

Edson dos Santos Cordeiro

**MODELAGEM DESCRITIVA ITERATIVA E
INCREMENTAL DE PROCESSO DE SOFTWARE:
UMA EXPERIÊNCIA EM UMA MICROEMPRESA
DE DESENVOLVIMENTO DE SOFTWARE**

Florianópolis – SC

2003

UNIVERSIDADE FEDERAL DE SANTA CATARINA

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO**

Edson dos Santos Cordeiro

**MODELAGEM DESCRITIVA ITERATIVA E
INCREMENTAL DE PROCESSO DE SOFTWARE:
UMA EXPERIÊNCIA EM UMA MICROEMPRESA
DE DESENVOLVIMENTO DE SOFTWARE**

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Dra. Christiane Gresse von Wangenheim

Florianópolis, agosto de 2003.

MODELAGEM DESCRITIVA ITERATIVA E INCREMENTAL DE PROCESSO DE SOFTWARE: UMA EXPERIÊNCIA EM UMA MICROEMPRESA DE DESENVOLVIMENTO DE SOFTWARE

Edson dos Santos Cordeiro

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistema de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Dr. Raul Sidnei Wazlawick

Banca Examinadora

Dra. Christiane Gresse von Wangenheim

Dra. Patrícia Vilain

Dr. Ricardo Pereira e Silva

DEDICATÓRIA

Este trabalho é dedicado ao meu pai que,
mesmo sob as maiores adversidades,
despiu-se do egoísmo comum a muitos de nós e
em momento algum de sua luta,
deixou de acreditar e de me incentivar.

AGRADECIMENTOS

À minha mãe que, em seu aparente silêncio, foi brava incentivadora.

À minha esposa Margarete que não se limitou a acreditar e fez da minha a sua jornada.

À minha família, em especial à Marlene, Alex, Silvana, Benedito, Nair e Georgette, pela fé e motivação.

À minha orientadora Christiane pela competência e compreensão diante das dificuldades pelas quais passei.

À empresa que possibilitou realização desse trabalho.

Aos professores Ricardo e Patrícia pela disponibilidade e imensuráveis contribuições.

Aos meus amigos Rodolfo, Eduardo, Marília e Regina que se prontificaram, por diversas vezes, em momentos difíceis.

SUMÁRIO

<u>INTRODUÇÃO</u>	1
1.1 PROBLEMA	3
1.2 JUSTIFICATIVA	6
1.3 OBJETIVOS	7
1.4 METODOLOGIA	8
1.5 ESTRUTURA DO TRABALHO	15
<u>2 CONCEITOS FUNDAMENTAIS</u>	17
2.1 O QUE SÃO PROCESSOS DE SOFTWARE?	17
2.2 QUAIS SÃO OS PRINCIPAIS COMPONENTES DE UM PROCESSO DE SOFTWARE?	18
2.3 O QUE SÃO MODELOS DE CICLO DE VIDA DE SOFTWARE?	24
2.4 O QUE SÃO MODELOS DE PROCESSO DE SOFTWARE?	34
2.5 O QUE É PROCESSO DE MANUTENÇÃO DE SOFTWARE?	36
2.6 MODELOS DE PROCESSOS E EXPERIÊNCIAS DIRECIONADOS À MANUTENÇÃO	41
<u>3 MODELAGEM DE PROCESSOS DE SOFTWARE</u>	44
3.1 O PAPEL DA MODELAGEM DE PROCESSO DE SOFTWARE E SUAS IMPLICAÇÕES	44
3.2 LINGUAGENS E PARADIGMAS PARA MODELAGEM DE PROCESSO DE SOFTWARE	46
3.3 MEIOS UTILIZADOS PARA REPRESENTAR OS MODELOS DE PROCESSO	48
3.4 ABORDAGENS E EXPERIÊNCIAS UTILIZADAS NA MODELAGEM DE PROCESSOS.....	49
3.5 REQUISITOS DE MICROEMPRESAS EM RELAÇÃO À MODELAGEM DE PROCESSOS	65
3.6 CONSIDERAÇÕES SOBRE AS NECESSIDADES E PERFIL DE MICROEMPRESAS.....	72
<u>4 ABORDAGEM ITERATIVA E INCREMENTAL</u>	75
4.1 VISÃO GERAL DA ABORDAGEM.....	75
4.2 ESTUDO DE CASO: ETAPA DE PLANEJAMENTO INICIAL	80
4.3 ETAPA DE DESCRIÇÃO DO ELEMENTO DO PROCESSO	96
4.4 RESUMO DA DESCRIÇÃO E VALIDAÇÃO DO PROCESSO DE SOFTWARE.....	108
4.5 ACURÁCIA DO GUIA DE PROCESSO DE SOFTWARE	135
<u>5 AVALIAÇÃO DA ABORDAGEM</u>	140
5.1 AVALIAÇÃO DA ABORDAGEM EM RELAÇÃO AOS REQUISITOS DA MICROEMPRESA....	144
<u>6 DISCUSSÃO</u>	149
<u>7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS</u>	153
<u>REFERÊNCIAS BIBLIOGRÁFICAS</u>	155
<u>ANEXO A: RESULTADO DA PESQUISA DO PERFIL DE MICROEMPRESAS</u>	162
<u>ANEXO B: INSTRUMENTOS DE COLETA DE DADOS</u>	192
<u>ANEXO C: AUTORIZAÇÃO PARA DIVULGAÇÃO DOS DADOS DA PESQUISA</u> ...	199

LISTA DE TABELAS

Tabela 1: Relacionamento entre Perspectivas x Propriedades do Método Elicit.	56
Tabela 2: Total de horas empregas na descrição do processo de software.....	141
Tabela 3: Distribuição de horas entre as principais atividades do estudo de caso.....	141
Tabela 4: Similaridade entre as etapas dos estudos de casos.....	150
Tabela 5: Principais características dos estudos ao descrever o processo	151

LISTA DE QUADROS

Quadro 1: Modelos de ciclo de vida: vantagens e desvantagens	34
Quadro 2: Notações gráficas utilizadas pela ferramenta Spearmint.	63
Quadro 3: Notações gráficas utilizadas para representar execução em paralelo.	63
Quadro 4: Requisitos presentes nos estudos apresentados na literatura.	76
Quadro 5: Resultado da coleta de dados da etapa de avaliação da abordagem.	139
Quadro 6: Avaliação da conformidade aos requisitos da empresa	145

LISTA DE FIGURAS

Figura 1: Conhecimento e uso de modelos de melhoria do processo de software.....	3
Figura 2: Percentual de empresas que documentam seus processos de software.....	4
Figura 3: Processo de avaliação da abordagem.....	15
Figura 4: Componentes de um processo de software.....	19
Figura 5: Relacionamentos da entidade agente.....	22
Figura 6: Relacionamentos da entidade papel.....	24
Figura 7: Modelo de ciclo de vida Cascata.....	26
Figura 8: Modelo Espiral.....	28
Figura 9: Processo de Prototipação.....	29
Figura 10: Modelo Prototipação Incremental.....	30
Figura 11: Modelo Prototipação Evolutiva.....	31
Figura 12: Modelo Prototipação Rápida Descartável.....	32
Figura 13: Processo de Manutenção IEEE.....	39
Figura 14: Modelo de Estágios para o Ciclo de Vida do Software.....	42
Figura 15: Modelo de Estágios Versionados.....	43
Figura 16: Abordagem aplicada pelo estudo conduzido na LG.....	53
Figura 17: Estrutura de Processos de software proposta.....	54
Figura 18: Abordagem Elicit para descrever elementos do processo de software.....	55
Figura 19: Passos do modelo descritivo de processo de software do IESE.....	58
Figura 20: Exemplo em Spearmint de execução de atividades em paralelo.....	64
Figura 21: Área de trabalho da ferramenta Spearmint (IESE).....	65
Figura 22: Etapas e fases da abordagem para descrição de processo proposta.....	78
Figura 23: Descrição do Processo de Software em profundidade (<i>top-down</i>).....	79
Figura 24: Descrição do Processo de Software em largura (<i>top-down</i>).....	80
Figura 25: Instrumento de coleta de dados para caracterização dos produtos.....	83
Figura 26: Estrutura do Guia de Processo de Software.....	89
Figura 27: Ocorrência e frequência das atividades de alto nível em relação aos produtos.....	92
Figura 28: Frequência das atividades de alto nível executadas pelos agentes.....	92
Figura 29: Frequência de execução das atividades entre os agentes.....	93
Figura 30: Versões do ciclo de vida da microempresa.....	94
Figura 31: Instrumento de coleta de dados 7 (atividades) aplicado no estudo de caso.....	99
Figura 32: Instrumento de coleta de dados 8 (artefatos) aplicado no estudo de caso.....	101
Figura 33: Instrumento de coleta de dados 10 (papéis) aplicado ao estudo de caso.....	103
Figura 34: Instrumento de coleta de dados 9 (ferramentas) aplicado ao estudo de caso.....	104
Figura 35: Relacionamento possíveis entre os componentes do processo de software.....	105
Figura 36: Instrumento de coleta de dados 11 (relacionamentos).....	106
Figura 37: <i>Abstraction Sheet</i> do elemento de processo atividade.....	108
Figura 38: Validação da descrição da atividade de suporte.....	110
Figura 39: Modelo descrito e validado da atividade de Suporte.....	112
Figura 40: Modelo descrito e validado da atividade Gerência de Tarefas.....	114
Figura 41: Representação gráfica dos elementos da atividade Implementação.....	116
Figura 42: Representação gráfica dos elementos da atividade Teste.....	119
Figura 43: Representação gráfica dos elementos da atividade Controle de Versões.....	120
Figura 44: Representação gráfica dos elementos da atividade Teste de Instalação.....	122
Figura 45: Representação gráfica dos elementos da atividade Logística.....	123
Figura 46: Representação gráfica da atividade Instalação e Treinamento.....	124
Figura 47: Ciclo de vida final.....	126
Figura 48: Fluxo de artefatos no ciclo de vida.....	127
Figura 49: Papéis envolvidos no ciclo de vida.....	127
Figura 50: Ferramentas utilizadas no ciclo de vida.....	128

Figura 51: Refinamento do fluxo de controle da atividade Suporte.....	128
Figura 52: Fluxo de artefatos da atividade Suporte.....	128
Figura 53: Refinamento do fluxo de controle da atividade Gerência de Tarefas.....	129
Figura 54: Fluxo de artefatos da atividade Gerência de Tarefas.....	129
Figura 55: Refinamento do fluxo de controle da atividade Implementação.....	129
Figura 56: Fluxo de produtos da atividade Implementação.....	129
Figura 57: Refinamento da sub-atividade Modificar Artefatos.....	130
Figura 58: Fluxo de artefatos da sub-atividade Modificar Artefatos.....	130
Figura 59: Refinamento do fluxo de controle da atividade Teste.....	130
Figura 60: Fluxo de artefatos da atividade Teste.....	131
Figura 61: Refinamento do fluxo de controle da atividade Controle de Versão.....	131
Figura 62: Fluxo de artefatos da atividade Controle de Versão.....	131
Figura 63: Refinamento do fluxo de controle da atividade Teste de Instalação.....	132
Figura 64: Fluxo de artefatos da atividade Teste de Instalação.....	132
Figura 65: Refinamento do fluxo de controle da atividade Logística.....	132
Figura 66: Fluxo de artefatos da atividade Logística.....	132
Figura 67: Refinamento do fluxo de controle da atividade Instalação e Treinamento.....	133
Figura 68: Fluxo de artefatos da atividade Instalação e Treinamento.....	133
Figura 69: Atividades realizadas na avaliação do GPS.....	134
Figura 70: <i>Abstraction Sheet</i> : plano de mensuração da abordagem.....	136
Figura 71: Abordagem Iterativa e Incremental com nova etapa.....	143
Figura 72: Número de causas da inconsistência provocadas por agentes e produtos.....	144

RESUMO

A modelagem descritiva de processo de software é uma atividade essencial para a melhoria do processo de desenvolvimento de software. Uma representação explícita de como ele é executado é a base para seu entendimento, análise e melhoria. No entanto, é necessária uma abordagem sistemática que capture o processo “como ele é” executado em um modelo descritivo. O objetivo desse trabalho foi desenvolver, aplicar e avaliar uma abordagem iterativa e incremental para modelar processos de software. O estudo foi conduzido em uma microempresa localizada na cidade de Londrina – Paraná. O desenvolvimento da abordagem foi parametrizado nos requisitos da microempresa em relação à modelagem de processos de software e experiências relatadas na literatura. A avaliação da abordagem ocorreu por intermédio da aplicação do paradigma GQM e atendimento aos requisitos definidos pela microempresa. A aplicação da abordagem permitiu representar o processo de software real da microempresa em um Guia de Processo de Software gerado pela ferramenta *Spearmint*. O estudo relata experiências, aspectos positivos e negativos decorrentes da aplicação da abordagem iterativa e incremental para modelagem de processo de software da microempresa.

Palavras-chave: modelagem descritiva de processo de software, abordagem para descrição de processo de software, melhoria de processo de software.

ABSTRACT

Descriptive software process modeling is an essential activity for the improvement of software process development. An explicit representation of how it is performed is the basis of its understanding, analysis and improvement. However, a systematic approach is necessary to capture the process “as it is” performed in a descriptive model. The objective of this work is to develop, apply and assess an interactive incremental approach for software process modeling. The study was carried out in a small organization in the municipality of Londrina – Paraná. The parameters for approach’s development were the requirements of a small organization as far as software process modeling and the experiences described in the literature are concerned. The assessment of the approach was carried out by applying the GQM paradigm and complying with the requirements determined by the small organization. The approach’s application has allowed to represent the real software process used in a small organization in a Software Process Guide generated from Spearmint tool. The study reports experiences, the positive and negative aspects resulting from interactive and incremental approach’s application for modeling the software process of the small organization.

Keywords: descriptive software process modeling, software process description approach, software process improvement.

INTRODUÇÃO

A descrição ou modelagem de processos de software é uma atividade fundamental para empresas de desenvolvimento de software que almejam qualidade no processo e produto (HÖLTJE et al, 1994; ALVES & FALBO, 2001; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001; MACHADO et al, 2001). A importância em descrever o processo de software tem sido discutida na literatura de engenharia de software como uma das formas de suporte à qualidade do produto de software, sistematizar as práticas empregadas durante o desenvolvimento, aumentar a maturidade do processo de software, estabelecer uma linha de base (*baseline*) para avaliação e melhoria etc. Dada a importância em formalizar o processo de software, diversas organizações adotam-na como uma estratégia que tem resultado em inúmeros benefícios.

A descrição de processo de software permite detalhar o processo de software real executado em organizações de desenvolvimento e, conseqüentemente, possibilitar que falhas no processo sejam detectadas e corrigidas. Além disso, a formalização de processo de software é considerada o primeiro passo para implantação de programas de melhoria como CMM, SPICE (futura norma ISO 15504), BOOTSTRAP entre outros, pelo fato de capturar como o processo de software é executado (em um guia de processo de software, por exemplo) e adquirir experiências e lições que poderão ser incorporadas e utilizadas em projetos futuros (BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 63).

No Brasil, entretanto, a maioria das empresas de desenvolvimento de software não modela seus processos de software (MCT/SEPIN, 2002, p. 63). A falta de descrição freqüentemente conduz a diversas conseqüências negativas às empresas de desenvolvimento de software, dentre as quais (CURTIS, KELLNER & OVER, 1992, p. 75; CONRADI, FERNSTRÖM & FUGGETTA, 1993, p. 2-3; BRIAND et al, 1998, p. 250; KELLNER et al, 1998, p. 3; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 64): atraso na entrega do produto, aumento no custo do projeto, baixa qualidade do

produto, comprometimento do planejamento do projeto, impossibilidade de estabelecer uma prática sistemática de desenvolvimento (processo repetível), os processos não são visíveis, há dificuldades na implantação de programas de mensuração, baixa produtividade.

Nesse contexto, a categoria de microempresas de desenvolvimento de software tem apresentado os piores índices em relação à descrição de processos de software (MCT/SEPIN, 2002, p. 63). Aparentemente, diversas causas têm contribuído para essa situação em microempresas: falta de recursos financeiros, indisponibilidade de pessoal para atividades da qualidade, pouca ou nenhuma experiência em tópicos relacionados à qualidade etc. Além disso, microempresas se destacam como a maior categoria de empresas no setor de software brasileiro, considerando a força de trabalho efetiva. No entanto, há poucas pesquisas direcionadas ao desenvolvimento de políticas relacionadas à melhoria do processo e produto de software que poderiam contribuir com o aumento da maturidade dessa categoria de empresas no cenário nacional (DURSKI, 2001, p. 105).

Apesar da literatura afirmar que a formalização de processo de software é o ponto de partida à implantação de programas de melhoria da qualidade de produto e processo (MADHAVJI et al, 1994; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001; BECKER-KORNSTAEDT & BELAU, 2001; MACHADO et al, 2001), poucas pesquisas são realizadas nesse sentido. Existem diversos trabalhos que apresentam ferramentas de suporte a descrição de processos de software (KELLNER et al, 1998; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001), experiências relacionadas à descrição de processos de software (MADHAVJI et al, 1994; HÖLTJE et al, 2001; BECKER-KORNSTAEDT & BELAU, 2001; MACHADO et al, 2001), estudos que apresentam notações gráficas voltadas à modelagem de processos de software (BRIAND et al, 1998), técnicas aplicadas à descrição de processos (BECKER, HAMANN & VERLAGE, 1997). Porém, poucas pesquisas na literatura abordam “como” formalizar um modelo que permita capturar as práticas de engenharia software empregadas no processo atual da organização (BECKER-KORNSTAEDT & BELAU, 2000). Embora pouco explorado, a adoção de abordagens que auxiliem na descrição de processos de software tem apresentado resultados positivos e se mostra como uma

alternativa viável para empresas de desenvolvimento que almejam formalizar seus processos de software (HÖLTJE et al, 1994; MADHAVJI et al, 1994; BASILI et, 1996; BECKER-KORNSTAEDT, 2001; MACHADO et al, 2001).

Considerando o cenário apresentado, torna-se fundamental a condução de estudos que investiguem, desenvolvam e apliquem abordagens direcionadas a “como” descrever o processo de software de empresas de pequeno porte e contribuam com a diminuição da informalidade estabelecendo um processo de software visível e sistemático.

1.1 Problema

A melhoria do processo de desenvolvimento de software, como um meio para alcançar qualidade e produtividade é objeto de estudos intensos nos últimos anos (CONRADI, FUGGETTA, 2001). No Brasil, o conhecimento e uso de modelos de melhoria do processo de software têm crescido nos últimos anos tanto no uso como na intenção de uso (vide Figura 1) conforme dados da pesquisa realizada pelo Ministério da Ciência e Tecnologia (MCT/SEPIN, 2001).

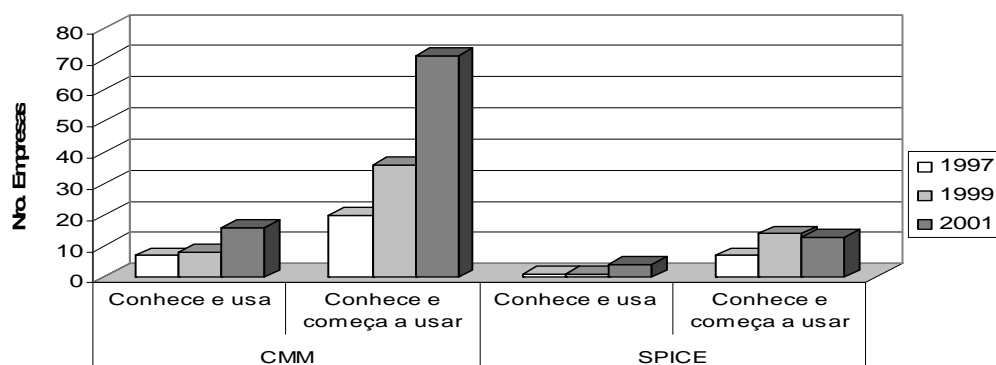


Figura 1: Conhecimento e uso de modelos de melhoria do processo de software

Algumas pesquisas afirmam que o primeiro passo para alcançar melhoria no processo de software por meio da implantação de modelos da qualidade é a formalização do processo de software (BECKER-KORNSTAEDT & BELAU, 2001, P. 1; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 63), ou seja, o processo de software real executado pela empresa deve ser descrito e representado em um modelo

descritivo (manual ou guia do processo de software). A modelagem descritiva de processos de software tem por objetivo capturar o processo de software executado (processo de software real) e representá-lo em um modelo ou guia (BECKER, HAMANN & VERLAGE, 1997, p. 1; BECKER-KORNSTAEDT & BELAU, 2001, p. 1). Os modelos de processo de software formam a base para o entendimento e análise do processo de software real, possibilitam a melhoria do processo de software existente, permitem re-projetar ou complementar os processos existentes formando uma base para mudanças ou disseminação do conhecimento sobre o processo de software.

No entanto, poucas empresas têm adotado essa abordagem como uma estratégia para compreender, disciplinar e melhorar seus processos de software. No Brasil, pesquisas apontam que apenas 35,5% das empresas (vide Figura 2) documentam seus processos de software (MCT/SEPIN, 2002, p. 63). Microempresas figuram como a categoria de empresas que apresenta o menor percentual em documentação do processo de software. Esse cenário demonstra que são necessárias políticas voltadas para microempresas de desenvolvimento de software no que se refere às políticas da qualidade.

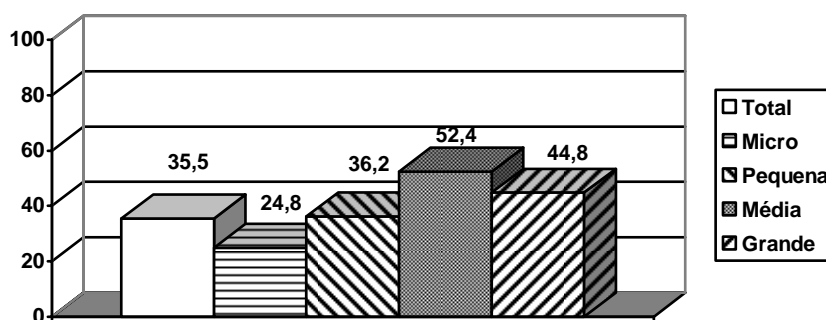


Figura 2: Percentual de empresas que documentam seus processos de software

Entretanto, o desenvolvimento de políticas da qualidade para microempresas deve considerar alguns empecilhos presentes nesse setor. A categoria de microempresas apresenta os piores indicadores relacionados diversos tópicos relacionados à qualidade em relação a outras categorias de empresas, dentre os quais: menor índice no uso de métricas, inexistência de responsável pela gestão da qualidade, baixo investimento em capacitação da força de trabalho, baixo investimento da organização em capacitação para a melhoria da qualidade, pouco conhecimento de

normas e modelos da qualidade etc (MCT/SEPIN, 2001, 115-135). Assim, a capacidade de investimento e força de trabalho de microempresas em qualidade de software requer a adequação de qualquer política da qualidade a esse perfil de empresas (DURSKI, 2001).

Frente à necessidade de alcançar a melhoria no processo de software e estabelecer estratégias para alcançar a melhoria no processo de software por meio da modelagem descritiva do processo de software real de uma organização de desenvolvimento, diversos estudos tem apresentado abordagens que sistematizam “como” descrever o processo de software de uma organização de desenvolvimento de software (HÖLTJE et al, 1994; MADHAVJI et al, 2001; BECKER-KORNSTAEDT & BELAU, 2001; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001; MACHADO et al, 2001). Entretanto, conforme os dados do MCT/SEPIN apresentados anteriormente, algumas dificuldades devem ser consideradas, citando as principais, por uma abordagem para modelagem de processo de software na maioria das microempresas no Brasil: baixa capacidade de investimento em programas da qualidade e capacitação da força de trabalho, pouca experiência e conhecimento em temas da qualidade, alto comprometimento da força de trabalho em atividades não relacionadas à qualidade (MCT/SEPIN, 2001, p. 115-135).

Além de considerar os dados do MCT/SEPIN, é necessário investigar quais são as necessidades de microempresas em relação à modelagem de processo de software, ou seja, quais são suas prioridades? Qual é sua capacidade de força de trabalho e investimento em relação à modelagem de processo de software? Qual estratégia seria mais apropriada às microempresas ao adotar uma abordagem para modelagem de processo de software? As abordagens para modelagem de processo de software existentes atendem ao perfil de microempresas? O perfil de microempresa se diferencia das demais categorias de empresas com relação à modelagem de processo de software? Microempresas consideram a modelagem de processo de software uma atividade importante e estão dispostas a investir, dentro de sua capacidade, em tecnologias para alcançar esse objetivo?

Essas questões, entre outras, são algumas preocupações que devem ser consideradas ao descrever processos de software de microempresas de desenvolvimento de software. A abrangência das questões apresentadas demonstra que existe um campo fértil e pouco explorado (se não urgente) para pesquisas relacionadas à modelagem de processos de software em microempresas de desenvolvimento de software. No entanto, poucas pesquisas estão sendo conduzidas nesse sentido apesar das experiências já relatadas na literatura apresentam resultados significativos (BECKER-KORNSRAEDT & BELAU, 2001; MACHADO et al, 2001). Portanto, devido às poucas pesquisas existentes, principalmente no Brasil, se faz necessário investigar as estratégias propostas pelas abordagens para modelagem de processos de software presentes na literatura frente ao perfil de microempresas brasileiras e apontar alternativas que incorporem essas soluções, seja em parte ou em sua totalidade.

1.2 Justificativa

Nos últimos anos o processo de software tem sido reconhecido, tanto pela indústria de software quanto pela academia, como um fator crítico para o desenvolvimento de sistemas (HÖLTJE et al, 1994; BASILI et al, 1996; BRIAND et al, 1998; BECKER-KORNSTAEDT & BELAU, 2000; BECKER-KORNSTAEDT, 2001). Entender, avaliar, padronizar, controlar, aprender, comunicar, melhorar, prever, mensurar, inspecionar, certificar etc são necessidades que estão relacionadas ao processo de software de organizações de desenvolvimento de software. No entanto, o primeiro passo para atender a essas necessidades é descrever o processo de software, ou seja, como o processo é executado ou deveria ser executado numa determinada organização de desenvolvimento. No Brasil, conforme dados da pesquisa bial realizada pelo Ministério da Ciência e Tecnologia (MCT) em 2001 (MCT/SEPIN, 2002), a documentação do processo de software é praticado por 35,5% das empresas (considerado o total de empresas independente de sua categoria), ou seja, uma grande parte não documenta seus processos de software. Segundo a mesma pesquisa, considerando apenas a categoria de microempresas, o percentual ainda é menor, 24,8%.

Conseqüentemente, a falta de descrição do processo de software pode dificultar que diversos benefícios sejam alcançados pela organização: a) guia para organizar, planejar, estimar e gerenciar o processo de software; b) pré-requisitos que determinem quais produtos intermediários devem ser entregues ao cliente; c) base para determinar tecnologias em engenharia de software apropriadas para prover suporte ao processo de software; d) infra-estrutura básica para análise e/ou padrões de mensuração (métricas) para alocação e consumo de recursos durante o processo de software e, e) base para condução de estudos empíricos que permitam averiguar a produtividade, custo, e qualidade de novas tecnologias empregadas ao desenvolver e/ou manter produtos de software (SCACCHI, 2001, p. 3).

Entre as possíveis alternativas exploradas pela literatura para contornar a falta de formalização de processos de software, encontra-se o desenvolvimento e utilização de abordagens que guiem organizações de desenvolvimento na descrição de seus processos de software. No entanto, poucas pesquisas estão sendo conduzidas com o objetivo de explorar ou desenvolver abordagens que auxiliem organizações de desenvolvimento no estabelecimento de seus processos de software (MADHAVJI et al, 1994; HÖLTJE, et al, 1994; BECKER-KORNSTAEDT, 2001).

1.3 Objetivos

Considerando a problemática relacionada à falta de documentação de processos de software, a necessidade de políticas voltadas especificamente para empresas de pequeno porte e as poucas pesquisas que exploram o desenvolvimento de abordagens que enfatizem “como” descrever os processos de software, o presente trabalho delineou os seguintes objetivos.

1.3.1 Objetivo geral

Desenvolver, aplicar e avaliar uma abordagem para descrever o processo de software de uma microempresa de desenvolvimento de software.

1.3.2 Objetivos específicos

Para atender ao objetivo geral, foram determinados quatro objetivos específicos:

1. Abordar os principais conceitos relacionados à modelagem de processos de software, entender as vantagens e desvantagens de cada um e identificar os possíveis contextos de aplicação;
2. Identificar as principais dificuldades (requisitos) de microempresas em relação à descrição de processos de software;
3. Levantar, analisar e comparar abordagens utilizadas na descrição de processos de software em empresas de desenvolvimento de software;
4. Desenvolver uma abordagem para descrever o processo de software de uma microempresa de desenvolvimento de software que considere as dificuldades (requisitos) apresentadas pela empresa;
5. Descrever o processo de software de uma organização classificada como microempresa utilizando a abordagem proposta;
6. Analisar e discutir a aplicação da abordagem para descrição dos processos de software.

1.4 Metodologia

As seções subseqüentes apresentam a metodologia aplicada à pesquisa. A metodologia foi dividida em etapas que apresentam o procedimento aplicado na condução do presente trabalho.

1.4.1 Procedimento

A pesquisa é dividida em cinco etapas para atender aos objetivos propostos: pesquisa do perfil de microempresas para levantar as características e requisitos dessas microempresas em relação à modelagem de processo de software; estudo bibliográfico com o objetivo de levantar conceitos, terminologias e experiências relacionadas à modelagem de processo de software; desenvolvimento da abordagem para descrição dos processos de software direcionado a uma microempresa de desenvolvimento de software amparado nos requisitos (pesquisa) e experiências apresentadas na literatura; aplicação da abordagem para descrição dos processos de software e análise e avaliação dos dados resultantes da aplicação da abordagem proposta.

Etapa 1: Pesquisa do perfil de microempresas

Esta etapa consiste na elaboração de um instrumento de coleta de dados (anexo A) na forma de um questionário para levantar dados sobre a situação atual em relação ao uso de modelos de processos de software em microempresas. A estrutura do questionário foi baseada na abordagem GQM (*Goal/Question/Metric*) (BASILI, 1992; BASILI & WEISS, 1984) a qual permitiu definir aspectos importantes que deveriam ser considerados e mensurados em microempresas de software no que se refere à modelos e modelagem de processos de software adequados ao perfil dessas empresas. O público-alvo da pesquisa consistiu de seis empresas de desenvolvimento de software que atendiam ao critério de microempresas (de 1 a 9 empregados) de acordo com o porte referente à força de trabalho efetiva definidos pelo Ministério de Ciência e Tecnologia – MCT (MCT/SEPIN, 2002, p. 16). Foram pesquisadas, empresas situadas em Florianópolis – Santa Catarina e em Londrina – Paraná. A seleção das microempresas de desenvolvimento de software para aplicação do instrumento de coleta de dados ocorreu de forma aleatória. Após a aplicação do instrumento de coleta de dados, os dados foram analisados e interpretados qualitativa e quantitativamente segundo a abordagem GQM (BASILI, 1992; BASILI & WEISS, 1984).

Etapa 2: Estudo bibliográfico

Nesta etapa foram identificados por meio de levantamento bibliográfico os diversos conceitos pertinentes à modelagem de processos de software com o objetivo de formar um vocábulo padronizado, apresentar divergências terminológicas entre os principais pesquisadores em processo de software e subsidiar a compreensão dos conceitos presentes no âmbito dessa área de pesquisa. Para atender aos requisitos dessa etapa, algumas questões foram listadas para conduzir o levantamento bibliográfico:

- a) Quais os principais conceitos relacionados à descrição de processo de software?
- b) Quais são as principais terminologias aplicadas ao processo de software?
- c) Na literatura existem abordagens para descrever os processos de software? Se existirem, são voltados para empresas de pequeno porte?
- d) A definição dos processos é uma prática considerada importante ao desenvolver e/ou manter produtos de software?
- e) Existem relatos de experiências relacionadas ao desenvolvimento e aplicação de abordagens para descrição de processo de software?
- f) Existem ferramentas automatizadas de suporte à descrição de processo de software?

Etapa 3: Desenvolvimento da abordagem para descrição dos processos de software direcionado a uma microempresa

O desenvolvimento da abordagem foi subsidiado pelos requisitos levantados na pesquisa do perfil de microempresas conduzida na etapa 1 e, por experiências relatadas na literatura. Inicialmente, a abordagem deveria prover uma estrutura organizada por etapas e, para cada etapa, fases que auxiliariam a sua aplicação. Para cada fase, definiram-se passos que pormenorizaram a sua execução estabelecendo sua seqüência, identificação, objetivo, finalidade e como deveriam ser executados. Para atender a essas especificações, foram adotados os seguintes passos:

1. Após a aplicação do instrumento de coleta de dados definido na etapa 1 do procedimento, os dados resultantes foram analisados qualitativa e quantitativamente com a finalidade de averiguar as necessidades de microempresas em relação à descrição de processos de software. Como resultado, foram levantados alguns requisitos que a abordagem deveria contemplar para atender às particularidades de uma microempresa em relação à descrição de seu processo de software;
2. No levantamento bibliográfico, foram pesquisados estudos que abordavam propostas e experiências aplicadas à descrição de processos de software. Preferencialmente, considerou-se a literatura que apresentava soluções voltadas às microempresas. No entanto, o estudo bibliográfico apontou apenas abordagens aplicadas a empresas de médio e grande porte. Como resultado desse passo, levantou-se cinco abordagens para descrição de processos de software tomadas como referência para o desenvolvimento da abordagem proposta neste trabalho;
3. O passo 3 consistiu no desenvolvimento da abordagem com base nos requisitos resultantes do passo 1 e abordagens para descrição de processos de software levantados no passo 2. O objetivo desse passo, portanto, foi adequar a abordagem proposta nesse trabalho aos requisitos levantados no passo 1 e, ainda, agregar experiências práticas relatadas na literatura em empresas de desenvolvimento;
4. Finalmente, no passo 4, foi definida uma notação gráfica para os principais elementos da abordagem: etapas, fases, passos e fluxos entre os elementos com o objetivo de facilitar sua compreensão.

Etapa 4: Aplicação da abordagem para descrição de processo de software

Nesta etapa foi conduzido um estudo de caso em uma microempresa situada na cidade de Londrina – Paraná, na qual a abordagem foi aplicada. A seleção da empresa obedeceu aos critérios: porte (microempresa), participação na pesquisa conduzida na etapa 1 da metodologia e, ainda, condições para condução do estudo de caso, por exemplo: tempo para reuniões com os participantes na descrição do processo, permitir acesso a suas dependências, acesso a documentos relacionados ao processo de software e permitir que fossem realizadas entrevistas junto aos empregados relacionados direta ou indiretamente aos processos de software da empresa.

Inicialmente, realizou-se um diagnóstico da empresa selecionada antes da aplicação da abordagem proposta nesse trabalho. O objetivo do diagnóstico era investigar a situação da empresa selecionada em relação ao processo de software. O diagnóstico foi conduzido por meio de entrevistas junto aos empregados, observação da execução dos processos de software da empresa, análise de documentos consumidos e produzidos durante a execução do processo de software, identificação de problemas relacionados à falta de descrição dos processos de software, identificação de necessidades da empresa que não eram atendidas em decorrência da falta de descrição do processo de software.

Os procedimentos e recursos necessários à aplicação da abordagem serão descritos detalhadamente no capítulo que apresenta a abordagem proposta neste trabalho. Portanto, é um dos objetivos da abordagem, possibilitar ao seu aplicador todas as informações necessárias à sua condução. A descrição do processo de software foi conduzida pelo pesquisador do presente trabalho apoiado pelos demais participantes dos processos de software da empresa. Foi definido um responsável na empresa pelo acompanhamento do estudo com o objetivo de transferir as experiências em descrição de processos de software.

