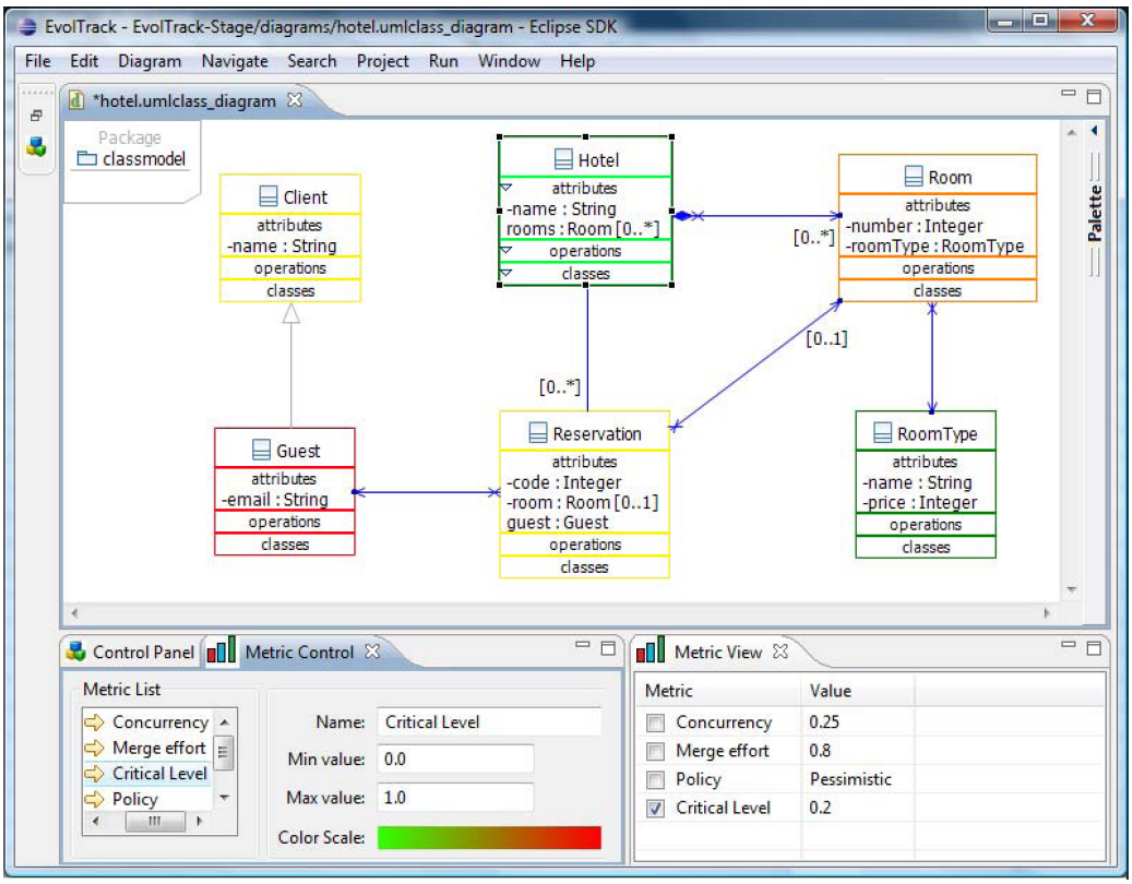


Figura 4.12 – Funcionalidade do EvolTrack de análise de métricas

Os dados destes projetos de software livre foram minerados. A partir destes dados foram extraídas as redes técnica do código e visualizadas com o EvolTrack para ajudar a entender a evolução do projeto ao longo do tempo. Na Figura 4.13 é possível perceber o que foi incluído ou alterado nesta rede de uma versão para outra (CEPEDA *et al.*, 2010).

Como resultado, este estudo mostrou a viabilidade de uso do EvolTrack e sinalizou algumas limitações de escalabilidade em relação à visualização de modelos de projetos muito grandes (CEPEDA *et al.*, 2010). Isso motivou a inclusão do requisito REQV6, pois originalmente o EvolTrack não disponibilizava as informações em diferentes granularidades. Atualmente, esta possibilidade já está em construção, como parte de outro trabalho de pesquisa que pretende ampliar os recursos de visualização do EvolTrack (SILVA, 2010).

A partir desta avaliação da viabilidade de uso do EvolTrack, iniciou-se então o projeto do EvolTrack-SocialNetwork, apresentado na próxima seção.

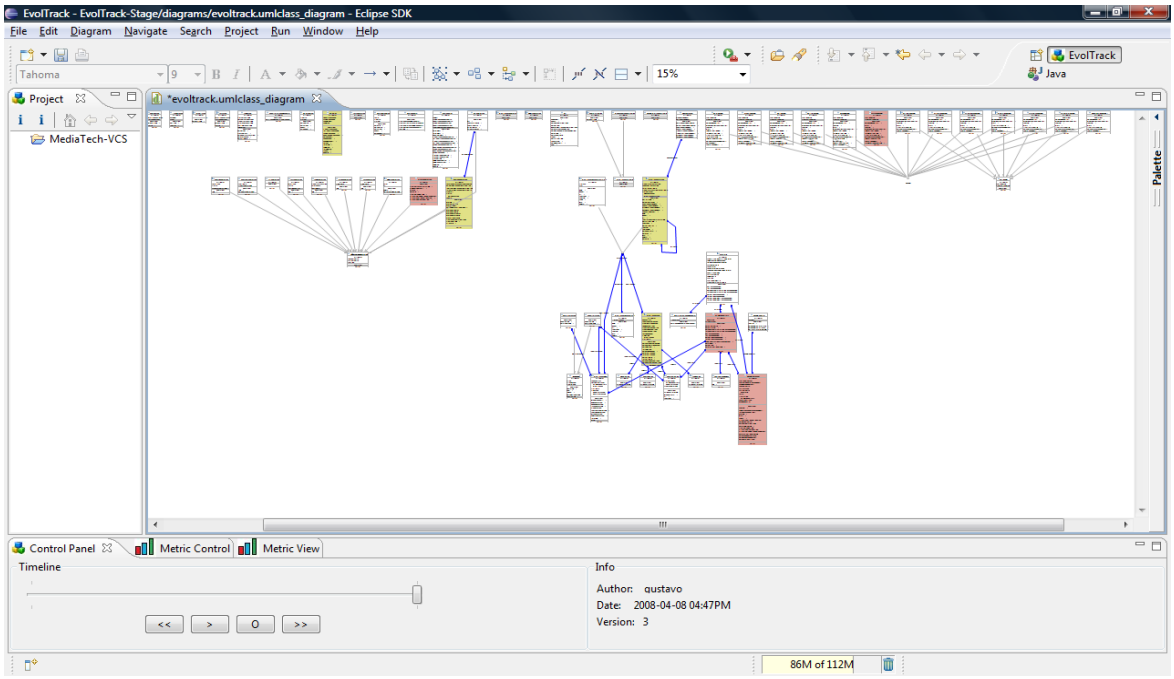


Figura 4.13 – Rede técnica do projeto MediaTech VCS (CEPEDA *et al*., 2010)

4.8.2. Arquitetura do EvolTrack-SocialNetwork

A arquitetura do EvolTrack-SocialNetwork (Figura 4.14) é composta por três módulos: o módulo de mineração de redes sociais, o módulo de visualização de redes sociais e o módulo de análise de redes sociais. Todos estes módulos se baseiam em um meta-modelo de redes sociais. Além disso, como foi aproveitada a infraestrutura do EvolTrack, também são utilizados os seus componentes, apresentados na seção anterior.

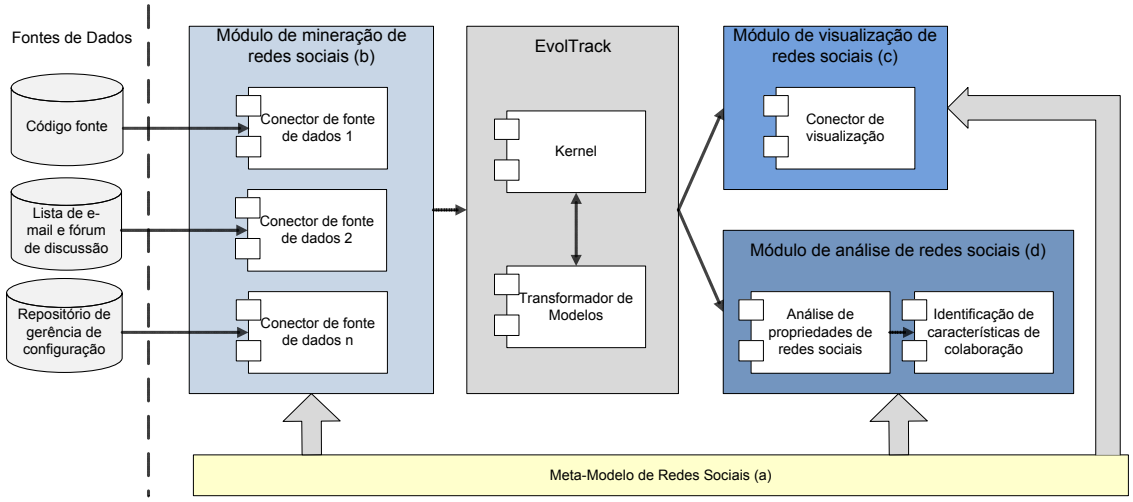


Figura 4.14 – Visão geral da arquitetura do EvolTrack-SocialNetwork

Originalmente, o EvolTrack utiliza o meta-modelo da UML. Ainda que este meta-modelo seja adequado para o tratamento das informações da rede técnica, ele não atende completamente às necessidades de informações das redes sócio-técnica e social. Assim, este meta-modelo precisará ser estendido no EvolTrack-SocialNetwork para atender ao requisito REQM1. Neste sentido, pode-se considerar o uso do **meta-modelo de redes sociais** já existente em (BALIEIRO *et al*., 2007), mas a escolha do meta-modelo deve ser feita com base nas análises que se deseja realizar com as redes sociais (Figura 4.14a).

O **módulo de mineração de redes sociais** (Figura 4.14b) recebe as informações extraídas pelos conectores de fontes de dados e pretende atender aos requisitos REQM2 e REQM3. O REQM2 é automaticamente satisfeito, pois o EvolTrack atualiza os dados sempre que existem modificações nas fontes de dados. Por outro lado, o REQM3 não é completamente satisfeito, visto que atualmente o EvolTrack possui conectores de fontes de dados apenas para ferramentas de gerência de configuração e ambientes de desenvolvimento. Entretanto, o EvolTrack já oferece a infraestrutura necessária para facilitar a criação de novos conectores. Neste caso, os conectores de fonte de dados que precisam ser construídos ou acoplados ao EvolTrack-SocialNetwork são: lista de e-mail e fórum de discussão.

O **módulo de visualização de redes sociais** (Figura 4.14c) recebe as informações processadas pelo *kernel*. Este módulo se concentra em atender aos requisitos REQV4 à REQV10. O REQV4 já está implementado no EvolTrack, pois o conector de visualização UML já expressa, através do diagrama de classes, exatamente aquilo que se quer representar com a rede técnica, onde os nós são as classes e as arestas representam as relações de dependências entre elas.

Os requisitos REQV5 e REQV7 precisarão ser construídos, pois o EvolTrack não oferece a visualização da rede sócio-técnica e social. O REQV5 foi o primeiro a ser tratado na implementação da abordagem proposta. Quanto ao REQV6, originalmente o EvolTrack não disponibilizava as informações em diferentes granularidades, mas esta possibilidade já está em construção, como parte de outro trabalho de pesquisa que pretende ampliar os recursos de visualização do EvolTrack (SILVA, 2010). Os recursos de percepção pretendidos nos requisitos REQV8 e REQV9 também precisam ser construídos

ainda. Em compensação, o REQV10 corresponde a uma funcionalidade nativa do EvolTrack que é a apresentação da evolução temporal da rede.

Por fim, o **módulo de análise de redes sociais** (Figura 4.14d) foca nos requisitos REQA11 à REQA15 que são a principal motivação para a construção do EvolTrack-SocialNetwork. Os REQA11 e REQA12 dizem respeito à necessidade de analisar as propriedades de redes sociais (SANTOS *et al*., 2009). Ainda que o EvolTrack não calcule essas propriedades, ele já é capaz de exibi-las e acompanhar a evolução de métricas, conforme visto na seção anterior (Figura 4.12).

O REQA13 remete à identificação do modelo (randômico, livre de escala ou mundo pequeno) correspondente à rede que está sendo analisada. O REQA14 tem como objetivo facilitar a integração com outras ferramentas que possuem recursos mais avançados para análise de outras propriedades das redes sociais. Por fim, o REQA15 está relacionado ao trabalho de Santos et al. (2010) sobre a identificação de características de colaboração e a sua associação aos níveis de maturidade em colaboração do CollabMM (MAGDALENO *et al*., 2009).

O EvolTrack-SocialNetwork está em desenvolvimento, como parte de um trabalho de iniciação científica, e espera-se que ele possa contribuir para prover informações de colaboração que ajudem a equipe de desenvolvimento a perceber e interferir no trabalho que está sendo realizado.

4.9. Considerações finais

Este capítulo mostrou que as redes sociais, obtidas como resultado das interações no desenvolvimento de software, podem fornecer informações úteis para o entendimento da colaboração existente entre os membros da equipe de desenvolvimento.

Em especial, o trabalho de Santos *et al*. (2010) mostrou que é possível combinar as propriedades de análise de redes sociais para identificar características de colaboração e compreender o nível de colaboração presente na execução de um projeto de desenvolvimento de software. Através destas características de colaboração, e com a ajuda de ferramentas de visualização e

análise de redes sociais, a equipe de desenvolvimento é capaz de perceber, refletir e interferir no trabalho que está sendo realizado.

Assim, este trabalho fornece um importante insumo para a presente proposta de tese que pretende se valer das características de colaboração identificadas para acompanhar a colaboração no projeto atual e planejar o nível de colaboração necessário em outros projetos similares.

Além das contribuições trazidas pelo trabalho de Santos *et al*. (2010) é importante também a implementação na ferramenta EvolTrack-SocialNetwork, dos requisitos para ferramentas de redes sociais apresentados, para que possam ser realizados estudos experimentais nos projetos de desenvolvimento de software que permitam o estudo das suas estruturas topológicas e modelos de formação.

Devido à importância da realização de estudos dessa natureza para a avaliação de um novo método, técnica, processo ou ferramenta na Engenharia de Software (PFLEEGER, 1999), é preciso planejar como o EvolTrack-SocialNetwork será avaliado. Ainda que um planejamento detalhado e completo deste estudo não seja objeto desta proposta de tese, foram traçadas algumas estratégias iniciais.

Seguindo a metodologia proposta por Shull *et al*. (2001), que se baseia em estudos experimentais para avaliar a introdução de tecnologias de software, deve ser adotada uma abordagem incremental. Inicialmente, deve ser realizado um estudo de viabilidade para determinar a viabilidade prática da aplicação da tecnologia. Este estudo de viabilidade deverá utilizar projetos de software livre, pois assim é possível aproveitar os dados publicados livremente.

Como segundo passo, pretende-se realizar, no contexto do desenvolvimento de um dos projetos do NP2TEC-UNIRIO inserido no ambiente acadêmico, um estudo de observação para avaliar a aplicabilidade e as formas de uso do EvolTrack-SocialNetwork. Somente então é possível pensar em realizar um estudo de caso, no contexto de um projeto de desenvolvimento de software real em ambiente industrial, para avaliar se o EvolTrack-SocialNetwork contribui efetivamente para o acompanhamento do nível de colaboração do projeto.

5. Linha de Processos de Desenvolvimento de Software

Este capítulo apresenta o conceito de linha de processos, originado da linha de produtos de software da área de reutilização. Em seguida, são discutidas as principais abordagens para linha de processos existentes na literatura. A partir da análise destes trabalhos, é apresentada a proposta para engenharia de linha de processos, que faz parte do enfoque de solução deste trabalho, juntamente com um exemplo da sua aplicação no domínio de processos de desenvolvimento de software.

Durante os primeiros anos de pesquisa, a Reutilização de Software[10] foi aplicada basicamente ao código fonte, com o objetivo de aumentar a qualidade e produtividade e reduzir os custos e o tempo de desenvolvimento (FRAKES e KYO KANG, 2005). Entretanto, a reutilização é inerente ao processo de solução de problemas dos seres humanos. Assim, à medida que soluções são encontradas, elas são utilizadas em problemas similares. Então, com o tempo, a ideia da reutilização de software estendeu-se a outros artefatos do desenvolvimento de software como a especificação, o projeto (do inglês *design*) e os testes. Atualmente, existem diferentes técnicas de reutilização de software, tais como: padrões (do inglês *patterns)* (GAMMA *et al*., 1994), *frameworks* (FAYAD *et al*., 1999), Desenvolvimento Baseado em Componentes (DBC) (BLOIS, 2006, BOSCH, 2000, SZYPERSKI, 2002, WERNER e BRAGA, 2005), Engenharia de Domínio (ED) (PRIETO-DAZ, 1991) e Linhas de Produto de Software (LPS) (BOCKLE *et al*., 2005, GARG *et al*., 2003, LINDEN *et al*., 2007, NORTHROP, 2002).

Do mesmo modo que existem analogias válidas entre produto de software e processo de software (OSTERWEIL, 1987), também existem analogias válidas entre reutilização de produtos de software e reutilização de processos de software. Como forma de promover a reutilização do conhecimento relacionado a processos de software, técnicas de reutilização, como as Linhas de Produtos de Software (LPS), apresentadas na próxima seção, têm sido adaptadas ao contexto de definição de processos de software (KELLNER, 1996, WASHIZAKI, 2006). Em suma, nestas abordagens, os processos são apenas mais um artefato no qual o conceito de reutilização pode ser aplicado.

[10] A reutilização de software é o uso do conhecimento ou dos artefatos de software já existentes para construir um novo software (FRAKES e KYO KANG, 2005).

5.1. Engenharia de Domínio e Linha de Produtos de Software

Um processo de desenvolvimento baseado em reutilização é caracterizado por duas etapas distintas: o desenvolvimento de artefatos reutilizáveis, denominado de processo de desenvolvimento para reutilização, e o processo de desenvolvimento de aplicações baseadas nestes artefatos, denominado desenvolvimento com reutilização (BRAGA, 2000).

O desenvolvimento *para* reutilização visa aplicar técnicas de Engenharia de Domínio (ED). A Engenharia de Domínio (ARANGO e PRIETO-DIAZ, 1991, BRAGA, 2000, WERNER e BRAGA, 2005) é uma "abordagem baseada em reutilização para definição do escopo, especificação da estrutura, e construção de recursos para uma classe de sistemas, subsistemas ou aplicações" (ISO/IEC, 1995). O desenvolvimento *com* reutilização diz respeito à Engenharia de Aplicações (EA) (MILER, 2000) que consiste na reutilização de artefatos gerados na ED. A ED e a EA trabalham em conjunto para promover a reutilização (Figura 5.1): a ED provê um conjunto de artefatos que serão utilizados, enquanto a EA constrói aplicações com base na instanciação dos artefatos gerados na ED (ATKINSON *et al*., 2001, OLIVEIRA, 2006).

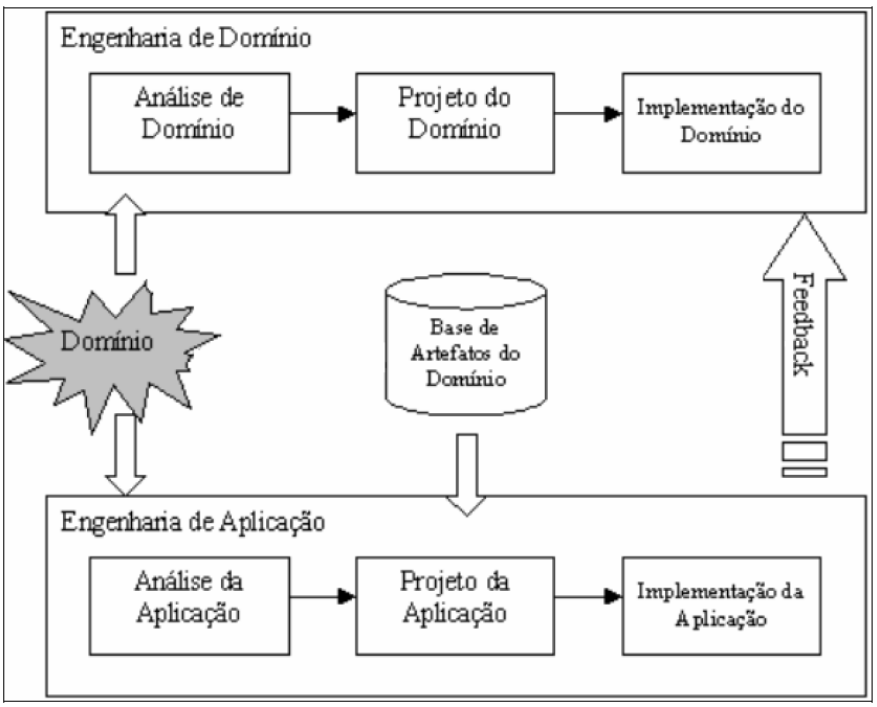


Figura 5.1 – Ciclo de vida de ED e EA (ATKINSON *et al*., 2001)

A ED consiste no desenvolvimento de um conjunto de ativos reutilizáveis que podem ser configurados e combinados para criar diferentes produtos

(HUBAUX *et al*., 2010). Pode-se dizer que existe um consenso da comunidade a respeito das etapas de ED:

- **Análise do domínio:** o domínio e o seu escopo são identificados e também são determinadas as características comuns e variáveis de uma família de aplicações;
- **Projeto do domínio:** os resultados da primeira etapa são utilizados para a construção de um projeto adaptável, fruto da identificação e generalização de soluções para as características comuns. As oportunidades de reutilização identificadas na análise do domínio são refinadas de forma a especificar as restrições do projeto;
- **Implementação do domínio:** nesta etapa são definidos mecanismos para a tradução dos requisitos (resultados da análise e projeto do domínio) em modelos implementacionais, que incluem a identificação, reengenharia e/ou construção, e manutenção de componentes reutilizáveis que suportam estes requisitos e soluções de projeto.

A EA tem fases similares às apresentadas na ED. No entanto, uma vez que a EA consiste no processo de reutilização de artefatos gerados na ED, as atividades relacionadas à modelagem das aplicações são mais específicas, enfatizando, sobretudo, as características particulares de cada aplicação.

Recentemente, o termo Linha de Produtos de Software (LPS) foi adotado para dar à ED um aspecto mais prático e mais voltado ao âmbito empresarial (GRISS, 1999, POULIN, 1995). Ambas, a LPS e a ED, têm em comum o desenvolvimento de uma família de aplicações através de uma abordagem de reutilização.

Uma LPS é um “conjunto de produtos de software que compartilham um conjunto de características comuns e controladas, que satisfazem necessidades de um segmento de mercado em particular, e são desenvolvidos a partir de artefatos, de forma predefinida” (NORTHROP, 2002). Uma LPS fornece um conjunto de características (KANG *et al*., 1990) que podem ser instanciadas em produtos específicos gerados a partir da linha de produtos.

Uma linha de produtos funciona como uma fábrica, que instancia produtos com características similares, mas, ao mesmo tempo, com algumas características específicas que os diferenciam, através da composição de componentes existentes (GARG *et al*., 2003, NORTHROP, 2002). Desta forma,

uma LPS pode ser vista como uma coleção de produtos similares (SUTTON e OSTERWEIL, 1996).

O conceito de variabilidade está fortemente presente na LPS e pode ser entendido como “a habilidade que um artefato possui de ser alterado, customizado, ou configurado para um contexto em particular” (BOSCH, 2004). A variabilidade em uma linha de produtos de software é importante para explicitar os pontos onde tais produtos se assemelham e, portanto, podem ser reutilizados, e os pontos onde eles diferem entre si, devendo receber tratamento específico (BECKER, 2003). Desta forma, a variabilidade oferece flexibilidade para diferenciar e diversificar os produtos (CHEN *et al*., 2009). A modelagem de características, detalhada na próxima seção, permite a representação de variabilidade em linhas de produtos de software.

5.1.1. Modelagem de características

A modelagem de características é uma abordagem que trata da complexidade em expressar requisitos em forma de características (MASSEN e LICHTER, 2004). Características (do inglês *features*) são definidas por Czarnecki e Eisenecker (2000) como “um aspecto, uma qualidade, ou uma característica visível ao usuário, proeminente ou distinta, de um sistema (ou sistemas) de software”. As características são tipicamente organizadas em um diagrama de características, através de uma estrutura hierárquica, como uma árvore ou grafo acíclico, dependendo da notação utilizada.

A modelagem de características é uma das técnicas mais aceitas para representar produtos em uma linha (LEE *et al*., 2002). O propósito da modelagem de características é “capturar e gerenciar as similaridades e diferenças, de forma a facilitar o entendimento de clientes e desenvolvedores no que se refere às capacidades gerais de um domínio, que são expressas através de características” (KANG *et al*., 1990).

Existem diversas notações que se propõem a representar a variabilidade no modelo de características. Alguns exemplos dessas notações são: FODA (*Feature Oriented Domain Analysis*) (KANG *et al*., 1990), FORM (*Feature Oriented Reuse Method*) (KANG *et al*., 1998) e Odyssey-FEX (FEX – *Feature Extended*) (OLIVEIRA, 2006). Neste trabalho optou-se por adotar a notação Odyssey-FEX que estende os elementos do modelo de características do

ambiente Odyssey (ODYSSEY, 2010), definidos originalmente na proposta de Miler (2000), para abranger elementos que representam conceitos, funcionalidades e tecnologias, incluindo a variabilidade e os relacionamentos entre eles. Oliveira (2006) realizou uma análise entre as notações mais comumente utilizadas de forma a resolver na Odyssey-FEX algumas das deficiências existentes nestas linguagens.

Em seguida, são apresentados alguns conceitos da modelagem de características. Alguns destes conceitos são comuns às diversas notações e inerentes à modelagem de características e variabilidade, enquanto outros existem apenas em algumas notações específicas, como a Odyssey-FEX.

Na notação Odyssey-FEX (OLIVEIRA, 2006), as características podem ser classificadas quanto à sua categoria, variabilidade e opcionalidade. Dentro do escopo de **categoria**, os tipos estão relacionados às diferentes fases de desenvolvimento do software. As características Entidade (representam atores que atuam sobre o domínio) e Domínio (se especializam em funcionais e conceituais) estão relacionadas à fase de análise do domínio. Já as características Tecnológicas, que permitem que aspectos tecnológicos sejam considerados e estão subdivididas em camadas (características de Ambiente Operacional, de Tecnologia de Domínio e de Técnicas de Implementação), estão relacionadas à fase de projeto.

Neste trabalho, estamos particularmente interessados nas Características de Domínio que estão ligadas à essência do domínio e representam conceitos (características conceituais) e funcionalidades (características funcionais) do domínio. As características conceituais dizem respeito aos conceitos que servem para o entendimento do domínio. As características funcionais descrevem as funcionalidades e requisitos do domínio, que serão primeiramente utilizados nas aplicações (BLOIS, 2006).

Alguns conceitos são inerentes à **variabilidade** e sua modelagem, tais como (OLIVEIRA, 2006):

- **Pontos de variação:** características que refletem a parametrização no domínio de uma maneira abstrata e podem ser configuradas através das variantes;
- **Variantes:** características que atuam como alternativas para se configurar um ponto de variação;

- **Invariantes:** características fixas que representam elementos não configuráveis em um domínio.

Estes conceitos são apresentados no exemplo ilustrado pela Figura 5.2. A característica funcional “ToquesMusicais” é um ponto de variação com três possíveis variantes: “ToqueMonofônico”, “ToquePolifônico” e “ToqueMP3”.

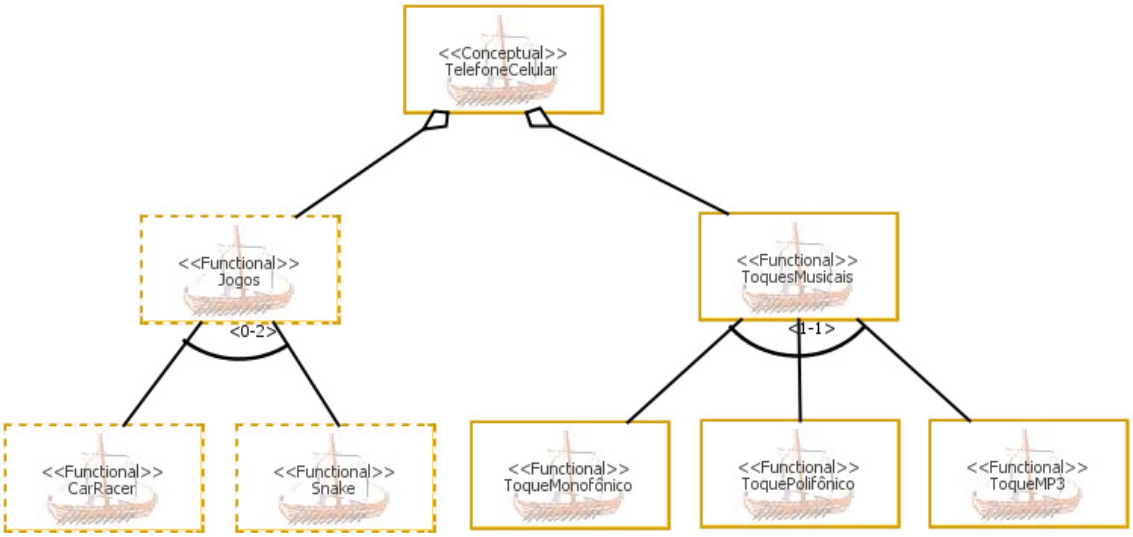


Figura 5.2 – Exemplo de variabilidade e opcionalidade na notação Odyssey-FEX

Ortogonalmente, em relação à **opcionalidade**, as características podem ser classificadas como (BLOIS, 2006, LINDEN *et al*., 2007, OLIVEIRA, 2006):

- **Elementos opcionais:** elementos que podem ou não estar presentes nos produtos derivados a partir da linha de produtos;
- **Elementos mandatórios:** elementos que devem obrigatoriamente estar presentes em todos os produtos derivados a partir da linha de produtos.

No exemplo da Figura 5.2, a opcionalidade é retratada através da linha pontilhada. Logo, as características funcionais “Jogos”, “CarRacer” e “Snake” podem ou não ser incluídas na instanciação da linha.

Essas classificações quanto à variabilidade e opcionalidade parecem ser um consenso para a modelagem de características, embora especificadas e representadas de diferentes maneiras (BOSCH, 2004, KANG *et al*., 1990, MASSEN e LICHTER, 2004).

A notação Odyssey-FEX (OLIVEIRA, 2006) valoriza ainda a semântica dos **relacionamentos** em um modelo de características, visando atingir uma maior capacidade de representação e expressão. Combina relacionamentos próprios do modelo de características (alternativo, ligação de comunicação, “implementado por”) com relacionamentos da UML (herança, composição,

agregação e associação). Alguns exemplos de uso destes relacionamentos também são apresentados pela Figura 5.2.

Outro conceito relacionado à modelagem de características, introduzido recentemente na notação Odyssey-FEX, é a **cardinalidade**, que é utilizada para definir o número mínimo e máximo de características que podem ser escolhidas a partir de um conjunto de alternativas de um ponto de variação (TEIXEIRA, 2008). No exemplo da Figura 5.2, é possível observar que apenas um toque musical pode ser escolhido, visto que a cardinalidade é <1-1>. No caso da característica “Jogos”, a cardinalidade mínima é zero, devido a sua opcionalidade, e existe um limite de no máximo dois jogos selecionados.

A especificação de **restrições** entre as características, que inclui os relacionamentos de dependência e mútua exclusividade, é uma forma de indicar a necessidade ou incompatibilidade da seleção conjunta de características (OLIVEIRA, 2006). Kang *et al*. (1990) propõem a representação de tais conceitos por meio do uso de regras de composição:

- **Inclusivas:** definem relações de dependência entre duas ou mais características, indicando que elas devem ser selecionadas em conjunto.
- **Exclusivas:** definem relações de mútua exclusividade entre características, onde duas ou mais características não devem ser escolhidas em conjunto.

Na notação Odyssey-FEX (OLIVEIRA, 2006), as regras de composição são expressas pela seguinte estrutura: antecedente + palavra-chave + consequente. A palavra-chave representa o tipo de regra: “requer” (*requires*), referente às regras inclusivas; e “exclui” (*excludes*), referente às regras exclusivas. Antecedente e conseqüente são expressões, que podem ser literais ou *booleanas* e representam uma característica ou combinação de características do domínio.

5.2. Linha de processos

A complexidade da adaptação dos processos levou à necessidade de caracterizar, usar e gerenciar as similaridades e diferenças entre os processos (SUTTON e OSTERWEIL, 1996). Neste sentido, a sistematização da reutilização e adaptação alcançada na LPS pode ser aplicada também a

processos, melhorando os resultados já alcançados pelas propostas de adaptação (em sua maioria *ad-hoc*) existentes (MONTERO *et al*., 2007).

Transferindo o conceito de linha de produtos para o contexto de processos, surge então a ideia de **linha de processos** (BARRETO, 2007, JAUFMAN e MUNCH, 2005, ROMBACH, 2006, WASHIZAKI, 2006), que são linhas de produtos cujos produtos são processos. Washizaki (2006) define uma linha de processo como "um conjunto de processos de um determinado domínio de problema ou com um determinado propósito, que tem características em comum e é construído baseado em ativos reutilizáveis de processos".