Etapa 5: Avaliação da abordagem (análise de dados)

A comparação do processo de software real ao processo de software descrito tem sido a estratégia adotada por diversos estudos conduzidos em atividades relacionadas à modelagem de processo de software (HÖLTJE et al, 1994; MADHAVJI et al, 2001; BRIAND et al, 1998; BECKER-KORNSTAEDT, 2000; BECKER-KORNSTAEDT & BELAU, 2001; BECKER-KORSTAEDT, NEU & HIRCHE, 2001; MACHADO et al, 2001). Tal estratégia consiste na comparação do processo de software real (atividades executadas, artefatos consumidos, produzidos e manipulados, papéis desempenhados etc) em relação ao modelo ou guia de processo no qual está descrito. Alguns estudos (HÖLTJE et al, 1994; MADHAVJI et al, 1994) introduziram algumas métricas por meio do Paradigma GQM (embora os estudos apresentem apenas algumas métricas) com o objetivo de disciplinar as observações realizadas durante a

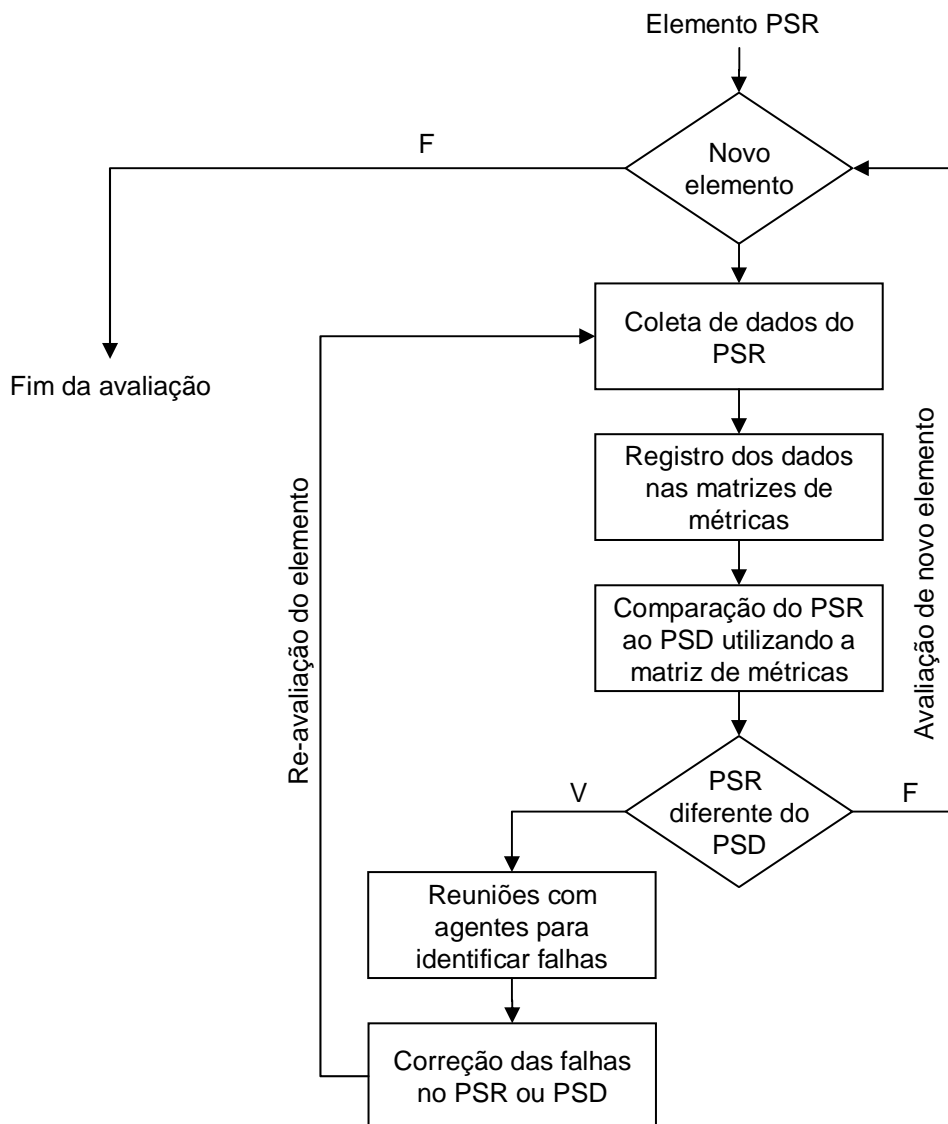
execução do processo e, deste modo, quantificar inconsistências oriundas dessas observações em relação ao processo de software descrito.

Considerando as experiências relatadas na literatura em relação à análise e discussão dos dados provenientes dos resultados da aplicação de abordagens para descrição do processo de software, foram estabelecidos os seguintes objetivos da: (a) averiguar a consistência entre o Guia de Processo de Software (GPS) e processo de software real e (b) verificar se a abordagem havia atendido aos requisitos (restrições) apresentados pela microempresa antes de iniciar a aplicação da abordagem. Para avaliar a fidelidade do GPS em relação ao processo de software real, foi aplicado o paradigma GQM (*Goal/Question/Metric*). Basicamente, foi definido um conjunto de questões e, para cada questão, foram identificadas métricas. O plano de métricas elaborado por meio do paradigma GQM considerou a relação bidirecional entre GPS e processo de software real e ainda inclui prováveis causas que poderiam motivar inconsistências nessa relação.

Com relação aos requisitos que deveriam ser atendidos pela abordagem, antes de iniciar a aplicação, foi elaborada uma lista baseada nos requisitos levantados na pesquisa do perfil de microempresas conduzida na etapa 1 e requisitos acrescentados pela microempresa na qual foi realizado o estudo. A lista apresentou, entre outros, os seguintes requisitos: elementos do processo de software que deveriam ser priorizados ao modelar o processo, tempo e período para realização de reuniões com todos os empregados, tempo e período para coleta de dados e falta de experiência dos empregados. Foram coletados dados relacionados aos respectivos requisitos durante todo o período da aplicação da abordagem até a versão final do GPS. A análise de dados desse item ocorreu por meio da comparação entre os parâmetros de cada um dos requisitos definidos pela empresa antes da aplicação da abordagem e os dados coletados sobre os mesmos durante a aplicação da abordagem.

Os seguintes passos foram adotados na condução da coleta e análise dos dados: 1) ao final da descrição de um elemento do processo de software, procedia-se a observação do processo de software real no qual eram aplicadas as métricas para averiguar a existência de inconsistências em relação ao processo de software descrito.

As métricas eram acondicionadas em matrizes que permitiam registrar diversos dados referentes ao processo de software em execução, por exemplo, número de atividades executadas, número de artefatos de entrada, número de artefatos de saída, número de artefatos manipulados, número de papéis desempenhados, número de ferramentas utilizadas etc; 2) após a coleta e registro dos dados do processo real na matriz de métricas, era realizada a comparação com o processo de software descrito, ou seja, se o processo de software real apresentava três artefatos de entrada em uma determinada atividade, o mesmo número de artefatos de entrada deveria constar no processo de software descrito; 3) após a comparação, constatado diferenças entre o processo real e o processo descrito, eram realizadas discussões entre os agentes envolvidos com o objetivo de averiguar se havia falhas na execução do processo (por exemplo, uma omissão do agente ao executar o processo) ou na descrição do processo de software (por exemplo, erro na descrição do processo de software seja por falha na coleta ou falha na descrição), caso não fossem observadas inconsistências entre o processo de software real e o processo de software real, novo elemento do processo de software real era avaliado aplicando novamente o passo 1; 4) identificadas as falhas, quando ocorriam, procedia-se a correção da descrição do processo de software ou correção na execução do processo de software real; 5) no caso da constatação de falhas, o processo de software real era novamente submetido a avaliação e os passos anteriores eram executados. Uma visão geral sobre o processo de avaliação da abordagem pode ser consultada na Figura 3.



PSR: Processo de Software Real
 PSD: Processo de Software Descrito

Figura 3: Processo de avaliação da abordagem.

1.5 Estrutura do Trabalho

Os demais tópicos do presente trabalho estão distribuídos em sete capítulos:

Capítulo 2: apresenta os principais fundamentos conceituais relacionados a processos de software. Deste modo, o capítulo apresenta conceitos, terminologias e experiências relacionadas a processos de software, comparações de terminologias entre diferentes autores e exemplos dos principais termos.

Capítulo 3: discute a modelagem de processos de software, linguagens e paradigma aplicados à modelagem de processos de software, meios utilizados para representar os modelos de processo de software, abordagens e experiências relatadas na literatura para descrição de processos de software, comparação entre as abordagens e experiências. Além disso, discute necessidades de microempresas em relação à modelagem de processos de software.

Capítulo 4: apresenta a abordagem iterativa e incremental para descrição de processos de software. Ainda, são descritos: estrutura da abordagem, estratégias utilizadas para elaborar a abordagem, etapas, fases e passos empregados na sua aplicação. Também é relatado o estudo de caso conduzido em uma microempresa na qual a abordagem foi aplicada. As etapas e fases da abordagem são pormenorizadas. Reformulações e a versão final do Guia de Processo de Software são apresentadas. Ao final, o resumo dos principais indicadores do estudo de caso.

Capítulo 5: avalia e analisa a aplicação da abordagem. São relatados os critérios utilizados para avaliar a abordagem: métricas definidas pela aplicação do paradigma GQM e, comparação entre as necessidades da microempresa em relação à modelagem de processo de software e atendimento desses requisitos pela abordagem.

Capítulo 6: discute a abordagem proposta neste trabalho em relação às abordagens relatadas na literatura. Aspectos comuns e distintos são tratados.

Capítulo 7: apresenta as considerações finais e trabalhos futuros.

2 CONCEITOS FUNDAMENTAIS

As pesquisas em processos de software são relativamente novas e, conseqüentemente, as terminologias utilizadas normalmente são conflitantes, redundantes e, em muitas situações, diferentes autores definem o mesmo objeto com diferentes terminologias. Este capítulo tem por objetivo apresentar os conceitos básicos relacionados a processos de software, diferentes interpretações e padronizar as terminologias empregadas nesta pesquisa. As definições empregadas neste trabalho serão apresentadas nos tópicos “o que é (são)”. As definições terminológicas apresentadas na literatura serão apresentadas no tópico “definição de terminologia” e, finalmente, quando oportuno, serão apresentado exemplos dos termos e conceitos apresentados.

2.1 O que são processos de software?

Processos de software são as diversas fases necessárias para produzir e manter um produto de software. Requerem a organização lógica de diversas atividades técnicas e gerenciais envolvendo agentes, métodos, ferramentas, artefatos e restrições que possibilitam disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software.

Definição de terminologia: processo de software

HUMPHREY & KELLNER (1989, p. 1) definem processos de software como sendo “a prática adotada por uma organização de desenvolvimento de software onde são aplicados métodos, técnicas, artefatos, ferramentas, agentes e etc para desenvolver e/ou manter um produto de software ou serviço”. De acordo com FEILER & HUMPHREY (1993, p. 31), um processo de software “é um conjunto de passos ordenados parcialmente para alcançar um objetivo”. LONCHAMP (1993, p. 4) define processos de software como sendo “um conjunto de passos ordenados parcialmente, que

relaciona um conjunto de artefatos, humanos, recursos computadorizados, estrutura organizacional e restrições, pretendidas para produzir ou manter um software”.

Exemplos de processos de software

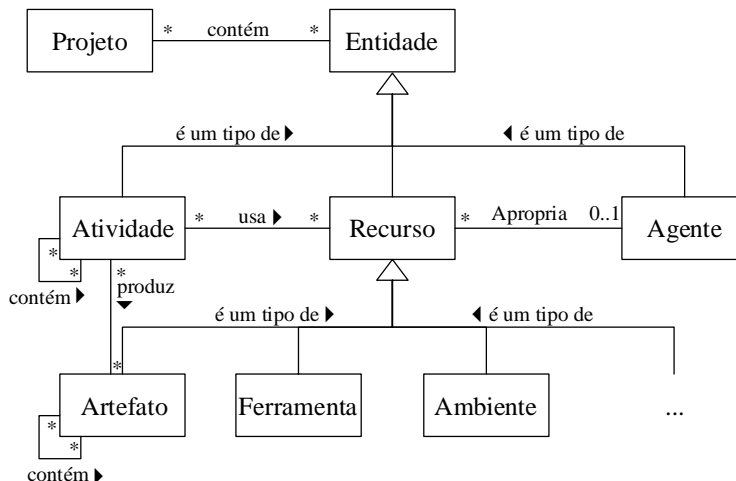
Pode-se considerar como exemplo de processo de software a determinação e especificação de sistemas e requisitos de software, análise e gerenciamento de riscos, prototipação de software, projeto, implementação, validação e verificação, controle e garantia da qualidade de software, integração de componentes, documentação, gerenciamento de configuração de versões de software, gerenciamento de dados, evolução de software, gerência de projeto etc (LONCHAMP, 1993, p. 4). Outro exemplo de processo de software é citado por SOMMERVILLE (1996): especificação de software, desenvolvimento, validação e evolução. PRESSMAN (2001) exemplifica processo de software por intermédio das seguintes atividades: definição, desenvolvimento e suporte.

2.2 Quais são os principais componentes de um processo de software?

Nesta seção, serão abordados os principais componentes de um processo de software e seus relacionamentos (Figura 4): atividades, agentes, recursos (artefatos, ferramentas) e; os principais conceitos relacionados a processos de software pertinentes a esta pesquisa: processo de software, modelo de processo de software, ciclos de vida. Processo de software refere-se a todas as atividades, bem como relacionamentos, artefatos, ferramentas, papéis etc, necessárias para construir, entregar e manter um produto de software. Já o ciclo de vida apresenta uma representação alto nível do processo de software executado (processo de software real) ou como deveria ser executado, ou seja, normalmente, ciclos de vida determinam as fases e o relacionamento entre as fases. Modelo de processo de software, diferentemente do ciclo de vida, representa o processo de software executado ou como deveria ser executado em um nível de abstração mais aprofundado que o ciclo de vida. Portanto, modelos de processo de software são representações que, além da representação de fases e relacionamento entre fases, descrevem as atividades executadas em cada fase e seus

relacionamentos, artefatos consumidos, produzidos e consumidos pelas atividades, papéis desempenhados, ferramentas utilizadas etc.

Figura 4: Componentes de um processo de software



Fonte: BECKER-KORNSTAEDT & WEBBY (1998, p. 5)

2.2.1 O que são atividades?

Atividades ou tarefas definem “como as coisas são feitas” ou “o que será feito” e para isso incorporam procedimentos, regras, políticas, agentes, recursos, papéis, artefatos (consumidos e produzidos). Assim, atividades são executadas por agentes e agentes executam as atividades, atividades usam e produzem artefatos e artefatos são usados e produzidos por atividades, atividades usam recursos e recursos são usados por atividades etc. Cada atividade deve definir claramente seu início e final.

Normalmente, atividades são organizadas em uma rede de duas dimensões: horizontal e vertical. A dimensão horizontal define os relacionamentos entre as atividades e a dimensão vertical decompõe a atividade em diversos níveis, ou seja, uma atividade composta por diversas outras sub-atividades (BECKER-KORNSTAEDT & WEBBY, 1998, p. 6). Exemplo de sub-atividades seria a decomposição da atividade gerenciamento de projeto nas sub-atividades: estimativas de tamanho, estimativas de custo, controle e acompanhamento.

Definição de terminologia: atividade

“Atividade é um passo do processo que produz mudanças de estado visíveis externamente no produto de software” (CONRADI, FERNSTRÖM & FUGGETTA, 1993, p. 5). LONCHAMP (1993, p. 44) define atividade como “um passo de processo elementar”. FEILER & HUMPHREY (1993, p. 31) definem atividade como sendo “um passo de processo, limitado, com uma duração finita e seu nível de abstração depende do contexto no qual é executada”.

Exemplos de atividades

Exemplos de atividades seriam: gerenciamento de projeto, teste de código, elicitação de requisitos, implementação, etc.

2.2.2 O que são recursos?

Recursos são entidades estáticas necessárias para executar uma atividade. A não utilização ou indisponibilidade de recursos necessários pode causar falha na execução de uma atividade ou até mesmo impedir sua execução. Recursos relacionam-se com diversos componentes do processo de software, por exemplo, atividades e agentes. Deste modo, uma atividade pode utilizar diversos recursos e um recurso pode ser utilizado por diversas atividades ou, um agente pode utilizar diversos recursos e um recurso pode ser utilizado por diversos agentes. Artefatos, ferramentas, equipamentos, ambientes (sala de reuniões, laboratórios etc) são considerados como tipos especializados (derivados) de recursos.

Definição de terminologia: recurso

BECKER-KORNSTAEDT & WEBBY (1998, p. 7) definem recursos como “descrição de uma classe abstrata de qualquer entidade estática a qual, quando não disponível, pode causar uma falha no projeto”. LONCHAMP (1993, p. 45) define

recurso como “um bem necessário para executar uma atividade”. KELLNER et al (1998, p. 5) definem recursos como “descrições de programas de computadores ou outros auxílios os quais podem ser usados para suporte ou automação de uma atividade”.

Exemplos de recursos

São exemplos de recursos: equipamentos, artefatos, ambientes e ferramentas.

2.2.3 O que são artefatos?

Artefato é um tipo de recurso produzido ou consumido em uma atividade. Nesse contexto, um artefato pode ser utilizado como uma entrada (matéria-prima) para uma determinada atividade e como uma saída de uma atividade (resultado da execução de uma atividade). Um artefato pode estar associado a diversas atividades e uma atividade pode estar associada a diversos artefatos. Artefatos não são tangíveis. Em um contexto de projeto de software, alguns artefatos existem apenas na forma de um arquivo lógico (arquivo executável, por exemplo). Geralmente, artefatos são persistentes e as possíveis modificações sofridas por um artefato podem produzir versões de um mesmo artefato (versionados). Artefatos podem ser representados como um agregado de sub-artefatos (módulos de código-fonte) ou artefatos mais complexos podem ser decompostos em sub-artefatos (Projeto Procedimental).

Definição de terminologia: artefato

KELLNER et al (1998, p. 5) definem artefatos como “descrições de um produto criado ou modificado durante a execução do processo ou, como o resultado final ou intermediário do processo ou, ainda, como resultado temporário, sendo que o resultado de uma atividade poderá ser utilizado por outras atividades”. CONRADI, FERNSTRÖM & FUGGETTA (1993, p. 5) definem artefatos como “(sub-) produtos e

matéria-prima de um processo”. LONCHAMP (1993, p. 45) define artefato como um “produto criado ou modificado durante um processo como um resultado requerido ou para facilitar um processo”.

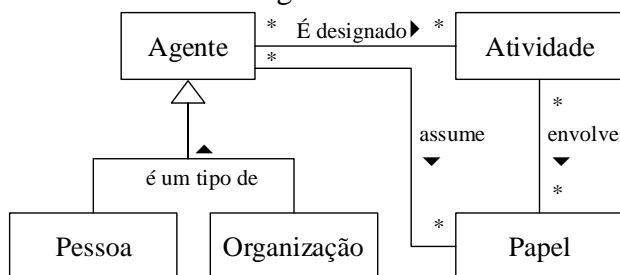
Exemplos de artefatos

Pode-se citar como exemplos de artefatos: código-fonte, código executável, manual de padrões, relatório de resultados, documento de requisitos, plano de trabalho.

2.2.4 O que são agentes?

Agentes ou atores são as entidades que executam atividades por intermédio de um papel (Figura 5). Essa entidade pode ser um pessoa ou grupo de pessoas. Geralmente, agentes estão relacionados a papéis e atividades, assim, um agente pode executar diversos papéis e um papel pode estar relacionado a diversos agentes. Atividades envolvem diversos agentes e um agente pode participar de diversas atividades.

Figura 5: Relacionamentos da entidade agente



Fonte: Adaptado de BECKER-KORNSTAEDT & WEBBY (1998, p. 9)

Alguns autores classificam ferramentas como um tipo de agente (LONCHAMP, 1993, p. 44; FEILER & HUMPHREY, 1993, p. 34). Outros autores (CONRADI, FERNSTRÖM & FUGGETTA, 1993, p. 6; BECKER-KORNSTAEDT & WEBBY, 1998, p. 6; KELLNER et al, 1998, p. 5). BECKER-KORNSTAEDT & WEBBY (1998) não classificam ferramentas como agentes por considerarem que ferramentas não são capazes de iniciarem uma atividade ou julgar se uma atividade foi

completada com sucesso ou não. Os autores consideram ainda que ferramentas sejam tipos de recursos necessários a execução de uma atividade e pelo fato de derivarem de recursos, são entidades estáticas. CONRADI, FERNSTRÖM & FUGGETTA (1993) e KELLNER et al (1998) definem agentes como indivíduos ou um grupo de indivíduos que executam uma atividade. Nessa pesquisa será utilizada a definição empregada por esses últimos autores.

Definição de terminologia: agente

FEILER & HUMPHREY (1993, p. 34) definem agente como uma “entidade que executa um processo definido”. LONCHAMP (1993, p. 44) define agente como o “executor de um processo podendo ser um humano ou uma ferramenta computadorizada”. CONRADI, FERNSTRÖM & FUGGETTA (1993, p. 6) definem agentes como sendo “humanos ou uma determinada pessoa, que executam uma atividade relacionada a um papel”.

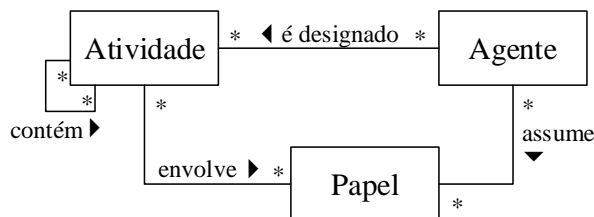
Exemplos de agentes

São considerados exemplos de agentes: programador “fulano”, equipe de programação “Alfa”, gerente de projeto “beltrano”, ferramenta “abc”.

2.2.5 O que são papéis?

Papéis representam um conjunto de responsabilidades, obrigações, permissões e habilidades necessárias para executar uma atividade ou sub-atividade. Geralmente, papéis são desempenhados por agentes humanos (Figura 6). Um sinônimo de papel seria cargo. Uma atividade pode exigir diversos papéis para ser executada e um papel pode ser aplicado em diversas atividades. Agentes podem assumir diversos papéis durante o desenvolvimento de software e um papel pode ser assumido por diversos agentes durante o desenvolvimento de software.

Figura 6: Relacionamentos da entidade papel



Fonte: Adaptado de KELLNER et al (1998, p. 5)

Definição de terminologia: papel

KELLNER et al (1998, p. 5) definem papel como um “conjunto de obrigações e permissões para executar uma atividade relacionada”. “Um papel descreve um conjunto de responsabilidades, direitos e habilidades necessárias para realizar uma atividade específica no processo de software” (CONRADI, FERNSTRÖM & FUGGETTA, 1993, p. 6). LONCHAMP (1993, p. 45) define papel como um “conjunto de permissões e obrigações associadas a um objetivo funcional”. Ainda, FEILER & HUMPHREY (1993, p. 37) caracterizam papel como “responsabilidade para executar um processo ou sub-processo”.

Exemplos de papel

Alguns exemplos de papéis seriam: gerente de projeto, projetista de banco de dados, programador, analista de sistemas, gerente de qualidade.

2.3 O que são Modelos de Ciclo de Vida de Software?

Modelos de ciclo de vida descrevem como um software deve ser desenvolvido. Basicamente definem a ordem global das atividades envolvidas em um contexto de projeto de software e propõe uma estratégia de desenvolvimento que pode ser aplicada a um determinado contexto de projeto de software. Modelos de ciclo de vida normalmente são vagos nas descrições de detalhes das condições de início e término de uma atividade, recursos utilizados, artefatos consumidos ou produzidos, papéis desempenhados etc.

Definição de terminologia: modelo de ciclo de vida de software

LONCHAMP (1993, p. 46) define modelo de ciclo de vida como um “modelo de processo de software semi-formal e muito abstrato”. SCACCHI (2001, p. 3) define modelo de ciclo de vida como um “modelo descritivo ou prescritivo de como um software é ou deveria ser desenvolvido”. A norma NBR ISO 12207 (NBR ISO/IEC 12207:1998, p. 3) define modelo de ciclo de vida como a “estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema desde a definição de seus requisitos até o término do seu uso”.

Exemplos de modelo ciclo de vida de software

Dentre os diversos modelos de ciclo de vida de software, pode-se citar: Cascata, Espiral, Prototipação Evolutiva, Prototipação Incremental, Prototipação Descartável, Refinamento Iterativo, Ciclo de Vida Progressivo (McCONNEL, 1996; TURPIN, 1996; SCACCHI, 2001).

Diferenças entre modelos de ciclo de vida

Ainda que muitos modelos de ciclo de vida sejam semelhantes entre si, existem alguns aspectos que os diferenciam (McCONNEL, 1996, p. 133-134):

1. Enfoque dado pelo modelo. Por exemplo, no Modelo Cascata o enfoque é dado na documentação e no Modelo Espiral o enfoque é dado nos riscos;
2. Estratégia de desenvolvimento. Define a disposição das atividades que deverão ser executadas para atingir um objetivo em um contexto de projeto de desenvolvimento de software. A disposição das atividades pode ser, por exemplo, linear (uma atividade após a outra) como no ciclo de vida Cascata puro ou iterativa (um conjunto de atividades é repetida várias vezes até atingir o seu objetivo) como nos modelos incrementais. Outras estratégias podem envolver a disposição das atividades em paralelo, a prototipação ou reunir as características de modelos de ciclo de vida lineares e iterativos.

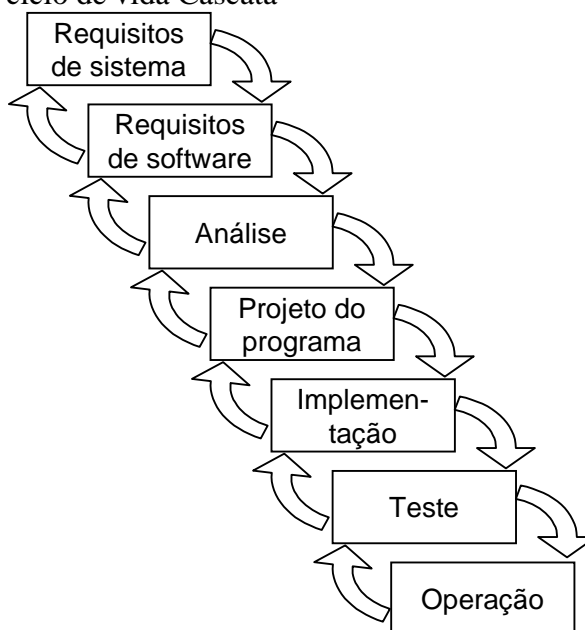
2.3.1 Modelos de ciclo de vida do software

Nesta seção serão brevemente resumidos alguns modelos de ciclo de vida mais conhecidos, suas características, vantagens e desvantagens: Cascata Puro, Espiral, Prototipação Incremental, Prototipação Evolutiva e Prototipação Rápida Descartável.

2.3.2 Modelo Cascata Puro

A proposta do modelo Cascata Puro (Figura 7) consiste na execução das atividades de desenvolvimento de software em uma seqüência ordenada. Desta forma, a passagem para determinada atividade exige como critério a finalização da atividade imediatamente anterior. As principais atividades do modelo são: requisitos de sistema, requisitos de software, análise, projeto de programa, codificação, teste e operação (ROYCE, 1970).

Figura 7: Modelo de ciclo de vida Cascata



Fonte: Adaptado de ROYCE (1970)

O modelo Cascata é recomendado para projetos nos quais há domínio dos requisitos do sistema que será desenvolvido e quando o pessoal envolvido no projeto é fraco tecnicamente (McCONNEL, 1996, p. 137).

No entanto, o modelo apresenta algumas desvantagens (HUMPHREY & KELLNER, 1989, p. 3-4):

1. Propõe uma seqüência entre etapas que não representa adequadamente um processo de desenvolvimento de software;
2. Não oferece suporte adequado para mudanças que são percebidas durante o processo e que requerem modificações em etapas anteriores (flexibilidade);
3. Não oferece facilidades para acomodar tecnologias recentes como prototipação rápida e,
4. Fornece poucos recursos para otimização de processo (detalhamento);

Devido a essas desvantagens, a utilização do modelo depende do contexto e pode resultar nas seguintes conseqüências (HUMPHREY & KELLNER, 1989, p. 3):

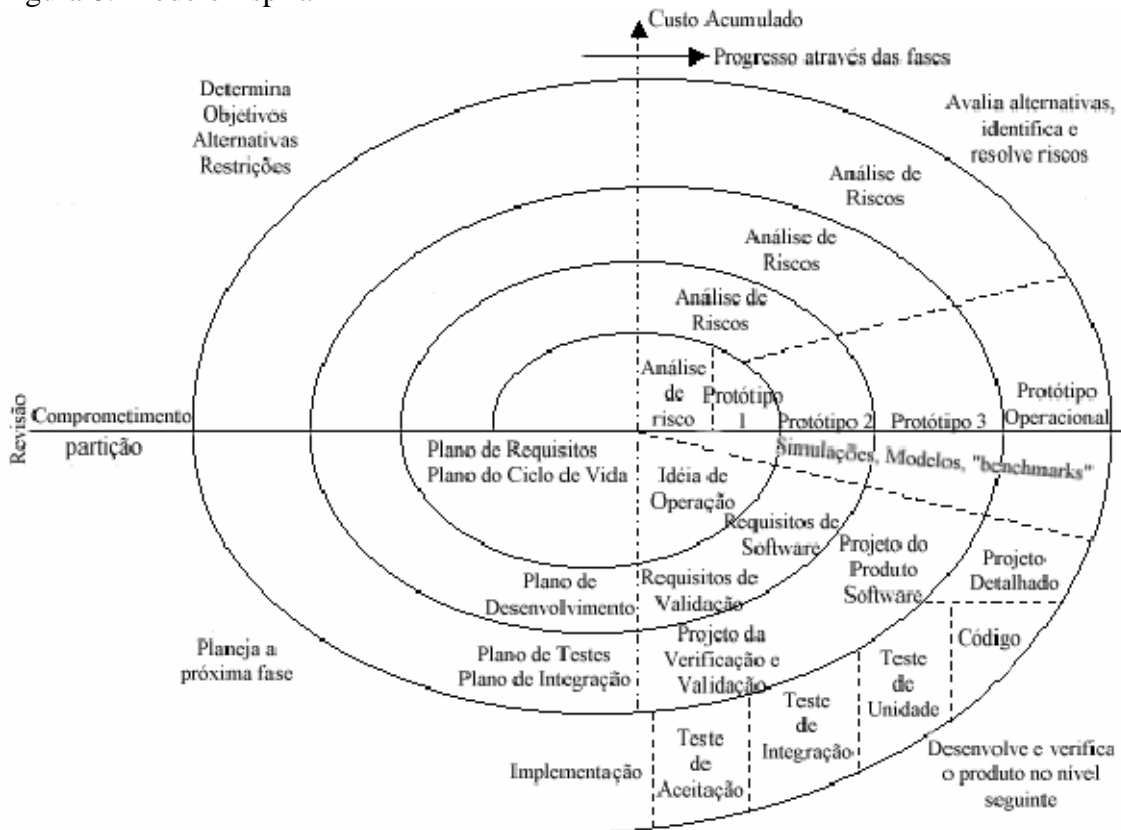
1. Pode não permitir a visão real do processo em andamento;
2. O modelo cascata, analogamente, cria dois universos, um universo se refere ao processo em andamento e o outro universo se refere às mudanças que deveriam ser aplicadas ao produto, mas não as são pelo fato do modelo não incorporar em sua dinâmica a revisão de etapas já concluídas durante o seu andamento;
3. Não é possível mensurar. O fato de normalmente não permitir uma visão real do processo também implica em uma visão irreal para a aplicação de métricas. Atividades não concluídas são rotuladas como concluídas.

2.3.3 Modelo Espiral

O modelo Espiral (Figura 8), proposto por Barry Boehm, reúne características dos modelos Cascata e Prototipação acrescentando ainda em sua base a análise de riscos. Cada giro na espiral (iniciando a partir do centro e avançando para fora) representa uma nova fase do processo. Esse processo evolutivo permite que novas versões possam ser construídas progressivamente. Tipicamente, o modelo pode ser dividido em 3 ou 6 regiões. PRESSMAN (2001) apresenta o modelo dividido em 6

regiões: comunicação com o cliente, planejamento, análise de riscos, engenharia, construção e evolução e, avaliação do cliente. O número de tarefas por regiões pode variar conforme o tamanho e complexidade do projeto. Desta forma, o modelo pode ser adaptado conforme as características do projeto.

Figura 8: Modelo Espiral



Fonte: Adaptado de BOEHM (2000, p. 2)

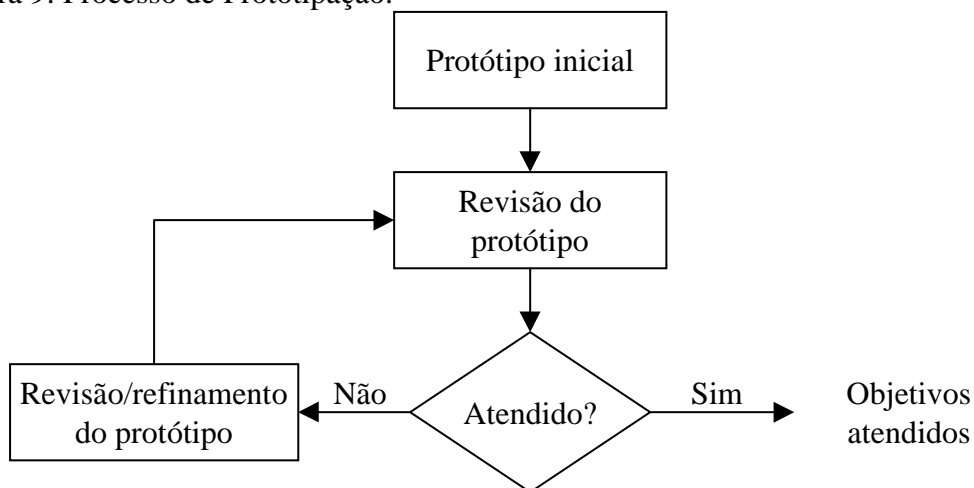
A adoção do modelo Espiral proporciona algumas vantagens. Teoricamente, quanto mais tempo e recursos forem destinados, ou seja, quanto mais iterações na espiral, menor será os riscos sobre o projeto. Outra vantagem em relação a modelos seqüenciais como o modelo Cascata, é a execução de atividades de verificação presentes ao final de cada iteração que permitem um melhor controle gerencial sobre o projeto. Uma desvantagem do modelo refere-se a situações em que o projeto é considerado simples e os riscos são modestos. Nesse contexto, muitos projetos não precisam da flexibilidade e gerenciamento de riscos proporcionados pelo modelo (McCONNEL, 1996, p. 143).

2.3.4 Prototipação

Organizações de desenvolvimento de software utilizam protótipos na construção de software com diferentes propósitos: na construção de modelos, simulações, implementações parciais ou ainda para testar aspectos técnicos de um sistema. A prototipação pode ser adotada como uma abordagem que compreenda todo o ciclo de vida de desenvolvimento de um software ou como um processo incorporado a um ciclo de vida (por exemplo, Modelo Espiral).

O processo de prototipação (Figura 9) consiste basicamente em diversos ciclos iterativos. Um protótipo é construído a partir de requisitos iniciais. É realizada uma avaliação crítica do protótipo a qual considerada os requisitos iniciais e requisitos que não foram mencionados inicialmente. Caso o protótipo não atenda aos requisitos pretendidos, novas iterações são realizadas produzindo novos protótipos. As iterações são finalizadas quando os requisitos forem atendidos.

Figura 9: Processo de Prototipação.



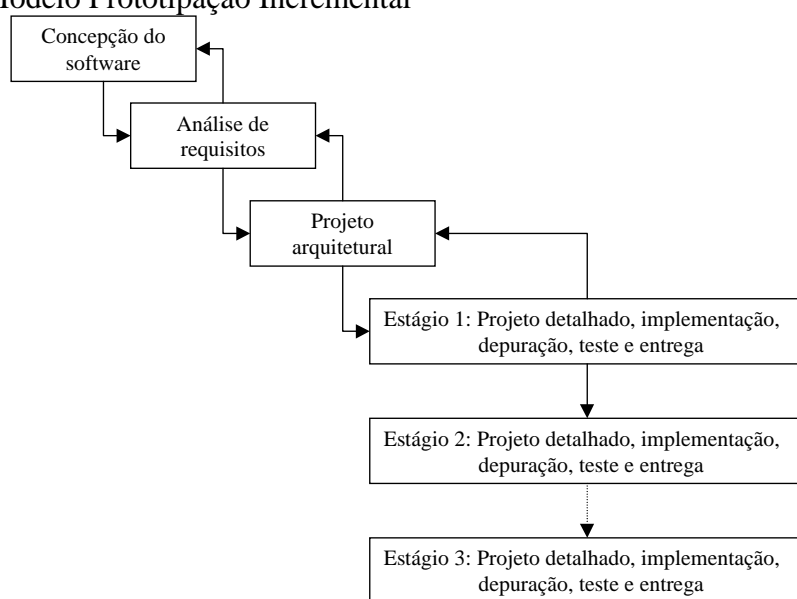
Fonte: Adaptado de CARR & VERNER (1997)

Existem diversos modelos de processo de desenvolvimento de software que empregam a prototipação em seus processos: Incremental Iterativo, Prototipação Rápida Descartável, Prototipação Evolutiva e Espiral (CARR & VERNER, 1997).

Prototipação Incremental

A Prototipação Incremental (Figura 10) também conhecida como Entrega por Estágio (McCONNEL, 1996) adota como estratégia o desenvolvimento por estágios. Normalmente os requisitos mais importantes são implementados primeiro e os demais são acrescentados em novas versões. O desenvolvimento ocorre gradualmente e, ao final de cada estágio, uma versão operável é produzida e incrementada nos demais estágios até a sua conclusão final.

Figura 10: Modelo Prototipação Incremental



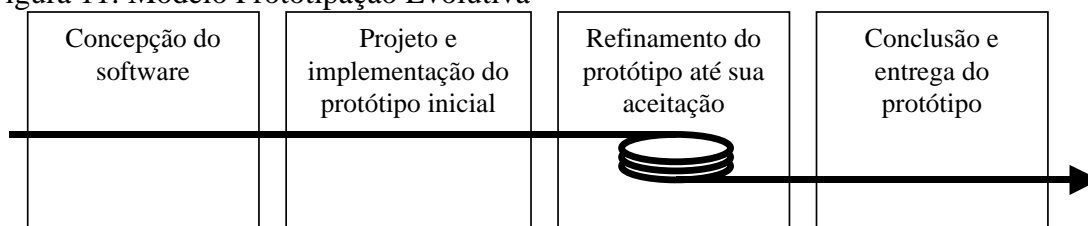
Fonte: Adaptado de McCONNEL (1996)

A Prototipação Incremental pode oferecer diversas vantagens: redução de riscos, maior visibilidade sobre o processo, problemas podem ser descobertos logo no início, auxilia na estimativa de tempo do projeto entre outras. No entanto, o emprego desse modelo exige grande esforço na atualização da documentação do usuário. Além disso, o desenvolvimento por estágio requer que as dependências entre os estágios sejam bem planejadas, pois um dos problemas comuns é descobrir que o estágio 2 depende de componentes do estágio 4, por exemplo.