Sutton e Osterweil (1996) sugeriram o conceito de "família de processos", que significa basicamente o mesmo que uma linha de processos (os autores apenas não usaram o termo "linha" para evitar sugerir que os itens relacionados formam uma sequência linear). Os autores destacam que a família de produtos e a família de processos são relacionadas. Os produtos podem ser caracterizados a partir dos processos usados para desenvolvê-los. Os processos, por sua vez, podem ser entendidos a partir dos produtos que eles geraram.

Como as definições propostas para linha de processos ainda não estão claras ou bem-definidas, neste trabalho adota-se, por enquanto, a seguinte definição para linha de processos:

> Um conjunto de elementos de processos que compartilham características comuns e variáveis dentro de um domínio específico e são desenvolvidas a partir de artefatos que podem ser reutilizados e combinados entre si, segundo regras de composição e recorte, para compor e adaptar processos.

Esta definição foi construída em conjunto com outros alunos de mestrado e doutorado envolvidos no estudo deste tema de pesquisa para produzir uma definição global de forma a garantir a integração dos resultados das pesquisas individuais. Para embasar e formalizar esta definição, bem como expandir o conhecimento sobre o uso e gestão de linhas de processos, uma revisão *quasi*-sistemática da literatura foi planejada e sua execução está em andamento, também como uma iniciativa em comum deste grupo de pesquisa.

Os objetivos de uso de linhas de processos são: aumentar a produtividade da atividade de adaptação de processos, diminuindo o esforço

necessário para realizá-la; aumentar a qualidade e adequação dos processos gerados (reutilização do conhecimento de especialistas e de dados sobre utilização); representar variabilidades e semelhanças entre processos para potencializar a reutilização; e diminuir os riscos de uma adaptação inadequada do processo (BARRETO *et al*., 2009, JAUFMAN e MUNCH, 2005, PEDREIRA *et al*., 2007, ROMBACH, 2006).

Os modelos de maturidade (CHRISSIS *et al*., 2006, SOFTEX, 2009) estabelecem que os processos também podem ser definidos com base em unidades menores e reutilizáveis de processos. Assim, os processos são decompostos em componentes de processo ou atividades (BARRETO *et al*., 2009). Esses dois conceitos têm significados bastante similares, sendo que a principal diferença é que um componente é definido para reutilização e a atividade é a menor atividade na decomposição.

Um componente de processo (do inglês *process component* ou *process bean* (MAGDALENO *et al*., 2007)) pode ser visto como um encapsulamento de informações e comportamento de processo em um dado nível de granularidade (GARY e LINDQUIST, 1999). Assim, um processo pode ser tratado como a integração de um conjunto de componentes de processo em diferentes níveis de granularidade (FUSARO *et al*., 1998).

Fusaro et al. (1998) consideram que um componente de processo pode representar: (i) uma técnica, isto é, um algoritmo ou série de passos cuja execução requer algum conhecimento e habilidade e que produz um determinado efeito; (ii) um método, isto é, um procedimento gerencial específico para aplicar uma técnica, organizado como um conjunto de regras que avaliam, selecionam e estabelecem como e quando usar e parar de aplicar as técnicas que inclui; ou (iii) um processo, isto é, um conjunto de métodos inter-relacionados necessários para atingir um objetivo específico.

Para o SPEM (*Software & Systems Process Engineering Meta-Model*) (OMG, 2008), um componente de processo é considerado um agrupamento de elementos de processo que é internamente consistente e pode ser reutilizado com outros componentes de processo para compor um processo completo. Deve ser auto-contido, ou seja, não deve haver referências de dentro de um componente para elementos que estão fora do componente. Com essa

definição abrangente, a estrutura de um componente de processo é bastante variável.

Barreto (2007) resume a situação mostrando que os componentes de processo são descritos e estruturados de diferentes formas por diferentes autores, não existindo um consenso sobre quais informações devem estar contidas em um componente nem sobre o nível de detalhe a ser utilizado, ficando essas informações dependentes do uso pretendido para os componentes por cada abordagem.

5.3. Propostas para linha de processos

Esta seção apresenta alguns trabalhos referentes ao tema linha de processos e relacionados aos interesses da presente pesquisa.

Jaufman e Munch (2005) apresentam uma abordagem para adaptação do processo padrão para projetos específicos, baseando-se em linhas de processos. O método consiste de dois passos principais. No primeiro passo, uma linha de processos específica de um domínio é usada para a adaptação *top-down* realizada no início do projeto. O propósito de se utilizar a linha de processos é fornecer o conhecimento do domínio necessário para definir um processo de software adequado. Em seguida, o processo definido é revisado com base nos dados coletados durante sua execução.

Como parte de uma abordagem de definição de processos baseada em reutilização, a proposta de Barreto *et al*. (2009) se preocupa em componentizar processos legados, definidos e executados nas organizações, para criar as linhas de processos e facilitar a sua reutilização e definição futura dos processos. Além disso, prevê a existência de uma biblioteca de componentes reutilizáveis de processos para auxiliar na definição tanto do processo padrão quanto do processo definido para o projeto.

Este foco do trabalho de Barreto *et al*. (2009) nos componentes de processos segue a mesma estratégia adotada por outros trabalhos sobre linhas de produtos que privilegiam a arquitetura (GARG *et al*., 2003) e a fase de projeto do domínio, em detrimento da fase de análise de domínio.

Rombach (2006) propôs a **engenharia de linha de processos**, cujo objetivo é, por meio de um processo de engenharia de domínio, criar um

conjunto de processos genérico que capturasse as semelhanças e variabilidades em um domínio. A engenharia de linhas de processo inclui: o processo de *engenharia de domínio*, que cria processos para reutilização; o processo de *engenharia de aplicação*, que cria processos específicos para um projeto; e um *repositório de artefatos* que disponibiliza processos reutilizáveis em todos os níveis de abstração. As propostas de Rombach (2006) e Barreto *et al*. (2009) têm em comum a preocupação com a existência de um repositório de componentes de processos.

A necessidade de uma abordagem sistemática e que separe as partes estáveis e variáveis do processo, apontada por Rombach (2006), também é defendida por Armbrust *et al*. (2008). A proposta, especificamente voltada para linha de processos de software, inclui as seguintes etapas: (i) determinar os membros da linha de processos; (ii) utilizar um processo de engenharia de domínio para construir um repositório de processos; (iii) instanciar a linha de processos extraindo uma instância de processo; e (iv) customizar/adaptar a instância do processo. Esta abordagem trata tanto da criação da linha de processo quanto da instanciação de processos a partir da linha.

Washizaki (2006) define a engenharia de linha de processos como “um sistema de estratégias inter-relacionadas e abordagens sistemáticas para construir, aplicar e gerenciar a linha de processos”. O autor propõe uma abordagem *bottom-up* para estabelecer linhas de processos e arquiteturas de linhas de processos a partir dos processos existentes, que incorporam semelhanças e variabilidades (Figura 5.3). Nesta proposta, as características do projeto também são levadas em consideração.

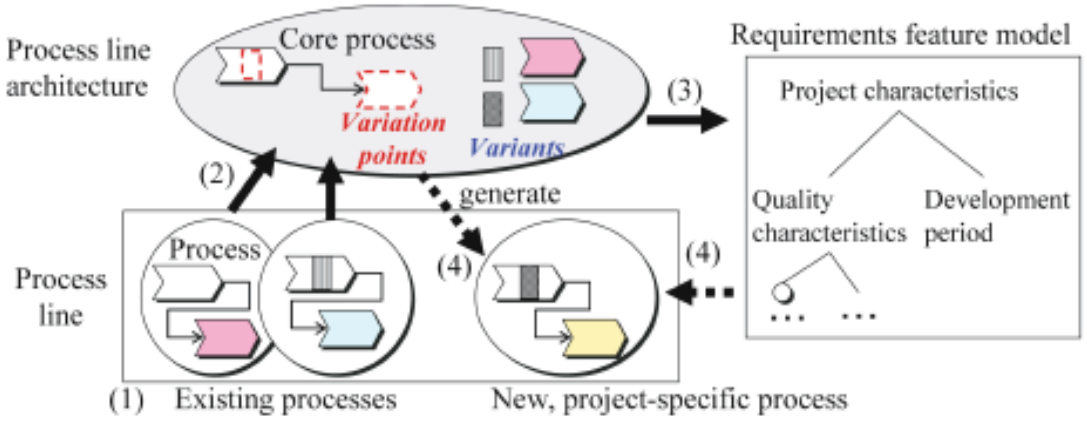


Figura 5.3 – Abordagem *bottom-up* para linha de processos (WASHIZAKI, 2006)

A análise destes trabalhos permitiu observar que o conceito de linha de processos é recente e que ainda há poucas publicações sobre o assunto. Além disso, é ressaltada a importância de se considerar tanto a criação da linha de processos quanto a instanciação de processos a partir da linha na abordagem de engenharia de linha de processos. Em particular, a fase de análise de domínio ainda não foi devidamente explorada, pois a maioria das propostas existentes se concentra na arquitetura dos componentes de processos.

5.4. Abordagem de engenharia de linha de processos

Os processos de software – apesar de também serem variáveis entre os projetos de software – ainda não são gerenciados da mesma forma sistemática que os produtos de software (ROMBACH, 2006). Neste trabalho, esta sistemática é estabelecida valendo-se de uma estrutura de reutilização de processos baseada em linha de processos. Desta forma, os processos de software das organizações também podem ser organizados de acordo com as suas similaridades e diferenças, facilitando e guiando a sua adaptação às necessidades específicas dos projetos.

Desta forma, conforme visto anteriormente, a segunda sub-hipótese (H1.2) formulada para esta pesquisa é:

É possível definir uma abordagem sistemática para apoiar a tomada de decisão sobre a adaptação de processos de desenvolvimento de software baseada na reutilização de processos.

Ainda que esta abordagem sistemática baseada em linha de processos seja necessária para apoiar a tomada de decisão, não é foco do presente trabalho de pesquisa construí-la ou implementá-la. Então, existem dois outros trabalhos, um de mestrado e outro de doutorado (NUNES *et al*., 2010b), relacionados a este tema.

A abordagem de Engenharia de Linha de Processos (ELP) foi concebida como um trabalho em comum deste grupo de estudo e compreende toda a estrutura necessária para construir, utilizar e gerir uma linha de processos. A ELP é formada por quatro elementos principais: um meta-modelo, uma notação, um método e um ferramental de apoio (Figura 5.4). Estes elementos são descritos nas próximas seções. Como o interesse do presente trabalho

está concentrado no método, este será o elemento explicado em mais detalhes.

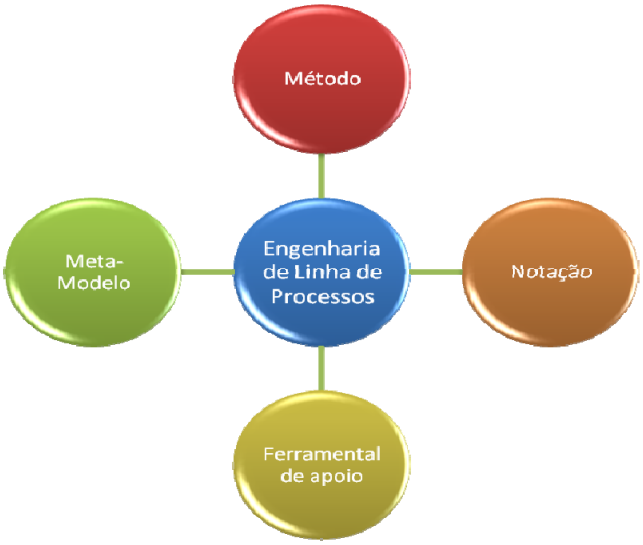


Figura 5.4 – Elementos da abordagem de Engenharia de Linha de Processos

5.4.1. Meta-Modelo

O meta-modelo (Figura 5.4) é formado pelos conceitos do domínio e visa estabelecer o vocabulário comum que é essencial para que as pessoas compreendam umas as outras. Portanto, é importante estabelecer relações semânticas entre os conceitos utilizados na definição dos processos, e assim viabilizar a utilização destes na organização.

Desta forma, o meta-modelo funciona como uma fonte unificada de referência, ou seja, como um repositório de conhecimento compartilhado. A representação deste meta-modelo pode variar de acordo com o seu propósito. Assim, podem existir representações com diferentes níveis de complexidade e formalismo. Por exemplo, modelos de classes, taxonomias e ontologias são representações possíveis para um meta-modelo.

Existem alguns exemplos de meta-modelo para o domínio de processos de desenvolvimento de software. Estes exemplos já lidam com o problema de estabelecer um entendimento comum sobre processos de software. Assim, diante das opções já existentes, não será proposto nenhum novo meta-modelo. Apenas são apresentadas as duas opções mais relevantes para este trabalho.

O meta-modelo para o domínio de processos de desenvolvimento de software mais conhecido e aceito atualmente é o SPEM (OMG, 2008) proposto pela OMG (*Object Management Group*) para representar processos de desenvolvimento de software e seus componentes. O escopo do SPEM é

limitado aos elementos mínimos necessários para definir um processo de desenvolvimento de software sem adicionar características específicas para determinados subdomínios ou disciplinas.

O SPEM pode fornecer os conceitos necessários para modelar, documentar, apresentar, gerenciar e executar processos de desenvolvimento. Além disso, ele descreve uma linguagem e um esquema de representação. O SPEM segue uma abordagem orientada a objetos para modelar famílias de processos relacionados e a sua especificação é estruturada como um perfil (do inglês *profile*) UML 2 (*Unified Modeling Language*) (OMG, 2008).

Outro meta-modelo que se destaca é a ontologia para processos de desenvolvimento de software, construída por Falbo (1998) e evoluída posteriormente (FALBO e BERTOLLO, 2005). Esta ontologia ajuda na identificação dos conceitos, e dos relacionamentos entre eles, presentes no domínio de processos de desenvolvimento de software. Esta ontologia já indicou ter correspondência com os elementos propostos pelo SPEM (OMG, 2008). Um modelo parcial dessa ontologia é apresentado pela Figura 5.5, onde os conceitos em cinza foram criados na evolução do trabalho.

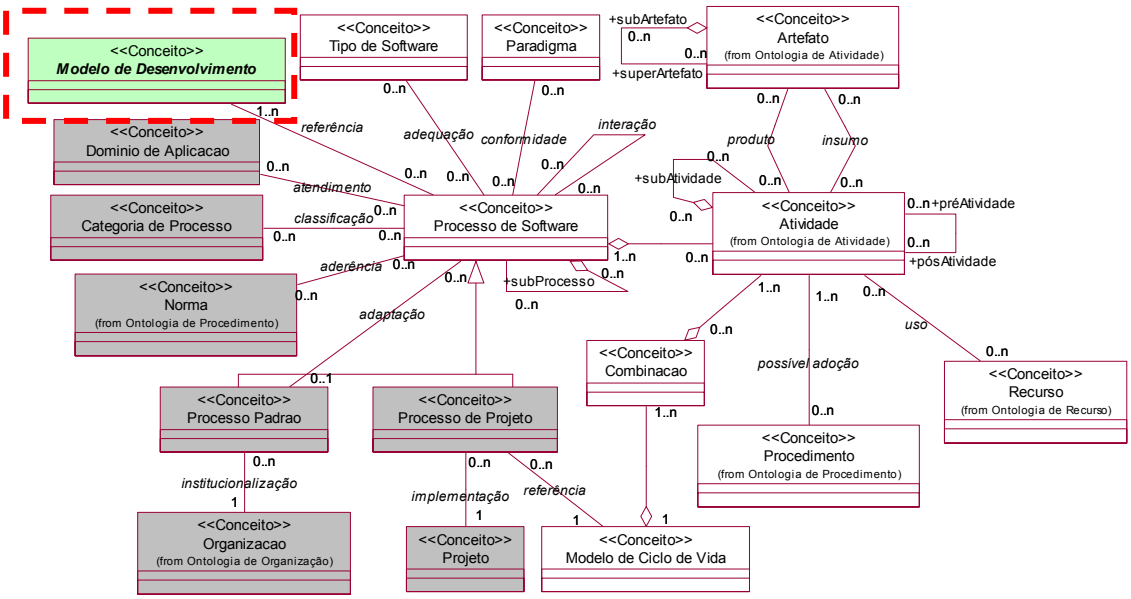


Figura 5.5 – Modelo parcial da ontologia de processos de software. Adaptado de: (BERTOLLO *et al.*, 2006)

Apesar de contemplar os principais conceitos relevantes para o domínio de processos de desenvolvimento de software, esta ontologia de processos de desenvolvimento de software não contempla os diferentes modelos de desenvolvimento atualmente existentes, tais como o desenvolvimento ágil e

livre, pois ela é direcionada para o desenvolvimento orientado ao planejamento. Desta forma, foi necessário introduzir um novo conceito “Modelo de Desenvolvimento” ligado a “Processo de Software” e destacado na Figura 5.5.

5.4.2. Notação

A notação (Figura 5.4) estabelece a simbologia para representar os conceitos definidos no meta-modelo. A notação OdysseyProcess-FEX que será adotada na abordagem de engenharia de linha de processos proposta está sendo definida em uma dissertação de mestrado sendo inspirada no SPEM.

5.4.3. Método

O método (Figura 5.4) estabelece a sequência de passos para a criação da linha de processos e para a instanciação de processos a partir da linha. De forma análoga a engenharia de linha de produtos, a engenharia de linha de processos também compreende duas grandes fases (MONTERO *et al*., 2007, ROMBACH, 2006, SCHNIEDERS e PUHLMANN, 2006): Engenharia de Domínio de Processos (EDP) e Engenharia de Aplicação de Processos (EAP) (Figura 5.6).

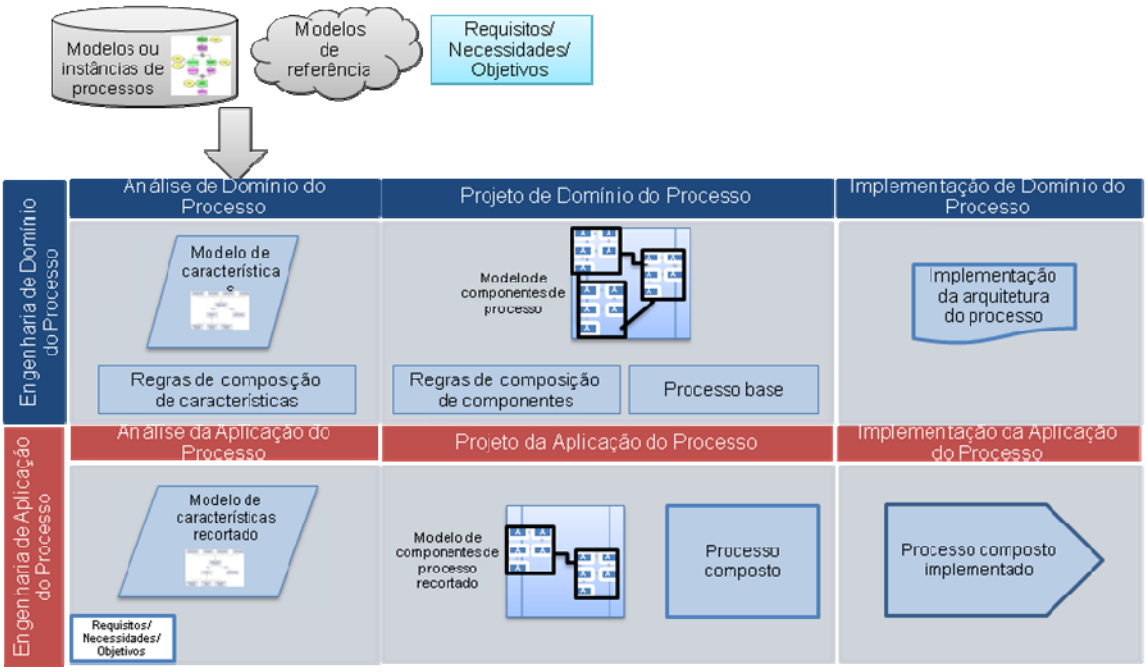


Figura 5.6 – Proposta de abordagem para engenharia de linha de processo

Como a engenharia de linha de processos foi estruturada de acordo com as macro-atividades e os produtos gerados em cada fase, nesta seção é apresentado o ciclo completo. Em particular, a fase de análise de domínio é bastante explorada, pois ela não é devidamente tratada nas demais propostas.

Devido à sua importância, o estudo desta fase é percebido pelo grupo de pesquisa como uma oportunidade e um diferencial desta abordagem.

Além disso, esta fase já tem sido explorada pelo grupo de pesquisa através de uma dissertação de mestrado que investiga como os conceitos e artefatos podem ser utilizados, adaptados ou implementados no ferramental de apoio já existente (ODYSSEY, 2010).

Porém, ressalta-se que esta abordagem está sendo utilizada nesta proposta de tese apenas como apoio à decisão na adaptação de processos. Portanto, as fases mais importantes para o presente trabalho de pesquisa são a análise e o projeto de aplicação do processo, pois é justamente nestas duas fases que o apoio à decisão irá atuar para ajudar na seleção de características e na composição dos componentes de processos.

5.4.3.1. Estratégias de captura de conhecimento

Existem duas estratégias de captura de conhecimento possíveis para a ELP: *bottom-up* e *top-down* (ROMBACH, 2006, WASHIZAKI, 2006) que se diferenciam basicamente pelos insumos utilizados. Na estratégia *bottom-up,* também adotada nas propostas de Washizaki (2006) e Barreto *et al*. (2009), o conhecimento existente nos modelos de processos da organização ou nas instâncias executadas dos projetos de desenvolvimento de software é recuperado, através de análise ou mineração dos processos (*process mining*) (AALST e GIINTHER, 2007, AALST *et al*., 2004), e utilizado para criar a linha de processos. As técnicas de mineração de processos podem automatizar e ajudar a reduzir substancialmente o esforço manual necessário para descobrir e modelar o processo. Já existem trabalhos que tratam inclusive da mineração de processos de desenvolvimento de software (JENSEN e SCACCHI, 2007). Porém, nesta estratégia enfrenta-se o risco da memória organizacional sobre o processo não estar estruturada ou acessível.

Por sua vez, a estratégia *top-down*, inicialmente adotada neste trabalho, reflete a necessidade de padronização de processos. Nesta estratégia, inicia-se a linha de processos a partir de modelos de referência, onde processos ou partes deles são adquiridos de forma a serem combinados para formar novos processos. Neste caso, podem ser utilizados, por exemplo, métodos ágeis, tais como XP (BECK, 1999), Crystal (COCKBURN, 2004) e Scrum (SCHWABER, 2004).

Entretanto, Washizaki (2006) alerta para o risco de se realizar a análise do domínio, a partir destes modelos de referência, e não conseguir garantir a sua completude devido a sua extensão. Segundo Jaufman e Munch (2005), nesta estratégia deve-se manter um alto nível de adaptabilidade do processo, uma vez que inicialmente a linha de processo criada é menos madura.

5.4.3.2. Engenharia de Domínio de Processo (EDP)

A primeira fase EDP provê um conjunto de artefatos e infraestrutura para a derivação dos processos na próxima fase de EAP. A EDP é composta pelas atividades de análise do domínio do processo, projeto do domínio do processo e implementação do domínio do processo (SCHNIEDERS e PUHLMANN, 2006) (Figura 5.6).

a) Análise de Domínio do Processo

A primeira macro-atividade, análise de domínio do processo, busca identificar, coletar e organizar informações relevantes do domínio, utilizando para tal o conhecimento existente do domínio e métodos para a modelagem de informações (KANG *et al*., 1990). Desta forma, o objetivo é gerar, a partir dos insumos descritos anteriormente, o **modelo de características**. As **regras de composição de características** também são estabelecidas para definir as restrições entre as características.

O modelo de características, visto na seção 5.1.1, comumente usado para representar variabilidade nas linhas de produtos, também será usado com este mesmo propósito nas linhas de processos. Porém, é necessário definir o que se pretende representar neste modelo.

A partir do conceito Processo de Software, também presente no meta-modelo, iniciou-se o modelo de características pela representação das disciplinas (Figura 5.7). Uma disciplina é uma coleção de atividades relacionadas a uma área de interesse ou área de conhecimento (IBM, 2009). Esta ideia de trabalhar com as disciplinas é uma forma simples de organizar o conteúdo e facilitar a compreensão.

Neste exemplo, foram retratadas algumas das disciplinas do RUP (*Rational Unified Process*) (IBM, 2009). Na Figura 5.7 as disciplinas foram representadas como características funcionais e quase todas, a exceção da "Implementação", foram definidas como opcionais para que seja possível no

momento da instanciação do processo selecionar apenas aquelas que interessam para o projeto de desenvolvimento em questão. A disciplina "Implementação" foi definida como obrigatória por ser considerada essencial para qualquer projeto da organização. Neste caso, ela não poderá ser excluída durante a adaptação do processo.

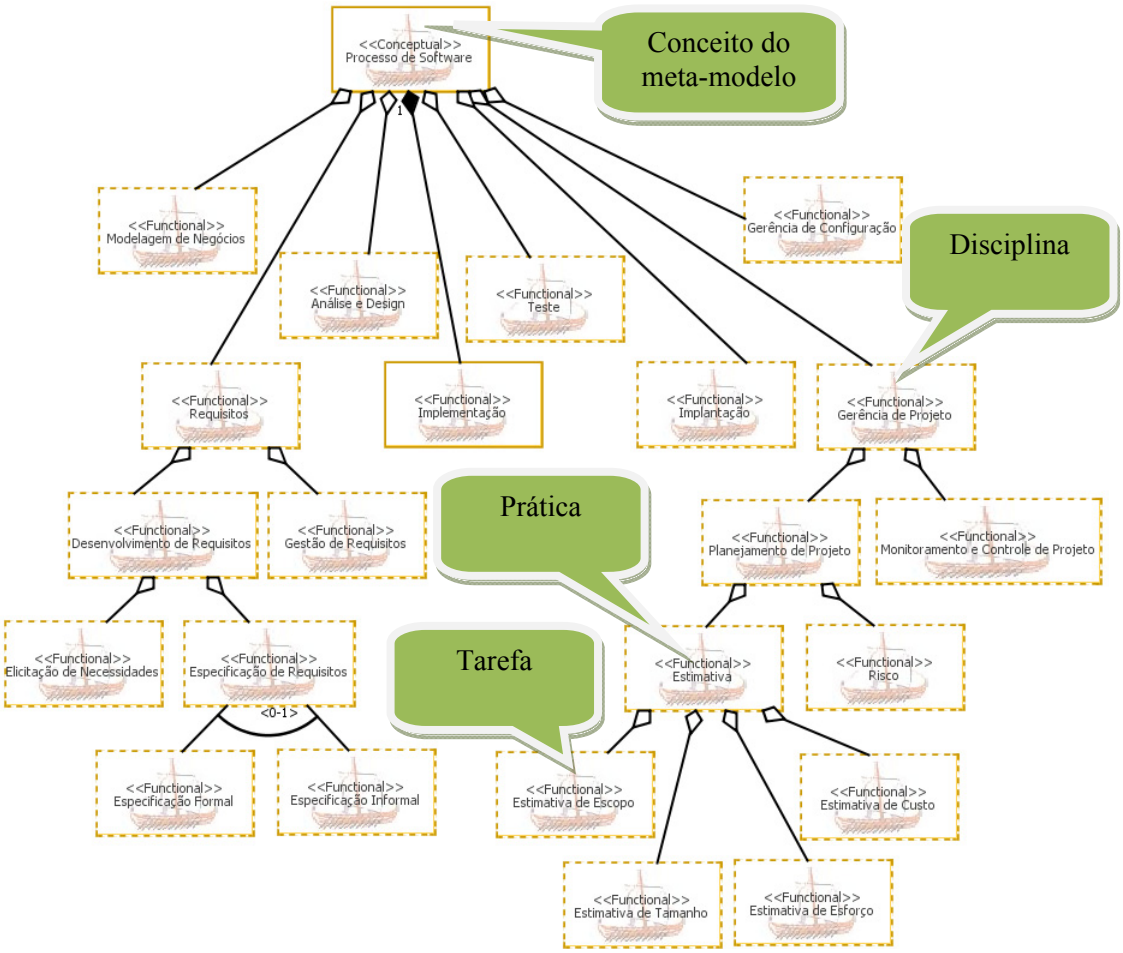


Figura 5.7 – Exemplo de modelo de características de processo na notação Odyssey-FEX representando disciplinas e práticas

Uma vez identificadas as disciplinas, é possível identificar as suas principais práticas. As práticas correspondem a atividades em alto nível de abstração, ou seja, àquilo que é feito no dia-a-dia de desenvolvimento. As práticas dependem da situação, do contexto e se esta situação muda é preciso selecionar práticas diferentes para abordar estas condições (TELES, 2004).

Neste sentido, alguns exemplos de práticas, focando nas disciplinas de gerência de projetos e requisitos, foram modelados no diagrama apresentado na Figura 5.7. No caso da prática "Planejamento de Projeto" foram identificadas quatro possíveis tarefas dedicadas a estimativas de escopo, tamanho, esforço e custo. Já a prática "Especificação de Requisitos" possui tarefas referentes à

“Especificação Formal” e “Especificação Informal” (Figura 5.7). Obviamente, este exemplo foi criado tendo como fonte de inspiração o RUP, mas ele é flexível o suficiente, já que grande parte do seu conteúdo é opcional ou variável, para permitir a posterior inclusão de outras disciplinas e práticas voltadas a outros modelos de desenvolvimento.

Este exercício de modelagem permitiu observar que os conceitos e elementos de variabilidade, opcionalidade e cardinalidade podem ser aproveitados das linhas de produtos, sem necessidade de adaptações. Porém considerando-se o uso da notação Odyssey-FEX (OLIVEIRA, 2006), os conjuntos de categorias e de relacionamentos precisarão ser revistos para representar e distinguir os novos conceitos (disciplina, prática e tarefa) relacionados a processos.

A representação atual, que não foi criada pensando em características para processos e sim para produtos, não permite esta distinção explícita, o que dificulta a leitura e o entendimento do modelo, como pode ser percebido no exemplo da Figura 5.7. A especificação completa da proposta de adaptação da notação Odyssey-FEX (OLIVEIRA, 2006), criando a OdysseyProcess-FEX para que esta possa representar processos de software, bem como a sua implementação no ferramental de apoio, é um trabalho de mestrado.

b) Projeto de Domínio do Processo

O projeto de domínio do processo é a etapa em que modelos de projeto são construídos, com base nos modelos de análise. Esta etapa compreende as atividades de criação de componentes, agrupamento destes e geração da arquitetura de referência do domínio, a qual será reutilizada por diferentes processos derivados deste domínio.

Esta é a principal atividade de criação da linha de processos uma vez que nela ocorre a criação da arquitetura do processo. A arquitetura de processos descreve os componentes de processos reutilizáveis e as suas interfaces. A variabilidade é representada na arquitetura também através dos pontos de variação, variantes e invariantes.

Semelhanças em uma arquitetura de linha de processos são representadas pelo **processo base** (*core process*) (Figura 5.6) (SIMIDCHIEVA *et al*., 2007, WASHIZAKI, 2006) ou *template* (REIS, 2002), que é formado por

partes comuns (elementos obrigatórios) de um conjunto de processos. O processo base é um conjunto de atividades comuns para um grande número de processos (MAGDALENO *et al*., 2007).

Neste caso, o processo base pode ser interpretado pelo conjunto de atividades mandatórias no modelo de características que indicam as similaridades entres os processos. Então, retomando o exemplo da Figura 5.7, o processo base seria composto apenas pela atividade de Implementação que é a única mandatória no modelo.

Diversas instanciações diferentes são possíveis para o mesmo processo base, ou seja, diferentes componentes ou atividades que atendam à estrutura estabelecida podem ser utilizados, dependendo da situação. Portanto, o processo base tem o objetivo de facilitar e guiar a adaptação de processos, estabelecendo uma estrutura básica a ser seguida. Entretanto, o processo base deve ser flexível o suficiente para permitir variadas instanciações, adaptadas a diferentes contextos que podem ocorrer em organizações ou projetos específicos (BARRETO, 2007).

Em seguida, é possível construir o **modelo de componentes de processo** (Figura 5.6) que apresenta os componentes contendo atividades invariantes e variantes, propriedades internas e interfaces com outros componentes. Um componente de processo é uma unidade básica de composição de processos e é considerado algo relevante para ser reutilizado em outras definições de processo.

Um componente de processo pode ser de dois tipos (BARRETO *et al*., 2009, MAGDALENO *et al*., 2007): (i) concreto (ou *blackbox bean*) quando não admite qualquer variabilidade, ou seja, precisa ser executado da forma que está descrito, sem qualquer variação. Em um componente concreto não há mais decisões a serem tomadas, pois tudo a ser definido no componente já o foi, e ele já pode ser utilizado diretamente em um projeto; ou (ii) abstrato (ou *framework bean*) quando admite variabilidades, não estando vinculado a uma única forma de realização e pode ser instanciado de diferentes maneiras.

No modelo de componentes de processos, a variabilidade também é representada pelos pontos de variação e variantes do processo (SCHNIEDERS e PUHLMANN, 2006). Pontos de variação são componentes de processos que podem ser modificados de acordo com as características de uma utilização

específica, ou seja, admitem diversas estruturas internas, que podem ser realizadas por diferentes variantes. Um componente abstrato em uma arquitetura de processos é um ponto de variação. Variantes de processo são os componentes candidatos que são aplicados nos pontos de variação, ou seja, uma variante é uma das possíveis implementações de um ponto de variação (BARRETO *et al*., 2009, WASHIZAKI, 2006).