Prototipação Evolutiva

A Prototipação Evolutiva (Figura 11) basicamente inicia com a concepção do sistema e emprega diversos ciclos de “re-projeto”, “re-implementação” e “re-avaliação” até a aceitação final do software. Cada novo protótipo confirma, refina ou adiciona novos requisitos até atingir a totalidade de requisitos do sistema.

Figura 11: Modelo Prototipação Evolutiva



Fonte: Adaptado de McCONNEL (1996)

O modelo pode ser adotado em contextos onde os requisitos são instáveis ou desconhecidos ou, ainda, quando há a necessidade de se testar um algoritmo antes de sua adoção. Dentre as vantagens proporcionadas pelo modelo Prototipação Evolutiva, destaca-se menor taxa de defeitos pela melhor definição das especificações do sistema, promove maior participação do cliente, permite maior visibilidade do progresso de desenvolvimento e é possível verificar a aceitação do sistema nos primeiros estágios do desenvolvimento. Por outro lado, o modelo pode agregar algumas desvantagens: pobre manutenibilidade provocada pela rapidez no desenvolvimento e/ou constantes mudanças na estrutura do protótipo, acréscimo de novas funcionalidades devido ao constante contato do cliente com o protótipo e, uso ineficiente do tempo destinado à construção do protótipo.

Diferentemente da Prototipação Incremental, o protótipo é entregue ao finalizar o desenvolvimento. Além disso, a Prototipação Incremental desenvolve partes do software que serão entregues à medida que são finalizadas, não ocorrendo mudanças nas especificações do projeto e, a maior parte dos requisitos deve ser conhecida logo no início do projeto.

2.3.5 Seleção de modelos de ciclo de vida

A escolha de um modelo de ciclo de vida deve considerar as características do contexto de projeto ao qual será aplicado. Diferentes projetos de desenvolvimento possuem diferentes necessidades que devem orientar a seleção do ciclo de vida mais adequado. McCONNEL (1996, p. 154), considerando as diferenças entre projetos de desenvolvimento, apresenta diversas questões que devem ser respondidas ao selecionar um modelo de ciclo de vida:

1. Qual o nível de compreensão do usuário e desenvolvedores em relação aos requisitos no início do projeto? É provável que a compreensão mude significativamente durante o desenvolvimento do projeto?
2. Qual o nível de compreensão dos desenvolvedores em relação a arquitetura do sistema? É provável que mudanças na arquitetura do sistema ocorram no meio do caminho?
3. Qual nível de confiança é necessário?
4. O quanto é necessário planejar e projetar durante o projeto prevendo mudanças em versões futuras?
5. Qual o nível de riscos implícitos no projeto?
6. Pode ser limitado em um cronograma?
7. É necessário habilidade para realizar correções no meio do projeto?
8. É necessário mostrar ao cliente o progresso durante o projeto?
9. É necessário demonstrar ao usuário aspectos gerenciais durante o projeto?

Para auxiliar na resposta a essas questões, McConnel apresenta uma tabela na qual são apresentadas as vantagens e desvantagens de diversos modelos de ciclo de vida em relação às questões apresentadas acima. Outras questões foram incluídas e possuem igual importância na escolha de um modelo de ciclo de vida para o projeto. O Quadro 1 apresentado nessa pesquisa não inclui todos os modelos de ciclo de vida abordados por McConnel. Os modelos de ciclo de vida constantes nessa tabela incluem apenas alguns modelos discutidos na seção anterior e abordados pelo autor e,

seu objetivo, é apenas exemplificar aspectos que devem ser considerados na seleção de um modelo de ciclo de vida demonstrando suas vantagens e desvantagens em relação a algumas questões que devem ser consideradas no momento da seleção.

McConnel utiliza uma escala para classificar os modelos que varia entre “deficiente”, “bom” e “excelente”. Em alguns casos, o autor refina a classificação utilizando mais de um valor, por exemplo, “bom para excelente”. Deve-se ressaltar ainda que, a seleção não se restringe somente pela consideração das vantagens oferecidas pelo modelo de ciclo de vida frente a um contexto de projeto, devem ser considerados outros aspectos como a maneira como o ciclo de vida será aplicado.

Quadro 1: Modelos de ciclo de vida: vantagens e desvantagens

Capacidade do modelo	Cascata Puro	Espiral	Prototipação Evolutiva	Prototipação Incremental
Trabalha com a compreensão deficiente dos requisitos	Deficiente	Excelente	Excelente	Deficiente
Trabalha com a compreensão deficiente da arquitetura	Deficiente	Excelente	Deficiente para bom	Deficiente
Produz sistemas de alta confiança	Excelente	Excelente	Bom	Excelente
É fácil modificar o sistema em versões futuras	Excelente	Excelente	Excelente	Excelente
Gerencia riscos	Deficiente	Excelente	Bom	Bom
Pode ser limitado em um cronograma predefinido	Bom	Deficiente	Deficiente	Bom
Permite correções no meio do projeto	Deficiente	Bom	Excelente	Deficiente
Possibilita ao cliente visibilidade do progresso	Deficiente	Excelente	Excelente	Bom
Possibilita ao cliente visibilidade do progresso do gerenciamento	Bom	Excelente	Bom	Excelente
Requer pouca gerência ou experiência para usá-lo	Bom	Bom	Deficiente	Bom

Fonte: Adaptado de McCONNEL, 1996, p. 156-157.

2.4 O que são Modelos de Processo de Software?

Modelos de processo de software são representações formais ou semiformais de processos de software que capturam necessidades existentes dentro de um contexto de projeto de software, considerando suas particularidades e relevâncias, provendo um meio para representar os processos aplicados (modelos descritivos) e/ou prever as ações necessárias à execução desses processos (modelos prescritivos).

O modelo de processo de software basicamente atua provendo mecanismos que permitem representar as práticas de Engenharia de Software que devem ser aplicadas em um determinado contexto de software e, definindo uma infraestrutura básica que dê suporte aos processos de software presentes nesse modelo contribuindo assim, com maior controle e gerenciamento das atividades presentes em um projeto de software.

Definição de terminologia: modelo de processo de software

CURTIS, KELLNER & OVER (1992, p. 76) definem modelo de processo de software como “uma descrição abstrata do processo atual ou proposto que representa elementos de processos selecionados que são considerados importantes para o modelo e pode ser executado por um homem ou máquina”. LONCHAMP (1993, p. 5), descreve modelos de processo como “uma descrição abstrata de um processo de software”. FEILER & HAMPHREY (1993, p. 33) definem modelo de processo como uma “representação abstrata da arquitetura, projeto ou definição de um processo”.

Aspectos comuns entre modelos de processo de software

Embora existam diferentes modelos de processo de software, muitos aspectos são comuns entre esses modelos (HUMPHREY & KELLNER, 1989; CURTIS, KELLNER & OVER, 1992; FEILER & HUMPHREY, 1993; SCACCHI, 2001):

1. São representações abstratas, mais ou menos formais, da arquitetura, projeto e definição de um processo;
2. Normalmente, apresentam uma ou mais perspectivas: funcional (o que é feito), comportamental (quando é feito e como é controlado), organizacional (quem faz e onde é feito) e ‘informacional’ (coisas usadas e produzidas);
3. Estabelecem relacionamentos entre as diversas entidades (atividades, artefatos e agentes) de um processo e,
4. Incorporam uma estratégia de desenvolvimento.

Benefícios da definição e utilização de modelos de processos

A descrição e utilização de modelos de processos de software resultam em diversos benefícios às organizações de desenvolvimento de software (CURTIS, KELLNER & OVER, 1992; HUFF, 1996; SCACCHI, 2001):

1. Permite que o conhecimento e experiências sobre um processo sejam organizados e conseqüentemente incorporados ao processo;
2. Auxilia na tomada de decisão em situações imprevistas;
3. Torna possível a identificação de tarefas críticas;
4. Atribui maior visibilidade aos processos;
5. Forma uma base para a elaboração de guias, scripts, manuais;
6. Auxilia no gerenciamento do projeto (organização, planejamento, alocação de recursos, estimativas etc);
7. Provê meio para uma efetiva comunicação e compreensão dos processos entre os agentes envolvidos;
8. Estabelece bases que podem ser avaliadas para análise e mudanças para melhoria do processo de software;
9. Possibilita que novas tecnologias sejam avaliadas e sejam incorporadas ao processo de software e,
10. Habilita a empresa na definição de programas de mensuração de qualidade e produtividade.

2.5 O que é Processo de Manutenção de Software?

Mudanças aplicadas a um produto de software após a sua entrega são praticamente inevitáveis. Essas mudanças sejam para corrigir um erro de lógica ou aperfeiçoar um recurso, exigem que diversas atividades sejam executadas. Esse conjunto de atividades executadas para modificar um produto de software com o objetivo de corrigir ou melhorar é denominado manutenção de software.

A manutenção de software pode ter origem em diversos motivos. Considerando esses motivos, LIENTZ & SWANSON apud BENNETT & RAJLICH (2000, p. 76) categorizam a origem da manutenção em quatro tipos:

1. Adaptativa: refere-se às mudanças relacionadas ao ambiente de software. A inserção de um produto de software em um contexto externo (local no qual o produto será utilizado como uma solução, por exemplo, uma empresa) sujeita-o às mudanças provenientes desse contexto. Nesse sentido, o ambiente original de um produto de software (regras de negócio, características de hardware, características de software etc) pode sofrer modificações decorrentes de mudanças originadas no contexto externo para o qual foi projetado, resultando em modificações no produto de software para acomodar essas mudanças. Exemplos de mudanças de um ambiente externo seriam: mudanças na conjuntura econômica de um país, mudanças nas regras de negócios de uma organização, um novo sistema operacional ou hardware entre outras;
2. Perfectiva: trata-se de mudanças ocasionadas por novos requisitos solicitados pelo cliente/usuário. À medida que um produto de software é usado, novos requisitos não previstos no projeto original podem ser adicionados com o objetivo de melhorar o produto. Assim, mudanças são realizadas no produto de software para ampliar os requisitos originais do produto. Adição de novos relatórios, inclusão de um novo módulo para interface com a Internet e validação de usuários seriam exemplos de novos requisitos percebidos pelo cliente/usuário durante o uso do software e que poderiam ser adicionados ao projeto original;
3. Corretiva: refere-se a correção de erros, normalmente relacionados a erros de implementação. Mesmo com um controle de qualidade rigoroso, defeitos podem ser descobertos durante o uso do produto pelo cliente/usuário. A finalidade da manutenção corretiva é corrigir os erros não percebidos no processo de software. Exemplos de manutenção corretiva são: erros de lógica, erros na coleta de requisitos, validação de entradas entre outros;
4. Preventiva: trata-se de manutenções relacionadas à prevenção de problemas que poderiam ocorrer futuramente no produto. Em essência, manutenções preventivas estão relacionadas às mudanças no produto de software para que possa ser mais fácil corrigi-lo, adaptá-lo e melhorá-lo. Modificar um algoritmo para aumentar a *performance*, melhorar as documentações e dividir um módulo em diversos outros (modularização) podem ser considerados exemplos de manutenções preventivas.

O processo de manutenção de software envolve diversas atividades (BAXTER, 1992; BASILI et al, 1996; IEEE Std 1219-1998; BENNETT & RAJLICH, 2000):

1. Análise e isolamento da requisição de mudança: análise (compreensão), análise de impacto, classificação e isolamento da solicitação de mudança;
2. Projeto: revisão e modificação dos projetos: procedimental, arquitetural, dados e interface;
3. Implementação: implementação da mudança;
4. Testes: teste de integração, aceitação e/ou regressão;
5. Gerenciamento de configuração: controle de mudanças envolvendo todos os artefatos e recursos pelos quais a mudança pode se propagar.

Definição de terminologia

“O processo de manutenção de software refere-se às atividades que são executadas após a entrega de um produto de software” (BENNETT & RAJLICH, 2000, p. 75). O padrão 1219-1998 da IEEE (IEEE Std 1219-1998, p. 4) define a manutenção de software como “modificações sofridas pelo produto de software depois da entrega, para correção de falhas, melhoria de performance ou outros atributos ou, ainda, para adaptar o produto a uma modificação ocorrida no ambiente”.

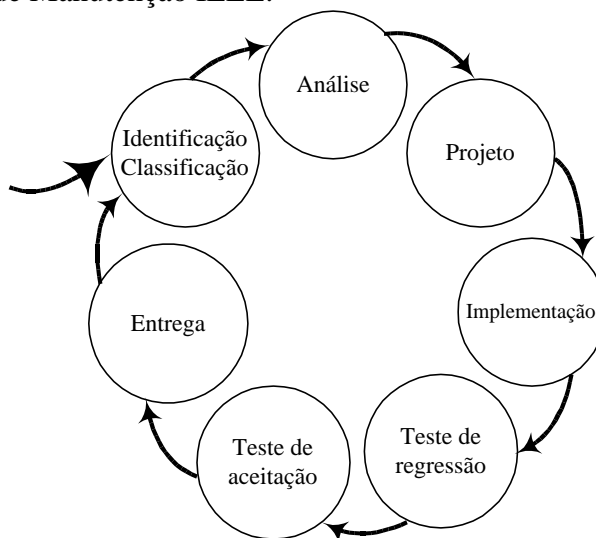
2.5.1 Qual a diferença entre novos desenvolvimentos e manutenção de software?

Diversas pesquisas tem centrado seus estudos nos processos relacionados a manutenção de software (LIENTZ & SWANSON, 1981; BASILI et al, 1996; MATTSSON, 1998; BENNETT & RAJLICH, 2000). Modelos de processos direcionados à manutenção de software podem ser encontrados em BASILI et al (1996), IEEE Std 1219-1998 (1998), BENNETT & RAJLICH (2000). A manutenção de software não pode ser compreendida como sendo apenas uma etapa do desenvolvimento de software cujo objetivo seja somente a correção de erros. A manutenção de software

inicia-se após a entrega de um produto de software e requer que diversas atividades sejam executadas com o objetivo de melhorar, adaptar, corrigir ou prevenir problemas que possam ocorrer nesse produto.

Muitos processos relacionados ao processo de desenvolvimento de software estão presentes nos processos de manutenção. Esta interseção pode ser observada nos processos de manutenção sugeridos pelo padrão 1219-1998 da IEEE (IEEE Std 1219-1998, p. 4) apresentado na Figura 13 ou, no modelo de processo adotado em BASILI et al (1996) onde é listada as seguintes atividades: análise de impacto do custo/benefício, isolamento, projeto de mudança, inspeção/certificação/verificação, codificação, teste de unidade, teste de integração, teste de aceitação, teste de regressão, documentação de sistema, outras atividades de documentação e, ainda, outras atividades diversas.

Figura 13: Processo de Manutenção IEEE.



Fonte: IEEE Std 1219 (1998, p. 4)

Apesar de muitas atividades serem similares ao desenvolvimento de software (desenvolvimento de novos projetos), as atividades envolvidas nos processos relacionados à manutenção podem incorporar todas as atividades presentes em um desenvolvimento de software e ainda adicionar novas atividades como, por exemplo, gerenciamento de configuração de software e suporte ao cliente. Além disso, deve-se considerar que o processo de manutenção de software somente é aplicado após o desenvolvimento inicial do software ter sido realizado com sucesso, ou seja, um

processo de manutenção de software nunca ocorre sem que um processo de desenvolvimento inicial esteja concluído.

McCONNEL (1996, p. 155) sugere que o ciclo de vida de desenvolvimento do software seja escolhido ou estruturado de acordo com as características do contexto, ou seja, diferentes projetos possuem diferentes necessidades e, portanto, podem exigir diferentes ciclos de vida. O autor sugere, ainda, que ao escolher um modelo de ciclo de vida, algumas questões, dentre outras, devem ser respondidas: os requisitos são pouco compreendidos? Há necessidade de gerenciamento dos riscos? Deve-se prover ao gerente do projeto o progresso do desenvolvimento? É possível que correções devam ocorrer durante o desenvolvimento?

O autor demonstra em seu trabalho, por exemplo, que os ciclos de vida Espiral e Prototipação Evolutiva são excelentes opções quando a resposta à questão “os requisitos são pouco compreendidos?” é afirmativa e, os ciclos de vida Cascata e Codifica-e-Corrige (Code-and-Fix) não seriam opções adequadas. O desenvolvimento de software pressupõe que inúmeros requisitos sejam implementados em uma única vez. Nesse caso, o processo de software utilizado para desenvolver o produto é estruturado de acordo com o conjunto de requisitos, considerando a complexidade, tempo de desenvolvimento, experiência no domínio da aplicação etc. Por outro lado, quando se trata de processos de manutenção de software, a resposta às questões que possibilitam a escolha de um ciclo de vida está mais sujeitas às variações contextuais do que em processos de desenvolvimento, mesmo tratando-se de um único produto. Para exemplificar as possíveis variações que podem ocorrer em uma organização cujos processos são caracterizados como de manutenção, qual seria a resposta mais adequada para as seguintes questões:

- a) Qual ciclo de vida seria mais adequado para solicitações de mudança de baixa complexidade: Cascata ou Espiral?
- b) Qual seria o ciclo de vida mais adequado para requisitos de alta complexidade: Cascata ou Espiral?

Empresas que mantêm produtos de software podem receber diariamente diversas solicitações de mudanças que variam, por exemplo, em

complexidade. Determinadas solicitações de mudanças tidas como complexas podem requerer que diversas atividades sejam executadas para garantir a sua adequada implementação, no entanto, determinadas solicitações de mudanças são menos complexas e, portanto, deveriam requerer, teoricamente, menor número atividades para implementá-las.

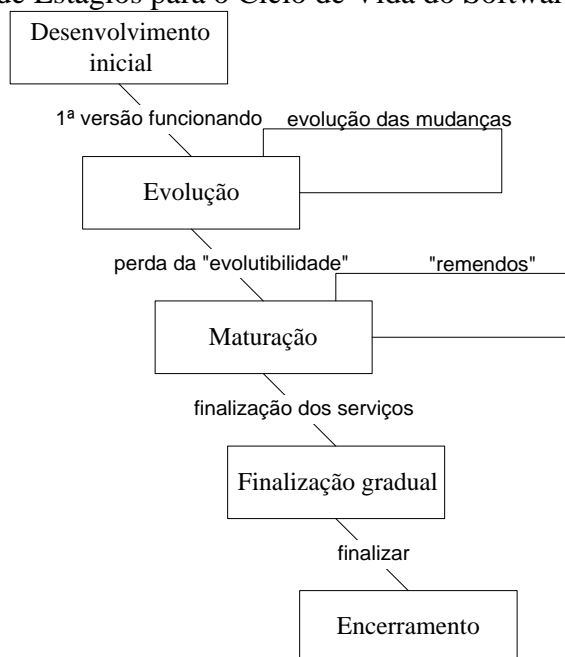
2.6 Modelos de Processos e Experiências Direcionados à Manutenção

A literatura de Engenharia de Software apresenta poucas experiências relacionadas diretamente a modelagem de processos de manutenção. Poucas pesquisas são realizadas e, conseqüentemente, não existem muitos modelos de processos voltados especificamente para o processo de manutenção. O objetivo desta seção é apresentar alguns modelos e experiências relacionadas ao processo de manutenção de software relatados na literatura.

2.6.1 Modelo de Estágio para o Ciclo de Vida do Software

O modelo de Estágios para o Ciclo de Vida do Software – *Staged Model for Software Lifecycle* – (BENNETT & RAJLICH, 2000) apresenta-se como uma proposta para sistematizar o processo de manutenção de software. Os autores enfatizam que a principal contribuição desse modelo consiste na separação da fase de Desenvolvimento Inicial da fase de Evolução (manutenção) e, acrescenta três outras fases: Maturação, Finalização Gradual e Encerramento (vide Figura 11).

Figura 14: Modelo de Estágios para o Ciclo de Vida do Software



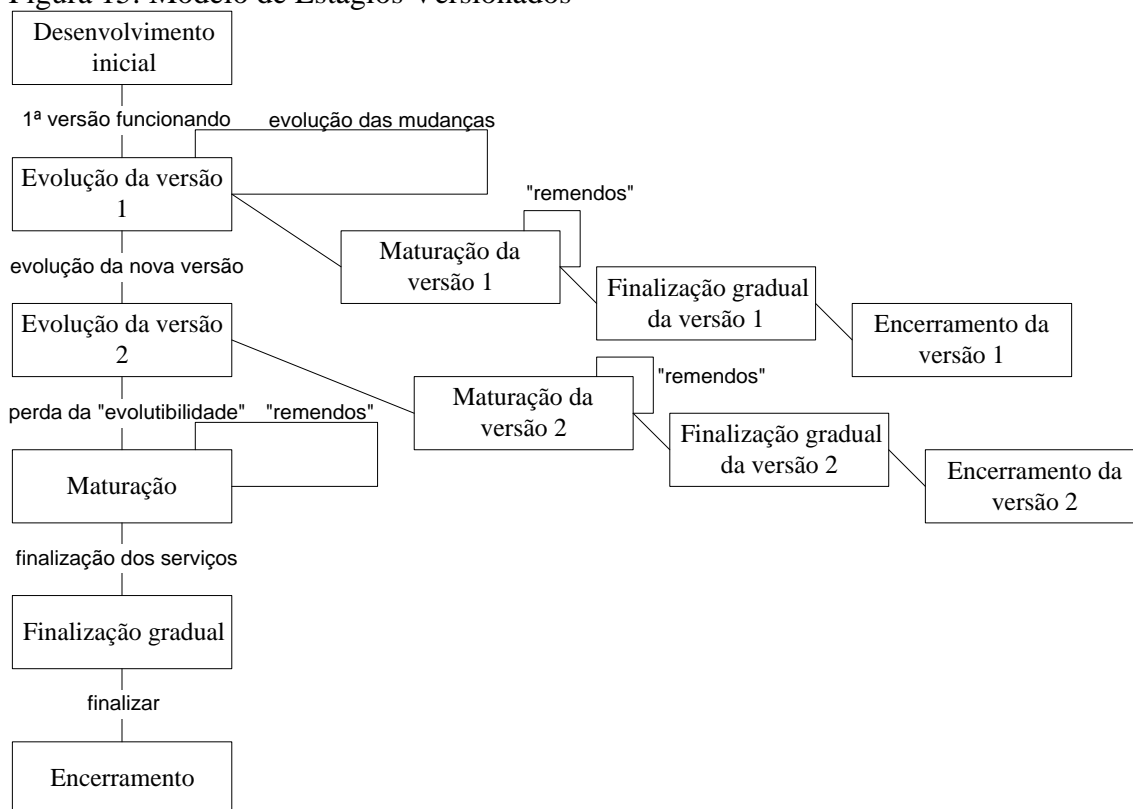
Fonte: Adaptado de software BENNETT & RAJLICH (2000, p. 78)

A fase de Desenvolvimento Inicial produz a primeira versão do produto de software. Nessa fase, a equipe de desenvolvimento adquire o conhecimento necessário para implementar o software. São empregadas nessa fase todas as atividades necessárias para produzir um produto de software (análise, projeto, implementação, testes etc). A fase de Evolução inicia-se somente quando a primeira versão for concluída com sucesso. Nessa fase, o produto de software é constantemente aperfeiçoado, falhas são corrigidas e adaptações são implementadas para atender a demanda do mercado. A empresa investe na melhoria pelo fato do produto ter uma boa aceitação no mercado e o retorno do investimento ser atraente. Na medida em que o produto evolui, poucas mudanças significativas são implementadas e, quando isso ocorre, o produto passa para a fase de Maturação. A fase de Finalização Gradual inicia-se quando a empresa de desenvolvimento não tem mais interesse no produto devido, por exemplo, ao pouco retorno financeiro ou o próprio produto já não atende as necessidades do mercado. Nessa fase, não são mais realizados quaisquer tipos de atualizações. Finalmente, na fase de Encerramento, os usuários do sistema substituem o produto e o ciclo de vida encerra-se.

O modelo de Estágios para o Ciclo de Vida do Software possui uma variação para “versionamento” (Figura 15). Essa variação permite que todas as

mudanças substanciais sejam implementadas em versões futuras do produto. Cada versão possui seu próprio ciclo e pequenas mudanças são realizadas até que a próxima versão seja concluída.

Figura 15: Modelo de Estágios Versionados



Fonte: Adaptado de BENNETT & RAJLICH (2000, p. 79)

As mudanças no produto de software são abordadas tanto na fase de Evolução quanto na fase de Maturação. Mudanças mais significativas são tratadas na fase de Evolução e mudanças mais superficiais, por sua vez, são tratadas na fase de Maturação. Para tratar essas mudanças, os autores sugerem tanto para a fase de Evolução quanto para a fase de Maturação, um mini-ciclo de mudanças. O mini-ciclo de mudanças é composto das seguintes fases: Solicitação de Mudança, fase de Planejamento incluindo Compreensão do Programa e Análise de Impacto da Mudança, fase de Implementação contendo Reestruturação para Mudança e Propagação da Mudança, fase de Verificação e Validação e, Atualização da Documentação.

3 MODELAGEM DE PROCESSOS DE SOFTWARE

Neste capítulo serão apresentados o papel e implicações da modelagem de processo de software em organizações de desenvolvimento de software, linguagens desenvolvidas especificamente para modelagem de processos de software, meios utilizados para representar processos de software, estudos relacionados à modelagem de processo de software em organizações de desenvolvimento de software. Ao final do capítulo, serão apresentadas as características comuns entre os estudos e características específicas de microempresas em relação à modelagem de processos de software.

3.1 O papel da modelagem de processo de software e suas implicações

A modelagem de processos de software tem sido abordada em diversas pesquisas nos últimos anos, dentre elas HUMPHREY & KELLNER (1989); HÖLTJE et al (1994); TURPIN (1996); BECKER, HAMANN & VERLAGE (1997); SUTTON (2000); BECKER-KORNSTAEDT (2001); MACHADO et al (2001). A ênfase dos estudos recaiu, sobretudo, na necessidade de definição e representação dos processos de software e na seleção adequada desses modelos frente aos processos necessários para desenvolver e manter produtos de software. Entre organizações de desenvolvimento ou entre projetos de uma mesma organização, a modelagem dos processos de software pode variar em largura e profundidade. A largura determina o alcance do processo podendo abranger todo ciclo de vida do software ou uma simples fase. A profundidade determina o nível de refinamento, ou seja, até que ponto um processo deve ser detalhado (HUFF, 1996).

As inúmeras variáveis envolvidas na elaboração de um produto de software determinam a sua complexidade. Dentre essas variáveis pode-se destacar: o produto que será desenvolvido, tecnologia aplicada, o número de pessoas envolvidas, métodos utilizados e etc. Conseqüentemente, o nível de complexidade do projeto de software determina o nível de complexidade dos processos necessários para desenvolvê-

lo. Assim, quanto maior a complexidade do projeto de software maior será a complexidade dos processos exigindo um controle mais rigoroso sobre esses processos (BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 63). A descrição de processos de software contribui com a resolução de diversos problemas presentes em organizações de desenvolvimento. HUMPHREY & KELLNER (1989) e HUFF (1996) ressaltam que a descrição de processos contribui sob vários aspectos no processo de desenvolvimento:

- A padronização do processo é necessária para permitir o treinamento, elaboração de guias, gerenciamento, revisão e automação;
- Com a padronização de métodos, cada experiência de projeto pode contribuir para uma melhoria global em uma organização;
- Processos padronizados provêm uma infra-estrutura básica para melhoria, avaliação e mensuração;
- Porque a definição de processos exige tempo e esforço para ser produzido, é impraticável produzir novos processos para cada novo projeto;
- Porque tarefas básicas são comuns em muitos projetos de software, apenas algumas customizações seriam necessárias para um processo padrão atender à maioria das necessidades de projetos.

Por outro lado, inúmeros problemas podem decorrer da falta da definição desses processos: aumento no custo de desenvolvimento, atraso na entrega do produto, baixa qualidade do produto, comprometimento na organização e gerência de projetos de desenvolvimento de software, inexistência de um processo repetível, impossibilidade de mensurar os processos, falta de visibilidade de processos (caixa-preta) entre outros (CURTIS, KELLNER & OVER, 1992; KELLNER et al, 1998). Portanto, as propostas apresentadas, pela maioria dos pesquisadores na área, focalizam dois aspectos: a) a necessidade de definir com maior fidelidade os processos de software em empresas de desenvolvimento de software e, b) o estabelecimento de uma prática evolutiva, isto é, que considere o processo de software como uma prática dinâmica exigindo melhoria contínua de suas representações, métodos, técnicas, ferramentas, procedimentos etc (MADHAJVI et al, 1994; BECKER, HAMANN & VERLAGE, 1997; BECKER-KORNSTAEDT, 2001).

3.2 Linguagens e Paradigmas para Modelagem de Processo de Software

Processos de software compreendem diversas atividades desempenhadas por agentes humanos. Dessa forma, são caracterizados pela criatividade, julgamentos, experimentos, completitude, informalidades, inexistência de estrutura e por atividades que são parcialmente ou não suportadas por qualquer tipo de apoio automatizado (SUTTON & OSTERWEIL, 1997).

Nesse contexto, são necessárias representações que permitam facilitar a definição de um projeto de software bem estruturado e de alta qualidade. Para atender esses requisitos, as linguagens de modelagem de processos de software disponibilizam uma notação “formal utilizada para expressar modelos de processo” (CONRADI, FERNSTRÖM & FUGGETTA, 1993) capaz de representar os modelos de processos de software por meio de uma sintaxe precisa e semântica bem definida. SUTTON & OSTERWEIL (1997) analisando diversas linguagens e processos de software, apresentam um conjunto básico de requisitos que uma linguagem de modelagem de processos de software deve conter:

1. Descrição das atividades: definir os passos contidos em um processo;
2. Descrição dos artefatos: definir o sistema de software que está sendo construído, incluindo o código-fonte e todos os outros artefatos produzidos;
3. Descrição dos recursos: especificar os recursos humanos e computacionais disponíveis para utilização nas atividades;
4. Relacionamento entre atividades: indicar os vários tipos de interconexões semânticas entre atividades, artefatos e recursos. Por exemplo, indicar que uma atividade pode preceder outra, um artefato (por exemplo, caso de teste) pode ser derivado de outro (por exemplo, uma especificação de requisitos), uma atividade pode modificar um artefato e utilizar um recurso;
5. Gerenciamento consistente: garantir a satisfação das condições requeridas sobre a interconexão entre atividades, artefatos e recursos e, entre atividades.

3.2.1 Classificações de Linguagens de Modelagem de Processo de Software

A literatura especializada apresenta diversas pesquisas nas quais são propostas ou usadas linguagens para modelagem de processo de software: STATEMATE (HUMPHREY & KELLNER, 1989), GRAPPLE (HUFF & LESSER, 1988), MARVEL (KAISER, 1990), SLANG (BANDINELLI, FUGGETTA & GHEZZI, 1993), SPELL (NGUYEN, WANG & CONRADI, 1997), JIL (SUTTON & OSTERWEIL, 1997), APEL (BIRK et al, 1997a), APPL/A (OSTERWEIL, 1997), PROSOFT (SILVA, 2001) entre outras.

CURTIS, KELLNER & OVER (1992), sugerem a classificação das linguagens considerando como referencial, a abordagem empregada pela linguagem. Dessa forma, classificam-nas em Modelos de Implementação, Modelos Funcionais, Modelos Baseados em Planos, Modelos Baseados em Redes de Petri e Modelos Quantitativos. AMBRIOLA, CONRADI & FUGGETTA (1997) classificam as linguagens de modelagem de processo tendo em vista a fase do ciclo de vida suportado e o nível de abstração oferecido pela linguagem:

1. Linguagens de Especificação de Processo (*Process Specification Languages – PSLs*): suportam a especificação de requisitos e avaliação de fases;
2. Linguagens de Projeto de Processo (*Process Design Languages – PDLs*): oferecem recursos úteis na fase de projeto;
3. Linguagens de Implementação de Processos (*Process Implementation Languages – PILs*): linguagens utilizadas principalmente na implementação e monitoramento das fases.

HUFF (1996) propõe a classificação considerando o paradigma que caracteriza a linguagem de modelagem de processo. Sob essa perspectiva, classifica as linguagens em: Paradigma Não-Executáveis, Paradigma Baseado em Estado, Paradigma Baseado em Regras e Paradigma Imperativo. Finalmente, SUTTON & OSTERWEIL (1997) sustentam que muitas linguagens de modelagem de processo baseiam-se em extensões de linguagens de programação convencional e sugerem a sua classificação em

Linguagens Funcionais, Baseadas em Regras ou Linguagens Reativas, Linguagens Imperativas e Redes de Petri.

3.3 Meios Utilizados para Representar os Modelos de Processo

Ao modelar processo de software, é necessário que as informações sobre o processo sejam organizadas e documentem o processo provendo auxílio aos participantes desse processo. Segundo KELLNER et (1998, p. 12) “a representação do processo é uma descrição, representação, imagem, retrato etc de um processo”. Os autores afirmam ainda que há três modos para representar o processo de software: modelos de processo, *templates* ou formulários e guias de processo.

3.3.1 *Templates* e formulários de processo

Templates e formulários de processo são apresentados na forma de um texto altamente estruturado. Elementos de informações são organizados e estruturados de tal forma que permitam o armazenamento e acesso as informações do processo. As informações são organizadas de maneira predefinida, em campos rotulados e, freqüentemente, são organizados em um modelo de hierarquia (dentro de tipos maiores) (KELLNER et al, 1998, p. 13). Experiências relacionadas à representação de processo de software utilizando *templates* e formulários podem ser consultadas em HÖLTJE et al (1994); MADHAVJI et al (1994).

3.3.2 Guias de processo

O propósito inicial de guias de processo é proporcionar aos participantes dos processos, descrições que permitam a sua execução, então, guias de processo são inicialmente desenvolvidos para pessoas diferentemente de modelos, *templates* e formulários os quais são normalmente desenvolvidos para engenheiros de

processo. Diversas características são associadas a guias de processo (KELLNER, 1998, p. 13):

- Guias de processo são estruturados;
- Orientados ao fluxo de trabalho;
- É um documento de referência para um processo específico;
- Provê suporte aos participantes do processo ao executar um determinado processo.

Geralmente, guias de processo utilizam gráficos e textos para representar o processo de software que apresentam, por exemplo, descrições detalhadas de métodos, técnicas e ferramentas que devem ser utilizadas, alertas sobre situações de riscos ou cursos alternativos, procedimentos a serem adotados, recursos a serem utilizados, modelos para serem utilizados, exemplos, etc. Experiências relacionadas à utilização de guias de processo podem ser consultadas em BECKER-KORNSTAEDT & BELAU (2000); BECKER-KORNSTAEDT (2001); BECKER-KORNSTAEDT, NEU & HIRCHE (2001).

3.4 Abordagens e Experiências Utilizadas na Modelagem de Processos

Nesta seção, serão apresentadas abordagens e experiências relacionadas para descrever processos de software em empresas de desenvolvimento de software.

3.4.1 Estudo 1: Experiências da NASA/GSFC na descrição dos processos de manutenção

BASILI et (1996) propõem um modelo de processo o qual aplica uma abordagem usada para modelagem de processos de manutenção. O estudo foi realizado em uma divisão (*Flight Dynamics Division – FDD*) ligada a Aeronáutica e a NASA (*National Aeronautics and Space Administration/Goddard Space Flight Center* (NASA/GSFC)) e conduzido pelo Laboratório de Engenharia de Software – SEL

(*Software Engineering Laboratory*) afiliado a FDD. A pesquisa foi iniciada em outubro de 1993 com a aplicação do *Quality Improvement Paradigm* (QIP), uma estratégia de melhoria de processo baseada em três passos interativos: compreensão, avaliação e “empacotamento” (*packaging*). Um aspecto importante na pesquisa é a combinação de abordagens quantitativas e qualitativas utilizadas para caracterizar a prática vigente na manutenção de software na FDD. A metodologia de análise qualitativa consistiu de seis passos: 1) identificação de entidades organizacionais; 2) identificação de fases; 3) identificação das atividades genéricas envolvidas em cada fase; 4) seleção de uma ou mais *releases* desenvolvidas para análise; 5) análise dos defeitos que ocorreram quando mudanças eram executadas nas *releases* selecionadas e; 6) estabelecimento de frequência e conseqüências de falhas em processos e organização pela análise das informações produzidas no passo 5.

O propósito da abordagem quantitativa foi definir e coletar dados significativos para caracterizar os processos e produtos do processo de manutenção. A análise dos dados deveria subsidiar o estabelecimento de uma linha de base para o processo de manutenção vigente que respondesse à algumas questões: qual é a distribuição do esforço entre as atividades durante a manutenção? Quais são as características de um release? Quais são as características dos erros de manutenção? Quais são as taxas de erros e taxas de mudanças? Para responder às questões, foi adotado o paradigma GQM (*Goal/Question/Metric*). A utilização do paradigma GQM possibilitou a elaboração de um conjunto de métricas que deveriam explorar as questões que se apresentavam ao estudo: esforço por atividade, esforço pelo tipo de manutenção, tempo gasto, fonte dos erros, classe de erros, tamanho do software antes da manutenção (em linhas de código) entre outras.

Os resultados do estudo permitiram uma melhor compreensão dos processos relacionados à manutenção de *releases* no FDD ao identificar 11 atividades distintas distribuídas em 5 grupos:

1. Análise/Isolamento: atividade análise de impacto do custo/benefício e atividade de isolamento;

2. Projeto: atividade de projeto de mudança e atividade de inspeção/certificação/verificação;
3. Implementação: atividade de codificação, teste de unidade e atividade de inspeção/certificação/verificação;
4. Teste: teste de integração, teste de aceitação, teste de regressão;
5. Outras atividades: atividade de documentação de sistema, outras atividades de documentação e, ainda, outras atividades diversas.

Diversos outros dados foram produzidos pelo estudo: percentual de erros por grupo de atividade, percentual de erros por tipo de manutenção, total de linhas de código adicionadas, alteradas ou excluídas entre outros. Os autores concluíram em seu estudo que a definição do modelo permitiu melhor compreensão dos processos executados ao manter os produtos de software, melhorou o gerenciamento e auxiliou os líderes de projeto provendo mais dados sobre as tarefas executadas pelas suas equipes. Além disso, a utilização de modelos mostrou-se como uma abordagem viável para empresas que desejam definir, compreender e melhorar seus processos de software.

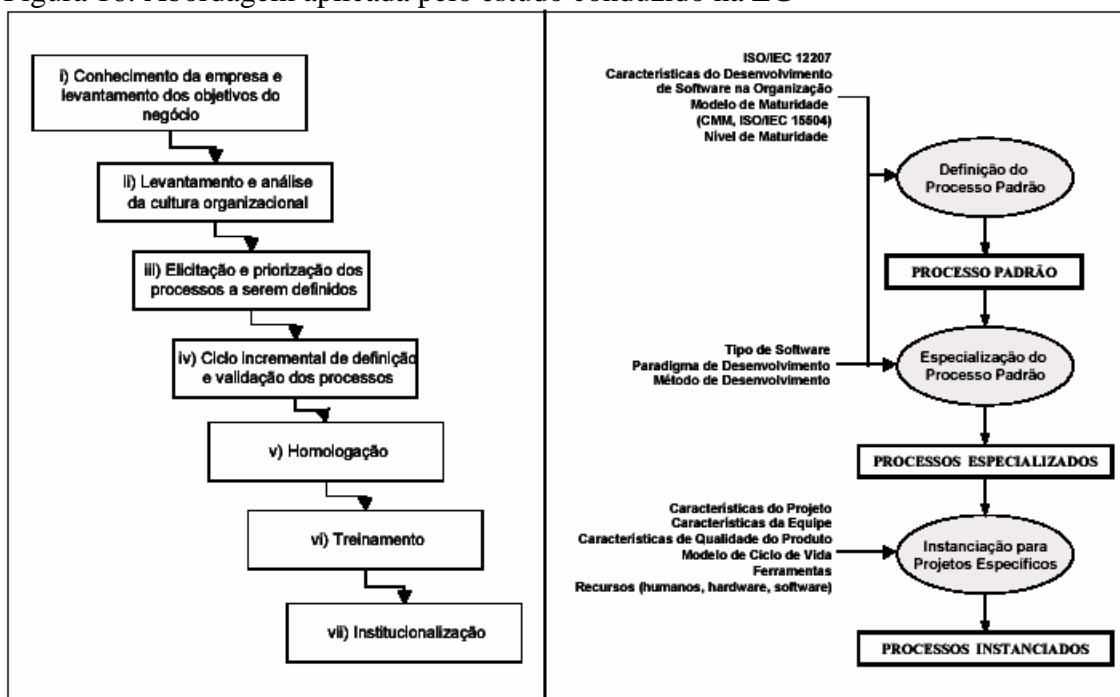
3.4.2 Estudo 2: Experiências em uma empresa de desenvolvimento de software de pacote

O estudo foi conduzido por pesquisadores da Universidade Federal do Rio de Janeiro (COPPE/UFRJ) em parceria com a empresa LG Informática, localizada no Rio de Janeiro, que desenvolve de software de pacote (MACHADO et al, 2001). A empresa já havia aplicado os conceitos de Gestão da Qualidade Total (GQT) em alguns setores relacionados diretamente aos clientes (departamentos de Customização e Implantação), no entanto, havia ainda a necessidade de ampliar o escopo do trabalho para incluir os demais setores e alinhar os conceitos da GQT às boas práticas de engenharia de software. A descrição do processo de software foi sustentada principalmente pela aplicação da Norma ISO/IEC 12207, ainda, foram consideradas algumas atividades presentes no modelo CMM e na futura Norma ISO/IEC 15504. A abordagem para descrever os processos de software da empresa foi dividida em sete atividades (Figura 16a):

1. Conhecimento da empresa e levantamento dos objetivos do negócio. Objetivo: conhecimento da área de atuação da empresa, produtos comercializados, histórico, estrutura organizacional, localização da matriz e suas filiais e fatores que motivaram a iniciativa de descrição de um processo de software;
2. Levantamento e análise da cultura organizacional. Objetivo: reconhecimento das particularidades do trabalho de cada departamento envolvido no projeto, buscando identificar os pontos fortes e o que precisaria ser melhorado em cada um deles;
3. Elicitação e priorização dos processos a serem definidos. Objetivo: priorizar e descrever os processos de software da empresa de acordo com a Norma ISO/IEC considerando ainda modelos como CMM e futura Norma ISO/IEC 15504;
4. Ciclo incremental de definição e validação dos processos. Objetivo: constituir um grupo de trabalho (GPES – Grupo de Processos de Engenharia de Software) nos moldes definidos pelo Modelo CMM responsável pela validação dos processos;
5. Homologação. Objetivo: homologar, garantir e manter por meio do GPES que os processos definidos sejam utilizados após sua implantação;
6. Treinamento. Objetivo: instituir uma cultura de processos na empresa tanto para os empregados efetivos quanto para futuros empregados quando incorporados ao quadro;
7. Institucionalização. Objetivo: incorporar as práticas descritas nos processos de software à rotina dos empregados, motivar sugestões por parte dos mesmos e realizar inspeções/auditorias.

A descrição do processo de software durou seis meses e envolveu intensamente o GPES que era constituído por dois diretores, quatro gerentes, dois analistas de sistemas da empresa e dois pesquisadores da COPPE/UFRJ. O processo padrão da empresa foi estabelecido (implantado) mediante o modelo genérico apresentado na Figura 16b.

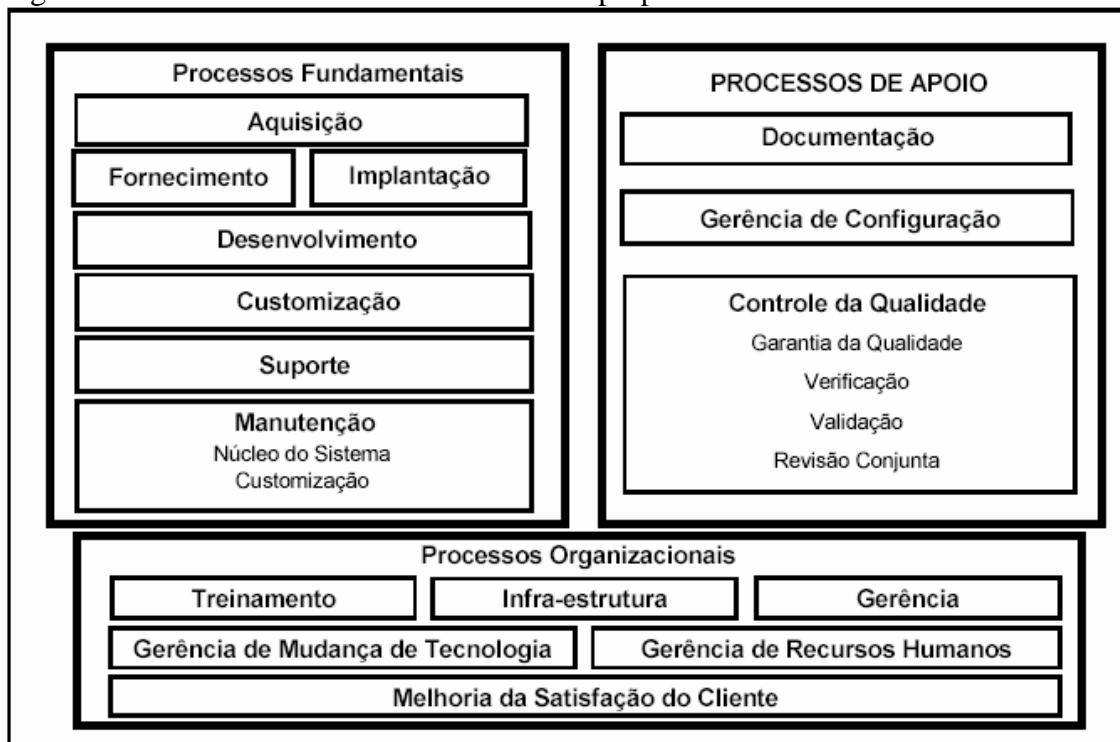
Figura 16: Abordagem aplicada pelo estudo conduzido na LG



a) Atividades para descrição de processos de software. b) Modelo para definição de processos de software
 Fonte: MACHADO (2001, p. 148)

O processo de implantação estava sendo realizado em vários departamentos da empresa em conjunto com um programa de mensuração. Os departamentos de Desenvolvimento, Manutenção e Suporte não possuíam metodologias de trabalho definidas, entretanto, percebeu-se por meio de entrevistas que havia uma cultura organizacional bem estabelecida e a preocupação em padronizar e incorporar boas práticas de engenharia de software já existentes aos processos que seriam descritos. Além dos processos previstos na Norma ISO/IEC 12207, foram propostos seis novos processos que deveriam atender às particularidades da empresa (Figura 17): Customização, Suporte, Implantação, Melhoria da Satisfação do Cliente, Gerência dos Recursos Humanos e Gerência de Mudanças de Tecnologia.

Figura 17: Estrutura de Processos de software proposta



Fonte: MACHADO (2001, p. 151)

O GPES criou um subgrupo o qual seria responsável pela definição de métricas a serem aplicadas nos processos com o objetivo de avaliar a performance das atividades do processo, a produtividade da equipe, a qualidade dos produtos gerados e o grau de satisfação dos clientes e produtos da empresa. A estruturação das métricas foi baseada no paradigma GQM (*Goal-Question-Metric*) tendo como eixo principal os objetivos da empresa. Entre as métricas inicialmente definidas para os processos de software da empresa, o estudo apresentou exemplos de métricas para os processos de Fornecimento, Customização, Suporte, Implantação, Melhoria da Satisfação do Cliente e Manutenção do Núcleo do Sistema. Cabe ressaltar que a aplicação do paradigma GQM foi utilizada, nesse trabalho, para avaliar o processo de software (qualidade e produtividade) e não avaliar a abordagem desenvolvida para descrever o processo de software real da empresa.

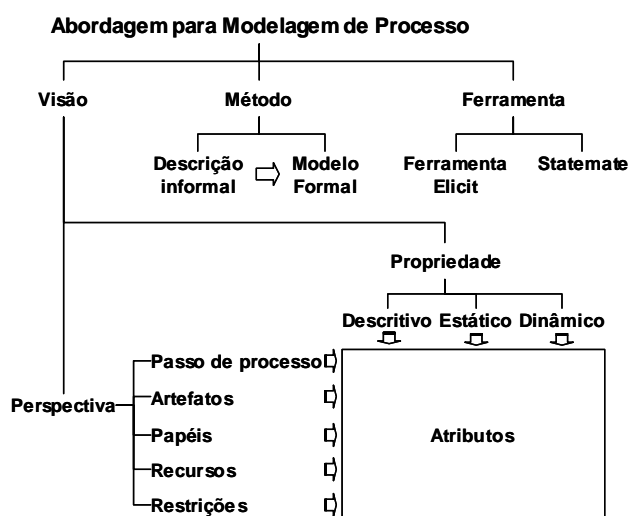
Nas considerações finais relatadas no estudo, os autores apontaram que a implantação do processo de software já havia ocorrido nos departamentos de Customização e Implantação, sendo que nos demais departamentos (Comercial, SAC, Desenvolvimento e Manutenção) ainda se encontravam em implantação. No entanto, os

autores consideraram que a experiência adquirida em conceitos da GQT pelo quadro de empregados antes da realização do estudo, facilitou a assimilação dos novos conceitos apresentados. A implantação dos processos é avaliada semanalmente em reuniões e modificações pontuais, sem muita gravidade, são discutidas com o objetivo de adequar as necessidades da empresa à estrutura de processos proposta. A utilização de um software de apoio ao processo agilizou o fluxo de trabalho principalmente em processos no qual há mais de um departamento envolvido. Além disso, o mesmo software está contribuindo com a automação de algumas métricas adicionando mais rapidez e precisão na coleta e análise dos dados. A implantação do programa de mensuração por intermédio do paradigma GQM mostrou-se intuitivo e útil para os diversos departamentos da empresa.

3.4.3 Estudo 3: Elicit, um método para descrever processo de software

HÖLTJE et al (1994) propõem a descrição do processo de software por meio do Método Elicit. A abordagem sugerida pelos autores descreve o processo de software em três dimensões: Visão, Método e Ferramenta. A dimensão Visão representa as diferentes visões de um modelo de processo de software. A dimensão Método e Ferramenta representam os métodos e ferramentas utilizadas ao descrever o processo de software (Figura 18).

Figura 18: Abordagem Elicit para descrever elementos do processo de software



Fonte: adaptado de HÖLTJE (1994, p. 2)

A dimensão Visão utiliza cinco perspectivas para visualizar o processo de software: passos de processo, artefatos, papéis, recursos e restrições. Para cada perspectiva, três propriedades são consideradas: descritiva, estática e dinâmica. Cada visão do processo (*perspectiva x propriedade*) é descrita por um conjunto de atributos. Por exemplo, a visão estática de um determinado artefato é descrita pelos seguintes atributos: entrada em um passo de processo, saída de um passo de processo, entrada de um usuário externo, saída para uma fonte externa, atribuído a um papel, controlado por restrições e contendo artefatos. O relacionamento entre perspectivas e propriedades por ser observado na Tabela 1.

Tabela 1: Relacionamento entre Perspectivas x Propriedades do Método Elicit.

Perspectiva	Propriedade descritiva	Propriedade estática	Propriedade dinâmica
Passo de processo	Identificador	Artefatos de entrada	Requer condições de entrada
	Alcançar objetivos	Artefatos de saída	Cumpre as condições de saída
	Tem o propósito	Executado pelo papel	Assume os estados
	Detalhado pelo procedimento	Apropriado pelo papel	
	Envia as mensagens	Recursos necessários	
	Recebe as mensagens	Controlado pelas restrições	
Artefatos	Caracterizado pelas métricas	Contém os passos (sub passos)	
	Identificador	Entrada para os passos de processo	Assume os estados
	Tem o propósito	Saída para os passos de processo	
	Armazenado no formato	Entrada para os usuários externos	
	É um artefato do tipo	Saída para as fontes externas	
	Caracterizado pela descrição	Apropriado pelo papel	
Papéis		Controlado pelas restrições	
		Contém os artefatos (sub artefatos)	
	Identificador	Executa os passos de processo	
	Tem as permissões	Apropriado para os passos de processo	
	Tem as obrigações	Apropriado para os papéis	
Recursos	Qualificações requeridas	Gerencia os recursos	
		Contém os papéis (sub papéis)	
	Identificador	Alocado no passo de processo	Assume os estados
	É um recurso do tipo	Gerenciado pelo papel	
Restrições	Caracterizado pela descrição	Contém os recursos (sub-recursos)	
	Identificador	Controla o passo de processo	Assume os estados
	É uma restrição do tipo	Controla os artefatos	
	Caracterizada pela descrição	Contém as restrições (sub-restrições)	

Fonte: HÖLTJE (1994, p. 4)

Na dimensão Método é utilizado o Método Elicit o qual é composto pelos seguintes passos: (1) Compreensão do ambiente organizacional, (2) Definição dos objetivos da descrição do modelo de processo, (3) Plano estratégico de descrição, (4) Desenvolvimento do modelo de processo, (5) Validação do modelo de processo, (6)

Análise do modelo de processo, (7) Análise após (pós-análise) a aplicação do método e (8) Agregação (*packaging*) de experiências obtidas.

A dimensão Ferramentas é caracterizada pelo uso de ferramentas para auxiliar na descrição do processo. No estudo foram utilizadas duas ferramentas: Elicit e Statemate. A ferramenta Elicit é responsável pela aquisição do conhecimento relativo a dimensão Visão, ou seja, coleta as informações da relação perspectiva x propriedades. A ferramenta Statemate é uma ferramenta comercial utilizada para projetar o desenvolvimento de sistemas reativos tais como sistemas de aviação e comunicação. As informações coletadas e organizadas na ferramenta Elicit foram usadas como entrada na ferramenta Statemate para executar atividades complementares de modelagem de aspectos estáticos e dinâmicos do processo. Maiores detalhes sobre os passos do Método Elicit podem ser obtidas em MADHAVJI et al (1994).

A análise do modelo é realizada considerando aspectos dinâmicos e estáticos. Com relação aos aspectos dinâmicos, o estudo utilizou a ferramenta Statemate para simular a execução do modelo por meio de animação visual do comportamento do modelo de processo descrito e, no caso dos aspectos estáticos, três características são avaliadas por meio do paradigma GQM:

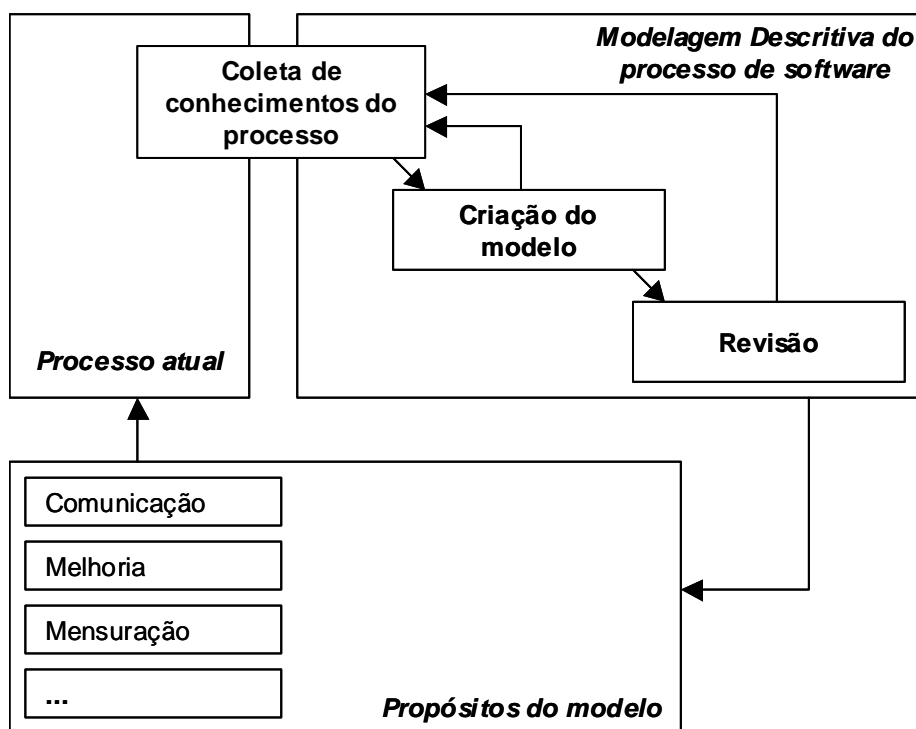
- Inconsistências: refere-se a informações contraditórias entre o modelo de processo descrito e processo real da empresa. Inconsistências, por exemplo, podem ocorrer em pares de atributos como *produz* (perspectiva atividade) e *é produzido por* (perspectiva artefato);
- Redundâncias: são informações textuais contendo a mesma informação ou informações complementares que poderiam ser descritas em um único atributo do processo. Redundâncias podem ocorrer ao modelar o processo de software principalmente quando as informações sobre o processo podem ter origem em diversas fontes e,
- Falta de completitude: no método Elicit, a completitude de um modelo de processo é avaliada pela utilização dos atributos de processo apresentados na Tabela 1. Três métricas foram definidas para avaliar a completitude do modelo de processo: (1) número de valores “em branco” por atributo para medir a utilização de cada atributo; (2) número e lista de entidades do

modelo de processo as quais tem atributos com valores “em branco” e, (3) número e lista de entidades do modelo de processo que não são descritas por um conjunto mínimo de atributos.

3.4.4 Modelo descritivo de processo de software

BECKER-KORNSTAEDT (2001) apresenta uma abordagem para modelagem descritiva de processo de software desenvolvida em pesquisas no Instituto Experimental de Engenharia de Software (IESE – *Institute for Experimental Software Engineering*). A abordagem apresenta três passos genéricos (Figura 19): coleta de conhecimentos do processo, criação do modelo e revisão.

Figura 19: Passos do modelo descritivo de processo de software do IESE.



Fonte: BECKER-KORNSTAEDT (2001, p. 314).

No primeiro passo são coletadas informações sobre atividades, artefatos, papéis, ferramentas, fluxos entre atividades, fluxo de artefatos, utilização de ferramentas e papéis envolvidos durante a execução do processo. No segundo passo, as

informações coletadas são utilizadas na formalização do modelo de processo atual e, finalmente, no terceiro passo, são conduzidas revisões no modelo descritivo criado. A abordagem também prevê ciclos de retorno ao passo 1 quando, por exemplo, faltam informações ao criar o modelo (passo 2) ou quando o modelo mostra-se incompleto ou incorreto no passo 3. O detalhamento dos três passos genéricos do modelo dependerá dos propósitos da modelagem, nível de detalhamento desejado e/ou intenções dos usuários do modelo de processo de software. As seções seguintes apresentarão brevemente os resultados de dois estudos nos quais o modelo genérico foi refinado.

Estudo 4: DaimlerChrysler

BECKER-KORNSTAEDT & BELAU (2000) relatam o estudo conduzido na Divisão de Infraestrutura Espacial da DaimlerChrysler Aeroespacial AG (Dasa). Nesse estudo, o modelo genérico foi refinado em oito fases:

Fase	Objetivo
Definição dos objetivos e escopo	Definição do escopo da modelagem de atividades, características do modelo e contexto organizacional da modelagem de atividades. Além disso, é determinado o nível de visibilidade do processo de acordo com os interesses dos usuários do modelo de processo.
Seleção ou desenvolvimento de um esquema de modelagem	Identificar e definir a estrutura das informações a serem coletadas pelo modelo bem como os relacionamentos entre essas entidades de informação. Essa fase deve estar alinhada aos objetivos e escopo definidos na fase anterior.
Seleção de linguagem	Determinar a notação por meio da qual o modelo de processo será descrito.
Seleção de ferramentas	Selecionar uma linguagem para modelagem que suporte o formalismo definido na fase anterior. Editores gráficos podem ser empregados para representar o modelo de processo.
Coleta de conhecimento do processo	Adquirir todas as informações necessárias para descrever o processo. Os meios de aquisição de informações podem ser entrevistas, observações e análise de documentos e produtos do processo.
Criação do modelo	Classificar, organizar e interpretar as informações coletadas para criar o modelo de processo de software. Diversas iterações são realizadas até os participantes concordarem com o modelo de processo criado. Se necessário, visões do processo são criadas para atender aos pontos de vista de cada usuário ou grupo de usuário do processo.
Análise do modelo de processo	Verificar o modelo de processo criado para averiguar sua completude, exatidão e consistência.
Análise do processo	Utilizar o modelo de processo descrito para verificar a fidelidade do modelo de processo em relação ao processo real.

A descrição do processo de software ocorreu, principalmente, por meio de entrevistas estruturadas (formulários estruturados). Foram estabelecidas duas entrevistas com duração máxima de duas horas para cada participante do processo de software as quais permitiram descrever atividades, artefatos, critérios de entrada/saída e papéis. Foram entrevistados sete participantes durante três dias consecutivos totalizando noventa minutos de entrevistas. Adicionalmente, foram consultadas documentações geradas durante o processo de desenvolvimento.

As informações coletadas, incluindo os questionários e consultas a documentos da organização, foram descritas em um arquivo texto. As informações fragmentadas nos questionários foram agrupadas por elementos do processo de software (atividades, sub-atividades, artefatos etc). O texto resultante permitiu a identificação de e correção de inconsistências (os entrevistados por diversas vezes se referiam ao mesmo elemento do processo de software com diferentes nomes). Além disso, foi observado que ocorria variações na execução de determinados processos, por exemplo, determinadas sub-atividades ocorriam apenas em determinados tipos de projetos.

A primeira versão do modelo descritivo do processo apresentou seis fases, 20 artefatos, 26 atividades, diversas sub-atividades e papéis para cada atividades (no estudo não foi explicitado o número de sub-atividades e papéis descritos). A versão do modelo foi enviada para avaliação da pessoa responsável pelo acompanhamento do estudo na empresa o qual apresentou sugestões que foram incorporadas gerando, desse modo, a segunda versão do modelo descritivo. Um novo ciclo de entrevistas foi realizado e mudanças foram incorporadas ao modelo. A fidelidade do modelo descritivo foi atestada pelo acompanhamento da execução do processo real. A comparação do modelo descrito com o processo real resultou nas seguintes observações: (1^a) variações no processo de software executado entre diferentes equipes e, (2^a) reincidência de atividades/sub-atividades em diferentes fases indicando a dificuldade em definir os limites entre as fases.

Diversas lições foram apresentadas pelos autores a partir do estudo conduzido na DASA. A limitação de bibliografia relacionada à descrição de processo de software dificultou a aquisição de informações e/ou experiências que poderiam ser

incorporadas ao estudo sugerindo, portanto, a necessidade de mais pesquisas nessa área. Uma única fonte de dados pode não ser suficiente para desenvolver um modelo descritivo de processo de software sendo, necessário, que diferentes fontes sejam utilizadas: artefatos, entrevistas de executores do processo, observação do processo de software real em execução ao entrevistar o executor do processo de software, considerar os diferentes papéis desempenhados durante o processo e, se possível, em diferentes projetos.

O conhecimento sobre o processo de software relatado pelos entrevistados, muitas vezes, diferenciava-se da prática, sugerindo, portanto, que a descrição do modelo de processo de software seja realizada por uma pessoa neutra (externa ao processo). O tempo gasto em entrevistas pela pessoa responsável pela descrição do processo pode ser reduzido se houver um conhecimento geral antecipado do processo, por exemplo, pela consulta a documentos e entrevista de um único executor do processo que tenha uma visão geral do processo de software para esclarecimentos pontuais. Essa constatação indica que a coleta de dados poderia ser realizada em duas etapas: entrevista preparatória que resultaria em uma visão geral do processo e, entrevista detalhada para obter dados mais específicos do processo de software. Essa abordagem para coleta de dados é particularmente interessante pelo fato dos executores nem sempre disporem de muito tempo para entrevistas.

Finalmente, a descrição do modelo de processo de software por meio de texto (linguagem natural) foi útil por facilitar a compreensão do modelo descritivo pelos executores do processo, entretanto, dificultou a verificação de inconsistências e análise de dados. Assim, o modelo descritivo de processo de software poderia ser realizado de duas formas: em linguagem natural para facilitar a compreensão dos executores do processo e, em linguagem mais formal, por exemplo, utilizando uma notação gráfica.

Estudo 5: Thales





O estudo (BECKER-KORNSTAEDT, NEU & HIRCHE, 2001) foi realizado na subsidiária alemã da Thales, cujas atividades estão voltadas para sistemas de segurança e aeroespacial. O motivo da parceria entre o IESE e a Thales foi a intenção da subsidiária melhorar o processo de software e alcançar um nível superior no modelo SPICE. Parametrizado nessa intenção, o Grupo de Processo de Engenharia de Software (GPES) da empresa elegeu a descrição do processo de software como um fator chave para alcançar o seu objetivo. A descrição do processo de software envolveu dois pesquisadores do IESE, três membros do GPES e diversos participantes do processo na Thales. O estudo de caso foi dividido em cinco estágios:

Estágio	Objetivo
Preparação	Fornecer uma visão geral do processo de software antes de iniciar as atividades de modelagem resultando na descrição em alto nível das principais atividades, artefatos e critérios de entrada e saída.
Treinamento	Apresentar os principais conceitos e terminologias relacionadas à modelagem de processos de software.
Modelagem inicial do processo	Transferir tecnologias para os participantes da empresa. Treinar os participantes na coleta e descrição do processo de software. A primeira versão descritiva de alto nível do processo de software deve ser gerada nesse estágio.
Elaboração do modelo	Detalhar o modelo descritivo de processo de software gerado no estágio anterior (Modelagem inicial do processo).
Revisão	Verificar inconsistências no modelo descritivo gerado no estágio anterior (Elaboração do Modelo). Deve ser gerada nesse estágio a versão final do modelo descritivo do processo de software.

A análise de documentos (Estágio de Preparação) durou aproximadamente dois dias. Foram descritos em linguagem natural os elementos de processo de alto nível: atividades, artefatos, papéis e critérios de entrada e saída. Além disso, também foi realizada a descrição de fluxos de produtos (consumidos, produzidos e modificados) entre atividades e artefatos e descrição dos papéis para cada atividade. O treinamento foi realizado por meio de workshop que ocorreu em um único dia (aproximadamente 8h00m). O workshop abordou conceitos relacionados à modelagem de processo de software, utilização de ferramentas relacionadas à modelagem de processo de software (ferramenta Sparmint) e apresentação de exemplos (modelos de processo descritos). No Estágio de Modelagem Inicial do Processo, as informações de alto nível coletadas no primeiro estágio foram modeladas por intermédio da ferramenta


Spearmint – *Software Process Elicitation, Analysis, Review and Modeling in an Integrated Environment* – vide Figura 21. A ferramenta utiliza ícones para representar os quatro elementos básicos de processo de software: atividade, artefato, papel e ferramenta e; seis tipos de relacionamentos entre os elementos: consome, produz, modifica, envolve, usa e contém (vide Quadro 2):

Quadro 2: Notações gráficas utilizadas pela ferramenta Spearmint.

	Atividade / sub-atividade
	Artefato
	Papel
	Ferramenta
	Consome: apenas é possível de artefato para atividade.
	Produz: apenas é possível de atividade para artefato.
	Modifica: apenas é possível entre atividade e artefato.
	Envolve: apenas é possível entre atividade e papel.
	Usa: apenas é possível entre atividade e ferramenta
	Contém: apenas visível em árvore, não há uma notação gráfica explícita para esse relacionamento.

A ferramenta Spearmint também possui representações para execução de atividades em paralelo (Quadro 3). Quatro possíveis estados são representados:

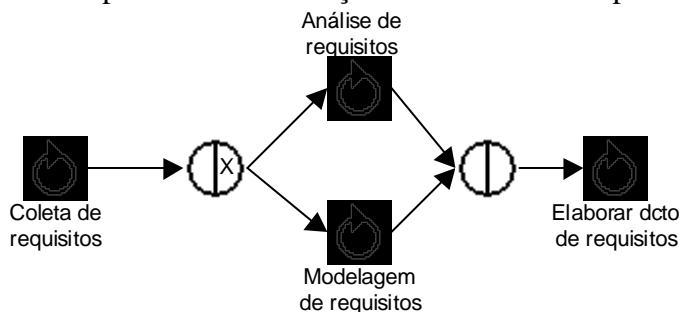
Quadro 3: Notações gráficas utilizadas para representar execução em paralelo.

	Utilizado para representar atividades em paralelo.
x	"and": significa que pelo menos duas (ou mais) atividades devem ser executadas antes ou depois do novo estado.
+	"or": significa que qualquer atividade de no mínimo duas atividades poderiam ser executadas antes ou depois do novo estado (disjunção inclusiva ou soma lógica).
\oplus	"xor": significa que exatamente uma de um grupo de atividade deveria ser executada antes ou depois do novo estado (disjunção exclusiva).
	"nil": significa que não há atividade envolvida.

O exemplo ilustrado na Figura 20 demonstra a utilização da representação de junção e disjunção. Nesse exemplo, o fluxo de controle é unificado em

um único ramo. A opção de estado "and" indica que a atividade “Elaborar documento de requisitos” pode ser executada se, e somente se, as atividades “Modelagem de requisitos” e “Análise de requisitos” forem executadas.

Figura 20: Exemplo em Spearmint de execução de atividades em paralelo.



A primeira versão do modelo descritivo do processo de software continha representações gráficas do processo, descrições textuais detalhadas de cada um dos elementos do processo coletados no primeiro estágio, fluxo de controle (ordem de execução das atividades) e fluxo de produtos (artefatos consumidos, produzidos e modificados). A primeira versão do modelo descritivo do processo gerado no Estágio Modelagem Inicial do Processo foi refinado no Estágio de Elaboração do Modelo. Foram criados dois sub-modelos (refinamento do processo de software devido à complexidade), identificados 30 artefatos, 24 atividades (9 atividades de alto nível) e quatro papéis. Finalmente, no Estágio Revisão, a consistência do modelo foi realizada pela ferramenta Spearmint que não apresentou erros na estrutura do modelo. No entanto, ao verificar a consistência entre as representações gráficas e a descrição textual, foram encontrados alguns erros, por exemplo, embora estivesse descrito na forma textual que um artefato deveria ser utilizado na Atividade “A”, a relação entre o artefato e a atividade não constava na representação gráfica.

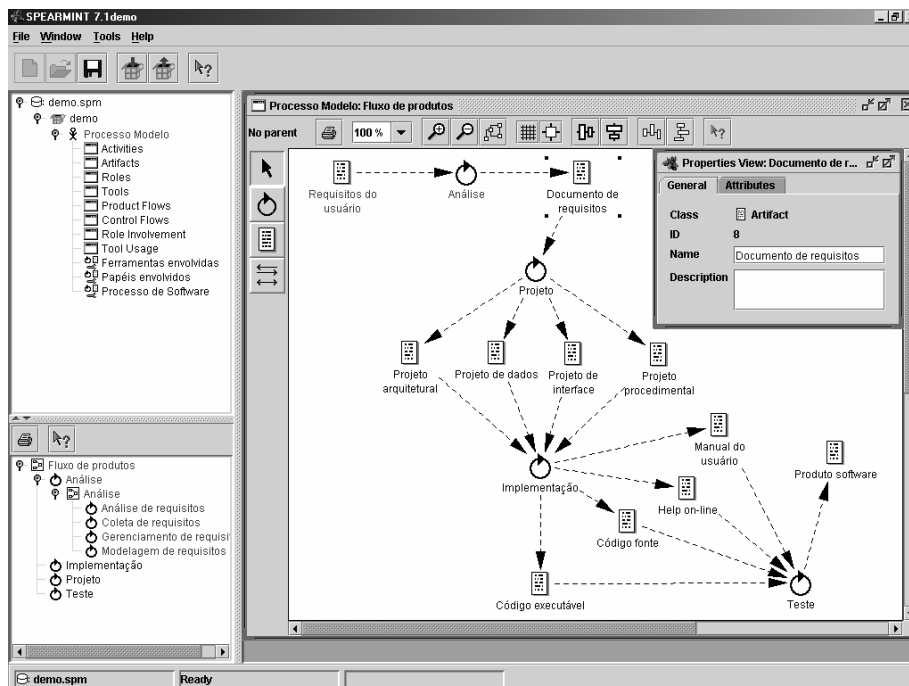


Figura 21: Área de trabalho da ferramenta Spearmint (IESE)

Um dado não coletado pelo estudo foi o esforço gasto para descrição do modelo de processo. Os autores apresentaram diversas lições aprendidas com o estudo: a necessidade de abordagens para modelar processos de software utilizar tanto representações gráficas como textuais, uma vez que a utilização de representações gráficas facilita a verificação do modelo e as descrições em linguagem natural facilitam a compreensão do modelo pelos participantes do processo; a transferência de tecnologias para a indústria não se resume apenas no desenvolvimento de ferramentas, é necessário prover conhecimento (treinamento) e acompanhamento aos participantes (suporte por telefone, por exemplo) da modelagem para garantir que os conceitos foram assimilados e incorporados às práticas de engenharia de software.

3.5 Requisitos de Microempresas em Relação à Modelagem de Processos

Nessa seção será apresentado brevemente o perfil de microempresas e necessidades relacionadas à modelagem de processos de software. Inicialmente serão apresentados dados da pesquisa bial realizada pelo Ministério da Ciência e Tecnologia sobre qualidade e produtividade no setor de software brasileiro. Posteriormente, serão apresentados os dados de pesquisa, conduzida pelo autor do

presente trabalho, em seis microempresas para levantamento das necessidades dessas empresas em relação à modelagem de processos de software.

3.5.1 Situação das microempresas no Brasil

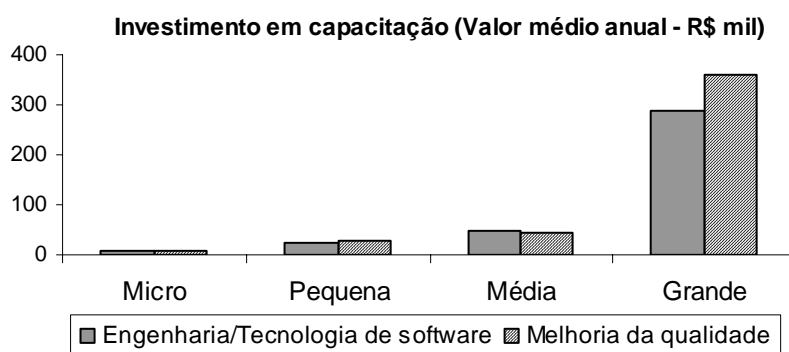
O Ministério da Ciência e Tecnologia por intermédio da Secretaria de Política de Informática e Automação (MCT/SEPIN) em sua última pesquisa, apresentou o diagnóstico da qualidade e produtividade em Software no Brasil. A pesquisa, divulgada ao final do ano de 2002, faz parte do Programa Brasileiro da Qualidade e Produtividade e é realizada a cada dois anos. Diferentemente dos resultados apresentados em outras pesquisas (1993, 1995, 1997 e 1999), a pesquisa divulgada em 2002 apresentou seus resultados desagregados segundo porte, considerando a força de trabalho efetiva das organizações, que inclui sócios, dirigentes e empregados efetivos. A amostra da pesquisa foi de 446 empresas.