Para executar cada uma das tarefas configuráveis do processo base, podem ser adotadas diferentes abordagens de acordo com as necessidades e padrões da organização e do projeto. Por exemplo, para compor a tarefa "EstimativaTamanho", existiriam três opções de componentes a serem escolhidos de acordo com a técnica que se deseja empregar: "EstimarLOC", "EstimarPPF" e "EstimarCOCOMO". Cada possibilidade constitui uma variante e é representada no modelo de componentes do processo (Figura 5.8). Portanto, será possível realizar a atividade "EstimativaTamanho" de três formas distintas nos projetos, dependendo do componente escolhido na adaptação de seu processo.

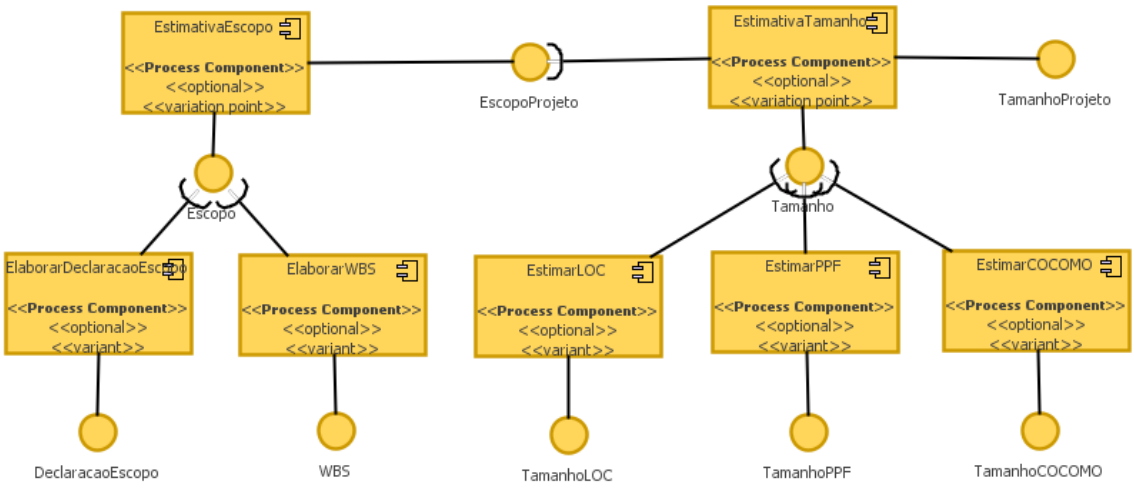


Figura 5.8 – Exemplo de modelo de componentes de processo

No exemplo da Figura 5.8, foi adotada a notação Odyssey-FEX, que segue as definições da UML2 para o diagrama de componentes (BLOIS, 2006), indicando os componentes de processo, seus tipos (através dos estereótipos: "concreto" ou "abstrato"), interfaces (artefatos) requeridas e produzidas e variabilidades.

Finalizando a atividade de projeto do domínio, são estabelecidas as **regras de composição de componentes** (Figura 5.6). Estas regras restringem ou obrigam a composição de determinados componentes. As regras

ajudam a garantir que sempre que um componente seja escolhido para reutilização, os componentes relacionados também possam ser selecionados. Do mesmo modo, dois componentes que sejam mutuamente exclusivos devem ser impedidos de serem utilizados conjuntamente. Por exemplo, pode existir uma regra determinando que o componente “ElaborarDeclaracaoEscopo” e o componente “ElaborarWBS” devem ser selecionados em conjunto para compor uma estimativa de escopo completa.

c) Implementação de Domínio do Processo

Na implementação de domínio do processo, a arquitetura de processos é efetivamente implementada no ambiente escolhido (Figura 5.6), para ser utilizado na composição e na adaptação dos processos. A linha de processos é, de fato, criada.

5.4.3.3. Engenharia de Aplicação do Processo (EAP)

Após criar a linha de processos na fase anterior, para cada novo projeto, ao invés de criar um processo manualmente, a organização usa os mecanismos de composição da linha de processos para ajudar na adaptação do processo. A EAP apresenta atividades análogas àquelas apresentadas na EDP. No entanto, uma vez que a EAP consiste na reutilização de artefatos gerados na EDP, são enfatizadas as características particulares de cada processo. Vale lembrar que é nesta fase, especificamente nas atividades de análise e projeto da aplicação, que se concentra o apoio à decisão da adaptação de processos.

a) Análise de Aplicação do Processo

A atividade de análise de aplicação do processo inicia a EAP, ou seja, inicia o uso da linha de processo para a composição dos processos que serão executados. A partir dos objetivos e requisitos para a adaptação do processo e do contexto do projeto, as características necessárias ao novo processo que será composto são selecionadas pelo gerente de projeto. As fases posteriores são regidas por estas escolhas, na medida em que elas restringem o escopo de informações do domínio que farão parte do processo composto. Portanto, foi identificada a necessidade de apoiar o gerente de projeto nesta tarefa.

A tarefa de obtenção de um processo a partir da linha de processos é chamada de derivação ou recorte. O artefato gerado é o **modelo de características recortado** (Figura 5.6), ou seja, contendo apenas as

características selecionadas para o processo em questão.

Considerando o exemplo do modelo de características apresentado na Figura 5.7, um possível recorte deste modelo de características é apresentado pela Figura 5.9, onde à medida que as características são selecionadas na árvore a esquerda, estas características são assinaladas com bandeirinhas na direita oferecendo, assim, maior visibilidade.

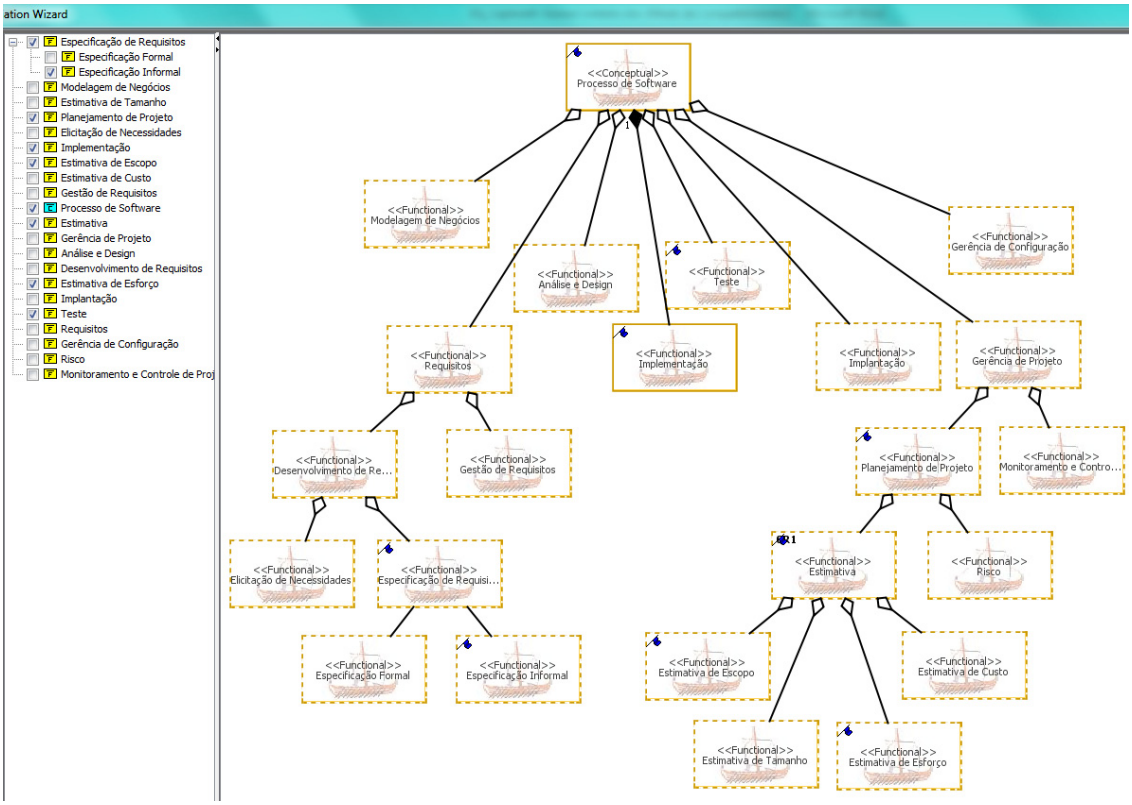


Figura 5.9 – Exemplo de modelo de características recortado

O modelo de características possui um alto nível de abstração, sendo utilizado como ponto de partida para o recorte necessário à instanciação de novos processos. A partir do modelo de características, o recorte vai descendo gradualmente em diferentes níveis de abstração. Assim, este recorte é fortemente influenciado pela variabilidade modelada no domínio e pela rastreabilidade estabelecida entre os diferentes artefatos.

b) Projeto de Aplicação do Processo

No projeto de aplicação do processo, a arquitetura criada na linha de processos é usada para instanciar a arquitetura correspondente ao processo atual. Isto significa que nesta fase cabe ao gerente do projeto, com o apoio do sistema de tomada de decisão, levando em consideração características da organização, do projeto e da equipe, **recortar os componentes de processos** (Figura 5.6),

selecionando aqueles que mais se adéquem ao contexto. Estes componentes serão utilizados para compor o processo específico do projeto. A partir do modelo de componentes recortado e das regras de composição existentes, o novo **processo** é **composto** (Figura 5.6).

c) Implementação de Aplicação do Processo

Na última macro-atividade da EAP, ou seja, na implementação de aplicação do processo, o **processo** composto é efetivamente **implementado** e está pronto para ser executado e adaptado em tempo de execução (Figura 5.6). Neste caso, pode-se optar, por exemplo, pela adoção de uma ferramenta de workflow ou BPMS (*Business Process Management System)* (ARAUJO e BORGES, 2001, MAGDALENO *et al*., 2007, PUNTAR *et al*., 2009) para automatizar a execução do processo.

Esta ferramenta pode ser, por exemplo, a Charon (MURTA, 2002) que oferece recursos para a modelagem, instanciação, simulação, execução, acompanhamento e evolução de processos de software. A ferramenta utiliza como notação para a representação gráfica de processos, o diagrama de atividades da UML. O modelo gráfico é traduzido para Prolog, pois esta é uma linguagem executável, declarativa e baseada em inferência. Além disso, a Charon adota uma arquitetura baseada em agentes inteligentes e uma ontologia para a comunicação entre estes agentes.

5.4.4. Ferramental de apoio

Por fim, como último elemento da abordagem de engenharia de linha de processos (Figura 5.4), propõe-se a adoção do Odyssey (ODYSSEY, 2010) como ferramental de apoio. O ambiente Odyssey, desenvolvido pelo Grupo de Reutilização de Software da COPPE/UFRJ, é uma infraestrutura de reutilização baseada em modelos de domínio e contempla atividades de ED e EA.

Apesar desta infraestrutura ter sido concebida com foco em produtos e não processos, ela já oferece diversos mecanismos para apoiar cada uma das fases da abordagem, ainda que alguns deles precisem de adaptações ou extensões. Tal fato pode ser observado pelos exemplos construídos e apresentados nas seções anteriores que utilizaram o Odyssey.

Além disso, a Charon (MURTA, 2002), citada como alternativa para o apoio à execução dos processos, foi construída dentro do ambiente Odyssey e

já associa o processo de desenvolvimento de software automatizado pela Charon ao desenvolvimento baseado em reutilização (ED e EA) apoiado pelo Odyssey.

5.5. Considerações finais

Este capítulo apresentou o conceito de linha de processos e seus respectivos elementos. Além disso, também foi apresentada e instanciada para o domínio de processos de desenvolvimento de software, a abordagem de engenharia de linha de processos, com foco nas fases de análise e projeto da aplicação.

Com o uso da linha de processos, espera-se que ao derivar um processo, o esforço requerido para a adaptação deste processo seja diminuído (WASHIZAKI, 2006). A pesquisa de Hollenbach e Frakes (1996) mostra que é possível diminuir em pelo menos dez vezes o tempo e o esforço necessário para criar a descrição de processo de um projeto quando se instancia um processo reutilizável, ao invés de criá-lo desde o início.

Também é importante destacar que não é intenção deste trabalho impor uma disciplina indevida ou indesejada aos processos adaptados para os projetos. Apesar da abordagem de engenharia de linha de processos ser sistemática, o que pode causar a falsa impressão de rigor na definição do processo, os componentes de processos representam práticas de desenvolvimento e estas práticas podem refletir qualquer modelo de desenvolvimento de software.

Por fim, conforme mencionado anteriormente, também é objetivo deste trabalho de pesquisa mostrar a interação da engenharia de linha de processos com a gestão de contexto, visando integrar as partes da solução deste trabalho, para que as informações de contexto de colaboração e disciplina possam ser geridas. Este tema é parte do próximo capítulo, dedicado à gestão de contexto.

6. Gestão de Contexto

Este capítulo apresenta os conceitos básicos e uma abordagem de gestão de contexto. Em seguida, são exemplificadas as dimensões, informações, situações e regras de contexto identificadas para o domínio de processos de desenvolvimento de software. Ainda neste capítulo é mostrada a integração da linha de processos com a gestão de contexto. Por fim, é ilustrado o ferramental de apoio que será adotado para apoiar a representação de contexto.

6.1. Contexto

Vários pesquisadores de diferentes áreas tentam formalizar a definição de contexto (BREZILLON e POMEROL, 1999, BREZILLON, 1999, CHEN e KOTZ, 2000, SCHMIDT *et al*., 1999), mas com visões diferentes. Bazire e Brézillon (2005) catalogaram mais de 150 definições de contexto e observaram que elas variam de acordo com os diferentes domínios.

Uma definição comumente adotada declara que contexto é qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade, onde uma entidade pode ser uma pessoa, lugar, ação ou objeto que seja considerado relevante para a situação (DEY *et al*., 2001). Neste trabalho, a definição adotada é a de Brezillon (1999), segundo a qual contexto é "uma descrição complexa do conhecimento compartilhado sobre circunstâncias físicas, sociais, históricas e outras dentro das quais ações ou eventos ocorrem".

Além do entendimento da definição de contexto, algumas características importantes de contexto também precisam ser levadas em consideração: (i) contexto não pode ser completamente descrito, pois é infinito (MCCARTHY, 1993); (ii) contexto é dinâmico (KOKINOV, 1999).

Devido a essas características, Brezillon e Pomerol (1999) propuseram um modelo que classifica o contexto de acordo com o *foco de atenção* de uma situação em particular. O foco pode representar uma tarefa (NUNES, 2007), um passo na solução de um problema ou uma tomada de decisão. Eles argumentam que o contexto não pode ser considerado de forma isolada ou ampla demais. Ao invés disso, é o foco de atenção que vai determinar o que é relevante para um determinado contexto.

De acordo com o foco de atenção, os autores classificam o contexto em três partes (Figura 6.1). O *conhecimento externo* representa a parte do conhecimento que não tem relevância para o foco definido. Por exemplo, suponha que o foco de um usuário é encontrar especialistas para ajudá-lo em uma tarefa de desenvolvimento de software. Neste caso, o conhecimento externo pode incluir informações dos especialistas, tais como: peso e estado civil. Ainda que estas informações façam parte do conhecimento sobre os especialistas, elas não são úteis para o foco de atenção atual.

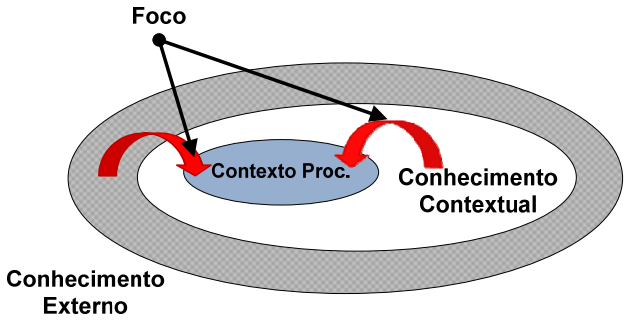


Figura 6.1 – Dinâmica do contexto. Adaptado de: (BREZILLON e POMEROL, 1999)

O *conhecimento contextual* representa o conhecimento que é imediatamente relevante para a tarefa e que possui uma forte relação com o foco de atenção em questão. No caso do exemplo anterior, o conhecimento contextual inclui informações sobre a localização, presença, disponibilidade, habilidade, reputação, experiência e linguagem de desenvolvimento dominada pelos especialistas.

Finalmente, o *contexto proceduralizado* é a parte do conhecimento contextual que é demandado, organizado e estruturado de acordo com o foco em questão. Esta é a parte do contexto que realmente é necessária para a situação atual. Assim, no caso do exemplo anterior isto significa saber que dois especialistas estão presentes, sendo que João está disponível e é especialista em Java, enquanto Carlos está ocupado e domina a linguagem Delphi.

Entretanto, para que o contexto possa ser realmente útil na prática, ele precisa ser gerido. Esta gestão do contexto é o tema da próxima seção.

6.2. Gestão de contexto

Na gestão de contexto, o contexto é estudado através de quatro diferentes aspectos (NUNES, 2007):

- **Definição de tipos de contexto, classificações e relacionamentos entre eles:** as informações sobre o domínio necessitam ser formalizadas para que o contexto possa ser gerenciado;
- **Mecanismos que reconheçam informação contextual:** como não existe uma forma única de reconhecimento e captura de informação, é necessário formalizar os mecanismos de captura que melhor se adequem à situação e informação contextual;
- **Aplicação de regras que identifiquem contextos relacionados ao atual:** para utilizar informações contextuais de atividades passadas, regras de inferência devem ser aplicadas de forma a selecionar aquelas situações iguais ou similares, que são passiveis de prover informações úteis;
- **Definição de modelos de apresentação de contexto que o tornem reutilizável:** as decisões sobre como será a explicitação dos contextos recuperados é um dos fatores de sucesso ou fracasso na reutilização e visualização das informações contextuais para um indivíduo ou grupo.

A partir destes aspectos, quatro questões referentes à gestão de contexto, apresentadas no ciclo da Figura 6.2, são examinadas (NUNES, 2007):

- Como reconhecer quais informações de contexto são necessárias?
- Como estabelecer os mecanismos de captura de contexto?
- Como representar o contexto?
- Como usar as informações de contexto?

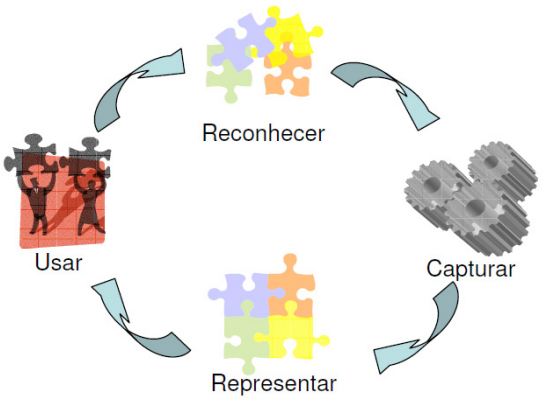


Figura 6.2 – Ciclo de transformação do contexto (NUNES, 2007)

A partir dos aspectos e questões e levantadas, é apresentada, na próxima seção, uma abordagem para a gestão de contexto.

6.3. Abordagem de gestão de contexto

O modelo de gestão de conhecimento baseado em contexto proposto por Nunes (2007), foi adaptado para ser utilizado na presente proposta de tese, ainda que esta abordagem possa vir a ser detalhada ou modificada durante o desenvolvimento da tese de doutorado de Nunes (NUNES *et al*., 2010b). A visão geral da abordagem de gestão de contexto é apresentada pela Figura 6.3. Esta abordagem é composta por cinco partes importantes, descritas brevemente a seguir e detalhadas em Nunes *et al*. (2010a).

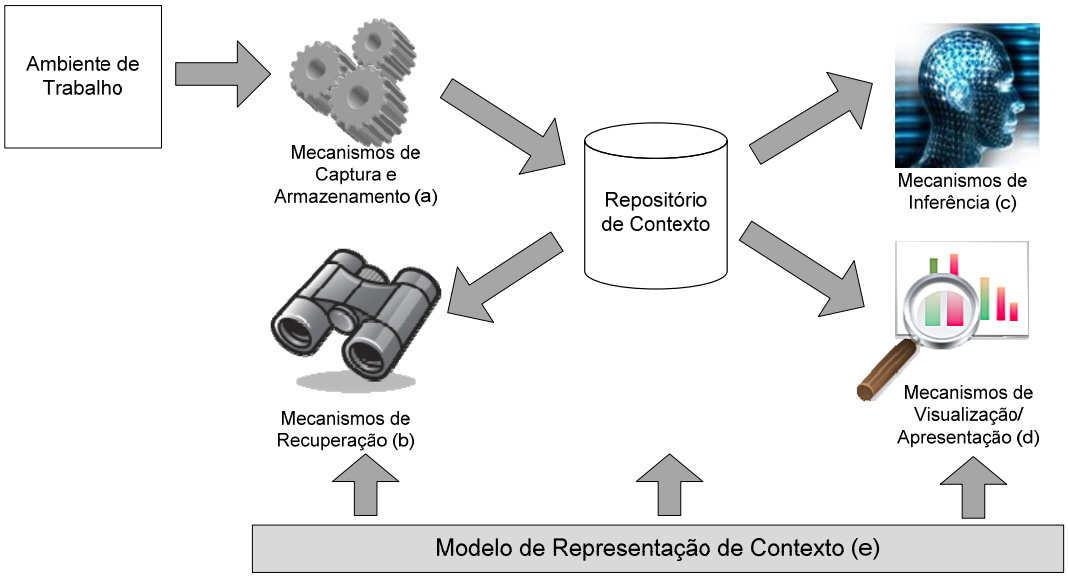


Figura 6.3 – Visão geral da abordagem de gestão de contexto

Nesta abordagem, as informações de contexto são capturadas do ambiente de trabalho, durante a adaptação ou execução dos processos, por mecanismos (Figura 6.3a) manuais (informações fornecidas pelo próprio agente humano), automatizados (agentes ou sensores capturam as informações que se encontram disponíveis no próprio ambiente) ou semi-automatizados (quando existe interação entre o profissional e o sistema).

Os mecanismos de armazenamento (Figura 6.3a) das informações de contexto capturadas alimentam o repositório de contexto, introduzindo contextos novos, que poderão ser consultados posteriormente quando ocorrerem situações semelhantes. Por sua vez, os mecanismos de recuperação (Figura 6.3b) são capazes de recuperar contextos, novos ou não, existentes na organização e registrados no repositório de contexto para auxiliar em situações similares. Esta recuperação permite que estes conhecimentos

prévios possam ser aproveitados na situação atual.

Com base nas informações de contexto recuperadas, os mecanismos de inferência (Figura 6.3c) podem processar regras de contexto existentes para eleger o contexto considerado útil para a situação em questão. Por exemplo, através da captura de informações contextuais os mecanismos de inferência podem sugerir uma tática específica para negociar com um cliente já conhecido na organização. Podem ainda sugerir a forma de tratar alguma exceção acontecida em contexto similar que foi bem sucedida.

Os mecanismos de visualização (Figura 6.3d) devem levar em consideração as diversas informações de contexto e os relacionamentos existentes entre elas, se houver. Os resultados das inferências processadas sobre as informações de contexto devem ser apresentados ao profissional ou grupo que está executando a atividade. A interação do indivíduo com a interface também pode gerar novas regras de contexto, contribuindo para que os mecanismos de inferência atuem no sentido de realizar uma seleção cada vez mais otimizada e proveitosa.

Por fim, para capturar e usar informações de contexto, é necessário primeiramente conhecer a sua semântica e especificar como elas serão representadas em um modelo compreensível e aceitável por todos (SANTOS, 2008). O modelo de representação de contexto (Figura 6.3e) explicita o conhecimento que está sendo capturado e deve adotar uma linguagem com semântica e formalismo que atendam as necessidades de representação.

A elaboração de um modelo de contexto com riqueza semântica ajuda a capturar a estrutura e o significado das informações de contexto e dos relacionamentos entre elas. Uma representação formal do contexto permite que mecanismos de raciocínio lógico possam ser usados para checar a consistência das informações de contexto, realizar comparações com outros contextos e inferir novas informações a partir dos contextos existentes.

6.4. Representação de contexto em processos de desenvolvimento de software

Um estudo realizado para identificar formas de representação das informações de contexto que caracterizam o domínio de processos de desenvolvimento de

software (NUNES *et al*., 2010a), incluiu tanto linguagens de representação quando ferramentas de apoio. Levando-se em consideração os resultados das análises realizadas por Fernandes (2008) e Santos (2008) sobre as vantagens e desvantagens das diferentes formas de representação de contexto existentes, duas linguagens foram selecionadas para serem exploradas: ontologias[11] e modelos de características (apresentado no capítulo anterior).

Em relação à ontologia, foi selecionada a linguagem OWL (*Web Ontology Language*) (SMITH *et al*., 2004) combinada com a ferramenta Protégé (PROTEGE, 2010). Entretanto, este par demonstrou uma complexidade de representação e capacidade de raciocínio que vai além das necessidades relacionadas ao objetivo do trabalho (NUNES *et al*., 2010a). Além disso, as ferramentas e padrões para manipular ontologias ainda não são de fácil uso e os mecanismos de inferência impactam diretamente no desempenho das aplicações (SANTOS, 2008).

Por outro lado, a modelagem de características é uma das técnicas mais aceitas para representar processos em uma linha (MONTERO *et al*., 2007). Assim, neste estudo de investigação, foi selecionada a linguagem Odyssey-FEX (OLIVEIRA, 2006) para a modelagem de domínio. Para a modelagem de contexto, foi utilizada a notação UbiFEX (FERNANDES, 2008). A UbiFEX é uma notação para modelagem de características que inclui de forma explícita elementos para a modelagem de contexto relevante para um determinado domínio. Além disso, inclui elementos para representar a influência dessas informações na configuração dos produtos (no caso deste trabalho, processos).

Neste cenário, a proposta de representação do conhecimento através de modelos de características, demonstrou possuir uma complexidade adequada de utilização e uma semântica suficientemente rica para atender aos propósitos do trabalho. Observou-se que este modelo de representação apresenta um grau de representatividade suficiente para abarcar as necessidades do trabalho que será realizado sobre processos de desenvolvimento de software, sem aumentar a complexidade da modelagem.

Assim, a abordagem UbiFEX é detalhada na próxima seção e utilizada como referência para a modelagem de contexto de processos de

[11] Uma ontologia é uma especificação formal de conceitos e termos do domínio com a definição de regras que regulam a combinação entre termos (GRUBER, 1995).

desenvolvimento de software apresentada nas seções seguintes.

6.4.1. Abordagem UbiFEX para representação de contexto

A abordagem UbiFEX (FERNANDES, 2008) tem como objetivo permitir a representação de contexto em modelos de características. Assim, a UbiFEX estende a notação Odyssey-FEX (OLIVEIRA, 2006) através da criação de duas novas características: entidade de contexto e informação de contexto.

A visão geral da abordagem, composta por quatro atividades, é apresentada na Figura 6.4. A primeira atividade é definir as entidades de contexto. As entidades de contexto (*context entity*) representam as dimensões de contexto relevantes para o domínio.

Em seguida, devem ser definidas as informações de contexto relevantes para o domínio em questão. As informações de contexto (*context information*) representam as características que devem ser coletadas para caracterizar as entidades de contexto do domínio (FERNANDES, 2008).

O terceiro passo é criar as definições de contexto (*context definition*), ou seja, caracterizar as situações específicas que podem acontecer. As definições de contexto descrevem situações relevantes para o domínio, tendo como base as entidades e informações de contexto modeladas anteriormente. A UbiFEX permite que uma nova definição de contexto seja definida a partir de outras previamente definidas, promovendo o reúso e facilitando a atividade de manutenção (FERNANDES, 2008).

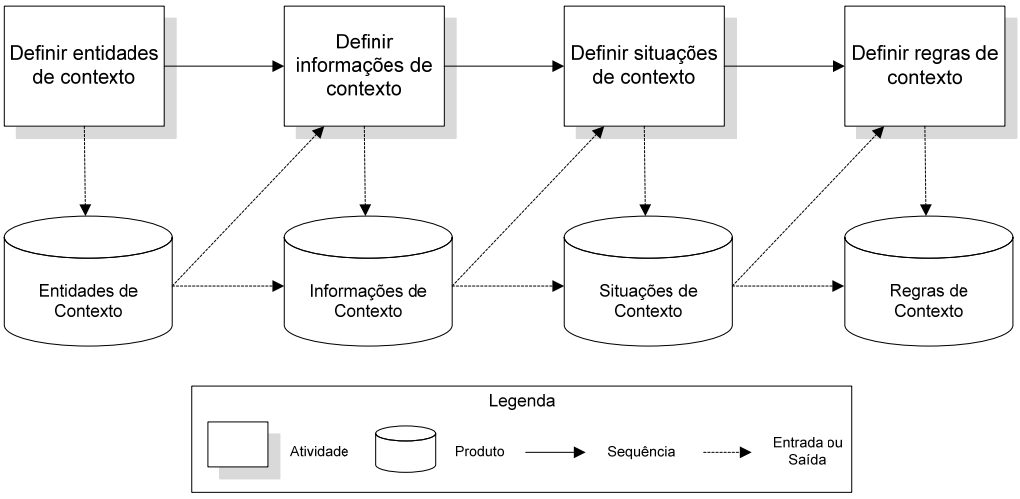


Figura 6.4 – Visão geral da abordagem UbiFEX

Por fim, as ações que devem ser tomadas para uma determinada situação podem ser especificadas pelas regras de contexto (*context rules*).

Estas regras definem como uma situação de contexto impacta na configuração de um processo, indicando, por exemplo, decisões a respeito da seleção de variantes em um ponto de variação.

As regras de contexto têm como propriedades: um identificador e uma expressão. Essa expressão é formada por um antecedente, o operador "implica" (*implies*) e um consequente. O antecedente pode ser formado por: uma definição de contexto; ou combinação de definições de contexto e operadores lógicos; ou combinação, por meio de operadores lógicos, de definições de contexto e características. Os operadores lógicos utilizados no antecedente podem ser do tipo E (*AND*), OU (*OR*), OU-Exclusivo (*XOR*) ou NÃO (*NOT*) (FERNANDES, 2008).

O consequente, por sua vez, é formado por uma característica ou uma combinação de características. Os operadores lógicos utilizados no consequente podem ser do tipo E (*AND*) ou NÃO (*NOT*). A presença de uma característica no consequente indica a sua adição. No entanto, a presença do operador lógico NÃO (*NOT*) indica a remoção da característica ao qual ele é aplicado (FERNANDES, 2008).

Esta abordagem UbiFEX, com todas as suas atividades e produtos, foi utilizada na presente pesquisa e a sua aplicação ao domínio de processos de desenvolvimento de software é apresentada nas próximas seções.

6.5. Dimensões de contexto de desenvolvimento de software

Seguindo a abordagem UbiFEX, o primeiro passo é definir as entidades ou dimensões de contexto para o domínio de processos de desenvolvimento de software. Neste sentido, alguns trabalhos agrupam os critérios de adaptação de processos ou fatores de contingência, aqui denominados informações de contexto, em diferentes categorias (pessoal, problema, produto, recurso e organizacional) (ALEXANDER e DAVIS, 1991) ou dimensões (tamanho, criticidade, dinamismo, pessoal e cultura) (BOEHM e TURNER, 2003). Porém, em geral, cada uma destas propostas trata apenas algumas categorias, ao contrário da proposta de Araujo *et al*. (2004), parcialmente adotada neste trabalho de pesquisa, que fornece uma visão completa de todas as dimensões envolvidas no desenvolvimento de software.

Como ponto de partida para a definição das informações de contexto relevantes para o foco do presente trabalho de pesquisa, Araujo *et al*. (2004) sugerem as seguintes dimensões de contexto para o desenvolvimento de software: indivíduo, papel, equipe, tarefa, projeto, organização, domínio da engenharia de software, produto de software, domínio do negócio e cliente/usuário (Figura 6.5).

Ainda que todas as dimensões agreguem informações de contexto importantes, as dimensões organização, projeto e equipe são particularmente relevantes para o presente trabalho de pesquisa. As dimensões organização e projeto devem ser consideradas, visto que a adaptação de processos, geralmente, é realizada nestes dois níveis (PEDREIRA *et al*., 2007). A dimensão equipe também é importante, dado o interesse deste trabalho de pesquisa no aspecto da colaboração.

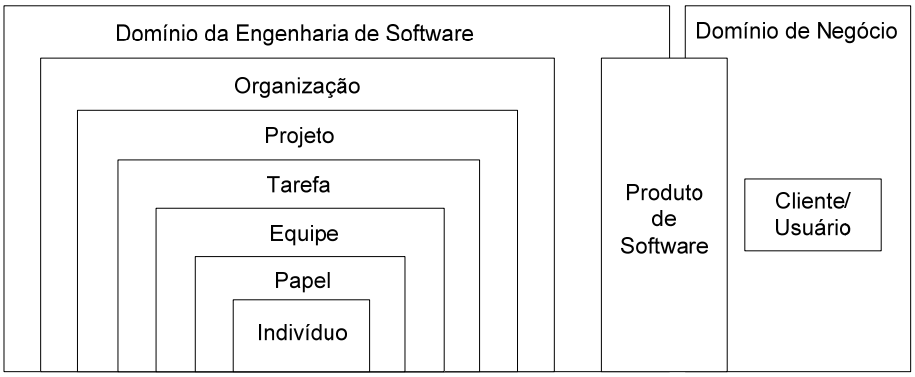


Figura 6.5 – Dimensões de contexto no desenvolvimento de software. Adaptado de: Araujo *et al*. (2004)

A dimensão organização descreve os objetivos de negócio e o processo padrão da empresa. Os resultados obtidos pelas organizações que prestam serviços aos seus clientes geram informações de: histórico de uso, sucesso ou insucesso, práticas e metodologias sugeridas no mercado ou em domínios específicos do negócio (ARAUJO *et al*., 2004, SANTORO *et al*., 2007).