Segundo a pesquisa, microempresas de desenvolvimento de software, considerando a força de trabalho efetiva das organizações, que inclui sócios, dirigentes e empregados efetivos, somam 36% das empresas de desenvolvimento de software, ou seja, é a maior categoria de empresas constatada pela pesquisa sob o critério força de trabalho se comparadas a pequena (33%), média (9%) ou grande (22%) empresas.

O número médio de domínios de software desenvolvidos no Brasil, por organização, foi de 3,8 e, segundo a pesquisa, as microempresas desenvolvem ou mantêm produtos de software para todos os domínios pesquisados. Os principais domínios de software são: administração privada (41%), indústria (36%), serviços (34%), comércio (33%), financeiro (29%) e educação (25%). A atuação em mais de um domínio pode exigir variação no processo de desenvolvimento de software e, conseqüentemente, pode requerer que os processos de desenvolvimento de software sejam flexíveis o suficiente para adaptarem-se às necessidades específicas de cada domínio.

O investimento das microempresas em capacitação em engenharia e tecnologia e, em melhoria da qualidade, mostrou-se bem inferior se comparado às outras categorias de empresas (Gráfico 1). Esse contexto cria dificuldades para microempresas adquirirem conhecimento de tecnologias em Engenharia de Software (métodos, técnicas, ferramentas etc) que permitam a melhoria dos seus processos de desenvolvimento. Por outro lado, demonstra que pode haver dificuldades das microempresas em destinarem recursos para qualificação da força de trabalho e, nesse sentido, faz-se necessário que sejam criadas oportunidades adequadas ao perfil dessa categoria com o objetivo de facilitar o acesso a essas tecnologias. Entre os diversos obstáculos que impedem microempresas a investirem na capacitação da força de trabalho, é provável a influência do custo financeiro, tempo necessário, incertezas sobre os reais benefícios que poderiam advir do uso desses conhecimentos entre outros.

Gráfico 1: Investimentos das organizações em capacitação em Engenharia/Tecnologia de Software.



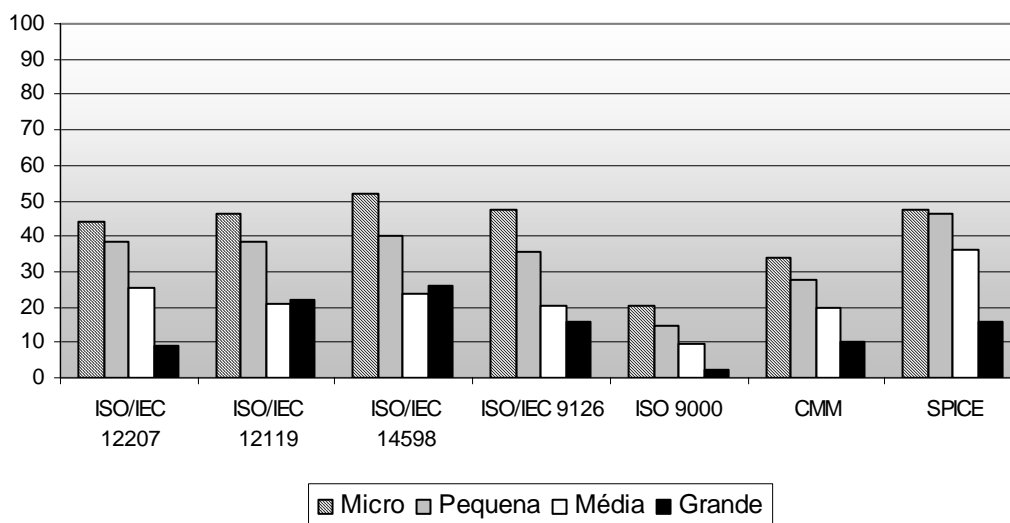
Nota: Os valores médios anuais dos investimentos em treinamento em engenharia/tecnologia de software e melhoria da qualidade por empresa foram estimados com base nas organizações que oferecem treinamento e registram os valores correspondentes.

Os reflexos dos baixos investimentos na atualização da força de trabalho pelas microempresas podem ser observados no

Gráfico 2, o qual demonstra o percentual de desconhecimento de normas e modelos nas empresas pesquisadas. É perceptível que o desconhecimento de normas e modelos é maior nas microempresas, como também é perceptível que pode haver uma relação inversamente proporcional entre os investimentos e conhecimentos ao comparar o Gráfico 1 e

Gráfico 2, ou seja, quanto maior o investimento em capacitação maior o conhecimento.

Gráfico 2: Desconhecimento de normas e modelos de qualidade nas organizações (%).



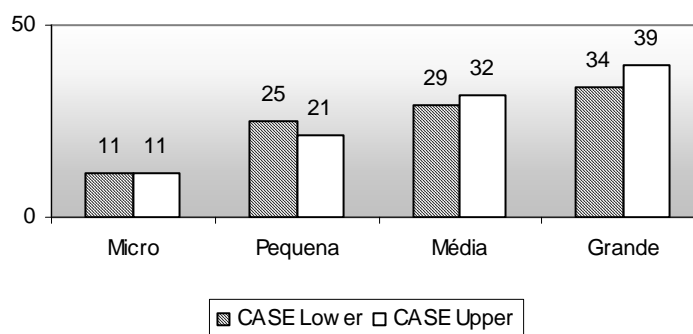
Mesmo considerando as proporções em investimentos em capacitação da força de trabalho por categoria de empresas e, que microempresas também oferecem meios de acesso ao quadro de empregados para promover a atualização da força de trabalho, o desconhecimento de normas nas microempresas é maior. No entanto, não é objetivo neste trabalho discutir quais fatores contribuem para essa situação sejam eles financeiros, políticas de capacitação, meios de acesso etc, mas sim, deixar claro que é necessário que qualquer tecnologia que seja desenvolvida para microempresas deva considerar esse contexto.

Dados sobre o processo de software indicam que a informalidade do processo de software é um fator preocupante nas empresas de software. A pesquisa do MCT/SEPIN apontou que 75,2% das microempresas não documentam todos os seus processos de software. Esse dado, contrastado com pequena (63,8%), média (47,6%) e grande (55,2%) empresa, atribui às microempresas o maior percentual entre as categorias de empresas pesquisadas.

Somente seis documentos adotados atingiram percentuais acima de 50% em microempresas: guia de instalação (50,3%), documentação de programas

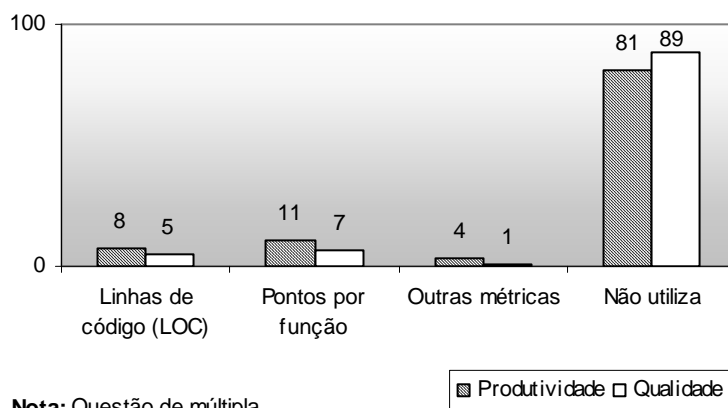
(52,3%), documentação de código (56,2%), help on-line (56,2%), manual do usuário (56,2%) e contratos e acordos (66,7%). Acompanhamento de prazos e descrição do produto para comercialização, ambos com 49,7%, também ficaram próximos aos 50%. O uso de ferramentas de desenvolvimento em microempresas também apresentou percentuais baixos e, nenhuma ferramenta de desenvolvimento pesquisada era utilizada por mais de cinquenta por cento das empresas. Os melhores percentuais observados foram geradores de relatórios (40,3%), gerador de código-fonte (28,2%) e depurador interativo (27,5%) sendo que as demais, atingiram percentuais inferiores a 25% de uso nas microempresas. O uso de ferramentas *CASE* (*Computer Aided Software Engineering*) que, em linhas gerais, apóia a execução de atividades de desenvolvimento de software de forma automatizada, também mostrou baixos índices para microempresas (vide Gráfico 3). Conforme os percentuais apresentados, praticamente uma entre dez microempresas utiliza ferramentas *CASE* no processo de software.

Gráfico 3: Percentual de uso de ferramentas *CASE* nas organizações.



O uso de métricas para medir a produtividade e qualidade dos processos de software entre as microempresas também apresentou percentual baixo (vide Gráfico 4). Para medir a qualidade dos processos de software, apenas 5% utilizavam linhas de código e 7% pontos por função. Já o uso de métricas para medir a produtividade apresentou percentual um pouco acima dos demais, 8% das microempresas utilizavam linhas de código e 11% pontos por função.

Gráfico 4: Métricas primitivas utilizadas para medir a produtividade e qualidade dos processos de software em microempresas (%).



3.5.2 Resumo da pesquisa sobre a situação atual das microempresas em relação à modelagem de processos de software

A pesquisa realizada pelo MCT/SEPIN apresentou diversos indicadores importantes, discutidos na seção anterior do presente capítulo, sobre o perfil e processo de software em microempresas brasileiras. No entanto, para atender ao objetivo do presente trabalho, além dos indicadores provenientes da pesquisa MCT/SEPIN, foram coletados dados mais específicos sobre a situação da modelagem de processos em seis microempresas de desenvolvimento de software localizadas em Florianópolis – SC e Londrina – PR. Nesta seção será apresentado de forma resumida o resultado da coleta realizada nas seis empresas. Maiores detalhes sobre a metodologia, processo de coleta, dados da coleta e análise dos dados podem ser consultados no Anexo A.

Entre as empresas pesquisadas, nenhuma documenta totalmente o ciclo de vida utilizado e, quatro responderam que documentam parcialmente. Apesar de cinco empresas informarem que o ciclo de vida não varia em projetos distintos, cinco não o documentam sistematicamente. Essa prática fomenta a individualização em detrimento da institucionalização do processo de desenvolvimento/manutenção de software. Percebeu-se também que existe a necessidade de adaptação, mesmo que informal, de modelos de ciclo de vida às necessidades das microempresas pelo fato de

nenhuma delas adotar, na íntegra, modelos de ciclos de vida existentes (vide Anexo A - Tabela 2).

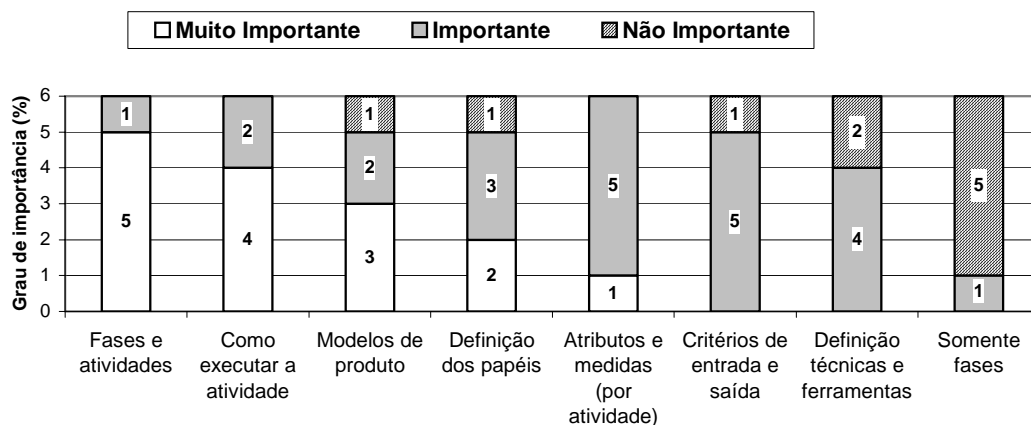
Considerando a documentação parcial do ciclo de vida adotado pelas microempresas, a formalização dos papéis e responsabilidades, constituiu-se o maior percentual (3) entre as empresas. A formalização das fases e atividades do ciclo de vida ocorria em uma das empresas e, quatro empresas documentavam parcialmente as atividades do ciclo de vida. Os critérios de entrada e saída eram formalizados por uma sendo que o restante das empresas não os formalizou. A formalização de métodos e técnicas não era realizada por nenhuma das empresas (maiores detalhes vide Anexo A - Gráfico 5).

Quanto à coleta de dados sobre o processo de desenvolvimento de software, entre as seis empresas pesquisadas, apenas uma fazia sistematicamente e outra eventualmente, sendo que as demais empresas (quatro) não coletavam (Anexo A - Gráfico 6). Entre os fatores que dificultariam a formalização do modelo de ciclo de vida, dois destacaram-se: custos e tempo (constatado em quatro empresas). A qualificação profissional, apesar de apresentar número menor (duas empresas), também se destacou entre as demais dificuldades: modelos inflexíveis, desconhecimento de benefícios, investimentos e modelos inadequados apresentaram o mesmo número de respostas (1) (maiores detalhes vide Anexo A - Gráfico 7). O uso de ferramentas mostrou-se sistemático em três das microempresas, as demais afirmaram que utilizavam eventualmente. A etapa de projeto apresentou o maior percentual no uso de ferramenta (observado em cinco empresas) seguida da etapa de análise (3). Sistematicamente, três das microempresas usavam as mesmas ferramentas e, quatro afirmaram que não há variação no uso de ferramentas entre projetos distintos.

Questionadas sobre o grau de importância de elementos de processos de software que deveriam ser modelados, cinco empresas atribuíram o grau “muito importante” e uma empresa atribuiu o grau “importante” para fases e atividades. A atribuição de “muito importante” e “importante” para descrição de procedimento de como executar a atividade obteve, respectivamente, quatro e duas respostas. Na ordem grau de importância “muito importante”, os elementos: modelos de produtos, definição

de papéis e atributos e medidas por atividade foram apontados, respectivamente, três, dois e um empresa. Os demais elementos, na ordem de grau de importância “importante”: critérios de entrada e saída, definição de técnicas e ferramentas e somente fases, foram apontados, respectivamente, cinco, quatro e uma empresa (Gráfico 5).

Gráfico 5: Grau de importância atribuído pelas microempresas ao definir elementos de processo de software.



É importante ressaltar que os dados apresentados nessa pesquisa implicam em um número insignificante perante o número total de microempresas no Brasil. Diversos aspectos relacionados a confiabilidade dos dados apresentados, podem ser consultados no Anexo A (capítulo 3) do presente trabalho.

3.6 Considerações sobre as Necessidades e Perfil de Microempresas

Microempresas apresentam, quase sempre, os menores índices em indicadores de qualidade e produtividade se comparadas às demais categorias de empresas. Diversos fatores que contribuem para a melhoria e produtividade do processo de software como: uso de ferramentas, implantação de programas de qualidade, adoção de práticas de Engenharia de Software, investimentos em capacitação do quadro de empregados, conhecimento de normas e modelos de qualidade entre outros, comprovam que é necessário definir políticas específicas para essa categoria de empresa (MCT/SEPIN, 2002).

A informalidade presente no processo de software agrava ainda mais a situação das microempresas impedindo, por exemplo, que diversas ações direcionadas à qualidade e produtividade sejam desenvolvidas. A formalização e execução sistemática do processo de software real podem auxiliar microempresas a melhorarem os índices de qualidade e produtividade. No entanto, alguns fatores, entre outros, devem ser considerados ao descrever o processo de software em microempresas conforme resultados das duas pesquisas apresentadas: (1º) microempresas dispõem de poucos investimentos; (2º) alto comprometimento da força de trabalho com atividades de desenvolvimento e/ou manutenção de software; (3º) pouca experiência do quadro de empregados em tópicos relacionados à qualidade (modelos, normas, padrões etc) e, (4º) necessidade de microempresas priorizarem determinados elementos do processo de software em relação a outros (NASCIMENTO & MARINHO, 2001; MCT/SEPIN, 2002).

Apesar das experiências e abordagens para descrever processos de software apresentadas na seção 3.4 apresentarem diversos pontos em comum e terem proporcionado melhorias ao processo de software das organizações nas quais foram aplicadas, é necessário considerar uma característica dessas abordagens em relação às necessidades de microempresas: a descrição do processo de software não ocorre progressivamente, ou seja, o processo de software é descrito em sua totalidade. Essa característica pode implicar em algumas dificuldades para microempresas de desenvolvimento de software:

1. Não permite que inconsistências sejam detectadas no início da descrição do processo de software. Organizações que possuam pouca experiência em modelagem de processo de software tendem a “errar” mais ao descrever o processo de software e, conseqüentemente, muito tempo seria despendido para corrigir o modelo de processo ao final de sua descrição;
2. Produz muitas informações que devem ser relacionadas, classificadas, organizadas, verificadas etc exigindo muita habilidade e tempo das pessoas envolvidas na descrição do processo;
3. Requer que muitas pessoas sejam mobilizadas em diversas atividades relacionadas à descrição do processo (em entrevistas, preenchimento de formulários, avaliando documentos etc)

dificultando a execução de atividades e prejudicando o cronograma da organização;

4. Dificulta que os custos (esforço, por exemplo) sejam escalonados para viabilizar a descrição do processo de software de acordo com a disponibilidade de investimento da organização;
5. Não é possível priorizar determinados elementos ao descrever o processo de software (por exemplo, descrever inicialmente somente atividades e depois todos os artefatos relacionados ou, descrever somente uma atividade complexa e depois as demais etc).

Modelos de processos provêm diversos benefícios para empresas de desenvolvimento de software. No entanto, a descrição de um modelo de processo de software deve considerar diversos aspectos ao modelar os processos de software em microempresas. Considerando o perfil e necessidades dessas empresas, o modelo de processo deve atender a alguns objetivos: consumir pouco tempo, baixo investimento e exigir pouca experiência.

As principais dificuldades apontadas pelas seis microempresas de desenvolvimento de software no levantamento de dados (Anexo A) foram tempo e custo. O tempo despendido pelos funcionários numa empresa de desenvolvimento figura entre os maiores custos da empresa, senão o maior. Assim, o tempo gasto na definição de modelos de processos implica em aumento de custos para a empresa. Um modelo de processo deve consumir pouco tempo e exigir pouco investimento. A falta de experiência de microempresas na descrição do processo de software exige que o modelo não seja complexo, no entanto, a complexidade é uma característica inerente ao processo de software, assim, alternativas precisam ser consideradas para diminuir o impacto sobre microempresas ao modelar seus processos de software. A necessidade de pessoas especializadas foi apontada como a terceira maior dificuldade pelas empresas de desenvolvimento no levantamento de dados (Anexo A). Além de pouca experiência, a utilização de modelos de processos complexos poderia também causar um impacto muito grande na prática informal instalada na empresa (cultura de desenvolvimento) e alimentar ainda mais a crença que modelos de processos de software são inúteis e apenas burocratizam o desenvolvimento de software.

4 ABORDAGEM ITERATIVA E INCREMENTAL

O presente capítulo tem por objetivo apresentar a abordagem iterativa e incremental para descrição de processo de software em microempresas de desenvolvimento de software. As seções seguintes apresentarão uma visão geral da abordagem e os requisitos que nortearam sua estrutura, estratégia de aplicação, estrutura da abordagem e procedimentos para aplicação para cada fase da abordagem. Paralelamente a apresentação das fases da abordagem, será demonstrada a aplicação da abordagem por meio de um estudo conduzido em uma microempresa de desenvolvimento de software localizada na cidade de Londrina – Pr que atua a pouco mais de dez anos no mercado de desenvolvimento de software. A microempresa possuía nove empregados efetivos e três estagiários. A empresa não desenvolve novos produtos de software (sob encomenda) e, se ocupa basicamente em manter os produtos já desenvolvidos. O processo de software executado pela empresa, portanto, se resume à correção, adaptação e melhoria de quatro produtos voltados para o domínio administrativo privado.

A seleção da empresa foi parametrizada em três critérios: (1) ser caracterizada como microempresa ao moldes do critério tamanho da empresa segundo força de trabalho efetiva do MCT (MCT, 2001, p. 115), (2) ter participado da pesquisa elaborada e aplicada nas etapas iniciais do presente trabalho (Anexo A) e, (3) possuir processo de software informal.

4.1 Visão Geral da Abordagem

A abordagem proposta neste trabalho tem por objetivo auxiliar microempresas a descreverem seus processos de software. A elaboração da abordagem foi amparada em experiências relatadas na literatura (seção 3.4) e em restrições que dificultam a descrição do processo de software observada em seis microempresas: pouca experiência do quadro de empregados, baixo investimento financeiro e falta de

disponibilidade de esforço de trabalho do quadro de pessoal para atividades relacionadas à qualidade e produtividade de software (seção 3.5.2).

Considerando as necessidades das seis microempresas pesquisadas (Anexo A) em relação à modelagem de processos de software, foi elaborado um conjunto de requisitos (Quadro 4) para subsidiar a estrutura da abordagem para descrever os processos de software de acordo com as necessidades apontadas pelas seis microempresas pesquisadas (Anexo A). O conjunto de requisitos foi submetido às abordagens aplicadas em estudos relatados na literatura com o objetivo de incorporar soluções já apresentadas nessas abordagens. A principal característica da abordagem, observados os requisitos apresentados no Quadro 4, é a capacidade de possibilitar a modelagem do processo de software real de maneira rápida propiciando mecanismos que auxiliem e guiem os participantes inexperientes a abstrair os elementos do processo de software. Além disso, validações e verificações devem ocorrer em etapas iniciais da modelagem com o objetivo de assegurar que as abstrações representem adequadamente o processo de software real.

Requisitos da abordagem	Requisitos				
	GSFC	LG	Elicit	Daimler	Thales
1. Treinar o quadro de empregados em modelagem de processos de software		✓	✓	✓	✓
2. Estabelecer um conjunto de atributos bem definidos para auxiliar na descrição dos elementos do processo de software			✓		
3. Validar o processo de software descrito	✓		✓	✓	✓
4. Verificar os elementos do processo de software à medida que são descritos					
5. Utilizar ferramenta automatizada para descrever o processo de software	✓		✓	✓	✓
6. Descrever inicialmente o processo de software em alto nível		✓		✓	✓
7. Priorizar determinados elementos do processo de software em relação a outros					
8. Permitir a descrição gradual do processo de software					
9. Possibilitar a revisão do modelo de processo de software descrito	✓	✓	✓	✓	✓
10. Compreender as características da microempresa (produtos, ferramentas, agentes, papéis etc)	✓	✓	✓		
11. Utilizar questionários estruturados para descrever o processo de software real			✓	✓	✓
12. Definir notações gráficas para representar os elementos e processo				✓	✓
13. Representar o processo de software real descrito por meio de um guia				✓	✓

Quadro 4: Requisitos presentes nos estudos apresentados na literatura.

Diversos requisitos propostos à abordagem foram contemplados pelas abordagens relatadas na literatura como pode ser observado no Quadro 4, entretanto, nenhuma abordagem atendeu plenamente a todos os requisitos propostos. Alguns requisitos (4, 7 e 8 – vide Quadro 4) considerados importantes às empresas pesquisadas não foram contemplados pelas abordagens propostas na literatura. Esse contexto sugeriu o desenvolvimento de uma abordagem que incorporasse as soluções já propostas na literatura e complementasse os demais requisitos não contemplados pelas mesmas.

Os três requisitos (requisitos 4, 7 e 8 – vide Quadro 4), não contemplados pelas abordagens propostas na literatura, possuem uma característica comum: a necessidade de decompor o processo de software em partes menores. Essa necessidade, apontada pelas seis microempresas pesquisadas, está relacionada diretamente com as suas principais dificuldades ao modelar o processo de software, ou seja, manipular partes menores do processo de software permite a alocação de um número menor de pessoas e recursos à atividade de modelagem do processo de software. Além disso, a modelagem progressiva do processo de software real possibilita que ocorram atividades de verificações e validações em estágios iniciais agregando consistência ao modelo de processo de software. Ao modelar partes menores do processo de software real, experiências podem ser incorporadas e reaplicadas em estágios seguintes colaborando com a aquisição de conhecimento pelas pessoas (inexperientes) envolvidas.

Com base nessas restrições, foi incorporada à abordagem a capacidade de definir os processos de software de forma iterativa e incremental. A aplicação da abordagem foi dividida em três etapas: Planejamento Inicial, Descrição do Elemento do Processo e Validação & Verificação. As próximas seções apresentarão em maiores detalhes as orientações que conduziram a elaboração da abordagem, estrutura, capacidades e fases que a compõem (vide Figura 22).

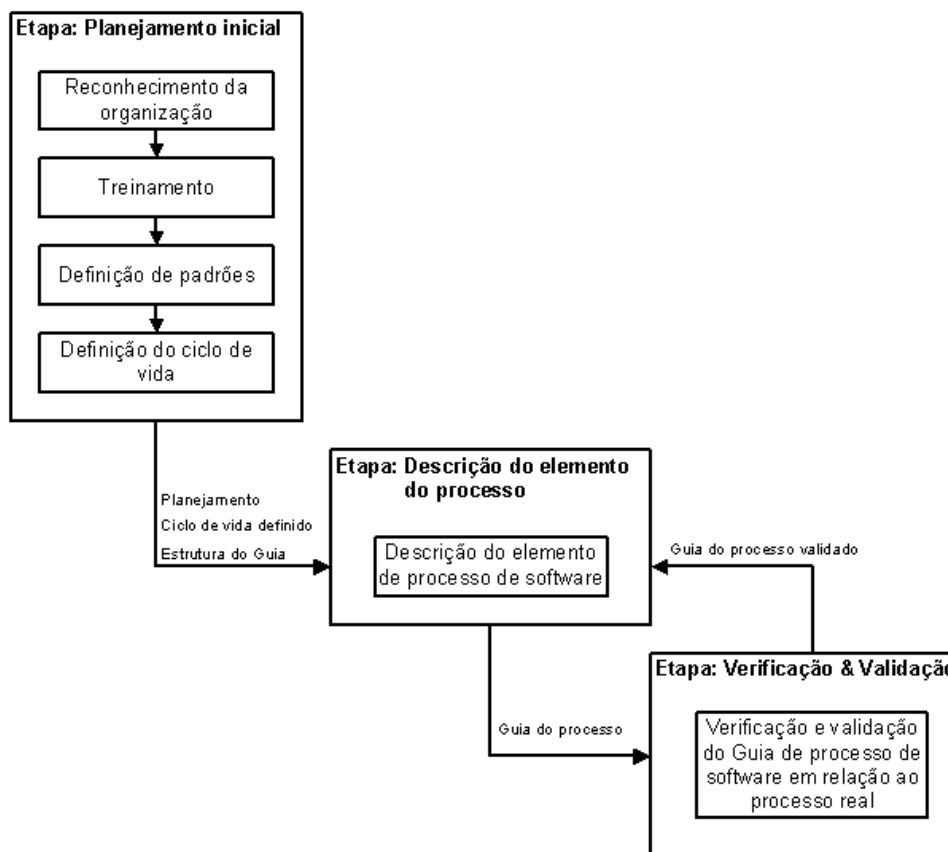


Figura 22: Etapas e fases da abordagem para descrição de processo proposta

As seções posteriores apresentarão a estrutura da abordagem bem como a justificativa de cada etapa proposta à abordagem.

4.1.1 Estrutura da Abordagem

A descrição do processo de forma incremental possibilita que o processo seja formalizado gradualmente. Considerando que microempresas têm preferências em descrever o processo de software iniciando pelas fases e atividades do processo, ou seja, pelos elementos de processo de software de alto nível, incorporou-se ao método a abordagem *top-down*.

A abordagem *top-down* pressupõe, no presente estudo, que o processo seja descrito de forma incremental e suas iterações se iniciem tendo em vista uma visão mais ampla do processo, ou seja, o processo pode ser descrito em camadas que manipulam os mesmos elementos do processo em diferentes fases do desenvolvimento

de software. Nesse caso, a descrição do processo poderia ser iniciada pelos elementos de alto nível, por exemplo, atividades (ciclo de vida). A Figura 23 mostra, a título de exemplo, a aplicação da abordagem *top-down* a qual se inicia, primeiramente, pela descrição de todas as atividades do processo (ciclo de vida) seguida pelas sub-atividades, papéis, ferramentas e artefatos de cada atividade do ciclo de vida (descrição do processo de software em profundidade).

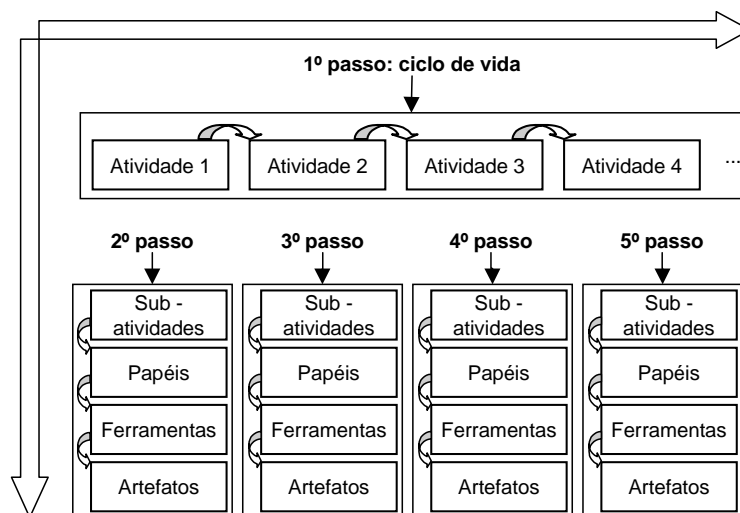


Figura 23: Descrição do Processo de Software em profundidade (*top-down*)

Deve-se considerar que a Figura 23 apresenta apenas uma visão ampla da abordagem *top-down*. Ao descrever o processo de software utilizando a abordagem *top-down*, outros elementos devem ser considerados, por exemplo, a transição entre as atividades, transição entre as sub-atividades, recursos, critérios de entrada e saída etc. Além disso, outras variações de descrição do processo de software poderiam ser aplicadas utilizando a abordagem *top-down*.

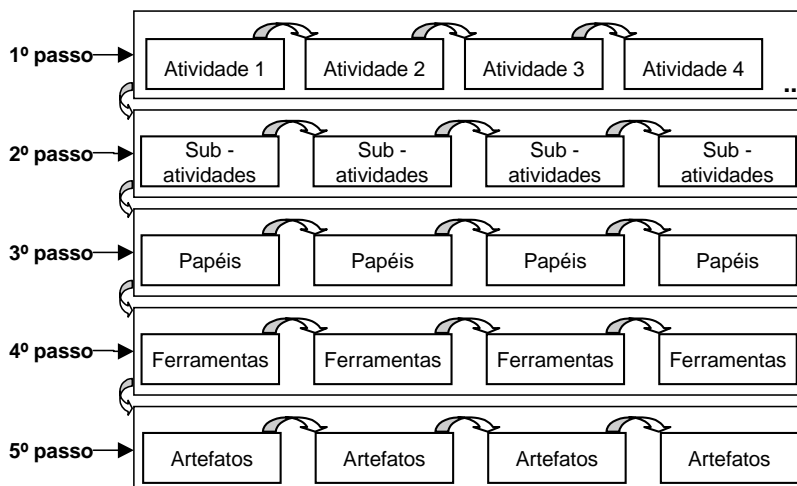


Figura 24: Descrição do Processo de Software em largura (*top-down*)

A Figura 24 apresenta uma variação na qual os elementos do processo de software são descritos em camadas (descrição do processo de software em largura). São descritas, inicialmente, todas as atividades do processo (ciclo de vida), depois, diferentemente do primeiro exemplo (Figura 23), são descritas todas as sub-atividades, após a descrição de todas as sub-atividades, é descrito todos os papéis envolvidos no processo de software, seguido da descrição de ferramentas e artefatos.

4.2 Estudo de Caso: Etapa de Planejamento Inicial

A descrição do processo de software não deve ocorrer de forma “aleatória”. A organização de desenvolvimento de software deve planejar a descrição de seus processos considerando suas características organizacionais: objetivo da descrição (mensuração, melhoria, certificação, padronização etc), mercado de atuação, tipo de produto desenvolvido, cultura organizacional, conhecimento e experiência em Engenharia de Software entre outras (MACHADO et al, 2001). As informações sobre a empresa, objetivo da descrição de processo, produtos, aspectos organizacionais e processo de software permitem delimitar o escopo da descrição do processo e dimensionar a capacidade da organização frente aos recursos necessários ao descrever o processo de software real. Considerando esse aspecto, são propostas quatro fases à Etapa de Planejamento Inicial: Reconhecimento da Organização, Treinamento,

Definição de Padrões e Definição do Ciclo de Vida. O produto resultante da Etapa de Planejamento deve ser um documento contendo cada uma das fases dessa etapa como tópicos, ou seja, o documento Planejamento deveria conter os seguintes tópicos: Reconhecimento da Organização, Treinamento, Definição de Padrões e Definição do Ciclo de Vida (HÖLTJE et al, 1994; MADHAVJI et al, 1994; BECKER-KORNSTAEDT & BELAU, 2001; MACHADO et al, 2001).

4.2.1 Fase de reconhecimento da organização

A primeira fase da Etapa de Planejamento Inicial destina-se à definição do objetivo da empresa frente à descrição do processo de software e capacidade da organização em relação à aplicação da abordagem (HÖLTJE et al, 1994, p. 4; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 71; MACHADO et al, 2001, p. 149). Descrever o processo de software envolve necessariamente o domínio de diversos conhecimentos relacionados à qualidade de software e, microempresas apresentam o maior índice de desconhecimento em temas relacionados à qualidade de software (MCT/SEPIN, 2002, p. 130-131). A média nacional de domínios de software desenvolvidos aproxima-se de 4 e 5,1 tipos de aplicações desenvolvidas em média (MCT/SEPIN, 2002, p. 37). Diferentes domínios e tipos de software podem exigir diferentes processos de software, portanto, é necessário caracterizar os produtos de softwares mantidos pela empresa. Para auxiliar a execução dessa fase, foram executados os seguintes passos na microempresa na qual o estudo foi conduzido:

Passos	Descrição
Passo 1:	Definir o objetivo da descrição de processo.
Objetivo:	Delimitar o escopo da descrição do processo.
Finalidade:	Alinhar a descrição do processo de software ao objetivo da organização.
Passo 2:	Levantar os produtos mantidos pela empresa.
Objetivo:	Definir o perfil dos produtos mantidos pela empresa.
Finalidade:	Caracterizar e definir as necessidades dos produtos mantidos pela empresa.
Como:	- Elaborar e aplicar instrumento de coleta de dados.

A primeira reunião com a empresa (julho/2002), após a aceitação da mesma em participar no estudo de caso, teve por objetivo determinar o interesse da empresa em relação à aplicação da abordagem para descrição do processo de software (passo 1). Inicialmente, foram levantadas as principais dificuldades da empresa em relação à informalidade do processo de software presente na empresa e, por fim, como a descrição do processo de software poderia contribuir para a melhoria do processo de software da empresa. A principal queixa da empresa em relação ao processo de software se referia à diversidade de práticas de engenharia de software entre os empregados da empresa. Embora muitas atividades fossem semelhantes, sua execução se diferenciava entre os empregados da empresa. Era comum a constatação de diversos erros após a execução de uma determinada atividade devido a não aplicação de um determinado procedimento. Por exemplo, após o atendimento e encaminhamento de uma solicitação de mudança de um cliente, era necessário que o agente encaminhasse um retorno (*feedback*) ao cliente relatando as providências a serem tomadas. No entanto, era comum a inexecução dessa atividade, ora por esquecimento, ora por negligência.

Diante dos fatos constatados, foi decidido que o objetivo da descrição do processo de software seria uniformizar as práticas empregadas durante a manutenção dos produtos da empresa (institucionalização do processo de software). Além do objetivo da descrição do processo de software, foi estabelecido que a descrição do processo de software fosse realizada em profundidade, ou seja, seria escolhida uma atividade do ciclo de vida. Para cada sub-atividades, seriam descritos os artefatos, ferramentas e papéis. Essa decisão foi fundamentada no fato de algumas atividades do ciclo de vida da empresa serem consideradas mais críticas pela empresa em relação a outras e necessitarem de maior atenção em um primeiro momento.