A dimensão projeto compreende as características e o escopo do desenvolvimento, seus objetivos e o processo a ser seguido. A dimensão equipe reflete a agregação de seus participantes e papéis. Assim, equipes formadas por diferentes participantes irão planejar e executar as suas tarefas de formas diferentes (ARAUJO *et al*., 2004, SANTORO *et al*., 2007).

6.6. Informações de contexto de desenvolvimento de software

Para cada uma destas dimensões de contexto, pode-se pensar em um conjunto de informações de contexto, de acordo com os aspectos de colaboração e disciplina dos processos de desenvolvimento de software. A determinação das informações de contexto não é uma atividade trivial. Segundo Nunes (2007), ainda não existe um consenso sobre quais conhecimentos são considerados contextuais. Vários tipos de informações contribuem com o contexto e a relevância de cada informação, conforme dito anteriormente, depende do foco de atenção em questão (BREZILLON, 1999). O contexto não pode ser entendido de forma isolada, mas sempre de acordo com um foco de atenção que vai determinar o que é relevante em uma situação específica (BREZILLON e POMEROL, 1999).

Esta dificuldade em definir as informações de contexto relevantes para uma determinada situação aponta para a necessidade de uma sistemática que guie este trabalho. Apesar deste ainda ser um problema em aberto na área, ele está fora do escopo do presente trabalho de pesquisa e será tratado como parte de outra tese de doutorado do grupo (NUNES *et al*., 2010b).

Parte do objetivo deste trabalho de pesquisa é suficiente identificar as informações de contexto relevantes para caracterizar um projeto e balancear a colaboração e a disciplina no domínio de desenvolvimento de software. Assim, um conjunto inicial de informações de contexto é apresentado nas próximas seções. Estes exemplos de informações de contexto foram obtidos através da revisão e adaptação de trabalhos já publicados na literatura (ARMBRUST *et al*., 2008, BEKKERS *et al*., 2008, BERGER, 2003, BOEHM e TURNER, 2003, BURNS e DENNIS, 1985, COCKBURN, 2000, 2001, GINSBERG e QUINN, 1995, LAANTI e KETTUNEN, 2005, LITTLE, 2005, MACHADO, 2000, MNKANDLA, 2008, PARK *et al*., 2006, PENG XU e RAMESH, 2008, QUMER e HENDERSON-SELLERS, 2008, SLOOTEN e BRINKKEMPER, 1993, WEERD, 2009) ou através da própria experiência da autora deste trabalho em definição de processos de desenvolvimento de software.

Porém, ainda são necessários estudos experimentais para ampliar e validar estas informações de contexto. Neste sentido, planeja-se realizar, no

escopo de uma dissertação de mestrado, uma revisão sistemática da literatura para ajudar a identificar outros critérios comumente adotados na adaptação de processos de desenvolvimento de software. Em seguida, este resultado deve ser validado com os especialistas da prática e da comunidade científica. Uma estratégia similar foi adotada por Bekkers *et al*. (2008), para determinar os fatores que mais influenciam a seleção do modelo de desenvolvimento de software.

As próximas seções descrevem o conjunto inicial das informações de contexto, agrupadas de acordo com as suas respectivas dimensões.

6.6.1. Dimensão organização

A dimensão organização descreve os objetivos de negócio e o processo padrão da empresa (ARAUJO *et al*., 2004, SANTORO *et al*., 2007).

- **Estrutura organizacional:** a estrutura organizacional é utilizada para estabelecer responsabilidades, distribuir autoridade e alocar os recursos da organização (VASCONCELLOS e HEMSLEY, 2003). A estrutura organizacional é caracterizada pelo seu *grau de complexidade*. Esta complexidade pode ser medida pelo número de níveis hierárquicos existentes, desde o topo aos níveis mais baixos da organização. A quantidade e a *rigidez* dos níveis hierárquicos influenciam diretamente na colaboração, pois impactam nos aspectos de comunicação e coordenação.
- **Cultura organizacional:** os valores e normas de uma organização são estabelecidos e reforçados ao longo do tempo. A cultura organizacional determina a forma de pensar e o comportamento de uma organização e difere de uma organização para outra (SCHEIN, 1990). A cultura organizacional exerce uma influência considerável no processo de tomada de decisões, nas estratégias de solução de problemas, nas práticas de inovação, nas negociações sociais, nos relacionamentos e nos mecanismos de planejamento e controle. Organizações que possuem culturas organizacionais mais *informais*, cooperativas e com maior autonomia de seus funcionários para *tomada de decisão* (coletivismo) são ambientes onde práticas colaborativas são mais bem recebidas, do que organizações com culturas *formais*, rígidas e burocráticas (individualismo), que, em geral,

prezam maior visibilidade e controle dos projetos e das equipes de desenvolvimento (HOFSTEDE *et al*., 2010).

- **Gestão de conhecimento:** processo pelo qual o conhecimento é adquirido, transformado e disseminado. O conhecimento se distingue em tácito e explícito. O conhecimento tácito é altamente pessoal, específico do contexto e difícil de *formalizar*, o que dificulta sua transmissão e compartilhamento com outros. O conhecimento explícito é *formal* e fácil de ser comunicado ou registrado em diferentes tipos de artefatos (NONAKA e TAKEUCHI, 1995).
- **Objetivo de negócio:** toda organização possui os seus objetivos de negócio, que estão relacionados ao seu planejamento estratégico. Os objetivos de negócio podem envolver diminuição de custos, aumento da qualidade, conquista de um novo segmento de mercado através do lançamento rápido de um produto, ou até inovação, através do uso de uma nova tecnologia.
- **Nível de maturidade atual:** uma preocupação comum das organizações ao adotarem um novo processo de desenvolvimento de software é manter o seu nível de maturidade atual para que possam preservar a(s) sua(s) certificação(ões). Como cada modelo de maturidade define os seus próprios níveis, foram considerados os nívels do CMMI (CHRISSIS *et al*., 2006) como referência.
- **Nível de maturidade desejado:** de forma análoga, se a organização tem o interesse em alcançar um novo nível de maturidade, tal fato também não pode ser ignorado durante a adaptação do processo, pois será necessário oferecer um processo que suporte esta iniciativa da empresa. Neste caso, também foram considerados os nívels do CMMI (CHRISSIS *et al*., 2006) como referência.

6.6.2. Dimensão projeto

A dimensão projeto compreende também o contexto do produto a ser construído, seus objetivos e o processo a ser seguido (ARAUJO *et al*., 2004).

- **Relacionamento com o cliente:** compreende o *nível de envolvimento* e *disponibilidade* do cliente e o nível de *confiança* existente entre a equipe de desenvolvimento e o cliente. Também é importante levar em consideração a existência de um único ou de múltiplos *representantes* do cliente, a

habilidade destes representantes em *expressar os requisitos* e o nível de *experiência no domínio*.

- **Tamanho do problema:** o tamanho do problema a ser resolvido pelo projeto pode resultar em uma necessidade de maior gerenciamento do processo ou na necessidade de dividi-lo em problemas menores que serão resolvidos em fases.
- **Complexidade do problema:** a complexidade do problema compreende o grau de dificuldade para a sua realização e está relacionado à natureza do projeto.
- **Duração:** o tempo previsto (em número de meses) para a duração de um projeto pode influenciar a adaptação do processo para o projeto. Em projetos longos, é necessário um cuidado maior com a gestão do conhecimento, pois aumenta o risco de mudanças nos membros da equipe de desenvolvimento. Além disso, também aumentam as chances do projeto ser afetado pela estabilidade dos requisitos e pela incerteza técnica.
- **Criticidade:** a criticidade do projeto está relacionada ao risco envolvido em caso de falha do produto desenvolvido. Assim, em projetos onde existe um alto risco, seja de vidas humanas ou de funções críticas ao negócio, é necessário maior controle sobre o processo de desenvolvimento de software que será utilizado do que nos casos onde o risco envolve somente pequenas perdas financeiras.
- **Novidade:** a novidade corresponde ao grau de inovação existente no projeto de desenvolvimento. Projetos com características de inovação não são o ambiente mais adequado para impor muito formalismo ao processo de desenvolvimento de software, pois a equipe precisa de mais liberdade para trabalhar de forma colaborativa e criativa.
- **Tecnologia de desenvolvimento do produto:** a escolha de uma tecnologia para o desenvolvimento do produto influencia na adaptação do processo de desenvolvimento, pois é necessário levar em consideração se esta tecnologia é uma *novidade* ou se já está difundida na organização e entre a equipe de desenvolvimento.
- **Ambiente tecnológico de desenvolvimento:** o ambiente tecnológico de desenvolvimento corresponde à *maturidade* do suporte computacional

(ferramentas, compiladores etc.) utilizado para a construção do produto.

- **Estabilidade dos requisitos:** a estabilidade dos requisitos está relacionada à frequência de mudanças destes requisitos. Em projetos com requisitos voláteis e dinâmicos, são mais indicadas características de agilidade e colaboração com o cliente, do que tentar manter um excesso de formalismo no processo para gerenciar estas mudanças de requisitos.
- **Tipo de desenvolvimento:** o tipo de desenvolvimento de software pode ser exploratório, interno, para o mercado ou para um cliente específico. Dependendo do tipo de desenvolvimento pode existir mais espaço para a colaboração ou mais necessidade de disciplina.
- **Frequência de lançamento de versões:** assiduidade prevista para o lançamento de versões de acordo com a necessidade de *feedback* dos usuários.
- **Nível de colaboração planejado:** nível de colaboração que é desejado ou necessário que o projeto possua.
- **Nível de disciplina planejado:** nível de disciplina que é desejado ou necessário que o projeto possua.
- **Nível de colaboração atual:** nível de colaboração atual que o projeto efetivamente possui.
- **Nível de disciplina atual:** nível de disciplina atual que o projeto efetivamente possui.

6.6.3. Dimensão equipe

O contexto de uma equipe reflete a agregação das características de seus participantes e papéis. As equipes são estabelecidas dentro de um projeto de desenvolvimento para executar tarefas específicas. Assim, equipes formadas por diferentes participantes irão planejar e executar as suas tarefas de formas diferentes (ARAUJO *et al*., 2004, SANTORO *et al*., 2007).

- **Tamanho da equipe:** o tamanho da equipe está relacionado à quantidade de pessoas que a compõem. Possivelmente, equipes maiores vão precisar de mais disciplina, enquanto equipes menores conseguem trabalhar de forma mais colaborativa.
- **Experiência técnica:** a *experiência da equipe no desenvolvimento de produtos de software* e a *experiência no domínio da aplicação* podem

determinar a possibilidade de oferecer mais colaboração ou mais disciplina ao processo definido para o projeto. As equipes mais experientes tecnicamente, talvez já não precisem mais de tanto rigor no seu processo de desenvolvimento de software, pois já são capazes de antecipar as atividades, o que não acontece com uma equipe de novatos.

- **Experiência gerencial:** a experiência do gerente de projeto para coordenar a equipe e inclusive gerenciar mais de um projeto ao mesmo tempo também deve ser considerada.
- **Experiência no trabalho em conjunto:** da mesma forma, equipes onde os seus membros já estão acostumados a trabalhar em conjunto podem já estar mais voltadas à colaboração, enquanto outras podem precisar ainda de incentivo a colaboração para que os seus membros se integrem.
- **Proximidade:** identificar se os membros da equipe estão localizados fisicamente próximos ou se esta equipe trabalha geograficamente distribuída é importante para balancear a dosagem de colaboração e disciplina necessárias. Uma equipe que trabalha reunida em um mesmo local tem maior facilidade de comunicação e coordenação do que uma equipe dispersa em locais diferentes.
- **Estabilidade:** a alta rotatividade dos membros da equipe de desenvolvimento pode caracterizar uma necessidade de maior formalismo para evitar perda de conhecimento e diminuir os riscos para o projeto e a organização.

A Tabela 6.1 resume as informações de contexto e seus atributos. Em relação à granularidade e complexidade, uma informação de contexto pode ser ou não atômica. Uma informação de contexto atômica já representa a menor unidade. Quando a informação de contexto não é atômica, ela é composta por outras informações de contexto.

Cada informação de contexto possui um conjunto de valores nominais possíveis de serem assumidos. Neste trabalho, os valores nominais foram organizados de acordo com a sua inclinação à colaboração, o que pode provocar inversões nas escalas.

Tabela 6.1 – Exemplos de informações de contexto

Dimensão	Informações de Contexto	Composição	Valores
Organização	Estrutura organizacional	Complexidade	Muito baixa; Baixa; Média; Alta; Muito alta
		Rigidez	Muito baixa; Baixa; Média; Alta; Muito alta
	Cultura organizacional	Formalismo	Muito baixo; Baixo; Médio; Alto; Muito alto
		Tomada de decisão	Descentralizada; Centralizada
	Gestão de conhecimento	Formalismo	- Baixo - conhecimento tácito não disseminado; - Médio - socialização - conhecimento tácito transmitido; - Alto - externalização - formalização do conhecimento tácito em explícito
	Objetivo de negócio	*Atômica*	- Inovação; - Conquista de um novo segmento de mercado; - Aumento da qualidade; - Diminuição de custos
	Nível de maturidade atual	*Atômica*	- Nível 1 - Inicial; - Nível 2 - Gerenciado; - Nível 3 - Definido; - Nível 4 - Gerenciado quantitativamente; - Nível 5 - Otimizado
	Nível de maturidade desejado	*Atômica*	- Nível 1 - Inicial; - Nível 2 - Gerenciado; - Nível 3 - Definido; - Nível 4 - Gerenciado quantitativamente; - Nível 5 - Otimizado
Projeto	Relacionamento com o cliente	Nível de envolvimento	Muito alto; Alto; Médio; Baixo; Muito baixo
		Disponibilidade	Muito alta; Alta; Média; Baixa; Muito baixa
		Nível de confiança	Muito alta; Alta; Média; Baixa; Muito baixa
		Número de representantes	Um; Múltiplos
		Experiência no domínio	- Muito alta - especialista; - Alta - muito experiente; - Média - experiente; - Baixa - pouco experiente; - Muito baixa - sem experiência
		Habilidade de expressar requisitos	- Muito alta - expressivo; - Alta - muito comunicativo; - Média - comunicativo; - Baixa - pouco comunicativo; - Muito baixa - sem habilidade
	Tamanho do problema	*Atômica*	Muito baixo; Baixo; Médio; Alto; Muito alto
	Complexidade do problema	*Atômica*	- Muito Baixa - projetos que não envolvem cálculos complexos; - Baixa - projetos que envolvem cálculos pouco complexos; - Média - projetos que envolvem cálculos complexos; - Alta - projetos que envolvem cálculos razoavelmente complexos; - Muito alta - projetos que envolvem cálculos muito complexos
	Duração	*Atômica*	- Até 2 meses; - De 2 a 6 meses;

Dimensão	Informações de Contexto	Composição	Valores
			- De 6 a 12 meses; - De 12 a 24 meses; - Mais que 24 meses
	Criticidade	*Atômica*	- Muito baixa - em caso de falha do sistema, o trabalho precisará ser feito manual ou informalmente; - Baixa - perda de valores financeiros discretos; - Média - perda do investimento do projeto; - Alta - perda de quantias significativas ou de funções críticas do negócio; - Muito alta - perda de vidas humanas
	Novidade	*Atômica*	Muito alta; Alta; Média; Baixa; Muito baixa
	Tecnologia de desenvolvimento do produto	Maturidade	Muito alta; Alta; Média; Baixa; Muito baixa
	Ambiente tecnológico de desenvolvimento	Maturidade	Muito alta; Alta; Média; Baixa; Muito baixa
	Estabilidade dos requisitos	*Atômica*	- Muito Baixa - requisitos muito instáveis; - Baixa - requisitos instáveis; - Média - requisitos com grau médio de estabilidade; - Alta - requisitos praticamente estáveis; - Muito alta - requisitos estáveis
	Tipo de desenvolvimento	*Atômica*	- Projeto de pesquisa; - Projeto interno à organização; - Desenvolvimento do produto para o mercado; - Desenvolvimento contratado por um cliente em particular
	Frequência de lançamento de versões	*Atômica*	Muito alta; Alta; Média; Baixa; Muito baixa
	Nível de colaboração planejado	*Atômica*	Muito alto; Alto; Médio; Baixo; Muito baixo
	Nível de disciplina planejado	*Atômica*	Muito alto; Alto; Médio; Baixo; Muito baixo
	Nível de colaboração atual	*Atômica*	Muito alto; Alto; Médio; Baixo; Muito baixo
	Nível de disciplina atual	*Atômica*	Muito alto; Alto; Médio; Baixo; Muito baixo
Equipe	Tamanho da equipe	*Atômica*	- Até 6 pessoas; - De 7 a 10 pessoas; - De 11 a 39 pessoas; - De 40 a 80 pessoas; - Mais de 80 pessoas
	Experiência técnica	Experiência no domínio da aplicação	- Muito alta - especialista; - Alta - muito experiente; - Média - experiente; - Baixa - pouco experiente; - Muito baixa - sem experiência
		Experiência no desenvolvimento de software	- Muito alta - especialista; - Alta - muito experiente; - Média - experiente; - Baixa - pouco experiente; - Muito baixa - sem experiência
	Experiência gerencial	*Atômica*	- Muito alta - especialista; - Alta - muito experiente; - Média - experiente;

Dimensão	Informações de Contexto	Composição	Valores
			- Baixa - pouco experiente; - Muito baixa - sem experiência
	Experiência no trabalho em conjunto	*Atômica*	Muito alta; Alta; Média; Baixa; Muito baixa
	Proximidade	*Atômica*	Localizada; Distribuída
	Estabilidade	*Atômica*	Muito alta; Alta; Média; Baixa; Muito baixa

Vale ressaltar que nenhuma dessas informações de contexto sozinha é determinante para definir se um processo terá mais colaboração ou mais disciplina. O balanceamento será obtido pela sua combinação através das regras de contexto.

6.7. Situações de contexto de desenvolvimento de software

Para o entendimento destas informações de contexto dentro do domínio de processos de desenvolvimento de software, é necessário estabelecer algumas situações que impactem no estabelecimento de regras ou sugestões de combinações de práticas de desenvolvimento de software. Estas situações de contexto representam circunstâncias que podem acontecer baseadas na combinação de valores de determinadas informações de contexto. Alguns exemplos destas situações são apresentados na Tabela 6.2.

Tabela 6.2 – Exemplos de situações de contexto

Situação	Dimensão	Informação de Contexto
Contenção de despesas	Organização	Objetivo de negócio: diminuição de custos
Ambiente organizacional favorável a colaboração	Organização	Estrutura organizacional – Complexidade: baixa Estrutura organizacional – Rigidez: baixa Cultura organizacional – Formalismo: baixo
Preocupação com certificação	Organização	Nível de maturidade atual: >= 2 Nível de maturidade desejado: >= 2
Negociação de contrato	Projeto	Criticidade: alto Tamanho do problema: alto Complexidade do problema: alta Relacionamento com cliente – envolvimento: baixo
Inovação	Organização	Objetivo de negócio: inovação
	Projeto	Estabilidade de requisitos: muito baixa
	Projeto	Novidade: alta

Situação	Dimensão	Informação de Contexto
Carência de gestão de conhecimento	Organização	Gestão de conhecimento – formalismo: baixo
	Projeto	Duração: >=30 meses
	Equipe	Estabilidade: baixa

A primeira situação é bem simples e representa uma situação de contexto organizacional denominada Contenção de despesas, onde o objetivo de negócio é reduzir custos.

A segunda situação (Ambiente organizacional favorável a colaboração) corresponde ao caso em que a estrutura organizacional da empresa tem complexidade e rigidez baixas (Tabela 6.2). Além disso, a cultura organizacional é informal, ou seja, os funcionários têm maior autonomia e as práticas colaborativas são mais bem recebidas.

A terceira situação (Preocupação com certificação) representa um cenário muito comum nas organizações que já trabalham orientadas a processos e possuem uma grande preocupação com a certificação dos seus processos de acordo com determinados modelos de maturidade, tais como o CMMI (CHRISSIS *et al*., 2006), e o MPS.BR (SOFTEX, 2009). Nesta situação, a organização já alcançou um determinado nível de maturidade e tem a intenção de manter ou até mesmo evoluir nesta maturidade (Tabela 6.2).

A próxima situação foi criada tendo como inspiração o cenário histórico que motivou a criação dos modelos de qualidade. Neste cenário, o projeto tem um alto risco, seja de perda de vidas humanas ou de comprometimento de funções críticas para o negócio. Além disso, trata-se de um projeto grande e complexo, onde o relacionamento entre cliente e equipe de desenvolvimento se caracteriza pelo baixo nível de confiança (Tabela 6.2). Desta forma, o projeto acaba sendo governado por definições contratuais (GLAZER *et al*., 2008).

A situação seguinte representa um cenário de inovação, onde o objetivo de negócio da organização é a inovação do produto ou tecnológica, o projeto tem um caráter de novidade e, por isso mesmo, a estabilidade dos requisitos é muito baixa (Tabela 6.2).

Por fim, a última situação representa um caso mais complexo que envolve informações de contexto das três dimensões. Neste cenário existe a Carência de Gestão de Conhecimento, pois a gestão do conhecimento na

organização é feita informalmente e a equipe tem uma estabilidade baixa, ou seja, nesta equipe é comum a rotatividade dos seus membros (Tabela 6.2). Além disso, trata-se de um projeto de longa duração que tem maiores chances de ser afetado por estas mudanças na equipe.

6.8. Regras de contexto de desenvolvimento de software

Após a definição dos contextos relevantes para o domínio, as ações que devem ser tomadas caso uma determinada situação ocorra podem ser especificadas por meio das regras de contexto. Para ajudar a adaptação do processo é necessário estabelecer regras de contexto que relacionem as situações de contexto à seleção de práticas do modelo de características da linha de processo. As regras de contexto restringem a escolha das práticas dos diferentes processos de desenvolvimento de software de acordo com o contexto.

Neste sentido, alguns exemplos de regras de contexto foram construídos utilizando algumas das situações de contexto listadas na Tabela 6.2 e o modelo de características da linha de processos, apresentado no capítulo anterior. A Tabela 6.3 resume estes exemplos de regras de contexto.

Tabela 6.3 – Exemplos de regras de contexto

# regra	Situação de Contexto	Tipo de Regra	Prática
CR1	Contenção de despesas	*Implies*	Estimativa de custo AND Risco
CR2	Negociação de contrato	*Implies*	Estimativa
CR3	Inovação	*Implies*	NOT Especificação formal

Vale ressaltar que estas regras de contexto foram derivadas empiricamente com base no conhecimento e experiência da autora, mas ainda é necessário mais trabalho para estabelecer correlações válidas entre as situações de contexto e as práticas de desenvolvimento de software.

6.9. Linha de processos gerida por contexto

Uma das premissas para o uso de linhas de processos está associada ao fato de que quando se entende o processo implementado na organização, é possível fazer mudanças durante a execução do processo. Neste sentido, a adaptação dinâmica de processos está associada ao uso de informações de contexto para identificar modificações no processo em tempo de execução, visando se adaptar a novas situações.

Assim, é papel da gestão de contexto perceber mudanças no contexto da organização, do projeto e da equipe ao longo da execução do projeto para permitir que a tomada de decisão sobre as sugestões de adaptação do processo possa ser revista e continuamente atualizada, de acordo com o novo contexto identificado. Chegamos então a terceira sub-hipótese (H1.3) investigada nesta proposta:

É possível apoiar a tomada de decisão sobre a adaptação de processos de desenvolvimento de software ao longo da execução do projeto através da gestão de contexto.

A abordagem proposta para a combinação de linha de processos e gestão de contexto (Figura 6.6) envolve a introdução das estruturas de representação de contexto na abordagem de engenharia de linha de processos. Como as fases e os produtos da abordagem de engenharia de linha de processos já foram apresentados no capítulo anterior, nesta seção vamos nos ater a explicar as inclusões referentes ao contexto.

Conforme visto na seção 6.4.1, a abordagem UbiFEX (FERNANDES, 2008) estabelece quatro elementos de contexto que devem ser representados: entidades, informações, situações e regras. A representação do contexto faz parte da criação da linha de processos e, portanto, acontece na fase de Engenharia de Domínio do Processo.

Primeiramente, deve ser criado o modelo de contexto contendo as dimensões e informações de contexto e as relações entre elas. Assim, este modelo de contexto, ilustrado pela Figura 6.9, é criado na etapa de análise de domínio do processo.

Dando continuidade a representação dos elementos de contexto indicados pela abordagem UbiFEX, devem ser criadas, na etapa de projeto de

domínio do processo, as situações e regras de contexto. Estas regras de contexto vão estabelecer as relações entre o contexto e as características e os componentes da linha de processos.

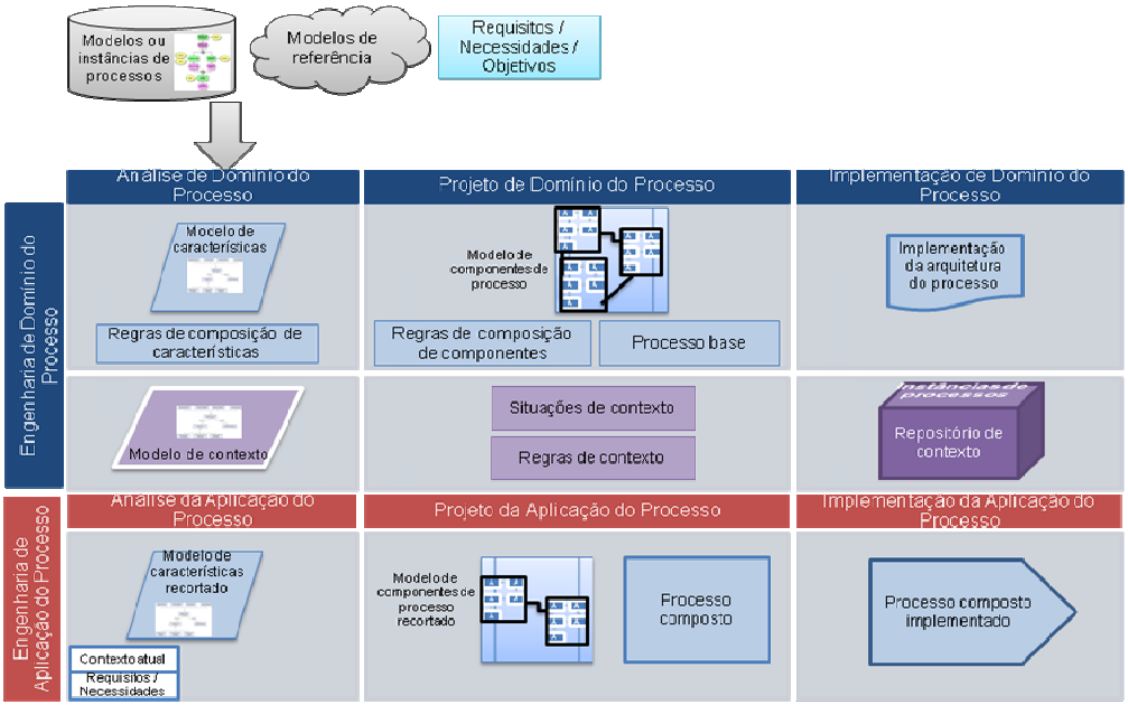


Figura 6.6 – Engenharia de linha de processos baseada em contexto com redes sociais

Por fim, a implementação de domínio do processo envolve a implementação do repositório de contexto. O repositório de contexto servirá para armazenar os contextos associados à organização, projeto, equipe e processo.

Na fase de Engenharia de Aplicação do Processo, o processo é instanciado e fica pronto para ser executado, associado ao seu contexto de execução, e adaptado dinamicamente durante a sua execução.

A partir daí, a gestão de contexto vai verificar continuamente se, a partir dos novos contextos que forem aparecendo, as situações de contexto mudaram para alertar ao ambiente de apoio à decisão que a adaptação indicada para o processo pode precisar ser revista.

6.10. Cenários de uso das informações de contexto

Neste trabalho são adotados dois cenários de uso das informações de contexto para aplicação da proposta. Estes cenários foram escolhidos, por representarem momentos importantes durante o desenvolvimento de um

projeto de desenvolvimento de software, onde o gerente do projeto precisa estar munido de informações para tomar decisões e ações sobre o futuro do projeto. Cada um dos cenários possui um conjunto de informações de contexto relevantes, de acordo com o foco do cenário em questão.

O primeiro cenário é mais simples e consiste na atividade de adaptação de um processo específico para um novo projeto, realizada pelo gerente no início do projeto. Neste cenário, são utilizadas como entrada as informações de contexto organizacionais e informações de contexto de outros projetos similares, recuperadas do repositório de contexto. Além disso, são capturadas, através de interação com o agente humano, as informações do contexto atual. Estas informações também serão alimentadas no repositório de contexto (Figura 6.7).

Com base em todas estas informações de contexto e nas características e componentes da linha de processos, são geradas sugestões para adaptação do processo do projeto, levando em consideração o balanceamento necessário entre colaboração e disciplina. Além disso, é oferecida ao gerente de projeto uma visualização do contexto organizacional, do contexto de projetos similares e do contexto atual (Figura 6.7).

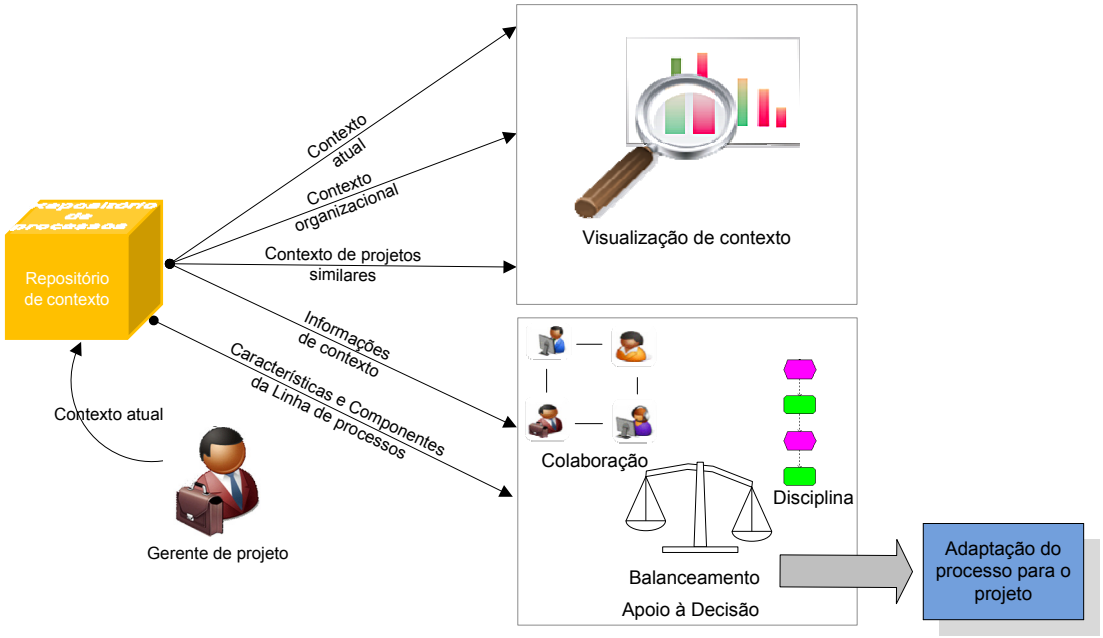


Figura 6.7 – Cenário 1 de Balanceamento

O cenário 2 ocorre nos marcos definidos para o acompanhamento do projeto. Neste momento, o gerente de projeto precisa ser munido de informações sobre o projeto para tomar decisões e possíveis ações corretivas.

Então, neste cenário, são utilizadas como entrada as seguintes informações de contexto: organizacionais; de outros projetos similares; do contexto anterior do projeto; e do contexto atual do projeto (Figura 6.8).

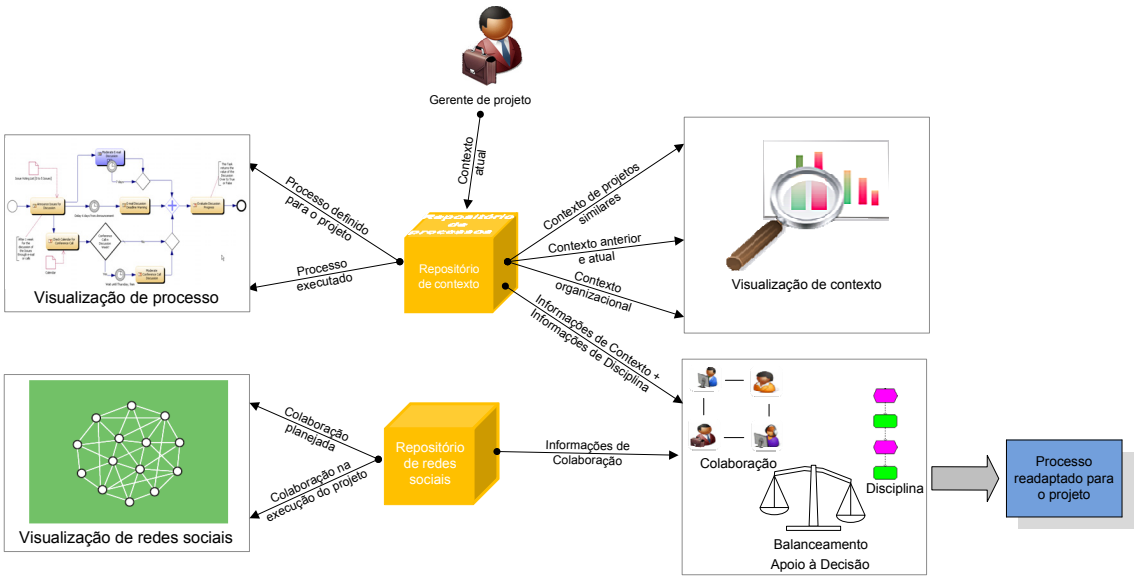


Figura 6.8 – Cenário 2 de balanceamento

As informações de contexto do projeto atual são recuperadas com base nas informações de contexto capturadas, através do gerente de projeto, no cenário anterior, mas também podem ser atualizadas neste cenário, caso tenham se modificado ao longo da execução do projeto.

Além da visualização de contexto, durante a realização desta atividade pretende-se oferecer a visualização do processo executado pelo projeto e a visualização das redes sociais existentes no projeto (Figura 6.8), a fim de aumentar a percepção do gerente do projeto sobre a situação atual da colaboração e da disciplina.

Com base em todas estas informações, o balanceamento entre colaboração e disciplina é atualizado com novas sugestões. Em particular, as sugestões referentes à colaboração poderão se valer da interpretação das características de colaboração presentes nas redes sociais (SANTOS *et al.*, 2010).

6.11. Ferramental de apoio

A construção do ferramental de apoio para a abordagem de gestão de contexto é objeto da pesquisa de tese de Nunes (NUNES *et al.*, 2010b). Assim, esta

seção apresenta apenas o ferramental de apoio adotado para a representação de contexto para o domínio de processos de desenvolvimento de software.

Para a modelagem de características utilizando a notação Odyssey-FEX (OLIVEIRA, 2006) e a modelagem de contexto utilizando a abordagem UbiFEX (FERNANDES, 2008), foi adotada a ferramenta Odyssey (ODYSSEY, 2010). Nesta parte do trabalho, o Odyssey foi escolhido porque permite que se tenha, de forma unificada no mesmo ambiente, o suporte para a modelagem do domínio através do modelo de características e para a modelagem contextual. Além disso, a ferramenta oferece a possibilidade de interação entre as duas abordagens.

Assim, seguindo a abordagem UbiFEX, as dimensões e as informações de contexto deste trabalho foram traduzidas respectivamente para as entidades de contexto (*context entity*) e informações de contexto (*context information*) como apresentado pelo exemplo na Figura 6.9.

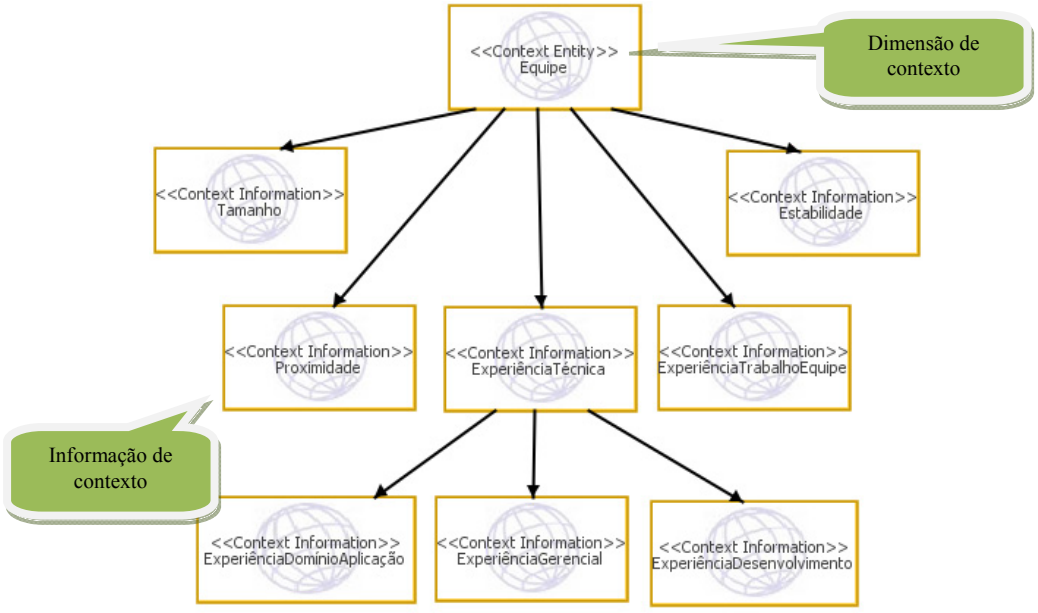


Figura 6.9 – Exemplo parcial de entidades e informações de contexto no Odyssey

O Odyssey permite ainda a criação das definições de contexto (*context definition*) que foram mapeadas para as situações de contexto apresentadas na seção 6.7. Por exemplo, a situação Negociação de Contrato ocorre quando o risco do projeto é alto, o tamanho do projeto é alto, a complexidade do projeto é alta e a confiança no relacionamento com o cliente é baixa. Na ferramenta, esta definição de contexto é apresentada na Figura 6.10 e sua semântica é definida como: “Negociação de Contrato = ((((Projeto.Risco = alto) AND

(Projeto.Tamanho = alto)) AND (Projeto.Complexidade = alta)) AND (Projeto.RelacionamentoCliente = baixa)).

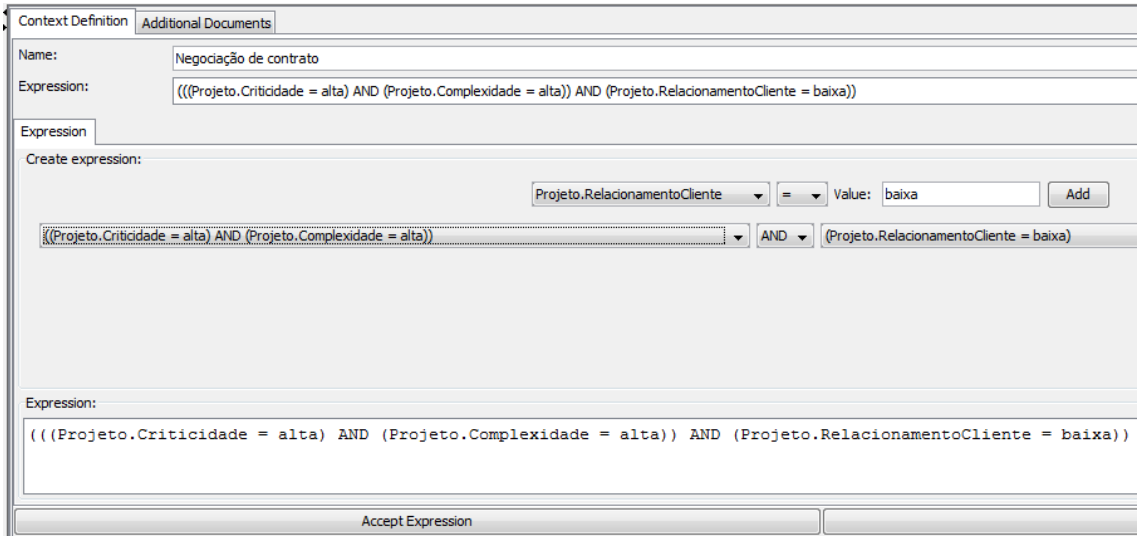


Figura 6.10 – Exemplo de definição de contexto no Odyssey

Por fim, as regras de contexto também podem ser modeladas no Odyssey. Usando um dos exemplos das regras de contexto da seção 6.8, esta regra poderia ser criada no Odyssey e sua semântica ficaria assim: “CR1: Negociação de contrato *implies* (((Estimativa de escopo AND Estimativa de custo) AND Estimativa de esforço) AND Estimativa de tamanho)”.

O próprio modelo de características, após o estabelecimento de situações e regras, apresenta as unidades de trabalho sobre as quais algum tipo de raciocínio pode ser realizado como apresentado na Figura 6.11. Na representação gráfica do modelo de características, as regras de contexto também podem ser visualizadas por meio das características que fazem parte do seu consequente, que são marcadas no canto superior esquerdo com o identificador da regra, permitindo a identificação das características que sofrem influência de contexto de forma explícita (FERNANDES, 2008).

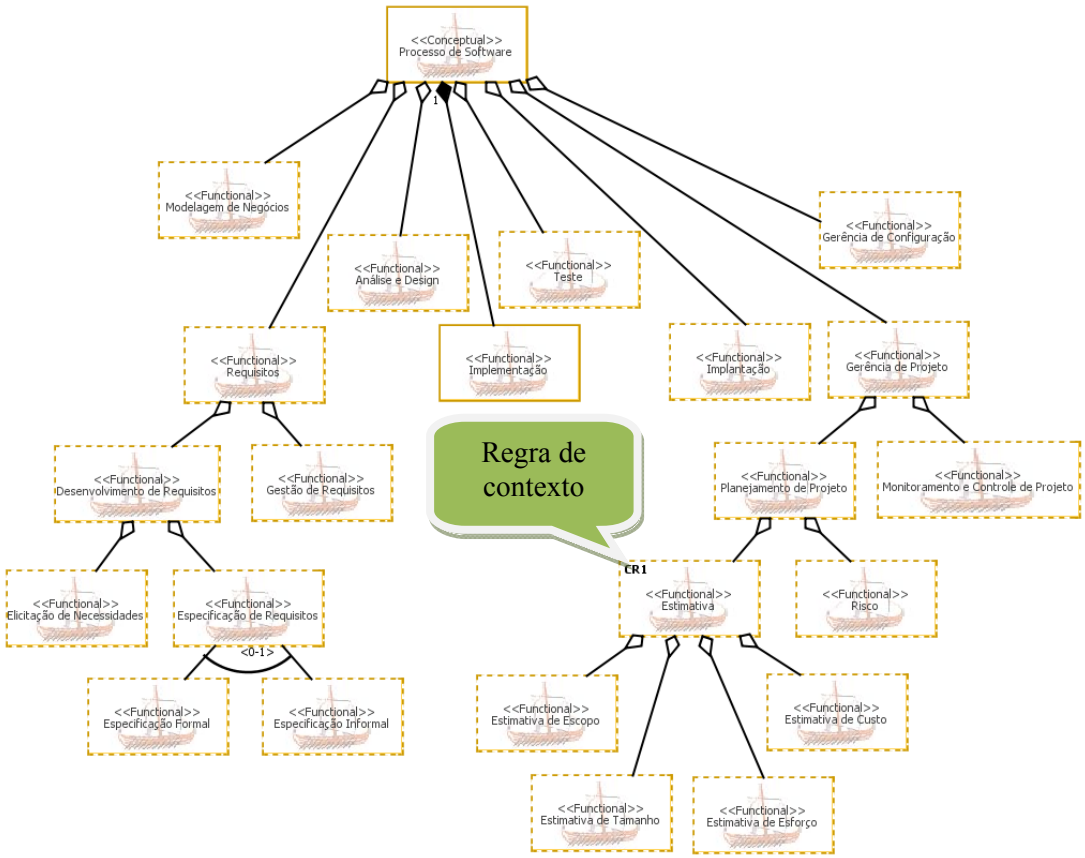


Figura 6.11 – Modelo de características de processo de software após o estabelecimento de situações e regras contextuais

6.12. Considerações finais

Este capítulo definiu o conceito de contexto e apresentou a visão geral da abordagem de gestão de contexto, composta por: mecanismos de captura e armazenamento; mecanismos de armazenamento; mecanismos de inferência; mecanismos de visualização; e o modelo de representação de contexto. Dentre estes elementos, este capítulo focou no modelo de representação de contexto para o domínio de processos de desenvolvimento de software.

O modelo de representação de contexto escolhido, após um estudo que levou em consideração duas linguagens de representação (ontologias e modelos de características) e suas respectivas ferramentas de apoio, foi o uso de modelos de características. Este modelo de representação demonstrou um grau de representatividade suficiente para abarcar as necessidades deste trabalho, sem aumentar a complexidade da modelagem (NUNES *et al.*, 2010a).

Ao optar por este modelo de representação, foi natural também adotar a abordagem UbiFEX (FERNANDES, 2008) que permite a representação de

contexto em modelos de características e foi desenvolvida dentro do Grupo de Reutilização de Software da COPPE/UFRJ. A UbiFEX define quatro elementos principais para serem representados: entidades, informações, situações e regras de contexto. Todos estes elementos foram então interpretados e definidos para o domínio de processos de desenvolvimento de software.

Em alguns casos, como nas informações, situações e regras de contexto, foram definidos apenas alguns exemplos iniciais para ilustrar a aplicabilidade da abordagem para o domínio em questão e permitir a construção da solução completa. Entretanto, nota-se ainda a carência de um método para sistematizar o levantamento destes elementos dentro de um determinado domínio.

Além disso, ainda são necessários estudos experimentais para ampliar e validar o conjunto inicial de informações de contexto proposto após a revisão bibliográfica. Neste sentido, planeja-se realizar uma revisão sistemática da literatura para ajudar a identificar outros critérios comumente adotados na adaptação de processos de desenvolvimento de software. Esta revisão sistemática, bem como a validação destes critérios com os especialistas, é objeto de uma dissertação de mestrado.

Por outro lado, este exercício de modelagem, nos permitiu observar a adequação do ferramental de apoio existente na ferramenta Odyssey (ODYSSEY, 2010) para viabilizar a modelagem destes elementos uma vez que eles tenham sido definidos para o domínio de processos de desenvolvimento de software.

Apesar de não ter sido explorado neste capítulo o uso dos mecanismos de inferência, pois eles ainda serão detalhados no escopo da tese de doutorado de Nunes (NUNES *et al*., 2010b), o papel deles neste trabalho seria o de perceber mudanças no contexto da organização, do projeto e da equipe e alertar o ambiente de tomada de decisão para que este possa rever as sugestões de adaptação do processo. Este ambiente de tomada de decisão é o tema do próximo capítulo.

7. Balanceamento de Colaboração e Disciplina

Este capítulo detalha a proposta deste trabalho para o apoio à decisão na adaptação de processos de software, balanceando colaboração e disciplina. Esta proposta é entendida como um problema de otimização e, por isso, uma breve revisão bibliográfica sobre o tema é apresentada. O problema foi definido, formalmente modelado e analisado. Por fim, são feitas algumas considerações sobre as perspectivas futuras para a implementação da solução e as possibilidades de avaliação da solução proposta.

O objetivo principal desta proposta de tese é apoiar o gerente de projeto na tomada de decisão sobre a melhor forma de adaptar o processo específico para o projeto, diminuindo o esforço necessário para sua execução e melhorando os resultados obtidos.

Este apoio deve fornecer ao gerente um melhor embasamento para esta tomada de decisão, uma vez que possibilita que seja considerado um universo maior de fatores e alternativas. O apoio sugerido diz respeito a propor ao gerente de projeto um processo que satisfaça as restrições e que otimize alguns dos fatores envolvidos no problema. O apoio à decisão também irá contribuir fornecendo informações que seriam difíceis de obter de outra forma e permitindo ao gerente testar diferentes soluções e medir o impacto de cada uma delas no projeto. Porém, a decisão final sobre o processo que será efetivamente adotado continua sendo dele.

A partir deste objetivo, observa-se a necessidade de um ambiente automatizado de apoio à decisão sobre a adaptação de processos de desenvolvimento de software. Este ambiente, que será usado pelo gerente de projeto, deve ser capaz de sugerir um processo específico para o projeto de acordo com o contexto e deve levar em consideração o balanceamento necessário entre colaboração e disciplina.

7.1. Apoio à Decisão

Segundo Spradlin (1997), uma decisão é uma alocação de recursos. O tomador de decisão tem autoridade sobre os recursos sendo alocados e toma a decisão considerando o objetivo que ele espera atingir com a alocação de recursos. O tomador de decisão toma decisões consistentes com seus valores,

ou seja, de acordo com os fatores que são importantes para ele, especialmente aqueles que são relevantes no contexto da decisão.

Segundo Clemen e Reilly (2000), um processo de tomada de decisão é formado pelas seguintes atividades: identificar o problema; identificar objetivos e alternativas; decompor o problema em modelos; escolher a melhor alternativa; analisar a sensibilidade para determinar a consistência das soluções; decidir se é necessária mais análise; e implementar a alternativa escolhida.

Porém, existem algumas dificuldades para a tomada de uma decisão (CLEMEN e REILLY, 2000):

- **Complexidade do problema:** os problemas a serem resolvidos normalmente são muito complexos e envolvem diversas possibilidades que devem ser consideradas;
- **Incerteza quanto à situação e a eventos futuros:** geralmente é preciso considerar uma série de eventos que, caso aconteçam, podem influenciar o problema;
- **Múltiplos objetivos a serem atingidos:** muitas vezes uma decisão é tomada na tentativa de atingir mais de um objetivo, que podem inclusive ser conflitantes, e pode ser difícil conseguir um equilíbrio entre eles;
- **Diferentes perspectivas:** o problema pode ser encarado de forma diferente dependendo de quem o está resolvendo.

Todos estes desafios também estão presentes na tomada de decisão na Engenharia de Software (RUS *et al*., 2003):

- O desenvolvimento de software é um processo complexo, com muitas atividades, artefatos, atores e interações;
- Uma vez que o desenvolvimento de software é dependente das pessoas envolvidas, ele está sujeito a muitas incertezas;
- O software deve ser construído mais rápido, mais barato e com maior qualidade. Os diferentes *stakeholders* envolvidos possuem diferentes objetivos e os gerentes de projetos têm que lidar com estes múltiplos e conflitantes objetivos;
- As pessoas tomam decisões com base na sua experiência, intuições, crenças pessoais e nas informações que têm disponíveis (que, podem ser

imperfeitas, insuficientes ou imprecisas).

A tomada de decisão é uma atividade importante e complexa na Engenharia de Software. Decisões precisam ser tomadas durante todo o ciclo de vida de desenvolvimento de software (RUHE, 2003). Por exemplo, o gerente de projetos constantemente toma decisões, tanto técnicas quanto gerenciais, e tem um grande conjunto de fatores a considerar, várias atividades para planejar e controlar e diferentes decisões a tomar: Como alocar tempo para as diferentes fases do projeto? Como alocar o pessoal? Como aumentar a qualidade do produto? Quanto de teste deve ser feito? Novas técnicas devem ser inseridas? O que funcionaria melhor para o projeto? (RUS *et al*., 2003).

Porém, normalmente, muitas destas decisões são tomadas de uma forma *ad-hoc*. Este processo não sistemático tipicamente se baseia em experiência e julgamento pessoal (RUS *et al*., 2003), em regras simplistas e não observa conhecimentos, modelos e experiências já existentes. Assim, não é levado em consideração o impacto que estas decisões irão provocar na qualidade do produto sendo desenvolvido (RUHE, 2003). Quanto mais cedo no ciclo de vida decisões ruins são tomadas, maior será seu impacto na qualidade do software.

As principais características de uma tomada de decisão que acontece muito cedo no desenvolvimento de software são (RUHE, 2003): (i) a quantidade e a qualidade da informação disponível nesta fase são tipicamente baixas; (ii) os processos e parâmetros para a tomada de decisão estão mudando dinamicamente; (iii) existe um número de interesses conflitantes dos *stakeholders*, com objetivos distintos, que precisam ser balanceados.

Para apoiar os usuários (sejam indivíduos ou grupos) nessa tomada de decisão, é possível adotar uma abordagem direta ou indireta. Assim, pode existir desde um apoio que considere todas as possibilidades e sugira as melhores opções até um apoio que apenas forneça informações ao tomador de decisão para que este as interprete e decida (BARRETO, 2005). Em ambos os casos, a ideia é apoiar e não substituir os usuários. A decisão final continua sendo feita por humanos (RUS *et al*., 2003).

7.2. Problemas de Otimização

Durante uma tomada de decisão podem existir dois tipos de problemas: problemas de decisão e problemas de otimização. Problemas de decisão são aqueles cujo objetivo é apenas verificar se existe ou não uma solução viável, que atenda a determinadas restrições. Os problemas de otimização têm por finalidade buscar soluções próximas da ótima e que atendam um conjunto de restrições impostas, para maximizar ou minimizar um determinado fator (PAPADIMITRIOU e STEIGLITZ, 1998). O problema de adaptação tratado neste trabalho é entendido como um problema de otimização, devido à necessidade de maximizar ou minimizar os aspectos de colaboração e disciplina.

Métodos de otimização podem fornecer soluções para problemas complexos e podem sugerir formas de encontrar soluções aceitáveis em situações onde soluções perfeitas são teoricamente impossíveis ou praticamente inviáveis (HARMAN e JONES, 2001).

Cada solução gerada para o problema é conhecida como solução candidata. Se uma determinada solução candidata satisfaz todas as restrições do problema, então ela é chamada de solução viável. As soluções ótimas são soluções viáveis que conduzem a função de custo para o valor ótimo. O conjunto de todas as soluções candidatas é conhecido como espaço de busca.

A qualidade de uma solução é medida de acordo com o valor gerado pela função de custo e a sua proximidade do valor ótimo. Quanto mais próximo do valor ótimo estiver o valor calculado pela função de custo para uma determinada solução candidata, melhor será a qualidade desta solução candidata.

Existem dois tipos de métodos para resolver um problema de otimização. Os métodos de otimização capazes de encontrar soluções ótimas para os problemas são conhecidos como **métodos exatos**. Alguns exemplos de métodos exatos são: busca sistemática, programação linear e *branch and bound*. As descrições e exemplos de aplicações de cada um destes métodos podem ser encontrados na literatura (HARMAN, 2007, NETTO, 2010).

Entretanto, dependendo da natureza do problema, é impossível

encontrar a solução ótima, em tempo exequível, a partir de um método exato que se proponha a avaliar todas as soluções possíveis de um dado problema e selecionar a melhor dentre elas. Estes algoritmos muitas vezes não são aplicáveis a problemas de Engenharia de Software reais, onde existem múltiplos critérios e funções de custo complexas. Nestes casos, uma alternativa é utilizar **métodos heurísticos**.

Segundo Reeves (1993), heurísticas são técnicas que têm por finalidade a busca de soluções boas (próximas da solução ótima) para problemas complexos cuja solução ótima não pode ser encontrada para todas as instâncias em tempo exequível, pois o espaço de busca cresce em escala exponencial. Consistem na aplicação de regras que tentam guiar a solução para um ótimo local (a melhor solução considerando um subconjunto do conjunto das soluções viáveis para o problema), na esperança de que este ponto também seja o ótimo global (a melhor dentre todas as soluções possíveis).

Estas técnicas também possuem algumas limitações. Em geral, não oferecem garantias de qualidade e viabilidade das soluções geradas, pois nem sempre a solução ótima global será obtida e, muitas vezes, não é possível afirmar quão próxima a solução gerada está da solução ótima. A previsibilidade também é baixa, pois cada execução pode obter resultados diferentes, mas sempre melhores que os anteriores (HARMAN, 2007).

Apesar destes desafios, os métodos heurísticos têm sido aplicados com bastante frequência nas áreas de engenharia, financeira e industrial. Contudo, estas iniciativas ainda são recentes na área de Engenharia de Software.

O termo Engenharia de Software Baseada em Busca (*Search Based Software Engineering - SBSE*) foi cunhado por Harman e Jones (2001) e consiste em uma área de pesquisa que estuda a Engenharia de Software sob a ótica de problemas de busca e otimização, com a aplicação de técnicas heurísticas para buscar soluções aproximadas. Desde então, houve um acentuado crescimento no número de trabalhos nesta linha de pesquisa (CLARKE *et al*., 2003, HARMAN *et al*., 2009, HARMAN, 2007), em diversas áreas da Engenharia de Software, tais como análise de requisitos, projeto arquitetural, testes, manutenção, gerência da configuração e gerenciamento de projetos (NETTO, 2010).

Para Clarke *et al*. (2003), a análise da aderência da aplicação de uma solução heurística deve considerar os seguintes critérios: (i) o espaço de busca do problema deve ser tão grande (crescimento exponencial) que inviabilize a busca da solução através de um método exato em tempo polinomial; (ii) inexistência de soluções completas, capazes de prover uma solução para o problema para qualquer instância em tempo polinomial, pois não faz sentido tentar resolver um problema que já foi resolvido; (iii) existência de uma função de custo adequada ao contexto do problema; (iv) as soluções candidatas devem ser encontradas em tempos exequíveis, isto é, o tempo de resposta do algoritmo não pode comprometer a continuidade do processo. Se um determinado problema atende a estes quatro critérios, então uma solução heurística pode ser adequada para buscar uma solução aproximada.

Muitos problemas relacionados ao desenvolvimento de software podem ser encarados como problemas de otimização, onde há muitas alternativas a selecionar com base em um conjunto de restrições e as soluções próximas do ótimo são suficientes desde que se encaixem em um limite de tolerância aceitável. Por exemplo, em Karlstrom e Runeson (2002), é proposta uma abordagem de otimização no que diz respeito à introdução de *Extreme Programming* (XP) nas organizações e à seleção de suas práticas, considerando a facilidade de adoção e o potencial efeito benéfico de cada prática. Um problema similar, de alocação de desenvolvedores em tarefas de correção de *bugs* e o sequenciamento destas tarefas, foi tratado por Netto (2010) utilizando algoritmos genéticos.

Seguindo a ideia de que problemas da Engenharia de Software podem ser formulados como problemas de busca e otimização, na próxima seção, é apresentada a modelagem do problema de adaptação como um problema de otimização.

7.3. Modelagem do Problema de Balanceamento na Adaptação do Processo do Projeto

Segundo Harman (2007), existem 2 ingredientes principais para aplicação de otimização aos problemas de Engenharia de Software: escolher a representação do problema e definir as funções de custo. Atendendo a estes

pré-requisitos, é possível aplicar técnicas de otimização a outras áreas de Engenharia de Software, além daquelas que têm sido tradicionalmente exploradas, e obter resultados interessantes e potencialmente importantes.

Assim, na próxima seção é apresentada a definição detalhada do problema de otimização que se quer resolver neste trabalho: como adaptar o processo específico do projeto a partir da linha de processos, usando as informações de contexto de forma a balancear a colaboração e disciplina? (MAGDALENO, 2010b).

7.3.1. Definição do Problema

Seguindo a abordagem de linha de processo gerida por contexto apresentada no capítulo anterior, o Problema de Balanceamento na Adaptação do Processo do Projeto (PBAPP) inicia na fase de Engenharia de Aplicação do Processo (EAP), quando começa a adaptação do processo a partir da linha de processos, usando as informações de contexto para compor o processo específico do projeto. A visão geral das etapas que compõem o problema de otimização PBAPP é apresentada pela Figura 7.1.

A primeira etapa do problema envolve o recorte das características da linha de processos. Neste ponto, as *informações de contexto* (da organização, do projeto e da equipe) atual (IC) e as *regras de seleção de características* (RSCAI e RSCAE) com base nas informações de contexto atuam para selecionar as características apropriadas para o projeto (Figura 7.1a). Em seguida, as *regras de composição de características* (RCCAI e RCCAE) recortam as características que foram selecionadas (Figura 7.1b). Esta seleção e recorte das características provocam um filtro no universo dos componentes da linha de processos, já que ambos estão relacionados.

Na etapa seguinte, as *informações de contexto* (IC) e as *regras de seleção de componentes* baseadas em contexto (RCSPI e RCSPE) filtram aqueles componentes de processos que são mais adequados para o projeto (Figura 7.1c). A partir daí, devem ser feitas as combinações possíveis dos componentes de processos de acordo com as *regras de combinação de componentes* (RCCPI e RCCPE) (Figura 7.1d).

Estas combinações serão avaliadas, considerando-se o seu potencial de colaboração e disciplina, para que seja escolhida a melhor combinação. Por

fim, será composto o processo adaptado para o projeto através do encaixe dos componentes selecionados à *sequência de atividades* (ATIV) do *processo base* (PB) (Figura 7.1e).

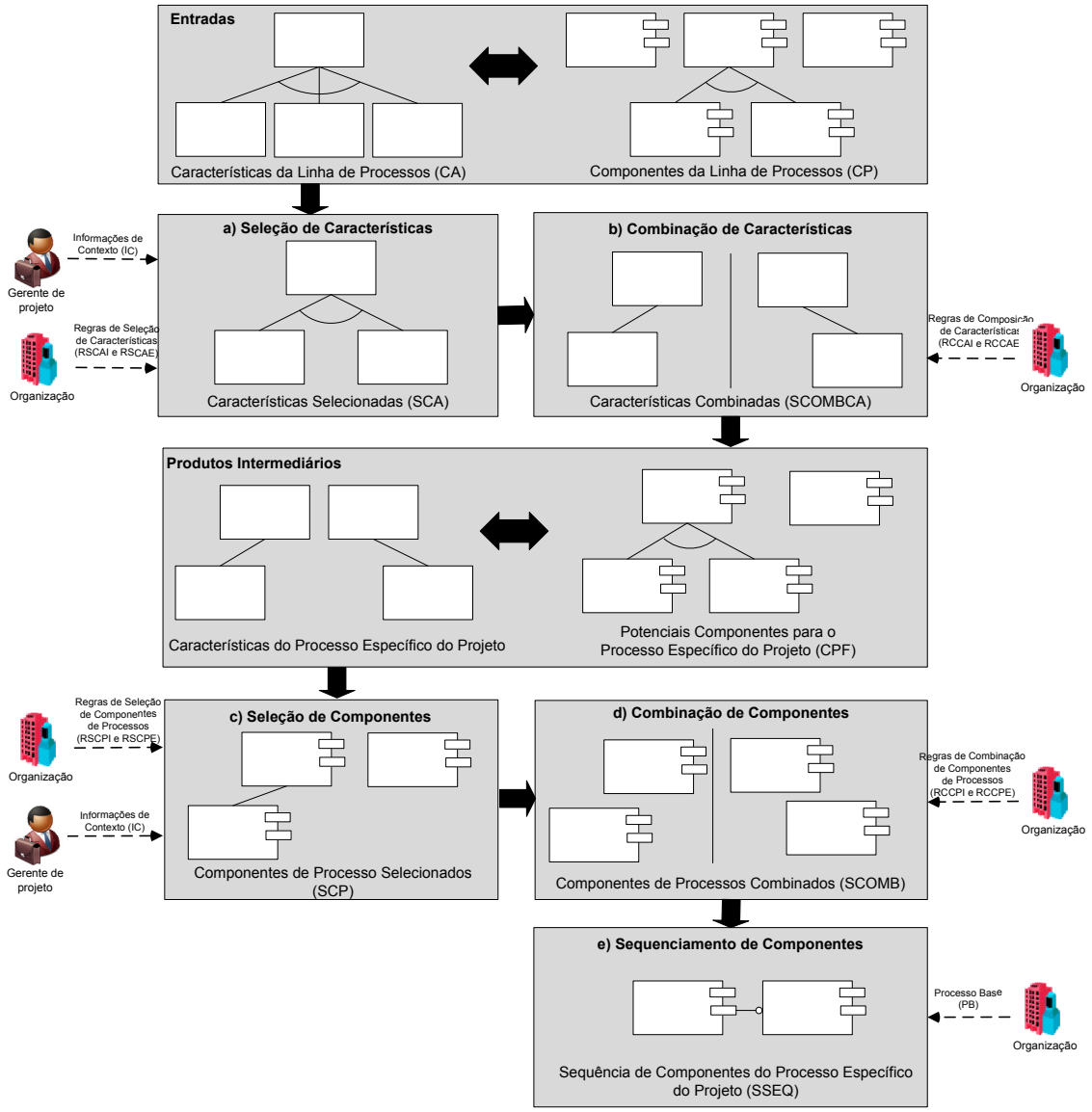


Figura 7.1 – Resumo das etapas do problema de balanceamento

7.3.2. Modelagem do Problema

Esta seção apresenta a modelagem formal de cada uma das fases do problema de otimização PBAPP. A modelagem inicial do problema está descrita em Magdaleno (2010b), mas desde então foi modificada com o acréscimo de novas etapas. Esta modelagem do problema é crítica para o entendimento da natureza do problema e também é útil para apontar as técnicas de otimização mais indicadas para serem implementadas.

Também vale ressaltar que esta modelagem se preocupou em garantir a

consistência entre os elementos de processos (atividades, artefatos, componentes e etc.) que forem adaptados em diferentes níveis e zelar para que todos os pré-requisitos das diferentes partes do processo sejam satisfeitos durante as operações de adaptação do processo.

7.3.2.1. Seleção das características do processo

A primeira etapa (Figura 7.1a) corresponde à seleção das características compatíveis com o contexto do projeto e pode ser modelada da seguinte forma:

- Seja CA o conjunto de todas as características (ou práticas de desenvolvimento de software) da linha de processos. Cada característica $CA_i \in$ CA possui um nome e um conjunto de componentes de processos.

  $CA = \{ CA_i \}$

  $CA_i = [nome_i, CP_i]$

  $\therefore CP_i \in CP$

- Seja CP o conjunto de todos os componentes de processos da arquitetura da linha de processos. Cada componente de processo $CP_i \in CP$ é composto por um nome, um potencial de colaboração, um potencial de disciplina, um conjunto de interfaces requeridas (artefatos de entrada) e um conjunto de interfaces produzidas (artefatos de saída).