Após a definição do objetivo e estratégia de aplicação, foram coletados dados sobre os produtos mantidos pela empresa (passo 2). Foram identificados quatro produtos distintos (identificados nesse estudo como 'A', 'B', 'C' e 'D'), porém, voltados para o mesmo domínio de aplicação. A aplicação do instrumento de coleta de dados para caracterizar os produtos de software mantidos pela empresa (vide Figura 25) revelou os seguintes dados:

Figura 25: Instrumento de coleta de dados para caracterização dos produtos

Produtos	A	B	C	D
Identificação do produto				
1. Descrição do sistema	A	B	C	D
2. Data da 1ª versão	10/1997	10/1994	?/1998	10/2000
3. Nº. da versão atual	5.2	2.07	6.1	2.07
4. Nº. de clientes	172	600	400	6
5. Nº. de módulos	85	428	107	361
6. Nº. de arquivos (BD)	15	80	19	80
Caracterização do processo				
Sim/Não				
7. Está relacionado aos outros produtos da empresa? Por exemplo, utiliza a mesma ou parte da base de dados, a mesma interface gráfica etc.	Não	Não	Não	Não
8. Os agentes responsáveis pela manutenção do produto são mantidos? Por exemplo: qualquer agente pode alterar o produto ou existe um agente responsável?	Sim	Sim	Sim	Sim
9. Há diferenças na quantidade de artefatos consumidos/produzidos em relação aos outros produtos da empresa?	Não	Não	Não	Não
10. Há diferenças no formato dos artefatos gerados em relação aos outros produtos da empresa?	Não	Não	Não	Não
11. Há diferenças na quantidade de atividades executadas em relação aos outros produtos?	Não	Não	Não	Não
12. Há diferenças nas características das atividades executadas em relação a outros produtos? Por exemplo: procedimentos adotados.	Não	Não	Não	Não
13. Há diferenças na quantidade de ferramentas de desenvolvimento utilizadas em relação aos demais produtos?	Não	Não	Não	Não
14. Há diferença nas atribuições e responsabilidade dos mesmos papéis em relação aos outros produtos da empresa?	Não	Não	Não	Não
15. Há diferenças em quantidade de papéis necessários para manter o produto em relação aos outros produtos da empresa?	Não	Não	Não	Não

Devido ao fato de cada produto possuir um agente responsável, foram conduzidas quatro entrevistas realizadas no horário de trabalho sendo que cada entrevista durou, aproximadamente, 0h30min em média. Durante três dias (08h00min por dia) foram coletados dados de documentos relacionados a cada produto para complementar e validar os dados coletados nas entrevistas. A consulta foi realizada em documentos produzidos antes da entrevista para evitar que os agentes fossem influenciados e passassem, por exemplo, a produzir documentos ou executar atividades não condizentes com sua prática diária. A consulta aos documentos produzidos em datas anteriores à entrevista confirmou as respostas dadas na entrevista pelos quatro agentes responsáveis, ou seja, o processo de software não se diferenciava entre os produtos, portanto, poderia ser produzido um único guia de processo de software para todos os produtos. Entretanto, constatou-se que havia inconsistência no processo executado,

confirmando a queixa inicial da empresa, ou seja, falta de uniformidade ao executar o processo de software entre os diferentes produtos mantidos pela empresa.

4.2.2 Fase de Treinamento

A participação efetiva dos envolvidos na descrição do processo de software é fundamental devido ao fato da cultura organizacional estabelecida apresentar-se como um dos principais obstáculos frente a propostas que envolverão mudanças no ambiente de trabalho (HUMPHREY, 2001). Especial atenção deve ser dada ao iniciar a descrição do processo de software devido às incertezas que normalmente se apresentam perante mudanças numa organização. É importante que reuniões sejam realizadas junto aos participantes para esclarecer o objetivo da descrição do processo, implicações (responsabilidades), benefícios etc (BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 71; MACHADO et al, 2001, p. 149). Como discutido anteriormente, mudanças na cultura da organização nem sempre são bem-vindas e a melhor alternativa para contornar essa situação é motivar os participantes, por meio de treinamentos que esclareçam e convençam a importância da descrição do processo de software à empresa e a eles próprios (HUMPHREY, 2001).

Complementar ao treinamento, deve-se propiciar domínio sobre os temas abordados durante a aplicação da abordagem para diminuir a insegurança e incentivar a participação (SHAW, 2000; BECKER-KORNSTAEDT, NEU & HIRCHER, 2001). Além disso, outros tópicos podem ser ministrados para complementar os conhecimentos em qualidade e engenharia de software. Treinamentos complementares dependerão do objetivo da empresa, por exemplo, se a empresa pretende certificar-se, é importante que treinamentos relacionados à qualidade sejam ministrados ou, se a empresa pretende descrever o processo de software para avaliar a qualidade do processo, treinamentos em técnicas, métodos e/ou tecnologias em engenharia de software poderiam ser ministrados aos participantes. A mesma estratégia poderia ser aplicada para programas de mensuração etc. Considerando os aspectos mencionados acima, um único passo foi aplicado:

Passo 1:	Contextualização e participação.
Objetivo:	Expor aos participantes (diretos e indiretos) na aplicação da abordagem, os objetivos pretendidos, conceitos e terminologias relacionadas a processos e modelagem de processos de software, benefícios que poderão advir da descrição do processo de software tanto para os participantes quanto para a organização, importância da participação efetiva dos envolvidos.
Finalidade:	Informar, motivar e envolver os participantes na aplicação da abordagem.
Como:	<ul style="list-style-type: none"> - Apresentar na forma de palestra ou curso, os principais conceitos e terminologias utilizadas na aplicação da abordagem. Sugerem-se como tópicos termos e conceitos relacionados a processo de software, elementos do processo de software e qualidade de software (processo e produto). - A palestra ou curso pode ser ministrado pelo responsável pela modelagem do processo de software ou, empresa especializada em treinamentos ou ainda em parcerias com instituições de ensino superior.

O treinamento, realizado em um dia (final de semana), foi dividido em duas etapas: apresentação do levantamento de dados (conduzido na fase 1 da primeira etapa) e realização de um mini-curso abordando os principais conceitos relacionados à modelagem de processo de software e aplicação da abordagem. Para subsidiar as discussões sobre o levantamento de dados, foi elaborado um relatório contendo o resultado da coleta de dados e observações conduzidas na empresa. O relatório apresentou diversos problemas presentes no processo de software da empresa e que estavam relacionados à falta de uniformidade ao executar o processo de software. Entre os problemas constatados estavam: dificuldade no acesso a informações, perda de informações, confiabilidade das informações, pouca uniformidade na estrutura dos artefatos, redundância de informações e inconsistências. Ao total, foram relatados vinte e um problemas e conseqüências relacionadas à falta de descrição do processo de software.

No período da tarde, na forma de um mini-curso, foram abordados os principais conceitos relacionados à modelagem de processo de software, entre os quais: principais componentes do processo de software (atividade, artefato, papel e ferramenta), finalidade da descrição do processo de software, vantagens da institucionalização do processo de software e responsabilidades ao modelar o processo de software. Na seqüência, foi apresentada a abordagem para descrever o processo de software discutida no presente trabalho. Ao final, foi definido que a abordagem seria aplicada utilizando a abordagem *top-down* em profundidade e, a primeira atividade a ser

descrita seria a atividade de suporte, considerada como a atividade mais crítica do processo de software da empresa. A descrição do processo de software em profundidade foi amparada nos seguintes argumentos: (1º) determinadas atividades, como suporte, são mais críticas e precisavam ser abordadas em um primeiro momento e, (2º) devido ao fato dos agentes não possuírem nenhuma experiência em descrição de processo de software, a descrição em profundidade poderia contribuir com a aquisição de conhecimento e experiências que poderiam ser aplicadas na descrição de outras atividades uma vez que todos os elementos do processo de software seriam abordados.

4.2.3 Fase de Definição de Padrões

A representação do processo de software é um ponto chave na definição do processo de software de uma organização (BECKER-KORNSTAEDT & BELAU, 2000, p. 4; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 67). Qualquer representação do processo de software (seja na forma de manuais, guias ou formulários) deve possibilitar aos participantes do processo de software auxílio eficiente e acurado sobre sua execução. A utilização de hipertexto como meio para representar o processo de software tem se mostrado como uma excelente alternativa pelo fato de ser amplamente utilizado (interface utilizada na Internet) e permitir acesso rápido às informações por meio de *hyper-links* (KELLNER et al, 1998).

Antes de iniciar a descrição de qualquer elemento do processo de software, é necessário uniformizar as representações que deverão ser empregadas durante toda a aplicação da abordagem (BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 63). Considerando a falta de uniformidade dos empregados da microempresa ao executarem o processo de software e recomendações observadas nos estudos, três passos foram aplicados para definir os padrões de representação do processo de software real da microempresa:

Passos	Descrição
Passo 1	Definir as propriedades dos elementos do processo a serem descritas
Objetivo:	Padronizar a descrição dos elementos do processo de software
Finalidade:	Evitar inconsistências e orientar a descrição dos elementos do processo.
Passo 2:	Definir a estrutura do guia
Objetivo:	Definir a estrutura do guia de processos de software.
Finalidade:	Propiciar um meio para facilitar, auxiliar, informar e conduzir os participantes na execução do processo de software.
Passo 3:	Definir a notação
Objetivo:	Definir os termos e notações que serão utilizadas durante a aplicação do método.
Finalidade:	Viabilizar a comunicação e compreensão de termos e representações entre os participantes.

Os elementos que deveriam ser descritos foram definidos em reunião com a participação dos agentes: atividades, artefatos, papéis, ferramentas e fluxos entre os elementos do processo. Para cada elemento do processo de software, foi determinado um conjunto de atributos que deveriam ser utilizados para descrevê-los. Para determinar os atributos, foi utilizado, como referência, o estudo realizado por HÖLTJE et al (1994, p. 3) o qual apresenta um conjunto de propriedades descritivas, estáticas e dinâmicas para modelar os elementos do processo de software. Para cada elemento do processo de software foram definidas as propriedades descritivas, estáticas e dinâmicas:

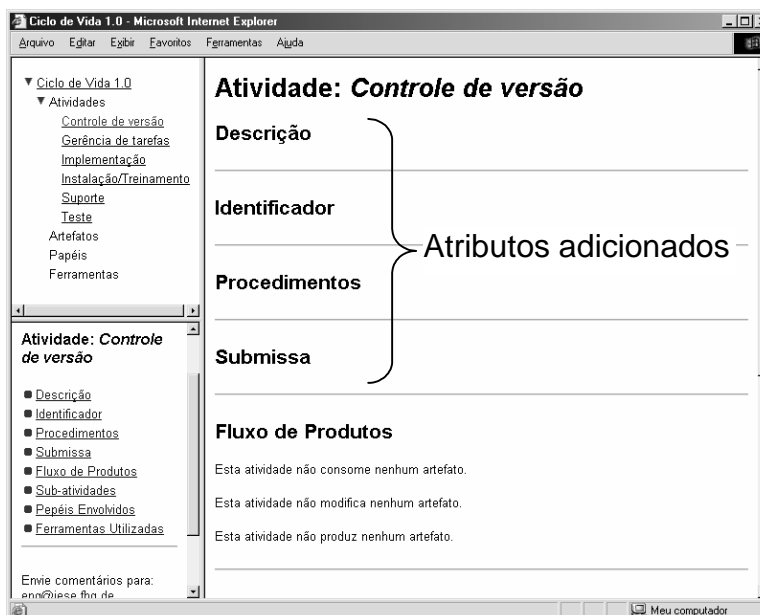
Elemento	Propriedades descritivas	Descrição
Atividade	Identificador Descrição Procedimentos Submissa	Identificador único da atividade. Descrição sucinta do objetivo da atividade. Descrição dos procedimentos da atividade. Atividade a qual está submissa.
Artefato	Identificador Descrição Caminho Modelo Tipo Formato	Identificador unido do artefato. Descrição sucinta do artefato. Local em disco no qual deve ser armazenado. Local em disco no qual está localizado o modelo. Tipo do artefato. Formato do arquivo com respectiva extensão.
Papéis	Identificador Descrição Submisso Atribuições Qualificações	Identificador unido do papel. Descrição sucinta do papel. Papel ao qual é subordinado. Atribuições (responsabilidade) do papel. Qualificações necessárias ao papel.
Ferramentas	Identificador Descrição	Identificador único da ferramenta. Descrição sucinta da aplicação da ferramenta.

Propriedades estáticas		Descrição
Atividade	Subatividades Artefatos de entrada Artefato de saída Finalidade Origem Destino Papéis envolvidos Fluxo de entrada Fluxo de saída Ferramentas utilizadas	Lista de sub-atividades (caso possua). Lista de artefatos de entrada. Lista de artefatos de saída. Finalidade do artefato de entrada/saída. Origem do artefato de entrada. Destino do artefato de saída. Papéis envolvidos na atividade. Fluxo originado em outras atividades. Fluxo de saída para outras atividades. Lista de ferramentas utilizadas na atividade.
Artefato	Sub-Artefatos Entidade produtora Entidade consumidora Entidade modificadora Ferramenta produtora	Lista de sub-artefatos. Descrição da entidade produtora. Descrição das entidades consumidoras. Descrição das entidades modificadoras. Descrição da ferramenta produtora.
Papéis	Artefatos manipulados Envolvido nas atividades Utiliza as ferramentas	Lista de artefatos manipulados pelo papel. Lista de atividades nas quais está envolvido. Lista de ferramentas utilizadas pelo papel.
Ferramentas	Artefatos gerados Utilizada nas atividades Manipulada pelos papéis	Lista de artefatos gerados pela ferramenta. Lista de atividades que utilizam a ferramenta. Lista de papéis que manipulam a ferramenta.

Propriedades dinâmicas		Descrição
Atividade	Critérios de entrada Critérios de saída Estado	Lista dos critérios de entrada. Lista dos critérios de saída. Lista dos possíveis estados da atividade.
Artefato	Critérios para produção Critérios para consumo Critérios para modificação Estado	Lista de critérios para produzir o artefato. Lista de critérios para consumir o artefato. Lista de critérios para modificar o artefato. Lista dos possíveis estados do artefato.
Ferramenta	Critérios de uso	Lista de critérios para o uso da ferramenta.

A estrutura e meio de representação do Guia de Processo de Software - GPS (passo 2) foi baseado no estudo conduzido por KELLNER et al (1998) o qual propõe a utilização de guias de processo eletrônicos (hipertextos) baseados em tecnologia Web para estruturar e disponibilizar o acesso ao mesmo. Com base nesses requisitos, foi selecionada a ferramenta Spearmint (BECKER-KORNSTAEDT, NEU & HIRCHE, 2001) para estruturar e gerar o guia de processo de software (maiores detalhes sobre a ferramenta vide 3.4.4 – Estudo 5: Thales). A ferramenta Spearmint permite estruturar o guia de forma hierárquica e gerar a mesma estrutura no formato html (Figura 26). A ferramenta também permitiu adicionar novos atributos a cada elemento do processo de software para complementar sua descrição adequando as necessidades de informação ao perfil da empresa:

Figura 26: Estrutura do Guia de Processo de Software



Naturalmente, para atender ao passo 3, foi selecionada a notação da própria ferramenta Spearmint para representar cada elemento do processo de software. A seleção da notação foi precedida por uma discussão entre os agentes. A discussão abordou cada representação com o objetivo de familiarizar os participantes com as representações adotadas pela ferramenta. Foram utilizados exemplos de guias de processo de software gerados pela ferramenta para exemplificar a estrutura e representação dos elementos do processo de software. Maiores detalhes sobre a notação utilizada pela ferramenta Spearmint pode ser encontrado nos Quadro 2 e Quadro 3 ou em BECKER-KORNSTAEDT, NER & HIRCHE (2001, p. 66-69). A aplicação da fase foi finalizada com uma reunião entre os agentes para aprovar a estrutura e notação que seriam utilizadas na descrição do processo de software real da empresa.

4.2.4 Fase de Definição do Ciclo de Vida

Independente de a empresa modelar um ou vários processos de software, é importante que a prática atual seja explicitada em alto nível, ou seja, o ciclo de vida utilizado no processo de software (BRIAND et al, 1998, p. 253; BECKER-

KORNSTAEDT & BELAU, 2000, p. 7; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 71).

A descrição em alto nível do processo de software (ciclo de vida) é uma fase importante, pois permite uma visão ampla das principais etapas empregadas no desenvolvimento/manutenção de software e é particularmente importante na abordagem proposta nesse trabalho, pelo fato da adoção da abordagem *top-down* como estratégia de descrição do processo de software. Ao descrever o processo de software utilizando a abordagem *top-down*, é necessário que o elemento de software a ser formalizado se enquadre no ciclo de vida de desenvolvimento da organização.

Outro fator relevante ao definir o modelo de ciclo de vida da empresa se refere à possibilidade da mesma possuir mais de um produto de software. Produtos distintos podem exigir, por exemplo, que algumas atividades sejam incorporadas ou não. Portanto, a definição do ciclo de vida deve ser uma atividade realizada com cautela evitando que generalizações acobertem necessidades específicas de determinados produtos. Deste modo, foram executados os seguintes passos para determinar o ciclo de vida da empresa:

Passos	Descrição
Passo 1:	Coletar dados sobre o ciclo de vida da empresa.
Objetivo:	Capturar as principais atividades de alto nível (ciclo de vida) executadas ao desenvolver/manter produtos de software.
Finalidade:	Compreender como os participantes executam as principais atividades (ciclo de vida) relacionadas ao processo de software.
Como:	<ul style="list-style-type: none"> - Selecionar os produtos de software; - Identificar os agentes responsáveis pela execução do processo de software (inclusive os agentes que executam atividades eventuais); - Acompanhar a execução das atividades realizadas por esses agentes.
Passo 2:	Representar o ciclo de vida.
Objetivo:	Representar o ciclo de vida do processo de software.
Finalidade:	Explicitar o ciclo de vida praticado ao manter o produto de software selecionado.
Como:	<ul style="list-style-type: none"> - Verificar se existem variações nas atividades executadas para manter os produtos. - No caso de produtos que possuam mais de um agente envolvido, verificar se existe variação na execução das atividades entre os agentes envolvidos na manutenção do mesmo produto. - Representar as principais atividades e fluxos de controle (seqüência) entre as atividades executadas por cada agente responsável utilizando a notação previamente definida na Fase Definição de Padrões;

Passo 3:	Formalizar o ciclo de vida
Objetivo:	Descrever o ciclo de vida do processo de software.
Finalidade:	Uniformizar o ciclo de vida praticado pelos responsáveis do produto selecionado.
Como:	<ul style="list-style-type: none"> - Verificar se existem diferenças entre as atividades do ciclo de vida executadas para manter cada produto. - Verificar se existem diferenças entre as atividades do ciclo de vida executadas pelos agentes do mesmo produto. - Descrever as atividades do ciclo de vida utilizando linguagem natural e representação gráfica.

A coleta de dados sobre o ciclo de vida (passo 1) foi executada durante quatro dias (4h00m por dia) no período da tarde. No primeiro dia, foram identificados os agentes responsáveis por cada produto da empresa, grau de envolvimento e atividades de alto nível executadas por cada um. Para cada agente e seu respectivo produto descrito no Instrumento de Coleta 2 (vide anexo B), foram definidas as atividades executadas por meio de observações do processo de software real executado por esse agente.

Após a coleta das atividades executadas por um agente ao manter um determinado produto, foi realizada a representação do fluxo entre essas atividades. Os dados coletados nos Instrumentos 2, 3 e 4 (vide Anexo B) foram sumarizados para permitirem o cruzamento de dados entre atividades e produtos e entre atividade, produto e agentes. O objetivo foi identificar as atividades de alto nível (fases) executadas por cada agente em relação aos produtos mantidos, o número de ocorrência das atividades e a sua frequência em relação aos quatro produtos (apresentados como A, B, C e D nesse trabalho) mantidos pela empresa. Os dados sumarizados podem ser observados nas Figura 27 e Figura 28.

Figura 27: Ocorrência e frequência das atividades de alto nível em relação aos produtos

Produtos ►		Produto A	Produto B	Produto C	Produto D
Atividades ▼	Frequência ►	S/E/N	S/E/N	S/E/N	S/E/N
1. Atendimento		S	S	S	S
2. Gerência de tarefas		S	S	S	S
3. Implementação		S	S	S	S
4. Testes		S	S	S	S
5. Controle de versão		S	S	S	S
6. Teste de instalação		S	S	S	S
7. Logística		S	S	S	S
8. Instalação e treinamento		S	S	S	S
Legenda:	S: Sistemático. A atividade ocorre independente de qualquer situação.				
	E: Eventual. A atividade pode ocorrer ou não, dependendo da situação.				
	N: Não Ocorre. A atividade não ocorre no ciclo de vida desse produto.				
	Obs: Somente uma opção deve ser assinalada para cada ocorrência da atividade ao manter o produto, ou seja, ou "S", ou "E" ou "N".				

Figura 28: Frequência das atividades de alto nível executadas pelos agentes

Produto: A, B, C e D.									
Atividade ▼	Agente ►	Agente 1 Produto A	Agente 2 Produto B	Agente 3 Produto C	Agente 4 Produto D	Agente 5 Produto ABCD	Agente 6 Produto ABCD	Agente 7 Produto ABCD	Agente 8 Produto ABCD
	1. Atendimento		S	S	S	S			
2. Gerência de tarefas		S	S	S	S				
3. Implementação		S	S	S	S				
4. Testes		S	S	S	S				
5. Controle de versão		S	S	S	S				
6. Teste de instalação						E			
7. Logística							E		
8. Instalação e Treinamento								E	E
Legenda:	S: Sistemático. A atividade é executada independente de qualquer situação.								
	E: Eventual. A atividade pode ou não ser executada, dependendo da situação.								
	N: Não Ocorre. A atividade não é executada pelo agente no ciclo de vida desse produto.								
	Obs: Somente uma opção deve ser assinalada para cada ocorrência da atividade do produto, ou seja, ou "S", ou "E" ou "N".								

No segundo e terceiro dia, foram conduzidas observações sobre o processo de software executado por cada agente (apresentados como Agente 1, 2, 3, 4, 5, 6, 7 e 8 no presente trabalho) para averiguar se as etapas relatadas nos instrumentos de coleta de dados eram executadas (vide Figura 29). No quarto dia, foram realizadas consultas a documentos produzidos por cada agente ao manter os produtos de software (documentos gerados em datas anteriores à entrevista para complementar os dados coletados nas entrevistas e observações realizadas). O passo 2 foi aplicado duas vezes até a versão final da representação do ciclo de vida de manutenção da empresa. Na primeira representação do ciclo de vida, foram identificadas cinco atividades comuns de

alto nível praticadas pelos agentes (não foram encontradas atividades específicas a um único produto).

Os dados coletados sobre a execução do processo de software real eram acondicionados no Instrumento 6 (vide Anexo B). A disposição dos dados na forma de matriz (Figura 29) permitiu classificar e identificar as atividades executadas pelos agentes. Além disso, atividades classificadas como eventuais indicaram que poderiam ocorrer fluxos condicionais entre as atividades. Esses dados indicaram a necessidade de observar quais eram os critérios que definiam se uma atividade deveria ou não ser executada.

Figura 29: Frequência de execução das atividades entre os agentes

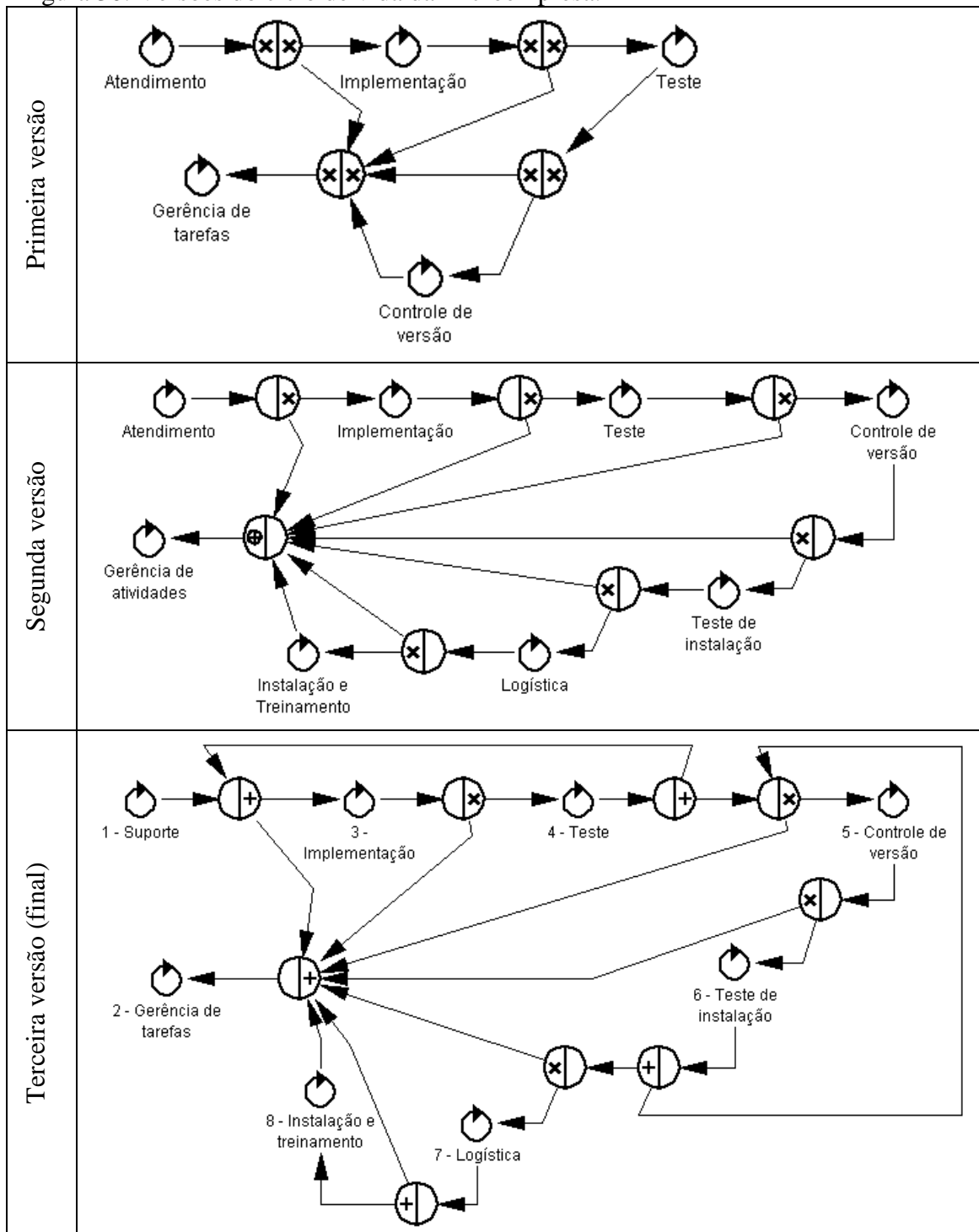
Produto: A, B, C e D.									
Atividade ▼	Agente ►								
	Agente 1 Produto A	Agente 2 Produto B	Agente 3 Produto C	Agente 4 Produto D	Agente 5 Produto ABCD	Agente 6 Produto ABCD	Agente 7 Produto ABCD	Agente 8 Produto ABCD	
1. Atendimento	S	S	S	S					
2. Gerência de tarefas	S	S	S	S					
3. Implementação	S	S	S	S					
4. Testes	S	S	S	S					
5. Controle de versão	S	S	S	S					
6. Teste de instalação					E				
7. Logística						E			
8. Instalação e Treinamento							E	E	
Legenda:	S: Sistemático. A atividade é executada independente de qualquer situação.								
	E: Eventual. A atividade pode ou não ser executada, dependendo da situação.								
	N: Não Ocorre. A atividade não é executada pelo agente no ciclo de vida desse produto.								
	Obs: Somente uma opção deve ser assinalada para cada ocorrência da atividade do produto, ou seja, ou "S", ou "E" ou "N".								

Até a versão final do modelo de ciclo de vida foram criadas três versões do ciclo de vida (Figura 30). O fluxo de controle seqüencial observado entre as atividades do ciclo de vida pode ser caracterizado como Cascata. As atividades eram executadas de forma seqüencial iniciando na atividade de Suporte e finalizando na atividade de Treinamento.

A primeira versão apresentou cinco atividades, a segunda versão apresentou oito atividades. A diferença entre as duas primeiras versões se deu pelo fato dos dados terem sido coletados apenas dos quatro primeiros agentes. Ao coletar os dados dos demais agentes, três atividades foram adicionadas ao modelo de ciclo de vida

da microempresa. A terceira versão se diferenciou pela divisão da atividade Instalação e Treinamento em duas atividades distintas e identificação de novos fluxos condicionais entre as atividades. A representação do modelo de ciclo de vida foi refeita e submetida novamente a nova avaliação.

Figura 30: Versões do ciclo de vida da microempresa.



O último passo (terceiro) foi realizado em 15 dias (4h por dia). Dados foram coletados por meio de observação do processo de software real em execução, consulta a documentos gerados durante o processo e entrevista junto aos agentes executores do processo. A cada três dias, os dados eram sumarizados para verificar inconsistências na execução do fluxo das atividades do ciclo de vida formalizado. Novas atividades de alto nível não foram observadas, entretanto, percebeu-se, no quinto dia de coleta de dados, que algumas atividades eram condicionais, ou seja, dependendo da circunstância, poderiam ou não ser executadas. As condições para execução ou não de uma atividade eram aplicadas a todos os produtos. Por exemplo, quando uma solicitação do cliente não envolvia qualquer alteração sobre o produto, somente era executada a atividade de Suporte, finalizando o ciclo na atividade Gerência de Tarefas. Então, ao ciclo de vida da terceira versão, foram adicionadas representações que condicionavam a execução de uma determinada atividade. Considerando essas observações, uma nova versão do ciclo de vida foi elaborada com a adição de elementos gráficos (junção e disjunção) entre as atividades que demonstravam os fluxos de controle condicionais entre as atividades (Figura 30 – terceira versão).

A atividades de Suporte, dependendo de determinadas condições poderia ser encerrada na atividade de Gerência de Tarefas ou passar o fluxo de controle para a atividade de Implementação (disjunção inclusiva – “or”). A atividade de Implementação, ao ser finalizada gerava obrigatoriamente dois fluxos de controle, um à atividade de Teste e outro à atividade Gerência de Tarefas. A atividade Teste poderia gerar dois fluxos diferentes: caso encontrasse erros nos testes, era gerado um fluxo de retorno à atividade de Implementação ou, caso contrário, gerava um fluxo à atividade de Controle de Versão e Gerência de Tarefas (disjunção exclusiva “xor” seguida de conjunção “and”). A atividade Controle de Versão, por sua vez, gerava dois: um à atividade Gerência de Tarefas e outro à atividade Teste de Instalação obrigatoriamente. A atividade Teste de Instalação ao ser concluída, poderia gerar dois fluxos de controle: um à atividade Controle de Versão caso constataste problemas no instalador ou (disjunção exclusiva “xor”) às atividades Gerência de Tarefas e Logística. A atividade de Logística poderia gerar dois fluxos de controle: somente à atividade Gerência de Tarefas caso não houvesse a necessidade de instalação e treinamento e/ou (disjunção inclusiva “or”) fluxo para Instalação e Treinamento. A atividade Instalação e

Treinamento, por sua vez, gerava um único fluxo de controle: para atividades de Gerência de Tarefas. Finalmente, a atividade de Gerência de Tarefas poderia receber fluxo de controle de pelos menos uma das demais atividades do ciclo de vida.

Após a modificação do ciclo de vida como observada na Figura 30 (terceira versão), prosseguiu-se a coleta de dados sobre a execução das atividades por mais dez dias. A execução das atividades pelos agentes conforme determinada no ciclo de vida se manteve e nenhuma inconsistência foi constatada.

4.3 Etapa de descrição do elemento do processo

O objetivo da Etapa de Descrição do Elemento do Processo é adquirir todas as informações necessárias para descrever o elemento do processo de software. As fontes de informações incluem entrevistas, observações, análise de documentos da organização e artefatos de entrada e saída do processo de software. Para alcançar o objetivo dessa etapa foi realizada a fase Descrição do Elemento de Processo de Software. Nessa seção, será apresentada em detalhes apenas a descrição e validação da atividade de Suporte da empresa (sub-atividades, artefatos, papéis e ferramentas). Ao final da seção, será apresentado um resumo da descrição e validação das demais atividades do ciclo de vida da empresa.

4.3.1 Fase de Descrição do Elemento do Processo de Software

Diversas informações são necessárias ao executar um processo de software: como, quem, quando, onde, o quê. Ao definir os elementos do processo de software, quatro perspectivas foram consideradas (CURTIS, KELLNER & OVER, 1992, p. 77):

- **Perspectiva Funcional:** *O que é feito.* Representa quais sub-atividades estão sendo executados e quais fluxos de entidades de informação (por exemplo: dados, artefatos, produtos) são relevantes a essa atividade do processo de software;

- **Perspectiva Comportamental:** *Quando é feito e como é controlado.* Representa quando as sub-atividades são executadas (por exemplo, em seqüência) bem como os aspectos de como elas são executadas por meio de *loops*, interações, condições de tomada de decisões complexas, critérios de entrada e saída;
- **Perspectiva Organizacional:** *Quem executa o elemento e onde ele é executado.* Representa onde e por quem (quais agentes) as sub-atividades do processo de software são executadas na empresa e, também, os mecanismos físicos de comunicação usados para transferir as entidades de informação, mídias físicas e a localização usada para armazená-las;
- **Perspectiva “Informacional”:** *Quais entidades de informação estão envolvidas e como as entidades de informação estão relacionadas.* Representam as entidades de informação produzidas ou manipuladas pelas sub-atividades do processo de software. Essas entidades incluem dados, artefatos, produtos (intermediários e finais) e objetos. Essa perspectiva inclui tanto a estrutura da entidade de informação como os relacionamentos entre elas.

Os diversos elementos do processo de software presentes em uma determinada perspectiva podem se relacionar a outros elementos presentes nas demais perspectivas (por exemplo, os fluxos de atividades presentes na perspectiva funcional podem se relacionar às entidades de informações produzidas ou consumidas na perspectiva informacional), dessa forma, dificilmente um elemento do processo de software poderia ser definido utilizando apenas uma perspectiva. Portanto, ao definir um elemento do processo de software, esse deve ser submetido às quatro perspectivas para averiguar se o mesmo atende as necessidades de informação de cada perspectiva. Por conseguinte, foram realizados os seguintes passos na microempresa:

Passo 1:	Descrever o elemento do processo de software
Objetivo:	Descrever o elemento do processo de software.
Finalidade:	Atribuir visibilidade ao processo de software progressivamente.
Como:	<ul style="list-style-type: none"> - Selecionar o elemento do processo de software de acordo com a estratégia de descrição definida no passo 1 dessa fase; - Elaborar instrumentos de coleta de dados estruturados. - Descrever o elemento do processo de software de acordo com a notação previamente definida na Fase de Definição de Padrões (Passo 2).

4.3.2 Instrumentos utilizados na descrição dos componentes do processo

O preenchimento e análise dos instrumentos de coleta de dados bem como a modelagem do processo com base nesses instrumentos será exemplificado somente uma vez para não tornar aplicação da abordagem repetitiva e extensa. Será apresentado um exemplo de cada instrumento com dados reais coletados durante a realização do estudo e as demais informações serão resumidas na seção seguinte. O preenchimento do Instrumento de coleta de dados sobre atividades inicia com a identificação da fonte (Agente entrevistado e observado), data e horário de início e término da coleta (vide Figura 31 – linha ‘A’). A identificação única da atividade observada, nesse caso, foi definida por meio de um nome único (linha ‘B’). A linha ‘C’ é utilizada para determinar a atividade de alto nível (uma fase do ciclo de vida, por exemplo) a qual está submetida à atividade descrita. Atividades complexas que demandam diversas ações distintas para cumprir um objetivo podem requerer sua divisão. Por exemplo, a atividade Engenharia de Requisitos poderia ser dividida em outras sub-atividades: Coleta, Análise, Modelagem, Verificação & Validação e Gerenciamento. Todas essas submissas à atividade de Engenharia de Requisitos.

A linha ‘D’ é utilizada para descrever, objetivamente, a atividade em questão. A linha ‘E’ é utilizada para classificar o tipo da atividade. Caso a atividade seja composta por mais de uma sub-atividade, essa é classificada como ‘Atividade’, caso contrário, a atividade não agrega outras atividades (atômica), é classificada como ‘Sub-atividade’. Atividades classificadas como sub-atividades devem, obrigatoriamente, ser submissas hierarquicamente a outra atividade (linha ‘C’). A linha ‘F’ depende da classificação da atividade descrita na linha ‘E’, ou seja, se a atividade é classificada como sub-atividade, essa é, portanto, tipo ‘Executável’. Atividades tipo ‘Executável’ são realizadas por agentes. Caso a atividade seja de alto nível (classificada como ‘Atividade’ na linha ‘E’), é tipo ‘Não-executável’. Atividades que agrupam outras atividades não são realizadas por agentes, apenas são utilizadas para agrupar um conjunto de sub-atividades com o objetivo, por exemplo, de organizar ou diminuir a complexidade. Normalmente, atividades ‘Não-executáveis’ são fases do ciclo de vida do desenvolvimento do software.

Figura 31: Instrumento de coleta de dados 7 (atividades) aplicado no estudo de caso.

A	Fonte: Agente 1 Data Coleta: 02/09/2002		Hora início: 14:00 Hora término: 18:00		
Propriedades Descritivas ("Perspectiva Informacional")					
B	Identificador:	Atendimento			
C	Hierarquia:	1. Não está submissa a outra atividade			
D	Descrição:	Atividade responsável pelo registro, análise e encaminhamento de solicitações (Ocorrências) de clientes no que se refere a erros do software, adição de novas funcionalidades, orientação a clientes etc.			
E	Classificação:	<input checked="" type="checkbox"/> Atividade	<input type="checkbox"/> Sub-Atividade		
F	Tipo:	<input type="checkbox"/> Atividade Executável	<input checked="" type="checkbox"/> Atividade Não executável		
G	Procedimento(s):	1. Preencher todos os dados da solicitação; 3. Identificar o usuário e funcionário que registrou a solicitação; 4. Encaminhar a solicitação ao agente responsável; 5. Definir a prioridade da solicitação; 6. Retornar ao usuário solicitante sobre os encaminhamentos;			
H	Subatividades:	1. Registrar Solicitação; 2. Classificar Solicitação; 3. Analisar Ocorrência; 4. Retornar Feedback; 5. Gerenciar Tarefa.			
I	Meta(s):	1. Registrar corretamente as solicitações e encaminhar a ocorrência.			
Propriedades Estáticas (Perspectiva Funcional e Organizacional)					
J	Atividade / Artefato	Solicitação	Ocorrência	Feedback	Controle de Tarefa
	Registrar Solicitação	Consumido	Produzido		
	Classificar Ocorrência		Consumido		
	Analisar Ocorrência		Modificado		
	Retornar Feedback		Consumido	Produzido	
	Atualizar Ocorrência		Modificado		
	Controlar Tarefa	Consumido	Consumido	Consumido	Produzido
L	Papéis:	Atendente e Cliente/usuário;			
M	Fluxo de entrada entre atividades:	Não possui			
N	Fluxo de saída entre atividades:	1. Implementação 2. Gerência de tarefas			
O	Ferramentas:	Gerenciador de Ocorrência, Gerenciador de E-mail e Gerenciador de Tarefas.			
Propriedades Dinâmicas (Perspectiva Comportamental)					
CRITÉRIOS DE ENTRADA					
P	Atividade / Artefato	Solicitação do Cliente	Ocorrência	Feedback	Controle de Tarefa
	Registrar Solicitação	Dados da solicitação			
	Classificar Ocorrência		Registrada		
	Analisar Ocorrência		Classificada		
	Retornar Feedback		Analísada		
	Atualizar Ocorrência		Encaminhada		
	Controlar Tarefa	Preenchida	Encaminhada	Preenchido	
CRITÉRIOS DE SAÍDA					
Q	Atividade / Artefato	Solicitação	Ocorrência	Feedback	Controle de Tarefa
	Registrar Solicitação		Registro completo		
	Classificar Ocorrência		Classificada		
	Analisar Ocorrência		Analísada		
	Retornar Feedback		Encaminhada	Usuário informado	
	Atualizar Ocorrência		Encaminhada		
	Controlar Tarefa	Consumido	Consumido	Usuário informado	Registro completo
R	Estados possíveis da atividade:	Em espera, Cancelada e Finalizada.			