  $CP = \{ CP_i \}$

  $CP_i = [nome_i, pcolab_i, pdisc_i, INTRCP_i, INTPCP_i]$

- Seja $INTRCP_i$ um conjunto de interfaces requeridas (artefatos de entrada) de um determinado componente de processo. Cada interface requerida $INTRCP_{ij} \in INTRCP_i$ possui um nome.

  $INTRCP_i = \{ INTRCP_{ij} \}$

  $INTRCP_{ij} = [nome_{ij}]$

- Seja $INTPCP_i$ um conjunto de interfaces providas (artefatos de saída) de um determinado componente de processo. Cada interface provida $INTPCP_{ij} \in INTPCP_i$ possui um nome.

  $INTPCP_i = \{ INTPCP_{ij} \}$

  $INTPCP_{ij} = [nome_{ij}]$

- Seja IC o conjunto de informações do contexto atual. Cada elemento $IC_i \in$ IC é composto por um nome e valor.

  $IC = \{ IC_i \}$

  $IC_i = [nome_i, valor_i]$

- Seja RSCAI um conjunto de regras de implicação para seleção de características. Estas regras indicam características que devem estar presentes em função das informações de contexto. Cada regra $RSCAI_i \in$ RSCAI é composta por um identificador, um antecedente (uma expressão de informações de contexto) e um consequente (um conjunto de características).

  $RSCAI = \{ RSCAI_i \}$

  $RSCAI_i = [id_i, ANTCAI_i, SEQCAI_i]$

  $\therefore ANTCAI_i = ecx$

  $SEQCAI_i = \{ CA_k \}$, onde $CA_k \in CA$

  $ecx = (ecx \text{ AND } ecx) \vee (ecx \text{ XOR } ecx) \vee (ecx \text{ OR } ecx) \vee CTX$

  onde, CTX = nome *op* valor

  onde, $\exists IC_i \in IC \rightarrow IC_i.nome = nome$

  e $op \in \{ =, \neq, >, \geq, <, \leq \}$

- Seja RSCAE um conjunto de regras de exclusão para seleção de características. Estas regras indicam características que não devem estar presentes em função das informações de contexto. Cada regra $RSCAE_i \in$ RSCAE é composta por um identificador, um antecedente (uma expressão de informações de contexto) e um consequente (um conjunto de características).

  $RSCAE = \{ RSCAE_i \}$

  $RSCAE_i = [id_i, ANTCAE_i, SEQCAE_i]$

  $\therefore ANTCAE_i = ecx$

  $SEQCAE_i = \{ CA_k \}$, onde $CA_k \in CA$

Para resolver esta primeira parte do problema, é necessário selecionar as características que são indicadas pelas regras de implicação baseadas em

contexto e não são indicadas pelas regras de exclusão baseadas em contexto. Esta seleção é expressa formalmente abaixo:

$$SCA = \{ SCA_i \in CA \mid \alpha (SCA_i) \wedge \beta (SCA_i) \}$$

A formulação acima indica que existe um conjunto de características, dentro do universo de características da linha de processos, para o qual as condições α e β são verdadeiras.

A primeira condição α é a seguinte:

$$\alpha(s) \leftarrow \exists\, RSCAI_i \in RSCAI \mid s \in RSCAI_i.SEQCAI \wedge eval(RSCAI_i.ANTCAI, IC) = true$$

onde *eval*(ecx, IC) assume um valor booleano caso as informações de contexto satisfaçam ou não o antecedente da regra.

A fórmula acima estabelece que existem regras de implicação para seleção de características para as quais existem características pertencentes ao consequente destas regras e o antecedente desta regra é satisfeito com as informações de contexto. Em outras palavras, as informações do contexto atual satisfazem a expressão de contexto que forma o antecedente destas regras e assim as características que pertencem ao consequente são selecionadas.

A segunda condição β expressa na fórmula do problema é a seguinte:

$$\beta(s) \leftarrow \nexists\, RSCAE_i \in RSCAE \mid s \in RSCAE_i.SEQCAE \wedge eval(RSCAE_i.ANTCAE, IC) = true$$

A expressão acima indica que não existem regras de exclusão para seleção de características para as quais existem características pertencentes ao consequente destas regras e o antecedente desta regra é satisfeito com as informações de contexto. Caso as informações do contexto atual satisfaçam a expressão de contexto que forma o antecedente destas regras de exclusão, as características que compõem o consequente das regras não poderiam ser selecionadas, pois seriam excluídas.

7.3.2.2. Combinação de características

A segunda etapa (Figura 7.1b) recebe como entrada um conjunto de características compatíveis com o contexto do projeto selecionadas na etapa anterior (SCA) e recorta estas características de acordo com as regras de composição, gerando como resultado uma lista de potenciais combinações destas características. Esta etapa pode ser modelada da seguinte forma:

- Seja $SCOMBCA_1$ o conjunto com todas as combinações de características vindas de SCA.

  $SCOMBCA_1 = \{ SCA_i \}$

  $SCA_i \subseteq SCA \wedge \nexists\ s \subseteq SCA \mid s \notin SCOMBCA_1$

- Seja RCCAI um conjunto de regras de implicação para composição de características. Estas regras indicam características que devem estar presentes em função da presença de outras características. Cada regra $RCCAI_i \in RCCAI$ é composta por um identificador, um antecedente (uma expressão de características) e um consequente (um conjunto de características).

  $RCCAI = \{ RCCAI_i \}$

  $RCCAI_i = [id_i, ANTCCAI_i, SEQCCAI_i]$

  $\therefore ANTCCAI_i = eca$

  $SEQCCAI_i = \{ CA_k \}$, onde $CA_k \in SCA$

  $eca = (eca\ AND\ eca) \vee (eca\ XOR\ eca) \vee (eca\ OR\ eca) \vee CAR$

  $\therefore CAR \in CA$

- Seja RCCAE um conjunto de regras de exclusão para composição de características. Estas regras indicam características que não devem estar presentes em função da presença de outras características. Cada regra $RCCAE_i \in RCCAE$ é composta por um identificador, um antecedente (uma expressão de características) e um consequente (um conjunto de características).

  $RCCAE = \{ RCCAE_i \}$

  $RCCAE_i = [id_i, ANTCCAE_i, SEQCCAE_i]$

  $\therefore ANTCCAE_i = eca$

  $SEQCCAE_i = \{ CA_k \}$, onde $CA_k \in SCA$

Para selecionar a lista de todas as combinações de características viáveis segundo as regras de composição de características, é necessário filtrar o conjunto $SCOMBCA_1$ e remover os elementos que não atendem a estas regras. Desta vez não é mais o contexto quem vai selecionar as características pertinentes, mas as restrições existentes entre elas é que vão limitar as combinações possíveis. Esta seleção é expressa formalmente a seguir:

$SCOMBCA = \{ SCOMBCA_i \in SCOMBCA_1 \mid \alpha (SCOMBCA_i) \wedge \beta (SCOMBCA_i) \}$

$$\alpha(r) \leftarrow \forall RCCAI_i \in RCCAI \mid evaluate(RCCAI_i.ANTCCAI, r) = true$$
$$\Rightarrow \nexists CA_k \in RCCAI_i.SEQCCAI \mid CA_k \notin r$$

$$\beta(r) \leftarrow \nexists RCCAE_i \in RCCAE \mid evaluate(RCCAE_i.ANTCCAE, r) = true$$
$$\Rightarrow \nexists CA_k \in RCCAE_i.SEQCCAE \mid CA_k \in r$$

onde *evaluate*(eca, SCOMBCA) assume um valor booleano de acordo com a forma como as características SCOMBCA satisfazem a expressão *eca.*

Ao final desta fase, o gerente de projeto pode avaliar a pertinência das características opcionais sugeridas para identificar se existem características que podem ou devem ser excluídas sem prejuízo para o projeto.

7.3.2.3. Seleção dos componentes de processos

A etapa de seleção dos componentes de processo compatíveis com o contexto do projeto (Figura 7.1c) pode ser modelada da seguinte forma:

- Seja CPF o conjunto de componentes de processos filtrados da arquitetura da linha de processos após o recorte das características. A definição do componente de processo $CPF_i \in CPF$ é feita da mesma forma que a definição dos componentes CP.

  $CPF = \{ CPF_i \}$

  $\therefore CPF_i \in CP$

- Seja RSCPI um conjunto de regras de implicação para seleção de componentes de processos. Estas regras indicam componentes que devem

estar presentes em função das informações de contexto. Cada regra $RSCPI_i \in RSCPI$ é composta por um identificador, um antecedente (uma expressão de informações de contexto) e um consequente (um conjunto de componentes de processos).

$RCSPI = \{ RCSPI_i \}$

$RCSPI_i = [id_i, ANTCI_i, SEQCI_i]$

$\therefore ANTCI_i = ecx$

$SEQCI_i = \{ CP_k \}$, onde $CP_k \in CPF$

- Seja RSCPE um conjunto de regras de exclusão para seleção de componentes de processos. Estas regras indicam componentes que não devem estar presentes em função das informações de contexto. Cada regra $RSCPE_i \in RSCPE$ é composta por um identificador, um antecedente (uma expressão de informações de contexto) e um consequente (um conjunto de componentes de processos).

  $RCSPE = \{ RCSPE_i \}$

  $RCSPE_i = [id_i, ANTCE_i, SEQCE_i]$

  $\therefore ANTCE_i = ecx$

  $SEQCE_i = \{ CP_k \}$, onde $CP_k \in CP$

Para resolver esta parte do problema, é necessário selecionar o conjunto de componentes de processo disponíveis para o projeto, que são indicados pelas regras de implicação de contexto e não são indicados pelas regras de exclusão de contexto. Seguindo a mesma lógica da primeira etapa, esta seleção é expressa formalmente abaixo:

$$SCP = \{ SCP_i \in CPF \mid \alpha (SCP_i) \wedge \beta (SCP_i) \}$$

$\alpha(u) \leftarrow \exists RSCPI_i \in RSCPI \mid u \in RSCPI_i.SEQCI \wedge eval(RSCPI_i.ANTCI, IC) = true$

$\beta(u) \leftarrow \nexists RSCPE_i \in RSCPE \mid u \in RSCPE_i.SEQCE \wedge eval(RSCPE_i.ANTCE, IC) = true$

7.3.2.4. Combinação dos componentes de processo

A etapa de combinação dos componentes de processo (Figura 7.1d) recebe como entrada uma lista de componentes de processo compatíveis com o projeto de acordo com suas informações de contexto (SCP) e gera como

resultado uma lista de potenciais combinações destes componentes que podem ser utilizadas na formação do processo base (SCOMB). Esta etapa pode ser modelada da seguinte forma:

- Seja $SCOMB_1$ o conjunto com todas as combinações de componentes de processos vindas de SCP.

  $SCOMB_1 = \{ SCP_i \}$

  $SCP_i \subseteq SCP \wedge \nexists\, s \subseteq SCP \mid s \notin SCOMB_1$

- Seja RCCPI um conjunto de regras de implicação para combinação de componentes de processos. Estas regras indicam componentes que devem estar presentes em função da presença de outros componentes. Cada regra $RCCPI_i \in RCCPI$ é composta por um identificador, um antecedente (uma expressão de componentes de processos) e um consequente (um conjunto de componentes de processos).

  $RCCPI = \{ RCCPI_i \}$

  $RCCPI_i = [id_i, ANTI_i, SEQI_i]$

  $\therefore ANTI_i = ecp$

  $SEQI_i = \{ CP_k \}$, onde $CP_k \in SCP$

  $ecp = (ecp \text{ AND } ecp) \vee (ecp \text{ XOR } ecp) \vee (ecp \text{ OR } ecp) \vee COMP$

  $\therefore COMP \in SCP$

- Seja RCCPE um conjunto de regras de exclusão para combinação de componentes de processos. Estas regras indicam componentes que não devem estar presentes em função da presença de outros componentes. Cada regra $RCCPE_i \in RCCPE$ é composta por um identificador, um antecedente (uma expressão de componentes de processos) e um consequente (um conjunto de componentes de processos).

  $RCCPE = \{ RCCPE_i \}$

  $RCCPE_i = [id_i, ANTE_i, SEQE_i]$

  $\therefore ANTE_i = ecp$

  $SEQE_i = \{ CP_k \}$, onde $CP_k \in SCP$

Para selecionar a lista de todas as combinações de componentes viáveis segundo as regras de combinações de componentes, é necessário filtrar o conjunto $SCOMB_1$ e remover os elementos que não atendem a estas regras. Seguindo a mesma lógica da segunda etapa, a solução da etapa atual pode ser expressa da seguinte forma:

$$SCOMB = \{ SCOMB_i \in SCOMB_1 \mid \alpha (SCOMB_i) \wedge \beta (SCOMB_i) \}$$

$$\alpha(t) \leftarrow \forall RCCPI_i \in RCCPI \mid evaluate(RCCPI_i.ANTI, t) = true$$
$$\Rightarrow \nexists CP_k \in RCCPI_i.SEQI \mid CP_k \notin t$$

$$\beta(t) \leftarrow \forall RCCPE_i \in RCCPE \mid evaluate(RCCPE_i.ANTI, t) = true$$
$$\Rightarrow \nexists CP_k \in RCCPE_i.SEQE \mid CP_k \in t$$

onde *evaluate*(ecp, SCOMB) assume um valor booleano de acordo com a forma como os componentes SCOMB satisfazem a expressão *ecp.*

7.3.2.5. Sequenciamento dos componentes de processo

A etapa de sequenciamento dos componentes de processo (Figura 7.1e) para preencher as atividades do processo base ocorre após a etapa de combinação dos componentes de processos (SCOMB). Cada atividade do processo base pode ser preenchida por um componente de processo ou ser executada só por ela mesma. Como os componentes de processos podem estar em diferentes níveis de granularidade, pode ser necessário que uma atividade do processo base seja preenchida por mais de um componente. Tanto as atividades quanto os componentes possuem interfaces (artefatos) providas e requeridas que devem ser respeitadas no momento do sequenciamento. Assim, esta etapa pode ser modelada da seguinte forma:

- Seja PB o processo base definido pela organização. O PB é composto por um nome e por um conjunto de atividades.

  PB = [nome, ATIV]

- Seja ATIV um conjunto de atividades pertencente a um dado processo base PB. Cada atividade $ATIV_i \in$ ATIV é composta por um nome, um conjunto de

pré-atividades, um conjunto de interfaces requeridas (artefatos de entrada) e um conjunto de interfaces produzidas (artefatos de saída).

$ATIV = \{ ATIV_i \}$

$ATIV_i = [nome_i, PRE\text{-}ATIV_i, INTRATIV_i, INTPATIV_i]$

$\therefore PRE\text{-}ATIV_i = \{ ATIV \}$

- Seja $INTRATIV_i$ um conjunto de interfaces requeridas (artefatos de entrada) de uma determinada atividade. Cada interface requerida $INTRATIV_{ij} \in INTRATIV_i$ possui um nome.

  $INTRATIV_i = \{ INTRATIV_{ij} \}$

  $INTRATIV_{ij} = [nome_{ij}]$

- Seja $INTPATIV_i$ um conjunto de interfaces providas (artefatos de saída) de uma determinada atividade. Cada interface provida $INTPATIV_{ij} \in INTPATIV_i$ possui um nome.

  $INTPATIV_i = \{ INTPATIV_{ij} \}$

  $INTPATIV_{ij} = [nome_{ij}]$

Para resolver o problema, é necessário associar os componentes de processo que farão parte de cada atividade do processo base.

$SCPATIV = \{ SCPATIV_i \}$

$SCPATIV_i = [AT_i, \{ C_{ij} \}]$

$\therefore AT_i \in ATIV$

$\forall ATIV_i \in ATIV \subset PB \mid \exists SCPATIV_i.AT = ATIV_i$

$C_{ij} \leftarrow componentes(AT_i, SCOMB_i)$

componentes(AT_i, $SCOMB_i$)

{

$SUB = \{ SUB_i \}$

$\therefore SUB_i \subseteq SCOMB_i \wedge \nexists\, s \subseteq SCOMB_i \mid s \notin SUB$

$\Phi \leftarrow \phi$; componentes selecionados

Para cada $SUB_i \in SUB$

$\forall r \in AT_i.INTRATIV \rightarrow \exists c \in SUB_i \mid r \in c.\ INTRCP$

$\forall s \in AT_i.INTPATIV \rightarrow \exists c \in SUB_i \mid s \in c.\ INTPCP$

$\forall c \in SUB_i \rightarrow \forall r \in c.\ INTRCP \mid r \in \cup\ SUB_i.INTPCP \vee r \in AT_i.INTRATIV$

Se todas estas condições são satisfeitas, então:

$\Phi \leftarrow \Phi \cup SUB_i$

}

Ao final é apresentada uma sugestão inicial de adaptação do processo para o projeto. Esta sugestão poderá ser modificada pelo gerente de projeto. Neste momento, o gerente de projeto pode incluir novos componentes de processo, quando julgar necessário, baseando-se nos componentes incluídos em projetos anteriores ou definindo um novo componente. Também é possível excluir componentes de processo que o gerente identifique como não sendo necessários ao projeto.

7.3.3. Funções de custo

Normalmente, há várias soluções candidatas para um problema de otimização. A otimização consiste em encontrar o conjunto de parâmetros para os quais as funções de custo têm um valor máximo ou mínimo. As funções de custo caracterizam o que é considerada uma boa solução. Cada função de custo terá uma solução ótima local própria, potencialmente distinta das soluções ótimas locais geradas para outras funções de custo. As funções de custo ajudam a identificar a técnica de busca mais aplicável e a entender melhor a natureza do problema das soluções candidatas em termos dos seus custos (HARMAN e JONES, 2001).

Como o problema deste trabalho de pesquisa envolve o balanceamento dos aspectos de colaboração e disciplina na adaptação do processo, estes aspectos podem ser usados como funções de custo para determinar fatores de qualidade para a solução do problema. Assim, simplificadamente, são consideradas as seguintes funções:

- **Maior Colaboração:** esta função é responsável por determinar a seleção de componentes de processos que satisfaz a todas as restrições do problema e representa o maior potencial de colaboração para o projeto. A colaboração é calculada pelo somatório do potencial de colaboração existente em cada componente de processo incluído no processo definido para o projeto.

- Seja S o conjunto de soluções possíveis que satisfazem a todas as restrições. Seja Colab(S_k) uma função que retorne a colaboração de um dado componente de processo S_k. O componente selecionado $S_i \in S$ é escolhido de acordo com a seguinte equação:

$$\nexists(S_j \in S)\ (\mathrm{Colab}(S_j) < \mathrm{Colab}(S_i))$$

- **Maior Disciplina:** esta função é responsável por determinar a seleção de componentes de processos que satisfaz a todas as restrições do problema e representa o maior potencial de disciplina para o projeto. A disciplina é calculada pelo somatório do potencial de disciplina existente em cada componente de processo incluído no processo definido para o projeto.
  - Seja S o conjunto de soluções possíveis que satisfazem a todas as restrições. Seja Disc(S_k) uma função que retorne a disciplina de um dado componente de processo S_k. O componente selecionado $S_i \in S$ é escolhido de acordo com a seguinte equação:

$$\nexists(S_j \in S)\ (\mathrm{Disc}(S_j) < \mathrm{Disc}(S_i))$$

7.3.4. Análise do espaço de busca

Para caracterizar o espaço de busca é importante analisar a sua complexidade e tamanho. Isso ajuda a entender a dificuldade do problema e a identificar a técnica mais adequada para aplicar.

A função complexidade do tempo de um algoritmo expressa o maior tempo necessário (pior caso) para solucionar um determinado problema. Em geral, esta função é representada na forma O(n), onde n significa o tamanho da instância do problema. Algoritmos de tempo polinomial são aqueles cuja função complexidade do tempo pode ser definida como O(p(n)) para uma determinada função polinomial p. Os demais algoritmos, cujos tempos de execução não podem ser delimitados, são conhecidos como algoritmos de tempo exponencial. No caso dos algoritmos exponenciais, a solução exata para instâncias grandes do problema não pode ser encontrada em tempo exequível. Nestes casos, devem ser utilizados algoritmos, que executam em tempo polinomial, para encontrar soluções aproximadas para o problema (NETTO, 2010).

No problema PBAPP, a complexidade é influenciada pelo número de

componentes de processos (*p*) e de atividades (*a*). Considerando estes fatores, a complexidade do problema é exponencial: $O(p!^a)$. Entretanto, quando as restrições são adicionadas na formulação do problema, o esforço necessário para encontrar as soluções tende a decrescer, devido ao número de caminhos que não atendem às restrições. Desta forma, o número de possíveis componentes de processos para preencher cada atividade do processo base diminui para *p'*, onde $p' < p$.

Na prática, o espaço de busca cresce proporcionalmente a $(p'!^a)$. Por exemplo, considerando um processo base, como o ciclo de vida em cascata, com uma média de 5 atividades e supondo, com base na abordagem ágil do OpenUP[12], que um modelo de desenvolvimento de software tem uma média de 20 componentes, podemos estimar que uma linha de processos preparada para trabalhar com 5 modelos de desenvolvimento distintos, vai possuir aproximadamente 100 componentes de processos. Assim, o espaço completo de busca tem aproximadamente $(100!^5)$ possíveis soluções.

O crescimento fatorial do espaço de busca inviabiliza a aplicação de métodos exatos, pois a solução não pode ser encontrada em um tempo exequível. Assim, uma solução utilizando métodos heurísticos, tais como os algoritmos genéticos, parece ser a opção mais indicada para implementar a solução deste problema PBAPP.

7.4. Considerações finais

Este capítulo abordou o problema da adaptação de processos de desenvolvimento de software como um problema de otimização. A definição e modelagem formal do problema foram apresentadas. Esta modelagem do problema é crítica para definir a natureza do problema de busca e apontar as técnicas de otimização mais indicadas para serem implementadas. A análise do espaço de busca também mostrou que ele é grande o suficiente para justificar o uso de métodos heurísticos.

Como próximos passos pretende-se definir mais claramente como as funções de custo devem ser calculadas, modelar computacionalmente o problema e selecionar qual técnica é a mais apropriada para ser aplicada na

[12] Site: http://www.eclipse.org/epf/openup_component/openup_vision.php

solução deste problema. Talvez a solução envolva inclusive a combinação de mais de uma técnica de otimização, como foi feito em outros trabalhos (BARRETO, 2005, NETTO, 2010). Clarke *et al*. (2003) sugerem que se comece pela técnica mais simples para verificar se os resultados são encorajadores antes de investir em outras técnicas mais sofisticadas que podem representar um esforço adicional desnecessário.

Como a implementação da solução ainda não está definida, não é possível planejar detalhadamente como esta abordagem será avaliada, mas alguns caminhos para serem explorados no futuro podem ser vislumbrados. Uma primeira alternativa é estabelecer um *baseline* aceitável, comparando a execução da técnica heurística com uma busca totalmente aleatória. A técnica heurística deve então encontrar soluções melhores e com menos esforço computacional (HARMAN, 2007). Também é possível avaliar a solução proposta comparando-a com outra técnica que implemente a solução do mesmo problema para analisar o tempo e a qualidade da solução.

A análise de sensibilidade é outra opção de estratégia de avaliação. A análise de sensibilidade permite investigar o grau de influência que cada um dos parâmetros de entrada provoca na solução e identificar quais parâmetros contribuem mais para a localização da solução ótima local. Esta análise consiste em selecionar cada um dos parâmetros, variar o valor do mesmo mantendo os valores dos demais fixos e observar a influência na qualidade da solução gerada e no tempo de resposta (HARMAN, 2007, NETTO, 2010).

Vale ressaltar também que independente da(s) alternativa(s) escolhida(s) para a avaliação, é preciso primeiramente popular uma base de dados experimental com os elementos que compõem os conjuntos definidos de características e componentes de processos. Em seguida, é necessário estabelecer cenários de teste com conjuntos de informações de contexto e regras de implicação e composição.

O próximo capítulo apresenta as considerações finais gerais desta proposta de tese.

8. Conclusão

Neste capítulo são apresentadas as considerações finais sobre esta proposta de tese, incluindo as principais contribuições, a metodologia aplicada ou prevista na continuidade desta pesquisa, os resultados preliminares obtidos, os resultados esperados e o cronograma de atividades previsto até a conclusão do trabalho.

8.1. Principais contribuições

Este trabalho oferece contribuições para diversas áreas de pesquisa. Do ponto de vista da área de conciliação de modelos de desenvolvimento de software, foi conduzido um estudo secundário (revisão *quasi*-sistemática) para caracterizar a conciliação de modelos de desenvolvimento de software (MAGDALENO *et al*., 2009). Esta caracterização é importante para os profissionais que desejam manter-se atualizados com o estado da prática. Esta revisão também ajuda a comunidade científica a construir um entendimento comum dos desafios que devem ser enfrentados, bem como a identificar os temas que têm sido pesquisados e onde a investigação ainda está faltando. Os resultados de tal investigação também são relevantes para a indústria de software que está buscando soluções nesta área.

Para a pesquisa em adaptação de processos de desenvolvimento de software, este trabalho se diferencia ao propor uma abordagem balanceada, sistemática e dinâmica para enfrentar os principais desafios da área e, ao mesmo tempo, suprir a carência por trabalhos que efetivamente orientem o gerente de projeto nesta tarefa.

Em relação à área de pesquisa em redes sociais, este trabalho contribuiu com o estudo de ferramentas de mineração, visualização e análise de redes sociais e com a identificação dos principais requisitos para ferramentas deste tipo (MAGDALENO *et al*., 2010). Além disso, pretende-se ainda construir a ferramenta EvolTrack-SocialNetwork para atender aos requisitos identificados e assim permitir mais pesquisas que explorem a colaboração através de redes sociais. Como primeiro passo neste sentido de analisar a colaboração através de redes sociais, o trabalho de Santos et al. (2010), que contou com a participação da autora desta proposta de tese,

mostrou como é possível combinar as propriedades de análise de redes sociais para identificar características de colaboração, mais especificamente de coordenação, nas redes sociais de desenvolvimento de software. Exemplos de uso destas características de colaboração no ambiente de apoio à decisão ainda serão explorados.

Ainda que não seja o foco principal deste trabalho, também foram feitas, em conjunto com outros alunos do grupo de pesquisa, contribuições para a área de reutilização de processos. A abordagem proposta se destaca ao integrar meta-modelo, notação, método e ferramental de apoio. Em particular, o método criado se diferencia das principais propostas existentes na área ao explorar a etapa de análise de domínio, onde é contruído o modelo de características, antes de se chegar ao modelo de componentes de processos na fase de projeto do domínio, onde a maioria dos trabalhos se concentra.

Como a gestão de contexto é foco de outra tese de doutorado e utilizada apenas como instrumento de apoio neste trabalho, as contribuições nesta área de pesquisa se limitam à sua instanciação para o domínio de processo de desenvolvimento de software. Assim, demonstrou-se como a abordagem UbiFEX (FERNANDES, 2008) e todos os seus elementos de dimensões, informações, situações e regras de contexto podem ser adotados em processos de desenvolvimento de software. Também foi realizado um estudo que indicou a combinação de linguagem e ferramenta de representação de contexto mais adequadas para este domínio (NUNES *et al*., 2010a).

Por fim, este trabalho também pretende contribuir para a área de apoio à decisão ou, mais especificamente, para a área de SBSE (HARMAN, 2007). Ao adotar técnicas de otimização no contexto de adaptação de processos, que praticamente ainda não tinha sido explorado, este trabalho aponta para um novo universo de possibilidades de pesquisa para a área. Além disso, devido às características do problema de adaptação que se quer resolver, também é possível que técnicas de otimização avançadas e ainda pouco exploradas em Engenharia de Software precisem ser adotadas, o que também acrescenta conhecimento novo sobre o assunto.

Ao resolver o problema de adaptação de processos de desenvolvimento de software de forma balanceada, sistemática e dinâmica, espera-se beneficiar tanto as organizações de desenvolvimento de software, quanto o gerente de

projetos. Ao reduzir o esforço requerido para a adaptação de processos e aumentar a adequação dos processos gerados, as organizações se aproximam dos seus objetivos de produtividade e qualidade. Por sua vez, o gerente de projeto será devidamente guiado e apoiado em sua tarefa de adaptar o processo e terá maior visibilidade e percepção do contexto do projeto atual, de outros projetos similares, da execução do processo e da colaboração entres os membros da equipe. Desta forma, o gerente de projeto torna-se mais capaz de avaliar as sugestões oferecidas pelo ambiente de apoio à decisão quanto à sua pertinência e relevância.

8.2. Metodologia de pesquisa

A metodologia de pesquisa utilizada para a realização deste trabalho inclui atividades realizadas antes da proposta de tese, para apoiar a elaboração da proposta, e atividades a serem realizadas após o exame de qualificação, para permitir o desenvolvimento e a conclusão da pesquisa proposta.

8.2.1. Atividades realizadas ou em andamento

O conjunto de atividades realizadas antes da proposta de tese e que apoiaram a sua elaboração é descrito a seguir:

- **Realizar revisão convencional da literatura:** esta atividade representa uma revisão *ad-hoc* da literatura sobre algumas áreas fundamentais: processos de desenvolvimento de software, adaptação de processos, redes sociais, linha de processos, gestão de contexto e apoio a decisão.

Esta pesquisa incluiu a busca por trabalhos relacionados a este e relevantes para a caracterização das áreas. Uma versão inicial deste corpo de conhecimento já está disponível. No entanto, ao longo do desenvolvimento desta pesquisa, este corpo de conhecimento deve ser evoluído e mantido atualizado.

- **Realizar revisão *quasi*-sistemática da literatura:** o objetivo desta revisão *quasi*-sistemática é caracterizar a conciliação dos modelos de desenvolvimento de software orientado ao planejamento, ágil e livre.

A primeira execução da revisão *quasi*-sistemática foi realizada em abril/2009, envolvendo quatro bases de dados digitais. Esta primeira execução foi concluída e os seus resultados foram publicados através de um relatório técnico (MAGDALENO *et al*., 2009) e submetidos para publicação de um artigo no *Journal of Systems and Software (JSS)*.

- **Definir proposta de tese:** com o corpo de conhecimento estabelecido e com os resultados obtidos da revisão *quasi*-sistemática, foi possível trabalhar na definição da proposta de tese.

Esta proposta inicial foi publicada em (MAGDALENO, 2010a) e discutida com a comunidade científica de Engenharia de Software no *Doctoral Symposium of International Conference on Software Engineering* (ICSE). Esta discussão contribuiu para revisão e detalhamento da proposta inicial, ao mesmo tempo em que demonstrou a relevância do tema.

- **Definir os instrumentos para planejar e acompanhar a colaboração e a disciplina:** o CollabMM, as redes sociais e a medição foram definidos como instrumentos para apoiar o planejamento e acompanhamento dos níveis de colaboração e disciplina no processo adaptado.

- **Acompanhar a proposição das características de colaboração:** para auxiliar o balanceamento entre colaboração e disciplina, a identificação das características de colaboração existentes nas redes sociais (SANTOS *et al*., 2010, 2009) e a sua associação aos níveis de colaboração do CollabMM (MAGDALENO *et al*., 2009) é um importante instrumento e será utilizado como parte da solução deste trabalho para analisar a colaboração.

- **Especificar o EvolTrack-SocialNetwork:** especificação dos requisitos e da visão geral da arquitetura do EvolTrack-SocialNetwork, com base nas ferramentas estudadas (MAGDALENO *et al*., 2010) e nos objetivos do presente trabalho de pesquisa. Como o EvolTrack (CEPEDA *et al*., 2008) será utilizado como ponto de partida para a construção da ferramenta de redes sociais, também foram realizados alguns testes iniciais de uso do EvolTrack em projetos de desenvolvimento de software livre para analisar a sua viabilidade (CEPEDA *et al*., 2010).

- **Definir o modelo de representação de contexto:** escolher o modelo de representação de contexto mais adequado para o domínio de processos de desenvolvimento de software.

Esta atividade foi realizada através de um estudo que analisou o uso de linguagens de representação de contexto (modelos de características e ontologias), levando em consideração também a disponibilidade de ferramentas de apoio (NUNES *et al*., 2010a). Como resultado, no presente trabalho optou-se por utilizar o modelo de características e a abordagem UbiFex (FERNANDES, 2008).

- **Definir conjunto inicial de informações de contexto:** o levantamento inicial das informações de contexto relevantes para o domínio do desenvolvimento de software foi realizado através da revisão e adaptação de trabalhos já publicados na literatura e da experiência da autora, mas ainda requer formalização, ampliação e validação.

- **Definir conjunto inicial de situações e regras de contexto:** a definição inicial das situações e regras de contexto para o desenvolvimento de software foi realizada apenas a título de exemplo.

- **Participar da definição da abordagem para engenharia de linha de processos:** discussão e estabelecimento, junto com outros alunos de doutorado e mestrado, da proposta para engenharia de linha de processos de acordo com as fases de engenharia de domínio e engenharia de aplicação (NUNES *et al*., 2010b).

- **Detalhar abordagem para relacionar linha de processos e gestão de contexto (*em andamento*):** a abordagem para relacionar engenharia de linha de processos com gestão de contexto foi iniciada, mas ainda precisa ser detalhada e complementada para incluir as informações de colaboração e disciplina.

- **Analisar alternativas para o problema de balanceamento:** as possibilidades de uso de soluções como *contextual graphs* (BREZILLON e POMEROL, 1999, SANTOS, 2008) e CSP (*Constraint Satisfaction Problem*) (BARRETO, 2005, RUHE, 2003) foram investigadas para apoiar a solução do problema de balanceamento. Como resultado, dentro da área de apoio à decisão, optou-se por trabalhar com problemas de otimização.