Os procedimentos a serem realizados na atividade são relacionados na linha ‘G’. O fato de serem listados numericamente não indica necessariamente a ordem de execução, no entanto, sinalizam que todos os itens relacionados devem ser cumpridos pelo agente para que a atividade seja concluída. Na linha ‘H’ são relacionadas as sub-atividades (caso a atividade descrita seja classificada como ‘Atividade’ (linha ‘E’) da atividade. A numeração das sub-atividades também não indica ordem de execução. O resultado global da atividade descrita é delineado na linha ‘I’, ou seja, o que se espera da atividade após finalizada.

Na linha ‘J’ (que envolve mais de uma linha dependendo do número de sub-atividades pertencentes à atividade descrita, portanto, o número de linha e coluna pode variar conforme a atividade) são descritas na forma de uma matriz a relação ‘Atividade x Artefato’. A relação entre esses dois elementos pode ser classificada em três tipos: produzido (artefato produzido pela atividade), consumido (artefato utilizado na execução da atividade) ou modificado (artefato utilizado e modificado pela atividade). Os agentes envolvidos na atividade são relacionados na linha ‘L’. Na linha ‘M’ são relacionados os fluxos de entradas da atividade descrita. No caso do exemplo, a atividade de Suporte é a primeira atividade do ciclo de vida da empresa e, portanto, não possui fluxo de entrada. As saídas para outras atividades são relacionadas na linha ‘N’ (os números não indicam seqüência). Na linha ‘O’ são descritas todas as ferramentas utilizadas na execução da atividade. São consideradas ferramentas: ferramentas automatizadas (CASES e não CASES), hardware (por exemplo, estação de teste, máquinas executando sistemas operacionais diferentes etc).

Os critérios que determinam o início e término de uma atividade são registrados nas linhas ‘P’ e ‘Q’ respectivamente. As informações são dispostas na forma de matriz. Nesse estudo de caso, os critérios foram baseados na relação atividade – artefato. No entanto, os critérios de entrada e saída podem abranger a relação entre outros elementos do processo de software. Além disso, o critério (seja ele de entrada ou saída) deve ser adequadamente especificado para evitar interpretações equivocadas. Por exemplo, a primeira relação na linha ‘P’ (entre a atividade Registrar solicitação e o artefato Solicitação do cliente) da Figura 31 indica “Dados da solicitação” como critério de entrada. Entretanto, quais são os dados? Quais informações são obrigatórias? Essas

questões não podem ficar sem respostas. Dessa forma, ao descrever o processo de software, os critérios de entrada e saída também devem ser especificados no Guia de Processo de Software. Finalmente, na linha ‘R’ são descritos os possíveis estados da atividade. Os estados também devem ser detalhados no Guia de Processo de Software para definir exatamente quando uma atividade entra ou sai de um determinado estado.

Após a descrição das atividades e sub-atividades da Etapa de Suporte (seguindo a ordem de descrição dos elementos do processo de software do estudo de caso: atividades/sub-atividades, artefatos, papéis e ferramentas) foi iniciado a descrição dos artefatos. Foi utilizado o instrumento 9. A Figura 32 apresenta um exemplo do instrumento aplicado ao artefato Ocorrência.

Figura 32: Instrumento de coleta de dados 8 (artefatos) aplicado no estudo de caso.

A	FONTE: BANCO DE DADOS DE OCORRÊNCIAS DATA COLETA: 03/09/2002		HORA INÍCIO: 08:00 HORA TÉRMINO: 08:21	
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)				
B	Identificador:	<i>Ocorrência.</i>		
C	Descrição:	<i>Documento no qual são registradas as informações de solicitação de suporte do cliente/usuário bem como seu andamento.</i>		
D	Caminho:	<i>//ServidorA/Ocorrência.exe (software que gerencia as ocorrências)</i>		
E	Modelo:	<i>Não possui modelo</i>		
F	Tipo:	<input type="checkbox"/> Arquivo-Fonte	<input type="checkbox"/> Componente	<input checked="" type="checkbox"/> Documentação
Propriedades Estáticas (Perspectiva Funcional e Organizacional)				
G	Sub-Artefatos:	<i>Não possui sub-artefato</i>		
H	Atividades produtoras:	<i>1. Registrar solicitação.</i>		
I	Atividades consumidoras:	<i>1. Classificar ocorrência; 2. Retornar feedback. 3. Controlar tarefa</i>		
J	Atividades modificadoras:	<i>1. Analisar ocorrência 2. Atualizar ocorrência.</i>		
L	Ferramentas produtoras:	<i>1. Gerenciador de ocorrências</i>		
Propriedades Dinâmicas (Perspectiva Comportamental)				
M	Estados:	<i>Registrada, Analisada, Classificada, Cancelada, Finalizada, Em Espera.</i>		

A linha 'A' identifica a fonte da qual os dados foram coletados, data, hora início e fim da coleta de dados. O identificador único do artefato é descrito na linha 'B'. Na linha 'C' é apresentada uma breve descrição do artefato. A localização do artefato (por exemplo, diretório, arquivo físico ou software que manipulado o artefato) é definida na linha 'D'. Na linha 'E' é especificado o formato padrão do artefato (modelo), caso exista.

A classificação do artefato (tipo) é definida na linha 'F'. Caso o artefato possua sub-artefatos, esses são relacionados na linha 'G'. Nas linhas 'H', 'I' e 'J' são relacionadas às atividades que podem produzir, consumir ou modificar o artefato, respectivamente. No caso de artefatos produzidos ou vinculados a uma determinada ferramenta (por exemplo, editor de texto, ambiente de programação etc), essa é descrita na linha 'L'. Finalmente, na linha 'M', são descritos os possíveis estados que o artefato pode assumir.

Os dados sobre os papéis foram realizados por intermédio do instrumento de coleta de dados 10. Um exemplo do aplicado ao estudo de caso pode ser observado na Figura 33. O instrumento de dados foi preenchido por meio de entrevista de todos os agentes que realizavam o mesmo papel. O preenchimento do instrumento inicia com a identificação da fonte, data e hora início e final da coleta de dados (linha 'A'). O papel é identificado com um nome único na linha 'B' e uma breve descrição do papel é especificada na linha 'C'.

Na linha 'D' é determinado o papel imediatamente superior ao papel descrito (a quem o papel descrito está subordinado, caso seja). As atribuições (responsabilidades) do papel são listadas na linha 'E'. Na linha 'F' são listadas todas as qualificações necessárias ao papel para cumprir suas responsabilidades. Os artefatos manipulados pelo papel são relacionados na linha 'G'. As atividades que devem ser realizadas pelo papel (atividades de responsabilidade do papel) são listadas na linha 'H' e, as ferramentas utilizadas durante a realização de suas atividades são relacionadas na linha 'I'.

Figura 33: Instrumento de coleta de dados 10 (papéis) aplicado ao estudo de caso

A	FONTE: AGENTE 1		HORA INÍCIO: 14:02
	DATA COLETA: 04/09/2002		HORA TÉRMINO: 14:37
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)			
B	Identificador:	<i>Atendente.</i>	
C	Descrição:	<i>Papel responsável pelo atendimento ao cliente, registro, encaminhamento, acompanhamento de ocorrências e retorno da situação da ocorrência ao cliente/usuário.</i>	
D	Hierarquia:	<i>Subordinado diretamente ao Gerente de Tarefas.</i>	
E	Atribuições:	<ol style="list-style-type: none"> 1. <i>Atender ao cliente (esclarecer dúvidas, instruir, orientar);</i> 2. <i>Registrar os dados das ocorrências solicitadas pelo cliente/usuário;</i> 3. <i>Analisar a viabilidade das ocorrências;</i> 4. <i>Classificar as ocorrências segundo o tipo.</i> 5. <i>Manter atualizado os dados da ocorrências;</i> 6. <i>Sempre manter o cliente informado sobre o andamento da ocorrência.</i> 7. <i>Registrar todas as atividades realizadas no controle de tarefas.</i> 	
F	Qualificações:	<ol style="list-style-type: none"> 1. <i>Conhecimento sobre as funcionalidades do software da empresa;</i> 2. <i>Conhecimento básico sobre programação em Delphi;</i> 3. <i>Conhecimento básico sobre o banco de dados Interbase;</i> 4. <i>Conhecimento sobre o sistema operacional Windows;</i> 5. <i>Conhecimento básico sobre configurações de periféricos;</i> 6. <i>Treinamento em atendimento ao público.</i> 7. <i>Domínio dos softwares: gerenciador de e-mails, gerenciador de ocorrências e gerenciador de tarefas.</i> 	
Propriedades Estáticas (Perspectiva Funcional e Organizacional)			
G	Artefatos manipulados:	<ol style="list-style-type: none"> 1. <i>Solicitação do cliente;</i> 2. <i>Ocorrência;</i> 3. <i>Controle de tarefas.</i> 4. <i>Feedback</i> 	
	Executa as Atividades:	<ol style="list-style-type: none"> 1. <i>Controlar tarefas;</i> 2. <i>Analisar ocorrência;</i> 3. <i>Classificar ocorrência;</i> 4. <i>Registrar ocorrência;</i> 5. <i>Atualizar ocorrência;</i> 6. <i>Retornar feedback.</i> 	
H			
I	Utiliza as ferramentas:	<ol style="list-style-type: none"> 1. <i>Gerenciador de e-mails;</i> 2. <i>Gerenciador de ocorrências;</i> 3. <i>Gerenciador de tarefas.</i> 	

A coleta de dados sobre as ferramentas utilizadas durante o processo da empresa foi realizada por meio do instrumento de coleta de dados 9. A Figura 34 apresenta um exemplo aplicado ao estudo. Na seqüência, cada item do artefato é detalhado. Os dados da identificação, data e hora da coleta são descritos na linha ‘A’. Um identificador único para a ferramenta é definido na linha ‘B’. A linha ‘C’ contém uma descrição sucinta da ferramenta e na linha ‘D’ é definido o tipo da ferramenta (se é utilizada para implementar/modificar o software ou apenas utilizadas em documentações relacionadas ao produto). Os artefatos relacionados às ferramentas são listados na linha ‘E’. Na linha ‘F’ são relacionadas as atividades que utilizam a ferramenta e, por último, na linha ‘G’, são descritos os critérios de uso da ferramenta (por exemplo, permissões, versões, regras de formatos etc).

Figura 34: Instrumento de coleta de dados 9 (ferramentas) aplicado ao estudo de caso

A	FONTE: AGENTE 1		HORA INÍCIO: 14:07
	DATA COLETA: 05/09/2002		HORA TÉRMINO: 14:26
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)			
B	Identificador:	<i>Gerenciador de ocorrências</i>	
C	Descrição:	<i>Ferramenta utilizada para registrar e manter as ocorrências de mudanças solicitadas pelos clientes/usuários.</i>	
D	Tipo:	<input type="checkbox"/> <i>Implementação</i>	<input checked="" type="checkbox"/> <i>Documentação</i>
Propriedades Estáticas (Perspectiva Funcional e Organizacional)			
E	Artefatos gerados:	<i>1. Ocorrência;</i>	
F	Utilizada nas Atividades:	<i>1. Registrar ocorrência;</i> <i>2. Analisar ocorrência;</i> <i>3. Classificar ocorrência;</i> <i>4. Retornar feedback</i> <i>5. Atualizar ocorrência.</i>	
Propriedades Dinâmicas (Perspectiva Comportamental)			
G	Crítérios para uso:	<i>Não possui.</i>	

Além do preenchimento dos instrumentos de coleta de dados específicos para cada componente do processo de software, era preenchido paralelamente o instrumento de coleta de dados 11 (Figura 35) responsável pelos relacionamentos entre os componentes do processo. Assim, na medida em que os dados de um componente eram coletados, as informações sobre seus relacionamentos eram dispostas no instrumento. O instrumento 11, de acordo com o tipo de componente, era atualizado toda vez que um mesmo componente do mesmo tipo tinha seus dados coletados. Por exemplo, se a coleta de dados era referente a atividades, após a coleta de dados de cada atividade era realizada uma atualização da matriz de relacionamentos das atividades. Foram considerados os seguintes relacionamentos entre os componentes do processo de software: entre atividades, entre atividades e artefatos, entre atividades e papéis e, entre atividades e ferramentas. O relacionamento entre os componentes era rotulado de acordo com o tipo de componentes que estavam se relacionando (vide Figura 35).

Figura 35: Relacionamentos possíveis entre os componentes do processo de software

Rótulo / componente	Contém (CT)	Entrada (EN)	Saída (SA)	Consome (CS)	Modifica (MD)	Produz (PD)	Usa (US)
Entre atividades	✓	✓	✓				
Entre atividade e papel							✓
Entre atividade e ferramenta							✓
Entre atividade e artefato				✓	✓	✓	

A título de exemplo a Figura 36 apresenta um instrumento aplicado no estudo de caso. Na linha 'A' eram dispostos os componentes com os quais as atividades se relacionavam, podendo ser atividades, artefatos, papéis ou ferramentas. Na linha 'B' sempre eram descritas as mesmas atividades e, na linha 'C' eram listados os rótulos que poderiam ser atribuídos ao relacionar a atividade ao componente. No cruzamento entre as linha e coluna (atividade e um determinado componente), era definido o tipo de relacionamento entre esses componentes.

Figura 36: Instrumento de coleta de dados 11 (relacionamentos)

ATIVIDADE: RELACIONAMENTOS							
A	Atividade	Registrar solicitação	Classificar ocorrência	Analisar ocorrência	Retornar feedback	Atualizar ocorrência	Controlar tarefa
B	Suporte	CT	CT	CT	CT	CT	CT
	Registrar solicitação		SA				SA
	Classificar ocorrência			SA		SA	SA
	Analisar ocorrência				SA	SA	SA
	Retornar feedback					SA	SA
	Atualizar ocorrência						SA
	Controlar tarefa						
C	CT: Contém		EN: Entrada	SA: Saída			

ARTEFATOS: RELACIONAMENTOS							
A	Atividade	Solicitação do cliente	Ocorrência	Feedback	Controle de tarefas		
B	Suporte						
	Registrar solicitação	CS	PD				
	Classificar ocorrência		MD				
	Analisar ocorrência		MD				
	Retornar feedback		MD	PD			
	Atualizar ocorrência		MD				
	Controlar tarefa	CS	CS	CS	PD		
C	CS: Consome		MD: Modifica	PD: Produz			

PAPEL: RELACIONAMENTOS							
A	Atividade	Atendente	Cliente				
B	Suporte						
	Registrar solicitação	US	US				
	Classificar ocorrência	US					
	Analisar ocorrência	US					
	Retornar feedback	US	US				
	Atualizar ocorrência	US					
	Controlar tarefa	US					
C	US: Usa						

FERRAMENTA: RELACIONAMENTOS							
A	Atividade	Gerenciador de Ocorrências	Gerenciador de tarefas	Gerenciador de e-mails			
B	Suporte						
	Registrar solicitação	US	US	US			
	Classificar ocorrência	US	US				
	Analisar ocorrência	US	US				
	Retornar feedback	US	US	US			
	Atualizar ocorrência	US	US				
	Controlar tarefa		US				
C	US: Usa						

4.3.3 Etapa de Verificação & Validação

A Etapa de Verificação & Validação tem por objetivo garantir que os elementos (atividades, sub-atividades, artefatos, fluxos de controle, fluxos de produtos, ferramentas, papéis etc) sejam descritos adequadamente e atendem às necessidades da organização e, ainda, avaliar o processo de aplicação da abordagem. Garantir, principalmente em etapas iniciais da descrição do processo de software, que as práticas de Engenharia de Software sejam uniformes e sistematicamente aplicadas, reforça a institucionalização do processo de software. Caso a descrição dos elementos do processo de software apresente muitas inconsistências e/ou problemas em sua representação, a reaplicação das etapas deveria ser considerada para garantir a descrição (formalização) consistente da mesma.

Como o processo de software é definido gradualmente pela abordagem proposta nesse trabalho, devida atenção deve ser dispensada ao unificar as diferentes perspectivas entre os elementos que estão sendo definidos e, em descrições de outros elementos do processo de software aplicados posteriormente. Também deve ser dada especial atenção ao integrar as atividades do ciclo de vida, principalmente quando são seqüenciais, uma vez que diversos elementos podem migrar (por exemplo, fluxo de controle e fluxo de produtos) e, portanto, deve haver consistência ao relacionar os elementos do processo de software. Além disso, à proporção que o processo de software é gradualmente definido, o Guia de Processo de Software (GPS) também deve ser atualizado e consistido para refletir o processo de software vigente na organização. A atualização do GPS e utilização sistemática pelos participantes do processo também reforça a cultura de institucionalização do processo de software. Portanto, considerando a descrição gradual do processo de software e a possibilidade do processo ser descrito ou em largura ou em profundidade, será apresentado um conjunto de métricas subdivididas por elementos do processo de software.

A elaboração das métricas foi orientada pelo paradigma GQM (BRIAND, DIFFERDING & ROMBACH, 1997) o qual permitiu definir diversas questões que devem ser respondidas ao consistir os elementos do processo de software

descritos. A flexibilidade oferecida pelo paradigma GQM ao estruturar um conjunto de métricas torna-o uma escolha natural e tem sido utilizado com sucesso em alguns estudos relacionados à modelagem de processo de software (HOLTJE et al, 1994; BECKER-KORNSTAEDT & BELAU, 2000). Foram elaboradas dez métricas para coletar dados sobre os elementos do processo de software (vide Figura 37). A elaboração das métricas foi baseada nos elementos do processo de software para adequá-las a estratégia utilizada ao descrever o processo de software, ou seja, se a descrição do processo de software iniciar-se pelas sub-atividades (descrição do processo pela largura – vide Figura 24), deve ser aplicado, ao final da descrição das atividades, o conjunto de métricas de atividades para validar a descrição dessas.

Figura 37: *Abstraction Sheet* do elemento de processo atividade.

Objeto:	Modelo Descritivo do Processo de Software	
Objetivo	Validar	
Enfoque de qualidade	Acurácia	
Ponto de vista	Processo real	
Contexto	Microempresa	
Metas ▼		
Fatores de Qualidade (FQ)		Fatores de Variação (FV)
Identificar inconsistências na descrição dos elementos do processo de software.		<ul style="list-style-type: none"> • Produto mantido.
1. Número total de fluxos de saída para outras atividades.		<ul style="list-style-type: none"> • Agente executor.
2. Número total de fluxos de entrada para outras atividades.		<ul style="list-style-type: none"> • Fonte de dados consultada.
3. Número total de papéis envolvidos em cada atividade.		
4. Número total de alocação de um papel no processo de software.		
5. Número total de ferramentas utilizadas em cada atividade.		
6. Número total de utilização de uma ferramenta no processo de software.		
7. Número total de artefatos de entrada da atividade.		
8. Número total de entradas de um artefato no processo de software.		
9. Número total de artefatos de saída da atividade.		
10. Número total de saídas do artefato no processo de software.		

4.4 Resumo da Descrição e Validação do Processo de Software

Nas próximas seções será apresentado o resumo da descrição do processo de software da empresa na qual foi realizado o estudo. Para cada etapa do ciclo de vida da empresa será destinada uma seção contendo a descrição e a validação do processo descrição em relação ao processo de software real.

4.4.1 Descrição e validação da etapa de suporte

A aplicação dos instrumentos de coleta de dados dos elementos de processo à atividade de Suporte foi realizada em quatro dias sendo destinado quatro horas por dia para coleta de dados de cada elemento do processo. Foram identificadas seis sub-atividades: “Registrar solicitação”, “Classificar ocorrência”, “Analisar ocorrência”, “Retornar feedback”, “Atualizar ocorrência” e “Controlar tarefa”; quatro artefatos: “Solicitação do cliente”, “Ocorrência”, “Feedback” e “Controle de tarefas”; dois papéis: “Atendente” e “Cliente” e três ferramentas: “Gerenciador de ocorrências”, “Gerenciador de e-mail” e “Gerenciador de tarefas”. Os dados coletados sobre os elementos da atividade de Suporte foram representados graficamente por meio da ferramenta Sparmint. Foram elaborados quatro gráficos: fluxo entre atividades, fluxo de artefatos, papéis envolvidos e ferramentas utilizadas. A aplicação dos instrumentos de coleta de dados e representações gráficas foi realizada em sete dias consecutivos totalizando 35h17min.

Relevando o fato do mesmo processo de software ser aplicado a todos os quatro produtos da empresa, considerou-se que cada um dos produtos deveria ser submetido ao conjunto de métricas para averiguar se a descrição da atividade de Suporte era consistente em relação ao processo de software real executado entre os agentes responsáveis pelos diferentes produtos. Em média, conforme dados colhidos do software de gerenciamento de ocorrências da empresa, eram atendidas 150 ocorrências ao mês pelos agentes (aproximadamente sete ocorrências por dia). Com base nesses dados, decidiu-se aplicar as métricas em pelos menos cinco ocorrências de cada um dos quatro produtos. Durante cinco dias seguidos (segunda a sexta-feira), foram coletados dados de uma ocorrência de cada produto por dia (vide resultado na Figura 38) totalizando 26h33min entre a coleta de dados e análise dos dados.

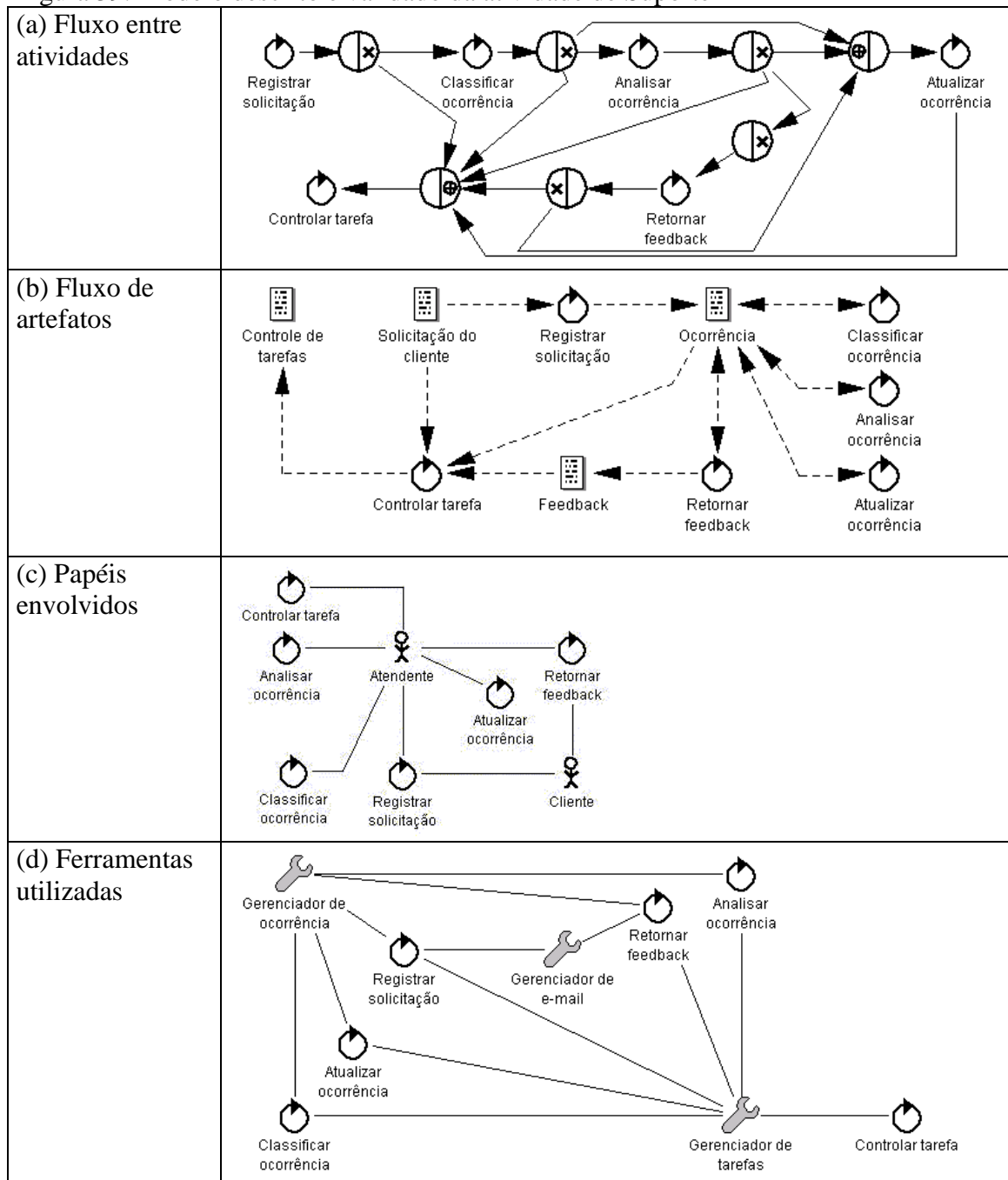
As matrizes de métricas foram elaboradas em uma planilha eletrônica programada para totalizar os dados coletados sobre o processo de software real da empresa. Os dados para preencher a planilha foram coletados durante a execução do processo de software real. Após o preenchimento das matrizes de métricas, as informações eram cruzadas com a descrição do processo (análise dos dados coletados). Por exemplo, se na matriz houvesse a indicação que uma determinada atividade apresentava duas saídas para outras atividades, a representação gráfica da atividade deveria conter dois fluxos de saída. Ocorrendo divergência entre os dados, a atividade era novamente descrita e uma nova validação era aplicada. A descrição do processo bem como a validação do mesmo somente era finalizada se não ocorressem mais divergências entre o processo de software descrito (Guia) e o processo de software real.

Duas inconsistências foram constatadas após a aplicação das métricas. A primeira inconsistência foi observada nos dados da métrica 1 e 2, mais especificamente no número de fluxos de entrada da sub-atividade “Atualizar ocorrência”. A sub-atividade “Atualizar ocorrência” apresentava apenas uma entrada, no entanto, a execução do processo de software real demonstrou que o artefato “Ocorrência” deveria ser atualizado assim que as sub-atividades “Classifica ocorrência”, “Analisar ocorrência” e “Retornar feedback” fossem finalizadas com o objetivo de manter o histórico das ações executadas em cada uma dessas sub-atividades. A segunda inconsistência se referia ao número de saídas das sub-atividades “Classificar ocorrência” e “Retornar feedback”. Na representação da atividade de Suporte, as sub-atividades “Classificar ocorrência” e “Retornar feedback” não apresentam nenhuma saída, no entanto, ao medir o número de saídas dessas sub-atividades por meio das métricas 7 e 9, foi percebido que deveria haver uma saída, ou seja, o artefato “Ocorrência” era modificado pelas sub-atividades “Classificar ocorrência” e “Retornar feedback”, portanto, deveria ocorrer uma saída do artefato em ambas as sub-atividades.

As demais representações, papéis envolvidos e ferramentas utilizadas não apresentaram inconsistências. As inconsistências conduziram a reaplicação da Etapa de Descrição dos Elementos do Processo de Software para corrigir as inconsistências constatadas por meio da aplicação das métricas (em um único dia – 2h15m). A Etapa de Verificação & Validação da atividade de Suporte foi conduzida nos mesmos moldes da

aplicação anterior, ou seja, foram aplicadas as métricas durante cinco dias consecutivos (totalizando 26h12m entre coleta e análise) sobre vinte ocorrências (cinco ocorrências por produto). O resultado da aplicação das métricas produziu o mesmo resultado apresentado na Figura 39 validando a nova representação.

Figura 39: Modelo descrito e validado da atividade de Suporte



4.4.2 Descrição, verificação e validação da atividade Gerência de Tarefas

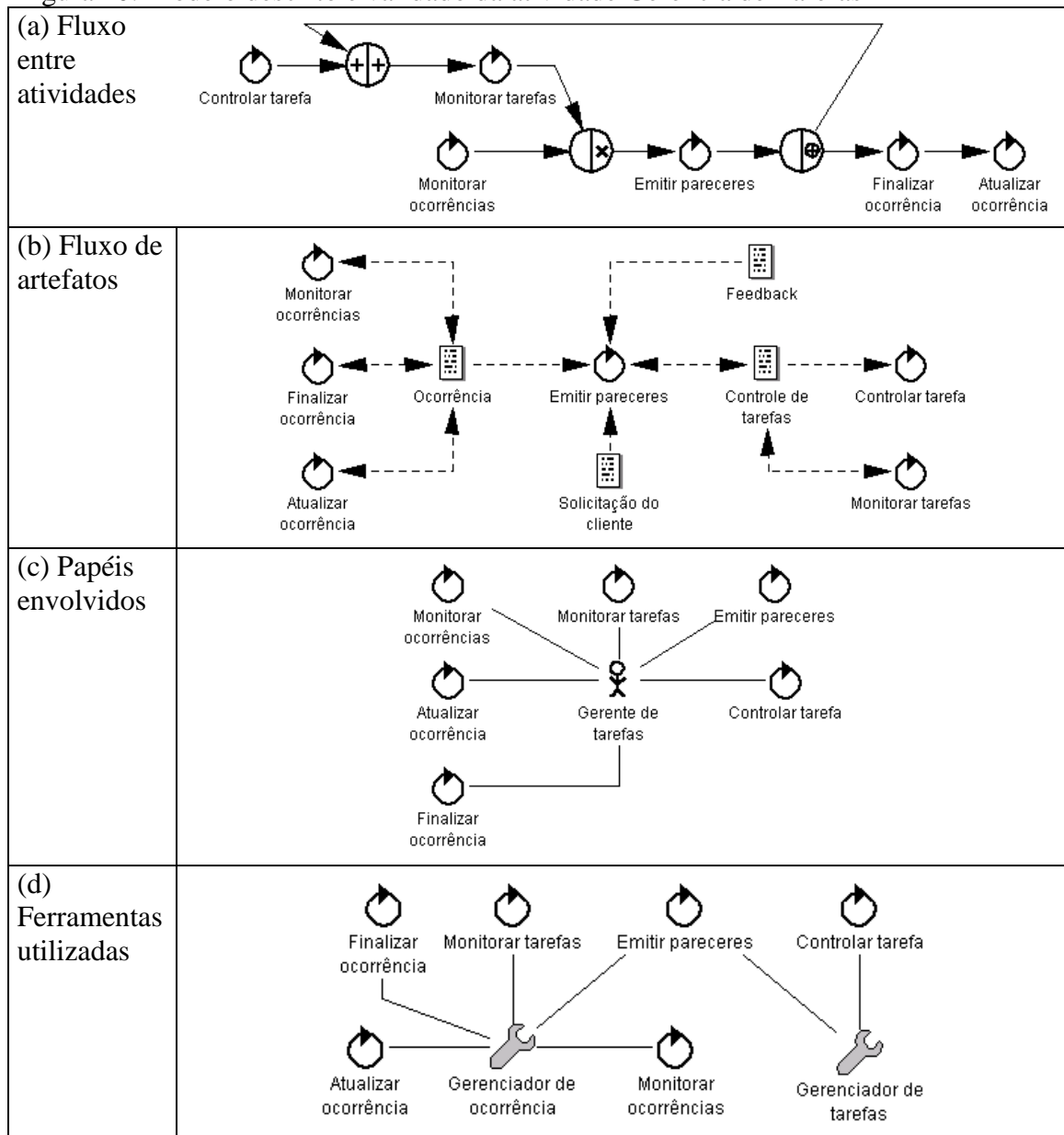
A descrição da Atividade de Gerência de Tarefas revelou a existência de seis sub-atividades sendo quatro sub-atividades próprias da Atividade de Gerência de tarefas: “Emitir pareceres”, “Monitorar tarefas”, “Finalizar ocorrências” e “Monitorar ocorrências” e duas sub-atividades já existentes na atividade de Suporte: “Atualizar ocorrências” e “Controlar tarefas”. Foram observados quatro artefatos já descritos: “Ocorrência”, “Feedback”, “Controle de tarefas” e “Solicitação do cliente”. Um novo papel foi constatado: “Gerente de Tarefas” e, com relação às ferramentas, foram observadas apenas a utilização de duas ferramentas já descritas: “Gerenciador de ocorrências” e “Gerenciador de Tarefa”. A aplicação dos instrumentos de coleta de dados foi realizada em quatro dias (quatro horas por dias) consecutivos sendo destinado um dia para cada elemento do processo de software totalizando 16h00m. A descrição e representação da atividade Gerência de Tarefas ocorreu nos dois dias seguintes perfazendo 10h35m.

A aplicação das métricas e análise dos dados (seis dias consecutivos totalizando 25h25m) mostrou apenas uma inconsistência relacionada à utilização da ferramenta “Gerenciador de Tarefas” que apresentava uma entrada e saída do artefato “Controle de tarefas” na sub-atividade “Emitir pareceres” e, no entanto, a ferramenta que manipulava o artefato (“Gerenciador de Tarefas”) não constava na métrica correspondente. De fato, ao consultar os agentes responsáveis, ficou comprovado que a ferramenta realmente era utilizada e, entretanto, não constava na representação gráfica correspondente.

A constatação da inconsistência exigiu a reaplicação da etapa Descrição do Elemento de Processo (realizada em um dia – 0h35m) e Verificação & Validação (totalizando 26h02m entre coleta e análise de dados) gerando uma nova representação do uso de ferramentas. As demais representações foram verificadas e validadas não requerendo mudanças na primeira versão. O resultado da aplicação das métricas também permitiu uma visão parcial do relacionamento e fluxo de entrada e saída dos elementos no processo de software da empresa considerando as duas

atividades descritas. Foi possível por meio da descrição incremental evidenciar, por exemplo, o relacionamento entre sub-atividades das duas atividades descritas: “Monitorar tarefa” e “Controlar tarefa”. A representação final da atividade pode ser consultada na Figura 40.

Figura 40: Modelo descrito e validado da atividade Gerência de Tarefas



4.4.3 Descrição, verificação e validação da atividade Implementação

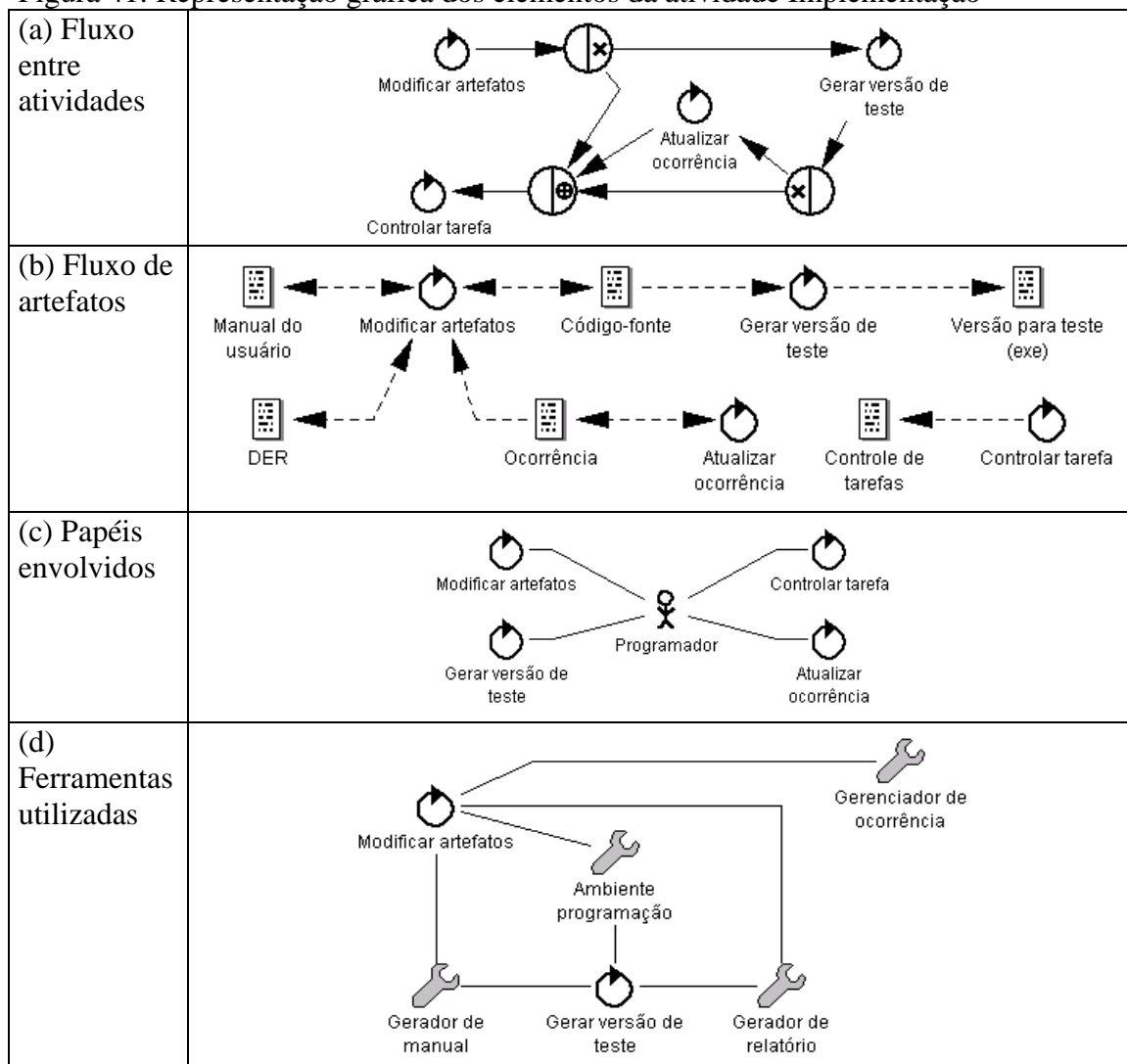
Inicialmente, a descrição da atividade de Implementação (24h37m) apresentou quatro sub-atividades sendo duas próprias: “Modificar artefatos” e “Gerar versão de teste” e duas já descritas na atividade de Suporte: “Atualizar ocorrências” e “Controlar tarefas”. Ainda foram descritos, na primeira versão da representação dessa atividade, seis artefatos: “Manual do usuário”, “DER (Diagrama de Entidade-Relacionamento)”, “Código-Fonte”, “Versão para teste (.exe)”, “Controle de tarefas” e “Ocorrência” sendo os dois últimos descritos ao modelar a atividade de Suporte; um papel: “Programador” e três ferramentas: “Ambiente de programação”, “Gerador de manual” e “Gerador de relatórios”.