- **Definir e modelar o problema de balanceamento:** o problema de balanceamento de colaboração e disciplina na adaptação do processo específico para o projeto foi definido e modelado como um problema de otimização (MAGDALENO, 2010b).

- Escrever e defender o **exame de qualificação** perante uma banca de professores que avaliam os riscos e a viabilidade da proposta.

- **Publicar os resultados:** Durante a fase de pesquisa, foram produzidos e publicados textos com a intenção de compartilhar a produção e o atual estágio de pesquisa com a comunidade acadêmica, divulgando os resultados inicias em conferências nacionais e internacionais. Estes são listados a seguir:
    - MAGDALENO, A. M., 2010, "An optimization-based approach to software development process tailoring". In: *PhD Track - International Symposium on Search Based Software Engineering (SSBSE)*, Benevento, Italy (to appear).
    - MAGDALENO, A. M., 2010, "Balancing Collaboration and Discipline in Software Development Processes". In: *Doctoral Symposium of International Conference on Software Engineering (ICSE)*, pp. 331-332, Cape Town, South Africa.

- CEPEDA, R. D. S. V.; MAGDALENO, A. M.; MURTA, L. G. P.; WERNER, C. M. L., 2010, "EvolTrack: Improving Design Evolution Awareness in Software Development", *Journal of the Brazilian Computer Society (JBCS)* (to appear).

- SANTOS, T. A. L.; ARAUJO, R. M. D.; MAGDALENO, A. M., 2010, "Identifying Collaboration Patterns in Software Development Social Networks", *Infocomp - Journal of Computer Science (*to appear*).*

- MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M., 2010, *Estudo de Ferramentas de Mineração, Visualização e Análise de Redes Sociais*. Relatório Técnico ES-735/10, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

- NUNES, V. T.; MAGDALENO, A. M.; WERNER, C. M. L., 2010, *Modelagem de contexto sobre o domínio de processos de desenvolvimento de software*. Relatório Técnico ES-734/10, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

- SANTOS, T. A. L.; ARAUJO, R. M.; MAGDALENO, A. M., 2009, "Padrões para Percepção da Colaboração em Redes Sociais de Desenvolvimento de Software". *Workshop de Desenvolvimento Distribuído de Software (WDDS)*, pp. 1-10, Fortaleza, CE, Brasil.

- MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M., 2009, *Revisão Quasi-Sistemática da Literatura: Conciliação de processos de desenvolvimento de software*, Relatório Técnico ES-730, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

Neste período outros artigos também foram submetidos para publicação e ainda encontram-se em revisão:

- MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M. D. “Reconciling Software Development Processes: A Quasi-Systematic Review”. *Journal of Systems and Software (JSS)*.

- MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M. D.; BARROS, M. “Formulando a Adaptação de Processos de Desenvolvimento de Software como um Problema de Otimização”. *I Workshop Brasileiro de Otimização em Engenharia de Software (WOES)*.

8.2.2. Atividades previstas

Após a defesa do exame de qualificação, o presente trabalho de pesquisa deverá prosseguir com o seguinte conjunto de atividades:

- **Refinar a proposta de solução:** ajustar a solução proposta de acordo com os comentários feitos pela banca durante o exame de qualificação.

- **Detalhar os instrumentos:** detalhar o uso dos instrumentos de planejamento e acompanhamento da colaboração e disciplina no processo adaptado.

- **Reexecutar a revisão *quasi*-sistemática sobre conciliação:** em 2010, é possível repetir a execução do protocolo, tentando capturar referências para trabalhos mais recentes que estendem o espaço de busca cronologicamente. Além disso, também é possível incluir outras máquinas de busca, que ainda não haviam sido utilizadas, para tentar recuperar documentos indexados apenas por estas máquinas, o que estenderia o espaço de busca geograficamente. Desta forma, aumenta-se a confiabilidade e a completeza do estudo.

- **Planejar e avaliar o EvolTrack-SocialNetwork:** seguindo as estratégias gerais traçadas no capítulo 4, planejar e realizar estudos experimentais para avaliar a viabilidade, a aplicabilidade e as possíveis contribuições trazidas pelo uso desta ferramenta de redes sociais em um projeto de desenvolvimento de software.

- **Participar da revisão sistemática sobre linha de processos:** o objetivo desta revisão sistemática é formalizar o conceito de linha de processo e identificar propostas de gestão e uso de linhas de processos. Esta revisão sistemática já foi planejada e será realizada em conjunto com outros alunos interessados no tema.

- **Definir um conjunto de características e componentes de processos para experimentos:** a partir dos modelos de desenvolvimento ágil e livre definir um conjunto experimental de características e componentes de processos para validar a solução de apoio à decisão proposta.

- **Participar da adequação do ferramental de apoio à abordagem de engenharia de linha de processos:** a ferramenta Odyssey (ODYSSEY, 2010) já atende parcialmente às necessidades da abordagem de engenharia de linha de processos, mas, como ela foi construída com o foco em linha de produtos, serão necessárias algumas adaptações para que possa satisfazer inteiramente estas necessidades. Esta adaptação está no escopo de uma dissertação de mestrado.

- **Participar da revisão sistemática sobre os critérios de adaptação de processos:** para ampliar o conjunto inicial de informações de contexto identificadas, esta revisão sistemática poderá ajudar a identificar os critérios comumente adotados na adaptação de processos de desenvolvimento de software. Esta revisão sistemática é objeto de uma dissertação de mestrado.

- **Participar da validação das informações de contexto:** as informações de contexto devem ser validadas através de uma pesquisa de opinião com especialistas (da academia e da indústria) com o objetivo de avaliar a pertinência e relevância das informações identificadas.

- **Construir o modelo de representação de contexto para processos de desenvolvimento de software:** modelar no ferramental de apoio selecionado, utilizando como base o modelo de representação escolhido, as dimensões, informações, situações e regras de contexto para o domínio de processos de desenvolvimento de software.

- **Instanciar o modelo de gestão de contexto:** o modelo de gestão de contexto proposto por (NUNES, 2007) deve ser instanciado para o domínio de processo de desenvolvimento de software através da especificação de mecanismos de captura, recuperação, apresentação e inferência das informações de contexto, que levem em consideração as informações de contexto e os cenários de balanceamento definidos.

- **Implementar a solução para o problema de balanceamento:** definir a técnica e implementar a solução para o problema de balanceamento utilizando algoritmos de otimização.

- **Planejar e avaliar a abordagem:** planejar e executar estudos experimentais que permitam avaliar a abordagem proposta. Esta avaliação deverá verificar se a proposta ajudou a reduzir o esforço necessário para a adaptação de processos e se ajudou a aumentar a adequação dos processos adaptados. Possivelmente, os resultados destes estudos vão indicar oportunidades de revisão da proposta.

  Existem diversas oportunidades para a realização de experimentos que visem a avaliação de viabilidade das hipóteses assumidas por esta proposta. Apesar de não ser foco deste trabalho apresentar um planejamento detalhado destes experimentos, é possível considerar algumas estratégias que ajudem a comprovar a viabilidade, eficácia e adequabilidade da proposta.

  De modo geral, a estratégia de validação adotada deverá seguir a metodologia para introduzir processos de software proposta por Shull *et al*. (2001) e evoluída por Mafra *et al*. (2006), que prevê uma realização incremental de estudos experimentais de viabilidade, de observação, estudos de caso em um ciclo de vida e, finalmente, estudos de caso na indústria. Esta validação deverá se limitar ao ambiente de apoio à decisão e suas interfaces com as abordagens de linha de processos e gestão de contexto. Assim, não está incluído no escopo avaliar de forma individual os mecanismos de gestão de contexto ou o ferramental de apoio usado para a linha de processos ou gestão de contexto. Mais detalhes sobre a avaliação do ambiente de apoio à decisão já foram fornecidos nas considerações finais do capítulo anterior.

- **Publicar os resultados:** divulgar e discutir os seus resultados alcançados com a comunidade acadêmica através da publicação de artigos nos seguintes eventos ou periódicos científicos de abrangência nacional e internacional:
    - Simpósio Brasileiro de Qualidade de Software (SBQS);
    - Simpósio Brasileiro de Engenharia de Software (SBES);
    - Simpósio Brasileiro de Sistemas Colaborativos (SBSC);
    - *International Conference on Software Engineering* (ICSE);
    - *International Conference on Software Process* (ICSP);
    - *International Conference on Product Focused Software and Process Improvement* (PROFES);
    - *European Systems and Software Process Improvement and Innovation* (EuroSPI);
    - *International Conference on Cooperative Information Systems (*CoopIS*);*
    - *International Conference on Computer Supported Cooperative Work in Design* (CSCWD);
    - *Collaboration Researchers' International Workshops on Groupware* (CRIWG).

- **Escrever a tese:** o texto da tese é elaborado utilizando como base o exame de qualificação e os artigos já produzidos.

- **Defender a tese** de doutorado perante uma banca de professores.

8.3. Resultados preliminares

A partir das atividades de pesquisa já realizadas, alguns resultados preliminares foram obtidos e são resumidos a seguir:

- Estabelecimento de uma versão inicial de corpo de conhecimento sobre processos de desenvolvimento de software, adaptação de processos, redes sociais, linha de processos, gestão de contexto e apoio à decisão;
- Realização de estudo secundário (revisão *quasi*-sistemática) para caracterizar a conciliação de modelos de desenvolvimento de software (seção 2.2.1) (MAGDALENO *et al*., 2009);
- Definição da visão geral da proposta de tese (seção 1.4), o que ajudou na delimitação do escopo das pesquisas de tese/dissertação de cada um dos membros do grupo de pesquisa (MAGDALENO, 2010a);
- Definição dos instrumentos para planejar e acompanhar a colaboração e disciplina (seção 3.5);

- Participação na definição das características de colaboração através de redes sociais (seção 4.6) (SANTOS *et al*., 2010, 2009);
- Análise de ferramentas de mineração, visualização e análise de redes sociais (MAGDALENO *et al*., 2010);
- Definição dos requisitos necessários para uma ferramenta de redes sociais (seção 4.7) (MAGDALENO *et al*., 2010);
- Análise de viabilidade do uso do EvolTrack em projetos de desenvolvimento de software livre (seção 4.8.1) (CEPEDA *et al*., 2010);
- Proposta de uma arquitetura inicial para a ferramenta de redes sociais EvolTrack-SocialNetwork (seção 4.8.2);
- Participação na definição de uma abordagem sistemática para engenharia de linha de processos (seção 5.4) (NUNES *et al*., 2010b);
- Investigação do modelo de representação de contexto para o domínio de processos de desenvolvimento de software (seção 6.4) (NUNES *et al*., 2010a);
- Identificação de um conjunto inicial de informações de contexto relevantes para o domínio de processos de desenvolvimento de software (seção 6.6);
- Detalhamento inicial da abordagem para relacionar linha de processos e gestão de contexto (seção 6.9);
- Investigação de alternativas para a solução do problema de balanceamento;
- Definição e modelagem do problema de balanceamento (seção 7.3) (MAGDALENO, 2010b);
- Elaboração do texto para o Exame de Qualificação.

8.4. Resultados esperados

Após concluir as atividades de pesquisa previstas para depois do exame de qualificação, são esperados os resultados listados a seguir:

- Atualização da revisão *quasi*-sistemática da literatura sobre conciliação de modelos de desenvolvimento de software;
- Detalhamento do uso dos instrumentos escolhidos para planejar e acompanhar a colaboração e disciplina;
- Participação na formalização da definição de linha de processos e identificação de abordagens alternativas através da realização de estudo secundário (revisão *quasi*-sistemática);
- Participação na adequação do ferramental de apoio à abordagem de engenharia de linha de processos como parte de uma dissertação de mestrado;
- Definir um conjunto de características e componentes de processos para

testes e experimentos do ambiente de apoio à decisão;

- Participação na identificação dos critérios de adaptação de processos de desenvolvimento de software através da realização de estudo secundário (revisão *quasi*-sistemática);
- Participação na avaliação com os especialistas (pesquisa de opinião) sobre as informações de contexto que influenciam a adaptação de processos, quanto à pertinência e ao nível de relevância;
- Construção do modelo de representação de contexto para processos de desenvolvimento de software com as dimensões, informações, situações e regras de contexto;
- Instanciação dos mecanismos da abordagem de gestão de contexto de acordo com as necessidades deste trabalho de pesquisa;
- Integração das informações de colaboração e disciplina à abordagem de linha de processos gerida por contexto;
- Definição da técnica de otimização que será adotada na solução do problema de balanceamento;
- Implementação da técnica de otimização para solução do problema de balanceamento;
- Planejamento e execução de avaliações de partes da solução, possivelmente resultando em revisões da proposta;
- Publicação de artigos conforme o andamento das atividades de pesquisa.

8.5. Cronograma proposto

Para o desenvolvimento das atividades descritas na metodologia deste trabalho de pesquisa, é prevista a seguinte distribuição de atividades ao longo dos anos de pesquisa restantes (Tabela 8.1):

	2010						2011												2012		
Atividades	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**
Refinar proposta de solução	X																				
Detalhar instrumentos	X	X*																			
Reexecutar revisão sobre conciliação			X	X																	
Planejar e avaliar EvolTrack-SocialNetwork			X	X	X*	X	X	X	X	X*											
Participar de revisão sistemática sobre linha				X	X	X*															

Atividades	2010						2011												2012		
	Jul	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar
de processos																					
Definir características e componentes de processos para experimentos		X	X	X																	
Participar da adequação do ferramental de apoio a engenharia de linha de processos	X	X	X	X	X*	X	X	X	X*												
Participar de revisão sistemática sobre critérios de adaptação			X	X	X	X	X	X*													
Participar da validação das informações de contexto							X	X	X	X	X	X*									
Construir o modelo de representação de contexto para processos de software											X	X	X	X*							
Detalhar abordagem para relacionar linha de processos e gestão de contexto	X	X	X*																		
Instanciar modelo de gestão de contexto										X	X	X	X	X*							
Implementar solução de balanceamento		X	X	X	X	X	X	X*	X	X	X	X	X	X	X*						
Planejar e avaliar abordagem				X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X*		
Escrever tese																	X	X	X	X	X
Defender tese																					X

** Indica oportunidades de publicação dos resultados*

Tabela 8.1 – Cronograma de atividades

Referências Bibliográficas

AALST, W.; REIJERS, H. A.; SONG, M., 2005, "Discovering Social Networks from Event Logs", *Computer Supported Cooperative Work (CSCW)*, v. 14, n. 6 (Jan.), pp. 549-593.

AALST, W.; GIINTHER, C., 2007, "Finding Structure in Unstructured Processes: The Case for Process Mining". *Application of Concurrency to System Design (ACSD)*, pp. 3-12.

AALST, W. V. D.; WEIJTERS, T.; MARUSTER, L., 2004, "Workflow Mining: Discovering Process Models from Event Logs", *IEEE Transaction on Knowledge and Data Engineering*, v. 16, n. 9, pp. 1128-1142.

ABRAHAMSSON, P.; WARSTA, J.; SIPONEN, M.; *et al*., 2003, "New directions on agile methods: a comparative analysis". In: *Proceedings of the 25th International Conference on Software Engineering*, pp. 244-254, Portland, Oregon, USA.

ABRANTES, J. F.; TRAVASSOS, G. H., 2007, *Revisão quasi-Sistemática da Literatura: Caracterização de Métodos Ágeis de Desenvolvimento de Software*, Relatório Técnico ES-714/07, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

ALBERT, R.; BARABASI, A. L., 2002, "Statistical mechanisms of complex networks", *Reviews of Modern Physics*, v. 74, n. 1, pp. 47-97.

ALBERT, R.; JEONG, H.; BARABÁSI, A. L., 1999, "Diameter of the World-Wide Web", *Nature*, v. 401, pp. 130-131.

ALEIXO, A. I. D. S., 2003, *Procedimentos para Implantar Equipes*. Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil. Disponível em: http://teses.eps.ufsc.br/defesa/pdf/8183.pdf.

ALEXANDER, L.; DAVIS, A., 1991, "Criteria for selecting software process models". *International Computer Software and Applications Conference (COMPSAC)*, pp. 521-528, Tokyo, Japan.

ANDERSON, D., 2005, "Stretching agile to fit CMMI level 3 - the story of creating MSF for CMMI® process improvement at Microsoft corporation". *Agile Conference*, pp. 193-201, Denver, USA.

ARANGO, G.; PRIETO-DIAZ, R., 1991, "Domain analysis: Concepts and research directions", *Domain Analysis: Acquisition of Reusable Information for Software Construction*, IEEE Computer Society Press, pp. 9-32.

ARAUJO, R., 2000, *Ampliando a Cultura de Processos de Software: Um*

Enfoque Baseado em Groupware e Workflow. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

ARAUJO, R.; BORGES, M. R. S., 2001, "Sistemas de Workflow". In: *Jornadas de Atualização em Informática (JAI) – Congresso da Sociedade Brasileira de Computação (SBC)*, pp. 1-17, Fortaleza, Ceará, Brasil.

ARAUJO, R. M. D.; SANTORO, F. M.; BRÉZILLON, P.; *et al*., 2004, "Context Models for Managing Collaborative Software Development Knowledge". In: *International Workshop on Modeling and Retrieval of Context (MRC)*, pp. 61-72, Ulm.

ARAUJO, R. M. D.; BORGES, M. R. S., 2007, "The role of collaborative support to promote participation and commitment in software development teams", *Software Process: Improvement and Practice*, v. 12, n. 3, pp. 229-246.

ARIMOTO, M. M.; MURAKAMI, E.; CAMARGO, V. V.; *et al*., 2009, "Adherence Analysis of Agile Methods According to MR-MPS Reference Model". *Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 249-263, Ouro Preto, MG, Brasil.

ARMBRUST, O.; KATAHIRA, M.; MIYAMOTO, Y.; *et al*., 2008, "Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards", *Making Globally Distributed Software Development a Success Story*, Berlin / Heidelberg: Springer, pp. 160-172.

ATKINSON, C.; BAYER, J.; BUNSE, C.; *et al*., 2001, *Component-Based Product Line Engineering with UML*. 1 ed. Addison-Wesley Professional.

BALIEIRO, M. A.; SOUZA JR, S. D.; PEREIRA, L. P.; *et al*., 2007, "OSSNetwork: Um Ambiente para Estudo de Comunidades de Software Livre usando Redes Sociais". In: *Experimental Software Engineering Latin America Workshop*, pp. 33-42, São Paulo, SP, Brasil.

BARABASI, A. L., 2003, *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Cambridge, Plume.

BARABASI, A. L.; BONABEAU, E., 2003, "Scale-Free Networks", *Scientific American*, pp. 50-59.

BARNETT, L., 2004, "Applying Open Source Processes In Corporate Development Organizations", *Forrester Research*, pp. 1-15.

BARRETO, A., 2007, *Uma Abordagem para Definição de Processos de Software Baseada em Reutilização*. Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BARRETO, A.; MURTA, L. G. P.; ROCHA, A. R., 2009, "Componentizando Processos Legados de Software Visando a Reutilização de Processos". *Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 189-203, Ouro Preto, Minas Gerais, Brasil.

BARRETO, A. S., 2005, *Apoio à Alocação de Recursos Humanos em Projetos de Software: Uma Abordagem Baseada em Satisfação de Restrições*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BARRETO, A. O. S.; ROCHA, A. R. C. D., 2009, "Definição e Monitoração de Objetivos de Software Alinhados ao Planejamento Estratégico". *Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 114-128, Ouro Preto, MG, Brasil.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D., 1994, "The goal question metric approach", *Encyclopedia of software engineering*, v. 1, pp. 528-532.

BAZIRE, M.; BREZILLON, P., 2005, "Understanding Context Before Using It", *Modeling and Using Context*, pp. 29-40.

BECK, K., 1999, *Extreme Programming Explained: Embrace Change*. Boston, MA, USA, Addison-Wesley.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; *et al*., 2001. Manifesto for Agile Software Development. Disponível em: http://agilemanifesto.org/. Acesso em: 15 Dez 2008.

BECKER, M., 2003, "Towards a general model of variability in product families". In: *Proceedings of the First Workshop on Software Variability Management*, pp. 19-27, Groningen, The Netherlands.

BEKKERS, W.; VAN DE WEERD, I.; BRINKKEMPER, S.; *et al*., 2008, "The Influence of Situational Factors in Software Product Management: An Empirical Study". *International Workshop on Software Product Management (IWSPM)*, pp. 41-48, Barcelona, Catalunya, Spain.

BERGER, P. M., 2003, *Instanciação de Processos de Software em Ambientes Configurados na Estação TABA*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BERTOLLO, G.; SEGRINI, B.; FALBO, R. D. A., 2006, "Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias". *Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 72-86, Vila Velha, ES, Brasil.

BIOLCHINI, J.; MIAN, P. G.; NATALI, A. C. C.; *et al*., 2005, *Systematic Review in Software Engineering*, Relatório Técnico ES-679, PESC-UFRJ. Disponível em: http://www.cos.ufrj.br.

BLOIS, A. P. T. B., 2006, *Uma Abordagem de Projeto Arquitetural Baseado em Componentes no Contexto de Engenharia de Domínio*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BOCKLE, G.; POHL, K.; LINDEN, F., 2005, "A Framework for Software Product Line Engineering", *Software Product Line Engineering*, Berlin / Heidelberg: Springer, pp. 19-38.

BOEHM, B., 2002, "Get ready for agile methods, with care", *IEEE Computer*, v. 35, n. 1, pp. 64-69.

BOEHM, B.; TURNER, R., 2003, *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA, USA, Addison-Wesley.

BOOCH, G., 1993, *Object-Oriented Analysis and Design with Applications*. 2 ed. Boston, MA, USA, Addison-Wesley.

BOSCH, J., 2000, *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Boston, MA, USA, Addison-Wesley.

BOSCH, J., 2004, "Software Variability Management". In: *International Conference on Software Engineering (ICSE)*, pp. 720-721, Scotland, UK.

BRAGA, R. M. M., 2000, *Busca e Recuperação de Componentes em Ambientes de Reutilização de Software*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BRANDES, U.; WAGNER, D., 2003, "Visone - Analysis and visualization of social networks", In: Junger, M., Mutzel, P. (eds), *Graph Drawing Software*, Heidelberg: Springer-Verlag, pp. 321-340.

BREZILLON, P.; POMEROL, J., 1999, "Contextual Knowledge Sharing and Cooperation in Intelligent Assistant Systems", *Le Travail Humain*, v. 62, pp. 223-246.

BREZILLON, P., 1999, "Context in problem solving: a survey", *Knowledge Engineering Review*, v. 14, n. 1, pp. 47-80.

BRIAND, L. C.; DIFFERDING, C. M.; ROMBACH, H. D., 1996, "Practical guidelines for measurement-based process improvement", *Software Process: Improvement and Practice*, v. 2, n. 4, pp. 253-280.

BRINKKEMPER, S., 1996, "Method engineering: engineering of information systems development methods and tools", *Information and Software Technology*, v. 38, n. 4, pp. 275-280.

BURNS, R. N.; DENNIS, A. R., 1985, "Selecting the appropriate application development methodology", *SIGMIS Database*, v. 17, n. 1, pp. 19-23.

CAMERON, J., 2002, "Configurable development processes", *Communications*

of ACM, v. 45, n. 3, pp. 72-77.

CAPILUPPI, A.; LAGO, P.; MORISIO, M., 2003, "Characteristics of Open Source Projects". In: *Proceedings of the 17th European Conference on Software Maintenance and Reengineering*, pp. 1-11, Washington, USA.

CATALDO, M.; HERBSLEB, J. D., 2008, "Communication networks in geographically distributed software development". In: *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pp. 579-588, San Diego, CA, USA.

CEPEDA, R. D. S. V.; MAGDALENO, A. M.; MURTA, L. G. P.; *et al*., 2010, "EvolTrack: Improving Design Evolution Awareness in Software Development", *Journal of the Brazilian Computer Society (JBCS)*(to appear).

CEPEDA, R. D. S. V.; MURTA, L. G. P.; WERNER, C., 2008, "Visualizando a Evolução de Software no Desenvolvimento Distribuído". *Workshop de Desenvolvimento Distribuído de Software (WDDS) - Simpósio Brasileiro de Engenharia de Software (SBES)*, pp. 41-50, Campinas, SP, Brasil.

CHAU, T.; MAURER, F.; MELNIK, G., 2003, "Knowledge sharing: agile methods vs. Tayloristic methods". *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 302-307, Linz, Austria.

CHEN, G.; KOTZ, D., 2000, *A Survey of Context-Aware Mobile Computing Research*, Technical Report TR2000-381, Dartmouth College.

CHEN, L.; BABAR, M. A.; ALI, N., 2009, "Variability management in software product lines: A systematic review". *International Software Product Lines Conference (SPLC)*, pp. 81-90, San Francisco, CA, USA.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S., 2006, *CMMI: Guidelines for Process Integration and Product Improvement*. 2 ed. Boston, MA, USA, Addison-Wesley.

CLARKE, J.; DOLADO, J. J.; HARMAN, M.; *et al*., 2003, "Reformulating software engineering as a search problem", *IEEE Proceedings-Software*, v. 150, n. 3, pp. 161–175.

CLEMEN, R. T.; REILLY, T., 2000, *Making Hard Decisions with DecisionTools Suite*. 1 ed. South-Western College Pub.

COCKBURN, A.; HIGHSMITH, J., 2001, "Agile Software Development: The People Factor", *IEEE Computer*, v. 34, n. 11, pp. 131-133.

COCKBURN, A., 2000, "Selecting a Project's Methodology", *IEEE Software*, v. 17, n. 4, pp. 64-71.

COCKBURN, A., 2001, *Agile Software Development*. Boston, MA, USA, Addison-Wesley.

COCKBURN, A., 2004, *Crystal Clear: A Human-Powered Methodology for Small Teams*. 1 ed. Boston, MA, USA, Addison-Wesley.

COHAN, S.; GLAZER, H., 2009, "An Agile Development Team's Quest for CMMI Maturity Level 5". In: *2009 Agile Conference2009 Agile Conference (AGILE)*, pp. 201-206, Chicago, USA.

CROSS, R.; ANDREW, P.; CROSS, R., 2004, *The Hidden Power of Social Networks: Understanding How Work Really Gets Done in Organizations*. Boston, Massachussetts, Harvard Business School Press.

CUBRANIC, D.; BOOTH, K. S., 1999, "Coordinating open-source software development". In: *Proceedings of the 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 61-66, Stanford, USA.

CUGOLA, G.; GHEZZI, C., 1998, "Software processes: A retrospective and a path to the future", *Software Process Improvement and Practice (SPIP) Journal*, v. 4, pp. 101-123.

CZARNECKI, K.; EISENECKER, U., 2000, *Generative Programming: Methods, Tools, and Applications*. Boston, MA, USA, Addison-Wesley.

DAVIS, G. B., 1982, "Strategies for information requirements determination", *Enterprise Analysis - IBM Systems Journal*, v. 21, n. 1, p. 4.

DEGENNE, A.; FORSE, M., 1999, *Introducing Social Networks*. 1 ed. Sage Publications.

DEMARCO, T., 1986, *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice Hall.

DEMARCO, T.; LISTER, T., 1999, *Peopleware: Productive Projects and Teams*. 2 ed. New York, USA, Dorset House.

DESHPANDE, A.; RIEHLE, D., 2008, "Continuous Integration in Open Source Software Development", *Open Source Development, Communities and Quality*, 1 ed, chapter 275, Boston, MA, USA: Springer, pp. 273-280.

DEY, A.; ABOWD, G.; SALBER, D., 2001, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", *Human-Computer Interaction*, v. 16, n. 2-4, pp. 97-166.

DIAS, M. D. S., 1998, *COPSE: Um Ambiente de Suporte ao Projeto Cooperativo de Software*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

DIAS, T. M. R.; OLIVEIRA, J. L. L., 2009, "Adoção da Metodologia Extreme Programming para Construção de Software". *Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS)*, pp. 10-23, Natal, RN, Brasil.

DOURISH, P.; BELLOTTI, V., 1992, "Awareness and coordination in shared workspaces". In: *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pp. 107-114, Toronto, Ontario, Canada.

EBERT, C., 2007, "Open Source Drives Innovation", *IEEE Software*, v. 24, n. 3, pp. 105-109.

ELLIS, C. A.; GIBBS, S. J.; REIN, G., 1991, "Groupware: some issues and experiences", *Communications of ACM*, v. 34, n. 1, pp. 39-58.

ERDOS, P.; RENYI, A., 1959, "On random graphs", *Publicationes Mathematicae (Debrecen)*, v. 6, pp. 290-297.

ESTOLANO, M. H. D. R., 2005, *Base de Métricas para a Estação TABA*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FALBO, R. A., 1998, *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FALBO, R. A.; BERTOLLO, G., 2005, "Establishing a Common Vocabulary for Software Organizations Understand Software Processes". *EDOC International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE)*, pp. 1-8, Enschede, The Netherlands.

FAYAD, M. E.; JOHNSON, R. E.; SCHMIDT, D. C., 1999, *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. 1 ed. John Wiley & Sons.

FEI DAI; TONG LI, 2007, "Tailoring Software Evolution Process". *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, pp. 782-787, Nagoya, Japan.

FELLER, J.; FITZGERALD, B., 2001, *Understanding Open Source Software Development*. Boston, MA, USA, Addison-Wesley.

FERNANDES, P. C. C., 2008, *UbiFEX: Uma Abordagem para Modelagem de Características de Linha de Produtos de Software Sensíveis ao Contexto*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FERREIRA, A. B. D. H., 2009, *Novo Dicionário Aurélio da Língua Portuguesa*. 3 ed. Curitiba, Editora Positivo.

FIORINI, S., 2001, *Arquitetura para Reutilização de Processos de Software*. Tese de Doutorado, PUC-RJ, Rio de Janeiro, RJ, Brasil. Disponível em: http://www-di.inf.puc-rio.br/~julio/tese-soeli.pdf.

FITZGERALD, B.; RUSSO, N.; OKANE, T., 2000, "An empirical study of system development method tailoring in practice". *European Conference on Information Systems (ECIS)*, pp. 187–194, Vienna, Austria.

FITZGERALD, B.; RUSSO, N. L.; O'KANE, T., 2003, "Software development method tailoring at Motorola", *Communications of ACM*, v. 46, n. 4, pp. 64-70.

FRAKES, W.; KYO KANG, 2005, "Software reuse research: status and future", *Software Engineering, IEEE Transactions on*, v. 31, n. 7, pp. 529-536.

FREITAS, C. M. D. S.; NEDEL, L. P.; GALANTE, R.; *et al*., 2008, "Extração de Conhecimento e Análise Visual de Redes Sociais". *Seminário Integrado de Hardware e Software (SEMISH) - Congresso da Sociedade Brasileira de Computação (SBC)*, pp. 106-120, Belém, PA, Brasil.

FRITZSCHE, M.; KEIL, P., 2007, "Agile Methods and CMMI: Compatibility or Conflict?", *e-Informatica Software Engineering Journal*, v. 1, n. 1, pp. 9-26.

FSF, 2008. The Free Software Definition. Disponível em: http://www.gnu.org/philosophy/free-sw.html. Acesso em: 31 Maio 2008.

FUGGETTA, A., 2000, "Software process: a roadmap". In: *Proceedings of the Conference on The Future of Software Engineering*, pp. 25-34, Limerick, Ireland.

FUKS, H.; GEROSA, M. A.; PIMENTEL, M. G., 2003a, "Projeto de Comunicação em Groupware: Desenvolvimento, Interface e Utilização". *Jornada de Atualização em Informática*, pp. 295-338, Campinas, SP, Brasil.

FUKS, H.; RAPOSO, A. B.; GEROSA, M. A., 2003b, "Do Modelo de Colaboração 3C à Engenharia de Groupware". *Simpósio Brasileiro de Sistemas Multimídia e Web (WEBMIDIA) – Trilha Trabalho Cooperativo Assistido por Computador (CSCW)*, pp. 1-8, Salvador, BA, Brasil.

FUSARO, P.; VISAGGIO, G.; TORTORELLA, M., 1998, "REP - ChaRacterizing and Exploiting Process Components: Results of Experimentation". In: *Proceedings of the Working Conference on Reverse Engineering (WCRE)*, pp. 20-29, Honolulu, Hawaii, USA.

GACEK, C.; ARIEF, B., 2004, "The Many Meanings of Open Source", *IEEE Software*, v. 21, n. 1, pp. 34-40.

GAMMA, E.; HELM, R.; JOHNSON, R.; *et al*., 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA, USA, Addison-Wesley.

GAO, Y.; FREEH, V.; MADEY, G., 2003, "Analysis and Modeling of Open Source Software Community". *North American Association for Computational Social and Organization Sciences Conference (NAACSOS)*, pp. 1-4, Pittsburgh, PA, USA.

GARG, A.; CRITCHLOW, M.; PING CHEN; *et al*., 2003, "An environment for managing evolving product line architectures". *International Conference on Software Maintenance (ICSM)*, pp. 358-367, Amsterdam, The Netherlands.

GARY, K.; LINDQUIST, T., 1999, "Cooperating process components". In: *Proceedings of the Twenty-Third Annual International Computer Software and Applications Conference (COMPSAC)*, pp. 218-223, Phoenix, AZ, USA.

GEROSA, M. A.; FUKS, H.; LUCENA, C. J. P., 2003, "Suporte à Percepção em Ambientes de Aprendizagem Colaborativa", *Revista Brasileira de Informática na Educação*, v. 11, n. 2, pp. 75-85.

GIBSON, D.; GOLDENSON, D.; KOST, K., 2006, *Performance Results of CMMI-Based Process Improvement*, Technical Report CMU/SEI-2006-TR-004, SEI-CMU. Disponível em: http://www.sei.cmu.edu/library/abstracts/reports/06tr004.cfm.

GINSBERG, M.; QUINN, L., 1995, *Process Tailoring and the Software Capability Maturity Model* CMU/SEI-94-TR-024, SEI-CMU. Disponível em: http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.024.html

GLASS, R. L., 2001, "Agile Versus Traditional: Make Love, Not War!", *Cutter IT Journal*, v. 14, n. 12, pp. 12-18.

GLAZER, H.; DALTON, J.; ANDERSON, D.; *et al*., 2008, *CMMI or Agile: Why Not Embrace Both!*, SEI-CMU. Disponível em: http://www.sei.cmu.edu/publications/documents/08.reports/08tn003.html.

GOLDENSON, D.; GIBSON, D., 2003, *Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results*, Technical Report CMU/SEI-2003-SR-009, CMU-SEI. Disponível em: http://www.sei.cmu.edu/library/abstracts/reports/03sr009.cfm.

GOTO, C. S.; ROSA, M. P.; DE SOUZA, C. D., 2008, "Um Estudo Exploratório sobre os Efeitos da Refatoração na Coordenação das Atividades de Desenvolvimento de Software Livre". *Workshop de Manutenção de Software Moderna (WMSWM)*, pp. 1-8, Florianópolis, SC, Brasil.

GRISS, M. L., 1999, "Architecting for large-scale systematic component reuse". In: *Proceedings of the 21st international conference on Software engineering*, pp. 615-616, Los Angeles, CA, USA.

GRUBER, T. R., 1995, "Toward principles for the design of ontologies used for knowledge sharing", *Int. J. Hum.-Comput. Stud.*, v. 43, n. 5-6, pp. 907-928.

GUTWIN, C.; GREENBERG, S., 1999, *A framework of awareness for small groups in shared-workspace groupware*, Technical Report 99-1, Department of Computer Science, University of Saskatchewan.

HAEFLIGER, S.; VON KROGH, G.; SPAETH, S., 2007, "Code Reuse in Open Source Software", *Management Science*, v. 54, n. 1, pp. 180-193.

HANSEN, M. T., 2009, "When Internal Collaboration Is Bad for Your Company", *Harvard Business Review*, v. 84, n. 3, pp. 83-88.

HANSSON, C.; DITTRICH, Y.; GUSTAFSSON, B.; *et al*., 2006, "How agile are industrial software development practices?", *Journal of Systems and Software (JSS)*, v. 79, n. 9, pp. 1295-1311.

HARDINGHAM, A., 2000, *Trabalho em Equipe*. São Paulo, SP, Brasil, Nobel.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y., 2009, *Search based software engineering: A comprehensive analysis and review of trends techniques and applications*, Technical Report

HARMAN, M., 2007, "The Current State and Future of Search Based Software Engineering". In: *2007 Future of Software Engineering*, pp. 342-357.

HARMAN, M.; JONES, B. F., 2001, "Search-Based Software Engineering", *Information and Software Technology*, v. 43, pp. 833-839.

HARMSEN, F.; BRINKKEMPER, S.; OEI, J. L. H., 1994, "Situational method engineering for informational system project approaches". In: *Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle*, pp. 169-194.

HARTMANN, D.; DYMOND, R., 2006, "Appropriate agile measurements: Using metrics and diagnostics to deliver business value". In: *Agile Conference*, pp. 126-131, Minneapolis, MI, USA.

HEALY, K.; SCHUSSMAN, A., 2003, "The Ecology of Open Source Software". In: *Annual Meeting of the American Sociological Association*, pp. 1-20, Atlanta, GA, USA.

HERBSLEB, J. D.; PAULISH, D. J.; BASS, M., 2005, "Global software development at siemens: experience from nine projects". In: *Proceedings of the 27th international conference on Software engineering*, pp. 524-

533, St. Louis, MO, USA.

HIGHSMITH, J.; COCKBURN, A., 2001, "Agile Software Development: The Business of Innovation", *IEEE Computer*, v. 34, n. 9, pp. 120-127.

HOFSTEDE, G.; HOFSTEDE, G. J.; MINKOV, M., 2010, *Cultures and Organizations: Software for the Mind, Third Edition*. 3 ed. McGraw-Hill.

HOLLENBACH, C.; FRAKES, W., 1996, "Software Process Reuse in an Industrial Setting". In: *Proceedings of the 4th International Conference on Software Reuse*, pp. 22-30.

HUBAUX, A.; CLASSEN, A.; MENDONÇA, M.; *et al*., 2010, "A Preliminary Review on the Application of Feature Diagrams in Practice". In: *International Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*, pp. 53-59, Linz, Austria.

HUMPHREY, W. S., 1989, *Managing the Software Process*. Boston, MA, USA, Addison-Wesley.

IBM, 2009. Rational Unified Process (RUP). Disponível em: http://www-01.ibm.com/software/awdtools/rup/. Acesso em: 12 Abr 2010.

ISO/IEC, 1995, *ISO/IEC 12207: Information technology - Software life cycle processes*. Geneve.

ISO/IEC, 2007, *ISO/IEC 15939: Software Engineering – Software Measurement Process*. Geneve.

JAUFMAN, O.; MUNCH, J., 2005, "Acquisition of a Project-Specific Process", *Product Focused Software Process Improvement*, Berlin / Heidelberg: Springer Verlag, pp. 328-342.

JENSEN, C.; SCACCHI, W., 2007, "Guiding the Discovery of Open Source Software Processes with a Reference Model", *Open Source Development, Adoption and Innovation*, chapter 234, Boston:, pp. 265-270.

JIANG, L.; EBERLEIN, A., 2008, "Towards a framework for understanding the relationships between classical software engineering and agile methodologies". In: *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral*, pp. 9-14, Leipzig, Germany.

JIN XU; YONGQIN GAO; CHRISTLEY, S.; *et al*., 2005, "A Topological Analysis of the Open Source Software Development Community". In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS)*, pp. 198-208, Waikoloa, HI, USA.

KAHKONEN, T.; ABRAHAMSSON, P., 2004, "Achieving CMMI Level 2 with

Enhanced Extreme Programming Approach", *Product Focused Software Process Improvement*, Heidelberg: Springer-Verlag, pp. 378-392.

KALINOWSKI, M., 2009, *DBPI: Abordagem de Prevenção de Defeitos de Software para Apoiar Melhoria de Processos e Aprendizado Organizacional*. Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

KANG, K.; COHEN, S.; HESS, J.; *et al*., 1990, *Feature-Oriented Domain Analysis* CMU/SEI-90-TR-21, CMU-SEI. Disponível em: http://www.sei.cmu.edu/domain-engineering/FODA.html.

KANG, K.; KIM, S.; LEE, J.; *et al*., 1998, "FORM: A feature oriented reuse method with domain-;specific reference architectures", *Annals of Software Engineering*, v. 5, n. 1 (Jan.), pp. 143-168.

KARLSTRÖM, D.; RUNESON, P., 2002, "Decision support for extreme programming introduction and practice selection". In: *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pp. 835-841, Ischia, Italy.

KELLNER, M., 1996, "Connecting reusable software process elements and components". *International Software Process Workshop (ISPW)*, pp. 8-11, Dijon, France.

KITCHENHAM, B., 2004, *Procedures for Performing Systematic Reviews*, Technical Report TR/SE-0401, Software Engineering Group, Department of Computer Science, Keele University. Disponível em: http://www.idi.ntnu.no/emner/empse/papers/kitchenham_2004.pdf.

KOCH, S., 2004, "Agile Principles and Open Source Software Development: A Theoretical and Empirical Discussion", *Extreme Programming and Agile Processes in Software Engineering*, 1 ed, chapter 3092, Heidelberg: Springer-Verlag, pp. 85-93.

KOKINOV, B., 1999, "Dynamics and Automaticity of Context: A Cognitive Modeling Approach", *Modeling and Using Context*, p. 830.

KRISHNAMURTHY, S., 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday*. Disponível em: http://firstmonday.org/issues/issue7_6/krishnamurthy/. Acesso em: 25 Maio 2008.

KUMAR, K.; WELKE, R. J., 1992, "Methodology Engineering: a proposal for situation-specific methodology construction", *Challenges and strategies for research in systems development*, John Wiley & Sons, pp. 257-269.

LAANTI, M.; KETTUNEN, P., 2005, "How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models", *Information and Software Technology*, v. 47, n. 9, pp. 587-608.

LEE, K.; KANG, K. C.; LEE, J., 2002, "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering". In: *Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools*, pp. 62-77

LINDEN, F. J.; SCHMID, K.; ROMMES, E., 2007, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. 1 ed. Berlin / Heidelberg, Springer.

LINDVALL, M.; BASILI, V.; BOEHM, B.; *et al*., 2002, "Empirical Findings in Agile Methods", *Extreme Programming and Agile Methods (XP/Agile Universe)*, Berlin / Heidelberg: Springer-Verlag, pp. 81-92.

LITTLE, T., 2005, "Context-adaptive agility: managing complexity and uncertainty", *Software, IEEE*, v. 22, n. 3, pp. 28-35.

LOPEZ-FERNANDEZ, L.; ROBLES, G.; GONZALEZ-BARAHONA, J. M.; *et al*., 2004, "Applying Social Network Analysis to the Information in CVS Repositories". *Mining Software Repositories Workshop - International Conference on Software Engineering (ICSE)*, pp. 101-105, Edinburgh, Scotland.

MACHADO, L. F. D. C., 2000, *Modelo para Definição de Processos de Software na Estação TABA*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MADEY, G.; FREEH, V.; TYNAN, R., 2002, "The open source software development phenomenon: An analysis based on social network theory". *Americas Conference on Information Systems (AMCIS)*, pp. 1806-1813, Dallas, TX, USA.

MAFRA, S.; BARCELOS, R.; TRAVASSOS, G. H., 2006, "Aplicando uma Metodologia Baseada em Evidência na Definição de Novas Tecnologias de Software". In: *Proceedings of the 20th Brazilian Symposium on Software Engineering (SBES)*, pp. 239-254, Florianópolis, SC, Brasil.

MAGDALENO, A.; NUNES, V.; ARAUJO, R.; *et al*., 2007, "Flexible Organizational Process Deployment", *Computer Supported Cooperative Work in Design III*, chapter 4402, Berlin / Heidelberg: Springer, pp. 679-688.

MAGDALENO, A. M., 2006, *Explicitando a Colaboração em Organizações através da Modelagem de Processos de Negócios*. Dissertação de Mestrado, UFRJ/IM/NCE, Rio de Janeiro, RJ, Brasil.

MAGDALENO, A. M., 2010a, "Balancing Collaboration and Discipline in Software Development Processes". In: *Doctoral Symposium of International Conference on Software Engineering (ICSE)*, pp. 331-332, Cape Town, South Africa.

MAGDALENO, A. M., 2010b, "An optimization-based approach to software development process tailoring". In: *PhD Track - International Symposium on Search Based Software Engineering (SSBSE)*, Benevento, Italy (to appear).

MAGDALENO, A. M.; ARAUJO, R. M. D.; BORGES, M. R. S., 2007, "Designing Collaborative Processes". *Workshop on Business Process Modeling, Development, and Support (BPMDS)*, pp. 283-290, Trondheim, Norway.

MAGDALENO, A. M.; ARAUJO, R. M. D.; BORGES, M. R. S., 2009, "A Maturity Model to Promote Collaboration in Business Processes", *International Journal of Business Process Integration and Management (IJBPIM)*, v. 4, n. 2, pp. 111-123.

MAGDALENO, A. M.; CAPPELLI, C.; BAIAO, F.; *et al*., 2008a, "A Practical Experience in Designing Business Processes to Improve Collaboration". *Business Process Design (BPD)*, pp. 156-168, Brisbane, Queensland, Australia.

MAGDALENO, A. M.; CAPPELLI, C.; BAIAO, F. A.; *et al*., 2008b, "Towards Collaboration Maturity in Business Processes: An Exploratory Study in Oil Production Processes", *Information Systems Management (ISM)*, v. 25, n. 4, pp. 302-318.

MAGDALENO, A. M.; WERNER, C. M. L., 2008, *Introdução aos processos de colaboração e reutilização em software livre*, Relatório Técnico 01/2008, FAPERJ/COPPE. Disponível em: http://www.uniriotec.br/padct/.

MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M. D., 2010, *Estudo de Ferramentas de Mineração, Visualização e Análise de Redes Sociais*, Relatório Técnico ES-735, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

MAGDALENO, A. M.; WERNER, C. M. L.; ARAUJO, R. M. D., 2009, *Revisão Quasi-Sistemática da Literatura: Conciliação de processos de desenvolvimento de software*, Relatório Técnico ES-730, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

MASSEN, T.; LICHTER, H., 2004, "Deficiencies in Feature Models". In: *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, pp. 59-72, Boston, MA, USA.

MCCARTHY, J., 1993, "Notes On Formalizing Context", pp. 555-560.

MCGARRY, J.; CARD, D.; JONES, C.; ET AL., 2001, *Practical Software Measurement: Objective Information for Decision Makers*. Addison-Wesley Professional.

MERRIAM-WEBSTER, 2004, *The Merriam-Webster English Dictionary*.

Merriam-Webster.

MILER, N., 2000, *A Engenharia de Aplicações no Contexto da Reutilização Baseada em Modelos de Domínio*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, Brasil.

MILLER, G. G., 2001, "The Characteristics of Agile Software Processes". In: *Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS)*, pp. 1-3, Santa Barbara, CA, USA.

MINICUCCI, A., 2001, *Técnicas de Trabalho de Grupo*. 3 ed. São Paulo, SP, Brasil, Atlas.

MNKANDLA, E., 2008, *A framework for agile methodology practices: a family of methodologies approach*. Doctoral thesis, Faculty of Engineering and the Built Environment, University of The Witwatersrand, Johannesburg. Disponível em: http://hdl.handle.net/10539/5666.

MONTERO, I.; PENA, J.; RUIZ-CORTÉS, A., 2007, "Business Family Engineering: Does it make sense?". In: *I JISBD Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS)*, pp. 34–40, Zaragoza, España.

MURTA, L.; CORRÊA, C.; PRUDÊNCIO, J. G.; *et al*., 2008, "Towards odyssey-VCS 2: improvements over a UML-based version control system". *International workshop on Comparison and versioning of software models*, pp. 25-30, Leipzig, Germany.

MURTA, L. G. P., 2002, *Charon: Uma Máquina de Processos Extensível Baseado em Agentes Inteligentes*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

NERUR, S.; MAHAPATRA, R.; MANGALARAJ, G., 2005, "Challenges of migrating to agile methodologies", *Communications of ACM*, v. 48, n. 5, pp. 72-78.

NETTO, F. D. C., 2010, *Um Método Automático para Geração de Cronogramas de Tarefas de Correção de Bugs*. Dissertação de Mestrado, Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Rio de Janeiro, RJ, Brasil.

NEWMAN, M. E. J., 2003, "The structure and function of complex networks", *SIAM Review*, v. 45, n. 2 (Mar.), pp. 167-256.

NONAKA, I.; TAKEUCHI, H., 1995, *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, USA.

NORTHROP, L., 2002, "SEI's software product line tenets", *IEEE Software*, v. 19, n. 4, pp. 32-40.

NUNAMAKER, J.; ROMANO, N.; BRIGGS, R., 2001, "A Framework for Collaboration and Knowledge Management". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS)*, p. 1060, Maui, HI, USA.

NUNES, V. T., 2007, *Um Modelo de Suporte à Gestão de Conhecimento Baseado em Contexto*. Dissertação de Mestrado, UFRJ/IM/NCE, Rio de Janeiro, RJ, Brasil.

NUNES, V. T.; MAGDALENO, A. M.; WERNER, C. M. L., 2010a, *Modelagem de contexto sobre o domínio de processos de desenvolvimento de software*, Relatório Técnico ES-734/10, PESC-COPPE. Disponível em: http://www.cos.ufrj.br.

NUNES, V. T.; WERNER, C.; SANTORO, F. M., 2010b, "Context-Based Process Line". In: *International Conference on Enterprise Information Systems (ICEIS)*, pp. 277-282, Funchal, Madeira, Portugal.

ODYSSEY, 2010. Odyssey SDE Homepage. Disponível em: http://reuse.cos.ufrj.br. Acesso em: 12 Jan 2010.

OGAWA, M.; MA, K., 2008, "StarGate: A Unified, Interactive Visualization of Software Projects". *IEEE Pacific Visualization Symposium*, pp. 191-198, Kyoto, Japan.

OLIVEIRA, R. F. D., 2006, *Formalização e Verificação de Consistência na Representação de Variabilidades*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

OMG, 2008. Software Process Engineering Metamodel. Disponível em: http://www.omg.org/technology/documents/formal/spem.htm. Acesso em: 23 Ago 2008.

ORR, K., 2002, "CMM Versus Agile Development: Religious Wars and Software Development", *Cutter Consortium*, v. 3, n. 7, pp. 1-34.

OSTERWEIL, L., 1987, "Software processes are software too". In: *International Conference on Software Engineering (ICSE)*, pp. 2-13, Monterey, CA, USA.

PAI, M.; MCCULLOCH, M.; GORMAN, J. D.; *et al*., 2004, "Systematic reviews and meta-analyses: an illustrated, step-by-step guide", *The National Medical Journal of India*, v. 17, n. 2, pp. 86-95.

PALMER, S. R.; FELSING, J. M., 2002, *A Practical Guide to Feature-Driven Development*. USA, Prentice Hall.

PAPADIMITRIOU, C. H.; STEIGLITZ, K., 1998, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

PARK, S.; NA, H.; PARK, S.; *et al*., 2006, "A semi-automated filtering technique for software process tailoring using neural network", *Expert Systems with Applications*, v. 30, n. 2 (Fev.), pp. 179-189.

PATEL, C.; LYCETT, M.; MACREDIE, R.; *et al*., 2006, "Perceptions of Agility and Collaboration in Software Development Practice". In: *Hawaii International Conference on System Sciences (HICSS)*, pp. 1-7, Kauai, Hawaii, USA.

PAULK, M., 2001, "Extreme programming from a CMM perspective", *Software, IEEE*, v. 18, n. 6, pp. 19-26.

PEDREIRA, O.; PIATTINI, M.; LUACES, M. R.; *et al*., 2007, "A systematic review of software process tailoring", *SIGSOFT Software Engineering Notes*, v. 32, n. 3, pp. 1-6.

PENG XU; RAMESH, B., 2008, "Using Process Tailoring to Manage Software Development Challenges", *IT Professional*, v. 10, n. 4, pp. 39-45.

PFLEEGER, S. L., 1999, "Albert Einstein and Empirical Software Engineering", *Computer*, v. 32, n. 10, pp. 32-38.

POULIN, J. S., 1995, "Populating software repositories: Incentives and domain-specific software", *Journal of Systems and Software (JSS)*, v. 30, n. 3 (Set.), pp. 187-199.

PRESSMAN, R. S., 2001, *Software Engineering: A Practitioner's Approach*. 5 ed. McGraw-Hill.

PRIETO-DÍAZ, R., 1991, "Implementing faceted classification for software reuse", *Communications of ACM*, v. 34, n. 5, pp. 88-97.

PROTEGE, 2010. The Protégé Ontology Editor and Knowledge Acquisition System. Disponível em: http://protege.stanford.edu/. Acesso em: 25 Jan 2010.

PRUDENCIO, J.; MURTA, L.; WERNER, C., 2009, "On the Selection of Concurrency Control Policies for Configuration Management". In: *Brazilian Symposium on Software Engineering (SBES)*, pp. 155-164, Fortaleza, Brazil.

PUNTAR, S.; IENDRIKE, H. D. S.; MAGDALENO, A. M.; *et al*., 2009, *Estudo Conceitual sobre BPMS*, Relatório Técnico 0007/2009, RelaTe-DIA, UNIRIO. Disponível em: http://seer.unirio.br/index.php/monografiasppgi.

PUTNAM, L. H.; MYERS, W., 2003, *Five Core Metrics: The Intelligence Behind Successful Software Management*. Dorset House Publishing Company, Incorporated.

QUMER, A.; HENDERSON-SELLERS, B., 2008, "An evaluation of the degree of agility in six agile methods and its applicability for method engineering", *Information and Software Technology*, v. 50, n. 4 (Mar.), pp. 280-295.

RAMAN, S., 2000, "It is software process, stupid: next millennium software quality key", *Aerospace and Electronic Systems Magazine, IEEE*, v. 15, n. 6, pp. 33-37.

RAYMOND, E. S., 2001, *The Cathedral & the Bazaar*. O'Reilly Media.

REEVES, C. R., 1993, *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press.

REIFER, D., 2003, "XP and the CMM", *IEEE Software*, v. 20, n. 3, pp. 14-15.

REIFER, D. J., 2002, "How Good are Agile Methods?", *IEEE Software*, v. 19, n. 4, pp. 16-18.

REIS, C. R., 2003, *Caracterização de um Processo de Software para Projetos de Software Livre*. Dissertação de Mestrado, USP, São Paulo, SP, Brasil. Disponível em: http://www.async.com.br/~kiko/dissert.pdf.

REIS, R. Q., 2002, *APSEE-Reuse: um meta-modelo para apoiar a reutilização de processos de software*. Tese de Doutorado, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brasil. Disponível em: http://hdl.handle.net/10183/1622.

ROBERTS, F. S., 2009, *Measurement Theory: Volume 7: With Applications to Decisionmaking, Utility, and the Social Sciences (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press.

ROCHA, A. R. C. D.; MALDONADO, J. C.; WEBER, K. C., 2001, *Qualidade de Software: Teoria e Prática*. São Paulo, SP, Brasil, Prentice Hall.

ROMBACH, D., 2006, "Integrated Software Process and Product Lines", *Unifying the Software Process Spectrum*, Berlin/Heidelberg: Springer-Verlag, pp. 83-90.

RUHE, G., 2003, *Software Engineering Decision Support*, iCORE Research Report Vol. 2.

RUS, I.; HALLING, M.; BIFFL, S., 2003, "Supporting Decision-Making in Software Engineering With Process Simulation and Empirical Studies", *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, v. 13, n. 5, pp. 531-545.

SANTANA, C. A.; TIMÓTEO, A. L.; VASCONCELOS, A. M. L., 2006, "Mapeamento do modelo de Melhoria do Processo de Software Brasileiro (MPS.Br) para empresas que utilizam Extreme Programming

(XP) como metodologia de desenvolvimento". In: *Anais do V Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 130-143, Vila Velha, ES, Brasil.

SANTORO, F.; BRÉZILLON, P.; ARAUJO, R., 2007, "Context Dynamics in Software Engineering Process", *Computer Supported Cooperative Work in Design III*, pp. 377-388.

SANTORO, F. M., 2001, *Um Modelo de Cooperação para Aprendizagem baseada em Projetos*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

SANTOS, M. B.; SOUZA, C. R. D., 2008, "Visualização Temporal de Redes Sociais com o OSSNetwork". *Workshop on Information, Visualization and Analysis in Social Networks (WIVA)*, pp. 79-88, Campinas, SP, Brasil.

SANTOS, T. A. L., 2010, *Identificação de Características de Colaboração em Processos de Desenvolvimento de Software através de Redes Sociais*. Dissertação de Mestrado, Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Rio de Janeiro, RJ, Brasil.

SANTOS, T. A. L.; ARAUJO, R. M. D.; MAGDALENO, A. M., 2010, "Identifying Collaboration Patterns in Software Development Social Networks", *Infocomp - Journal of Computer Science (to appear)*

SANTOS, T. A. L.; ARAUJO, R. M. D.; MAGDALENO, A. M., 2009, "Padrões para Percepção da Colaboração em Redes Sociais de Desenvolvimento de Software". *Workshop de Desenvolvimento Distribuído de Software (WDDS)*, pp. 1-10, Fortaleza, CE, Brasil.

SANTOS, V. V., 2008, *CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive Systems*. Tese de Doutorado, Universidade Federal de Pernambuco (UFPE), Recife, PE, Brasil.

SARMENTO, A., 2002, *Impacto dos sistemas colaborativos nas organizações : estudo de casos de adopção e utilização de sistemas workflow*. Tese de Doutorado, Departamento de Sistemas de Informação - Universidade do Minho, Minho, Portugal. Disponível em: http://repositorium.sdum.uminho.pt/handle/1822/285.

SCACCHI, W., 2007, "Free/open source software development: recent research results and emerging opportunities". *Meeting on European software engineering conference*, pp. 459-468, Dubrovnik, Croatia.

SCHEIN, E. H., 1990, "Organizational Culture", *American Psychologist*, v. 45, n. 2, pp. 109-119.

SCHMIDT, A.; AIDOO, K. A.; TAKALUOMA, A.; *et al*., 1999, "Advanced Interaction in Context". In: *Proceedings of the 1st international*

symposium on Handheld and Ubiquitous Computing, pp. 89-101, Karlsruhe, Germany.

SCHNIEDERS, A.; PUHLMANN, F., 2006, "Variability mechanisms in e-business process families". *International Conference on Business Information Systems (BIS)*, pp. 583-601, Klagenfurt, Austria.

SCHOLTES, P. R.; JOINER, B. L.; STREIBEL, B. J., 2003, *The Team Handbook*. 3 ed. Joiner/Oriel Inc.

SCHWABER, K., 2004, *Agile Project Management with Scrum*. Washington, DC, USA, Microsoft Press.

SHARP, A.; MCDERMOTT, P., 2001, *Workflow Modeling: Tools for Process Improvement and Application Development*. 1 ed. Artech House Publishers.

SHULL, F.; CARVER, J.; TRAVASSOS, G. H., 2001, "An empirical methodology for introducing software processes", *SIGSOFT Softw. Eng. Notes*, v. 26, n. 5, pp. 288-296.

SILVA, M. A., 2010, *IAVEMS : Infraestrutura de Apoio à Visualização na Evolução de Métricas de Software*. Projeto Final, UFRJ/IM, Rio de Janeiro, RJ, Brasil (em andamento).

SIMIDCHIEVA, B.; CLARKE, L.; OSTERWEIL, L., 2007, "Representing Process Variation with a Process Family", *Software Process Dynamics and Agility*, Minneapolis, MN, USA: Springer, pp. 109-120.

SLOOTEN, K. V.; BRINKKEMPER, S., 1993, "A Method Engineering Approach to Information Systems Development". In: *Proceedings of the IFIP WG8.1 Working Conference on Information System Development Process*, pp. 167-186.

SMITH, M.; WELTY, C.; MCGUINNESS, D. L., 2004. OWL Web Ontology Language Guide. Disponível em: http://www.w3.org/TR/owl-guide/. Acesso em: 26 Jan 2010.

SOFTEX, 2009, *Melhoria de Processo do Software Brasileiro – Guia Geral*, Modelo de Qualidade Versão 2.0 Disponível em: http://www.softex.br.

SOLINGEN, R. V.; BERGHOUT, E., 1999, *Goal/Question/Metric Method*. McGraw-Hill Inc.,USA.

SOMMERVILLE, I., 2004, *Software Engineering*. 7 ed. Addison Wesley.

SOUSA JUNIOR, S. F.; BALIEIRO, M. A.; DE SOUZA, C. D., 2008, "Análise Multidimensional de Redes Sociais de Projetos de Software Livre". In: *Anais do Simpósio Brasileiro de Sistemas Colaborativos (SBSC)*, pp. 23-33, Vitória, ES, Brasil.

SOUSA JUNIOR, S. F.; DE SOUZA, C. D., 2008, "Visualização Integrada de Múltiplas Métricas de Redes Sociais". *Workshop on Information, Visualization and Analysis in Social Networks (WIVA)*, pp. 60-68, Campinas, SP, Brasil.

SOUZA, C. D.; FROEHLICH, J.; DOURISH, P., 2005, "Seeking the source: software source code as a social and technical artifact". In: *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pp. 197-206, Sanibel Island, FL, USA.

SOUZA, C. R. D.; QUIRK, S.; TRAINER, E.; *et al*., 2007, "Supporting collaborative software development through the visualization of socio-technical dependencies". In: *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pp. 147-156, Sanibel Island, FL, USA.

SPRADLIN, T., 1997. Lexicon of Decision Making. Disponível em: http://dssresources.com/papers/features/spradlin/spradlin03052004.html. Acesso em: 21 Dez 2009.

STAPLETON, J.; CONSTABLE, P., 1997, *DSDM Dynamic Systems Development Method: The Method in Practice*. Boston, MA, USA, Addison Wesley.

SUTTON, S.; OSTERWEIL, L., 1996, "Product families and process families". *International Software Process Workshop (ISPW)*, pp. 109-111, Dijon, France.

SZWARCFITER, J., 1986, *Grafos e Algoritmos Computacionais*. Rio de Janeiro, RJ, Brasil, Campus.

SZYPERSKI, C., 2002, *Component Software: Beyond Object-Oriented Programming*. 2 ed. Boston, MA, USA, Addison-Wesley.

TAURION, C., 2004, *Software Livre: Potencialidades e Modelos de Negócio*. 2 ed. Rio de Janeiro, RJ, Brasil, Brasport.

TEIXEIRA, E. N., 2008, *Flexibilização para Representação de Características no Ambiente Odyssey*. Projeto Final, UFRJ/IM, Rio de Janeiro, RJ, Brasil.

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

TELLERIA, K. M.; LITTLE, D.; MACBRYDE, J., 2002, "Managing processes through teamwork", *Business Process Management Journal*, v. 8, n. 4, pp. 338 - 350.

THEUNISSEN, M.; KOURIE, D.; BOAKE, A., 2008, "Corporate-, Agile- and Open Source Software Development: A Witch's Brew or An Elixir of Life?", *Balancing Agility and Formalism in Software Engineering: Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques (CEE-SET)*, Springer-Verlag, pp. 84-95.

TRAINER, E.; QUIRK, S.; SOUZA, C. D.; *et al*., 2005, "Bridging the gap between technical and social dependencies with Ariadne". *Workshop on Eclipse technology eXchange - International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pp. 26-30, San Diego, CA, USA.

TRAVASSOS, G. H.; KALINOWSKI, M., 2008, *iMPS*, Relatório Técnico, SOFTEX.

TRAVASSOS, G.; DOS SANTOS, P.; NETO, P.; *et al*., 2008, "An Environment to Support Large Scale Experimentation in Software Engineering". In: *International Conference on the Engineering of Complex Computer Systems (ICECCS)*, pp. 193-202, Belfast, United Kingdom.

TSIRAKIDIS, P.; KOBLER, F.; KRCMAR, H., 2009, "Identification of Success and Failure Factors of Two Agile Software Development Teams in an Open Source Organization". In: *Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*, pp. 295-296, Limerick, Ireland.

TURK, D.; FRANCE, R.; RUMPE, B., 2002, "Limitations of agile software processes". In: *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*, pp. 43-46, Alghero, Italy.

TURNER, R.; JAIN, A., 2002, "Agile Meets CMMI: Culture Clash or Common Cause?", *Extreme Programming and Agile Methods — XP/Agile Universe 2002*, 1 ed, chapter 2418, Heidelberg: Springer-Verlag, pp. 153-165.

TURNU, I.; MELIS, M.; CAU, A.; *et al*., 2006, "Modeling and simulation of open source development using an agile practice", *Journal of Systems Architecture (JSA)*, v. 52, n. 11, pp. 610-618.

VASCONCELLOS, E. P. G. D.; HEMSLEY, J. R., 2003, *Estrutura das organizações*. São Paulo, SP, Brasil, Pioneira Thomson.

VIEGAS, F.; DONATH, J., 2004, "Social network visualization: Can we go beyond the graph", *Workshop on Social Networks, CSCW*, v. 4, pp. 6-10.

VINEKAR, V.; SLINKMAN, C. W.; NERUR, S., 2006, "Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View", *Information Systems Management*, v. 23, n. 3, pp. 31-42.

WARSTA, J.; ABRAHAMSSON, P., 2003, "Is Open Source Software Development Essentially an Agile Method?". In: *Proceedings of the Workshop on Open Source Software Development*, pp. 143-147, Portland, OR, USA.

WASHIZAKI, H., 2006, "Building Software Process Line Architectures from Bottom Up", *Product-Focused Software Process Improvement (PROFES)*, chapter 4034, Amsterdam, The Netherlands: LNCS, pp. 415-421.

WASSERMAN, S.; FAUST, K., 1994, *Social Network Analysis: Methods and Applications*. 1 ed. Cambridge, United Kingdom, Cambridge University Press.

WEERD, G. C. V. D., 2009, *Advancing in software product management: An incremental method engineering approach*. Tese de Doutorado, Utrecht University, Netherlands. Disponível em: http://igitur-archive.library.uu.nl/dissertations/2009-0826-200144/UUindex.html.

WERNER, C.; BRAGA, R., 2005, "A Engenharia de Domínio e o Desenvolvimento Baseado em Componentes", In: Gimenes, I. M. D. S., Huzita, E. H. M. (eds), *Desenvolvimento Baseado em Componentes: Conceitos e Técnicas*, Rio de Janeiro, RJ, Brasil: Ciência Moderna, pp. 57-103.

WWF, 2003, *Redes: uma introdução às dinâmicas da conectividade e da auto-organização*, Relatório Técnico, WWF - Brasil. Disponível em: http://www.wwf.org.

YAMAUCHI, Y.; YOKOZAWA, M.; SHINOHARA, T.; *et al.*, 2000, "Collaboration with Lean Media: how open-source software succeeds". In: *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 329-338, Philadelphia, PA, USA.