Entretanto, a aplicação e análise das métricas (realizada em 25h36m) após a descrição e representação gráfica no Guia de Processo de Software expuseram duas inconsistências: número de artefatos de entrada e número de ferramentas utilizadas na sub-atividade “Modificar artefatos”. A primeira inconsistência percebida por meio das métricas 7, 8, 9 e 10 foi a inexistência da entrada do artefato “Ocorrência” na sub-atividade “Modificar artefato”. A entrada do artefato, conforme observado após a constatação da inconsistência, era necessária para determinar o produto e modificações que deveriam ser realizadas nesse produto. A segunda inconsistência, relacionada à primeira e observada nas métricas 5 e 6, se referia à não utilização da ferramenta “Gerenciador de ocorrências” na sub-atividade “Modificar artefatos”. Além da constatação da necessidade do uso da ferramenta nessa sub-atividade ao aplicar as métricas 5 e 6, sua utilização era justificada ainda pela necessidade dessa ferramenta para manipular o artefato “Ocorrência”. A coleta de dados por meio das métricas da Etapa de Verificação & Validação ocorreu em cinco dias consecutivos e, no sexto dia, foi realizada a análise dos dados coletados.

As inconsistências conduziram a reaplicação da Etapa Descrição do Elemento de Software (1h22m) a qual reformulou as representações inconsistentes e Etapa Verificação & Validação (25h17m somando coleta e análise dos dados coletados). Após a reaplicação, as novas representações não apresentaram mais inconsistências. A

versão final das representações modificadas pode ser consultada na Figura 41. Foram necessários seis dias para reaplicar as duas etapas.

Figura 41: Representação gráfica dos elementos da atividade Implementação



4.4.4 Descrição, verificação e validação da atividade Teste

Preliminarmente, foram identificadas cinco sub-atividades pertencentes à atividade de Teste: “Testar código-fonte”, “Testar interface”, “Verificar artefatos”, “Atualizar ocorrência” e “Controlar tarefa” (as duas últimas descritas na atividade de Suporte); oito artefatos: “Versão para teste (exe)”, “Código-fonte”, “Manual do usuário” e “DER” descritos na atividade Implementação, “Controle de tarefas” e “Ocorrência” descritos na atividade de Suporte, “Relatório de erros de teste” e “Versão para instalação (exe)” descritos nessa atividade. Um único papel foi descrito:

“testador”. Seis ferramentas eram utilizadas: “Ambiente de programação”, “Gerador de manual”, “Gerador de relatórios” descritas na atividade de Implementação, “Gerenciador de ocorrência” e “Gerenciador de tarefas” descritas na atividade de Suporte, “Editor de texto” descrita nessa atividade. A coleta por meio dos instrumentos e análise dos dados ocorreu em 23h21m.

Ao total, foram identificados dois tipos de inconsistências ao finalizar a aplicação das métricas (26h57m) na atividade de Teste: número de artefatos de entrada e número de ferramentas utilizadas. As métricas 5 e 6 apontaram o uso de cinco ferramentas na sub-atividade “Testar código-fonte”, duas ferramentas nas sub-atividades “Testar interface” e quatro ferramentas na sub-atividade “Verificar artefatos”.

As representações gráficas apresentaram quatro, uma e duas respectivamente. A outra inconsistência se referia ao número de artefatos consumidos e produzidos pelas sub-atividades “Testar código-fonte” (uma entrada e uma saída na representação e duas entradas e uma saída nos resultados das métricas), “Testar interface” apresentou na representação gráfica uma entrada e uma saída se diferenciando nas métricas que apresentou três entradas e uma saída, “Verificar artefatos” que apresentou na representação gráfica duas entradas e duas saídas ao passo que nas métricas se observou quatro entradas e duas saídas.

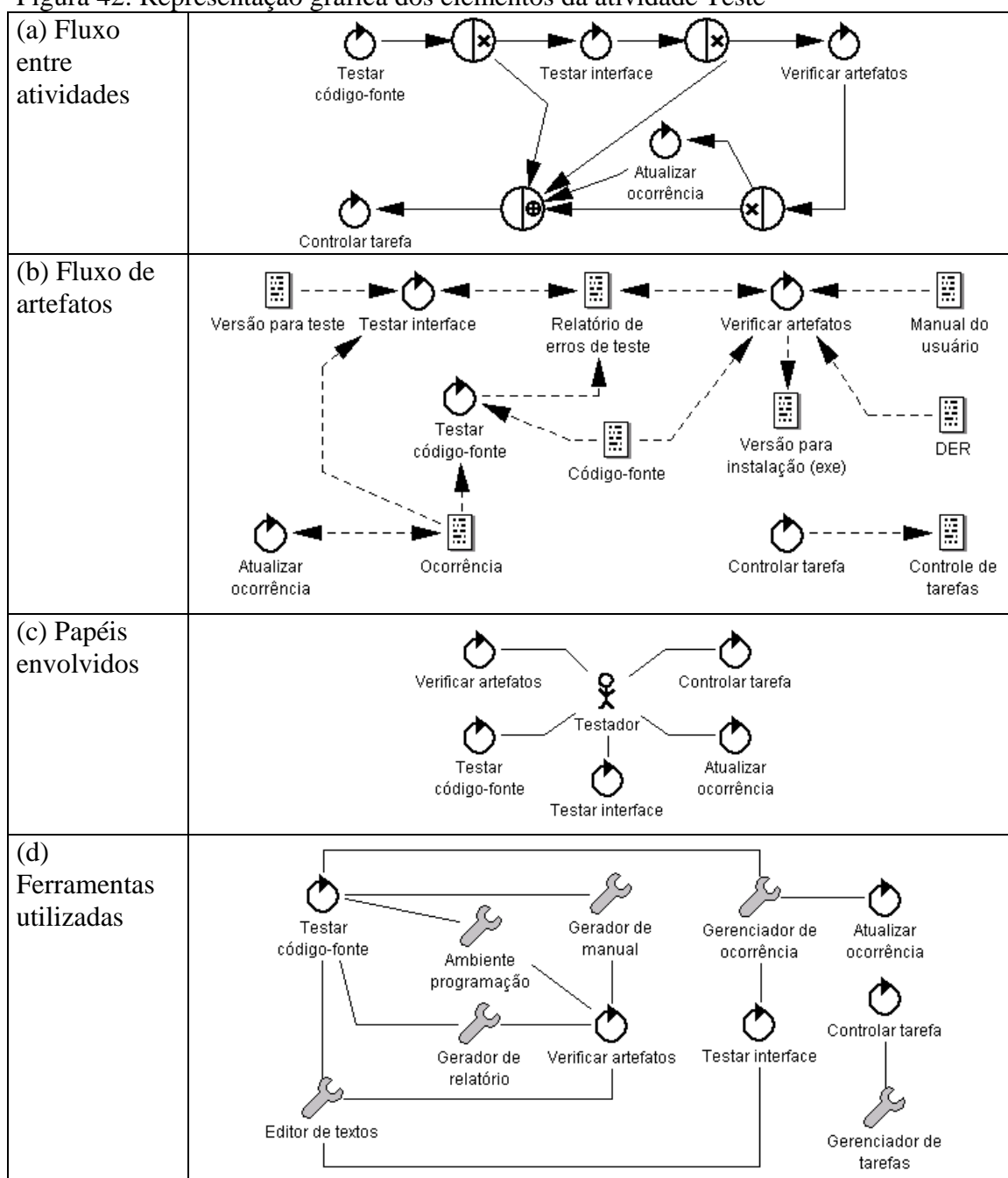
Novas consultas à atividade de Teste referendaram as inconsistências percebidas no resultado das métricas. De fato, as sub-atividades “Testar código-fonte” e “Testar interface” requeriam o artefato “Ocorrência” para sua execução. Os testes eram realizados com base nas solicitações de mudança explicitadas no artefato “Ocorrência” e, por conseguinte, também era necessária a ferramenta “Gerenciador de ocorrência” para manipular o artefato nessas atividades como constatado nos resultados das métricas 5 e 6. O artefato “Relatório de erros de teste” era modificado pela sub-atividade “Teste de interface” e, o artefato “Código-fonte” era necessário à sub-atividade “Verificar artefatos” para gerar a versão de instalação do produto caso não fossem encontrados erros na implementação.

Deve-se salientar que a atividade de Teste gera um fluxo de controle (vide terceira versão do ciclo de vida na Figura 30) à atividade Implementação caso sejam encontrados erros nos testes. Além disso, o artefato “Relatório de erros de teste” também deveria ser consumido pela sub-atividade “Modificar artefatos” caso os erros fossem encontrados. Essa constatação, percebida por meio da aplicação das métricas 7, 8, 9 e 10 complementou a descrição da atividade Implementação.

No entanto, a nova entrada na atividade Implementação, mas especificamente na sub-atividade “Modificar artefatos” criou um dilema, o novo artefato de entrada nem sempre era utilizada pela sub-atividade, somente quando eram detectados erros na Implementação. A solução foi refinar a sub-atividade “modificar artefatos” em duas novas sub-atividades: “Implementar alterações”, responsável pela primeira modificação e “Corrigir erros de implementação” responsável pelas correções de eventuais erros detectados na atividade de Teste.

As inconsistências foram corrigidas nas representações do fluxo de artefatos e ferramentas utilizadas. Em virtude da constatação das inconsistências, as Etapas Descrição dos Elementos de Software (0h37m) e Verificação & Validação (24h02m) foram executadas novamente (um novo giro na espiral) para averiguar novas irregularidades na representação da atividade de Teste. A reaplicação das duas etapas, principalmente da Etapa de Verificação & Validação não apresentaram novas inconsistências. A representação final dos elementos do processo de software da atividade de Teste pode ser consultada na Figura 41.

Figura 42: Representação gráfica dos elementos da atividade Teste

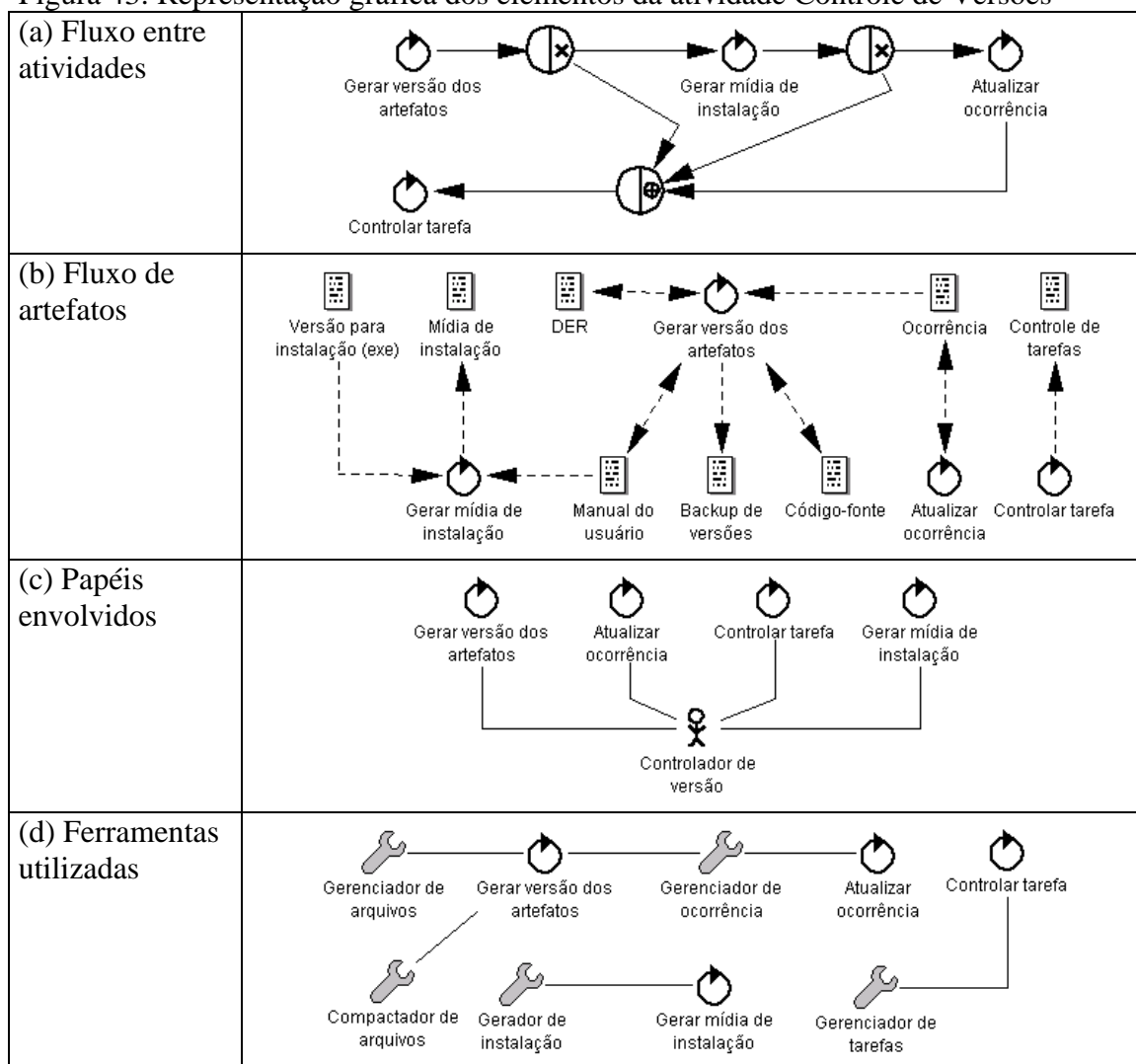


4.4.5 Descrição, verificação e validação da atividade Controle de Versão

Foram identificadas, ao descrever a atividade Controle de Versão (21h27m), quatro sub-atividades: “Gerar versão dos artefatos”, “Gerar mídia de instalação”, “Atualizar ocorrência” e “Controlar tarefas” sendo as duas últimas descritas na atividade de Suporte; oito artefatos: “DER”, “Manual do usuário”, “Código-fonte”, “Versão para instalação” descritos na atividade de Implementação, “Ocorrência”,

“Controle de tarefas” descritos na atividade de Suporte, “Backup de versões” e “Mídia de instalação” descritos nessa atividade; um papel: “Controlador de versão” e cinco ferramentas: “Gerenciador de ocorrências”, “Gerenciador de tarefas” descritos na atividade de Suporte, “Gerenciador de arquivos”, “Gerador de instalação” e “Compactador de arquivos” identificadas nessa atividade. Seguida da Etapa Descrição dos Elementos do Processo de software foi executada a Etapa Verificação & Validação (26h57m) a qual não apresentou inconsistências entre as representações constantes no Guia de Processo de Software e dados resultantes da aplicação das métricas. A versão final da representação gráfica da atividade pode ser consultada na Figura 43.

Figura 43: Representação gráfica dos elementos da atividade Controle de Versões



4.4.6 Descrição, verificação e validação da atividade Teste de Instalação

A descrição da atividade de Teste de Instalação (22h27m) revelou cinco atividades: “Testar instalador”, “Verificar personalização”, “Gerar cópia do instalador”, “Atualizar ocorrências” e “Controlar tarefa”, essas duas últimas descritas anteriormente na atividade de Suporte; cinco artefatos: “Mídia de instalação” descrito na atividade de Teste, “Controle de tarefas”, “Ocorrência” descritos na atividade de Suporte, “Relatório de teste de cópia” e “Solicitação de cópia de versão” descritos pela primeira vez nessa atividade; um papel: “Testador” descrito da atividade de Teste e quatro ferramentas: “Gerenciador de ocorrências”, “Gerenciador de tarefas” descritas na atividade de Suporte, “Gerenciador de arquivos” descrita na atividade Controle de Versão e “Editor de textos” descrita da atividade de Teste.

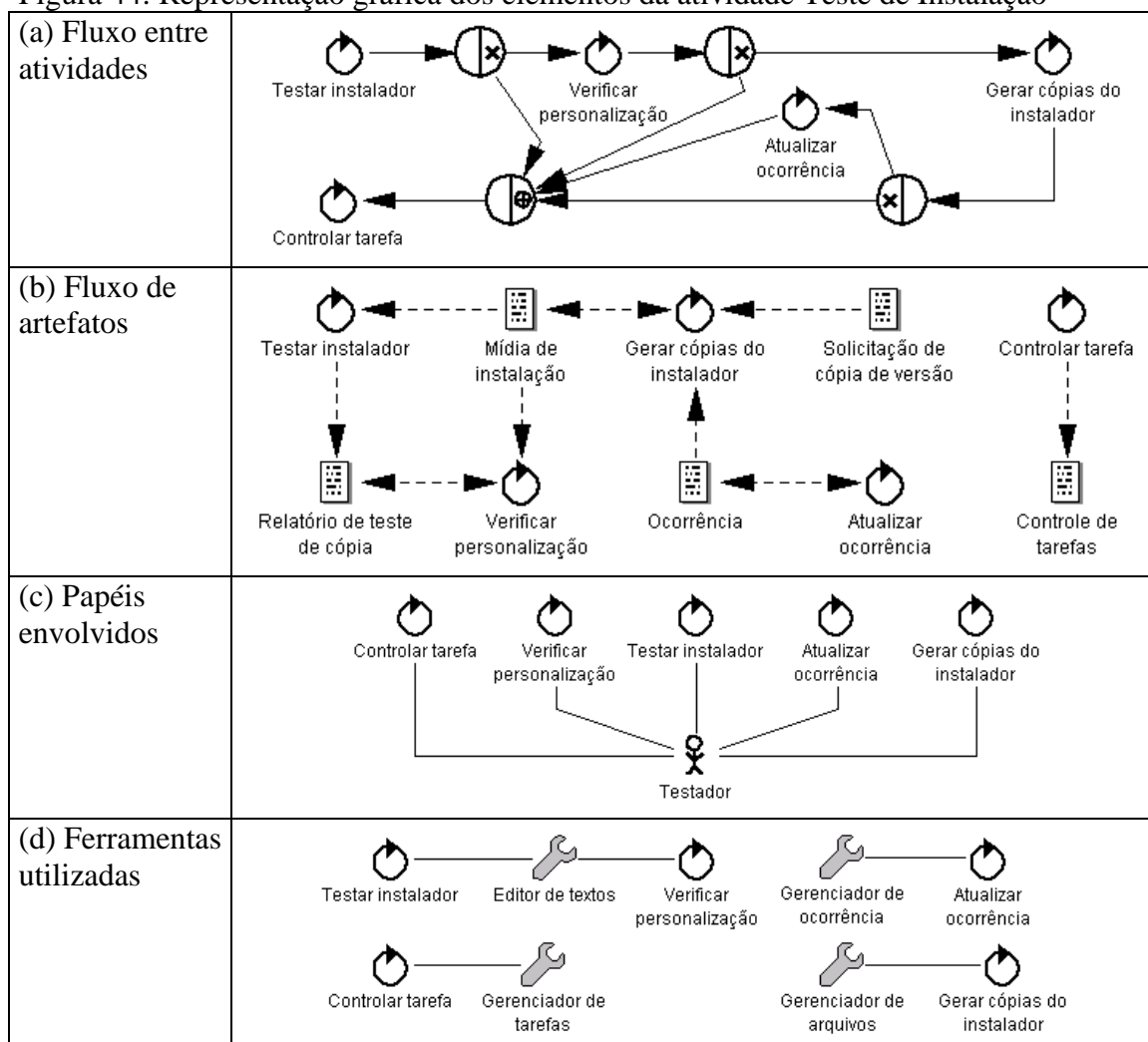
Apenas uma inconsistência foi observada na comparação entre as representações gráficas do Guia de Processo de Software e os resultados coletados por meio das métricas da Etapa de Verificação & Validação (25h05m): número de artefatos de entrada e saída das sub-atividades “Testar instalador”, “Verificar personalização” e “Gerar cópias do instalador”. Inicialmente, os dados coletados na Etapa de Descrição indicaram que o artefato “Relatório de teste de cópia” era produzido pela sub-atividade “Verificar personalização”. Na verdade, conforme observações após a constatação da inconsistência, o artefato era produzido pela sub-atividade “Testar instalador” e, na seqüência, deveria ser modificado pela sub-atividade “Verificar personalização”. A inconsistência presente no número de artefatos de saída da sub-atividade “Gerar cópias do instalador” deveu-se a um erro no preenchimento dos dados na matriz de métricas.

Com relação à inconsistência da sub-atividade “Gerar cópias do instalador”, após o término das verificações e, não constatado nenhum problema no instalador, o artefato “Mídia de instalação” deveria apresentar um “selo” de controle atestando que estava apto a ser reproduzido e enviado aos clientes. Caso o artefato “Mídia de instalação” apresentasse problemas ao ser instalado e/ou os elementos de personalização não conferissem com os dados do artefato “Solicitação de cópia de

versão”, o artefato “Mídia de instalação” deveria ser gerado novamente pela atividade de Controle de Versão (sub-atividade “Gerar mídia de instalação”).

O retorno do fluxo de controle à atividade Controle de Versão, como constatado ao definir o ciclo de vida da empresa, também era acompanhado do artefato “Relatório de teste de cópia” o qual apresentava a relação de problemas encontrados ao testar o instalador e que deveriam ser corrigidos gerando novamente o artefato “Mídia de instalação”. Detectadas as inconsistências, foi realizada a correção da representação gráfica (0h17m) referente ao fluxo de artefatos e, para assegurar a inexistência de novas inconsistências, as Etapas de Descrição e Verificação & Validação foram reaplicadas à sub-atividade (25h29m). Após a reaplicação das etapas, não foram identificadas novas inconsistências e a descrição da próxima atividade foi iniciada (vide Figura 44).

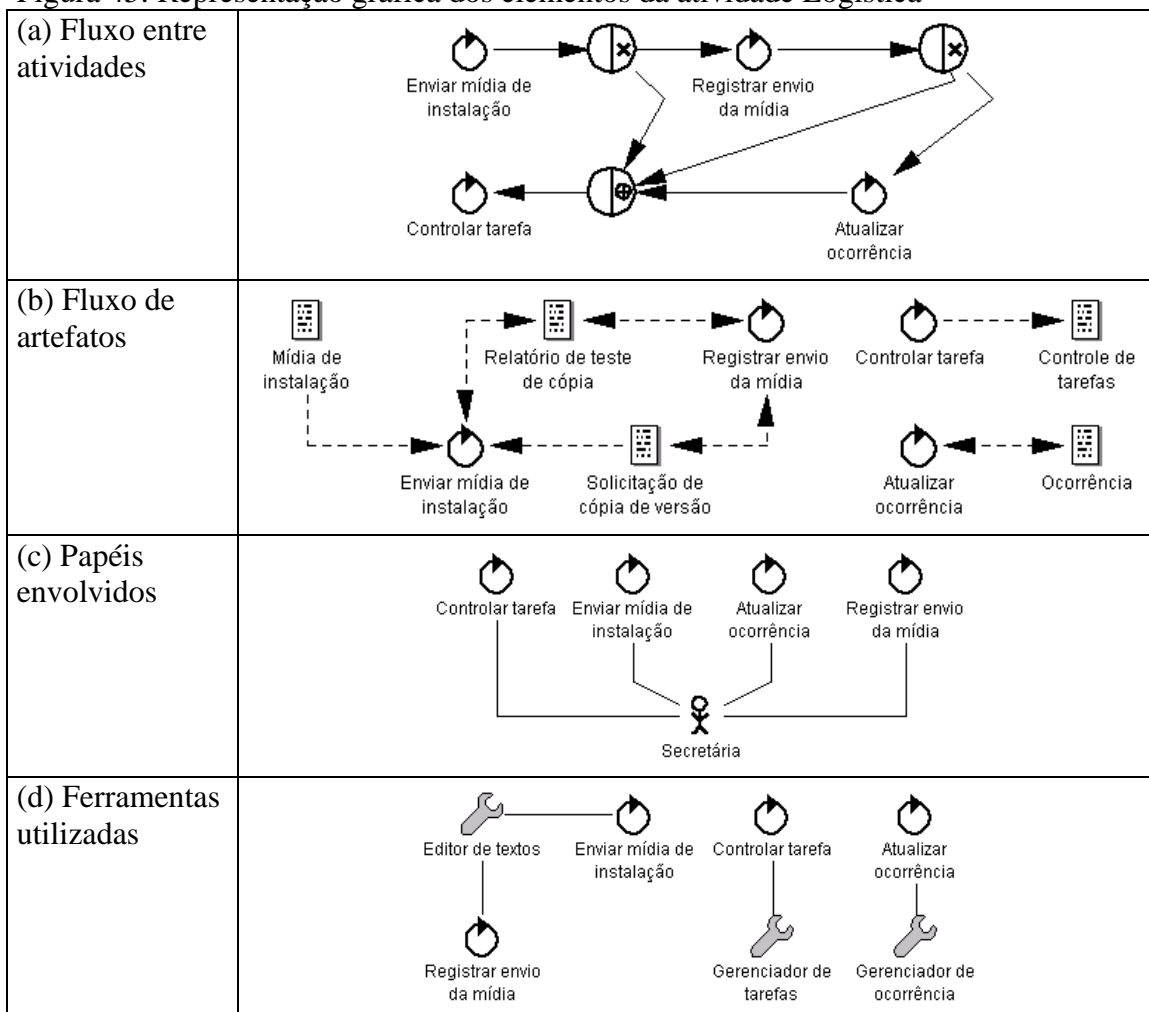
Figura 44: Representação gráfica dos elementos da atividade Teste de Instalação



4.4.7 Descrição, verificação e validação da atividade Logística

A Etapa de Descrição dos Elementos de Software da Atividade de Logística (19h16m) explicitou quatro atividades: “Enviar mídia de instalação”, “Registrar envio da mídia”, “Atualizar ocorrência” e “Controlar tarefas” sendo as duas últimas descritas na atividade de Suporte; quatro artefatos já descritos anteriormente: “Mídia de instalação”, “Relatório de teste de cópia”, “Ocorrência” e “Controle de tarefas”; um papel: “Secretária” e três ferramentas descritas em etapas anteriores: “Editor de textos”, “Gerenciador de ocorrência” e “Gerenciador de tarefas” (vide Figura 45). A aplicação das métricas presentes na Etapa de Verificação & Validação (23h10m) não apresentou inconsistências entre a representação e o processo real.

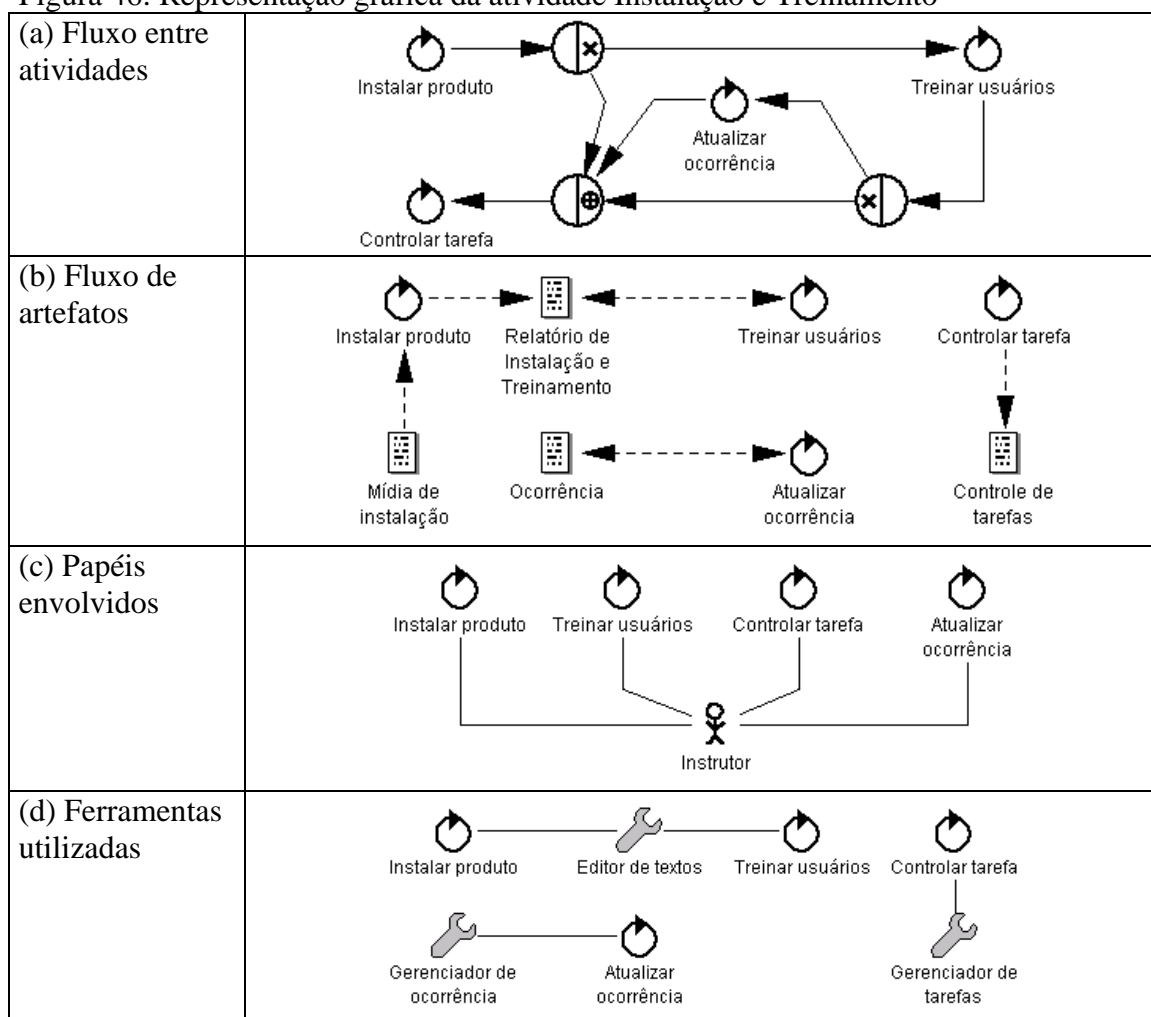
Figura 45: Representação gráfica dos elementos da atividade Logística



4.4.8 Descrição, verificação e validação da atividade Instalação e Treinamento

A atividade Instalação e Treinamento (vide Figura 46) apresentou na Etapa de Descrição dos Elementos do Processo de Software quatro sub-atividade: “Instalar produto”, “Treinar usuário”, “Atualizar ocorrência” e “Controlar tarefa” sendo as duas últimas descritas na atividade Suporte; quatro artefatos: “Relatório de instalação e treinamento” identificado nessa atividade e “Mídia de instalação”, “Ocorrência” e “Controle de tarefas”; um papel: “Instrutor” também identificado nessa atividade e três ferramentas já descritas em etapas anteriores: “Editor de textos”, “Gerenciador de ocorrências” e “Gerenciador de tarefas”. A comparação entre a representação gráfica do GPS e os dados coletados por meio das métricas aplicadas na Etapa de Verificação & Validação (24h11m) não apresentou inconsistências.

Figura 46: Representação gráfica da atividade Instalação e Treinamento



4.4.9 Reformulação do processo e Guia de Processo de Software

Ao término da descrição do processo de software, foi realizado um seminário (realizado em um sábado – 7h23m) para apresentar e discutir o Guia de Processo de Software elaborado. Após a apresentação e discussão, algumas decisões foram tomadas:

- 1^a) As sub-atividades “Controlar tarefas” e “Atualizar ocorrência” deveriam ser executadas sempre que uma sub-atividade fosse finalizada (com exceção das sub-atividades presentes na atividade Gerência de Tarefas). No entanto, como observado no Guia de Processo de Software e conforme o próprio processo de software real, a sub-atividade “Atualizar ocorrência” era executada em apenas algumas atividades do ciclo de vida. Com base nesse argumento, decidiu-se que as duas sub-atividades seriam incorporadas à atividade Gerência de Tarefas do ciclo de vida, uma vez que essas sub-atividades deveriam ocorrer toda vez que uma atividade do ciclo de vida finalizasse.
- 2^a) Para assegurar que a re-alocação das sub-atividades não gerasse inconsistências no Guia de Processo de Software, decidiu-se por uma reaplicação das métricas propostas na Etapa de Verificação & Validação para assegurar a consistência entre o processo de software real e o Guia de Processo de Software.

A reformulação discutida em reunião foi aplicada ao Guia de Processo de Software e demandou mudanças nos papéis envolvidos, ferramentas utilizadas, fluxos de controle e artefatos consumidos e produzidos. Durante uma semana (4h por dia totalizando 20h), foram coletados dados por meio das métricas apresentadas na etapa de Verificação & Validação. Os resultados obtidos atestaram consistência entre o processo de software real executado e Guia de Processo de Software.

4.4.10 Apresentação resumida do Guia de Processo de Software final

Nesta seção, serão apresentadas as principais representações gráficas do Guia de Processo de Software final. As representações gráficas estão dispostas conforme a ordem apresentada no ciclo de vida de manutenção de software da empresa. Inicialmente, serão apresentadas as representações gráficas de alto nível: ciclo de vida, fluxo de artefatos do ciclo de vida, papéis envolvidos no ciclo de vida e ferramentas utilizadas no ciclo de vida e, finalmente, as representações gráficas referentes ao fluxo de controle e fluxo de artefatos de cada atividade do ciclo de vida.

Figura 47: Ciclo de vida final

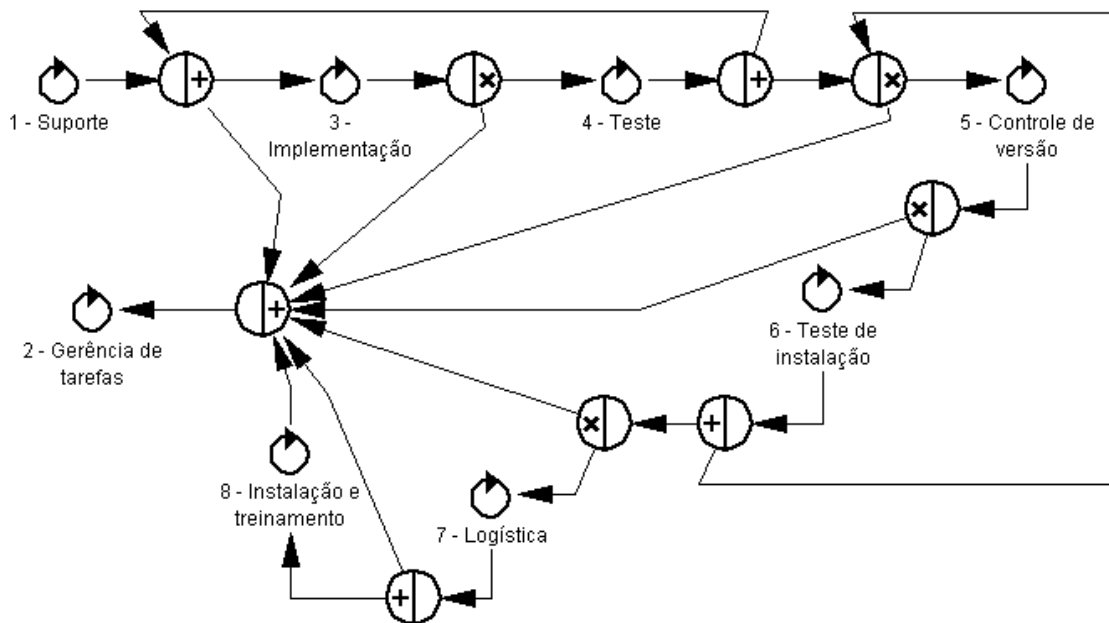


Figura 48: Fluxo de artefatos no ciclo de vida

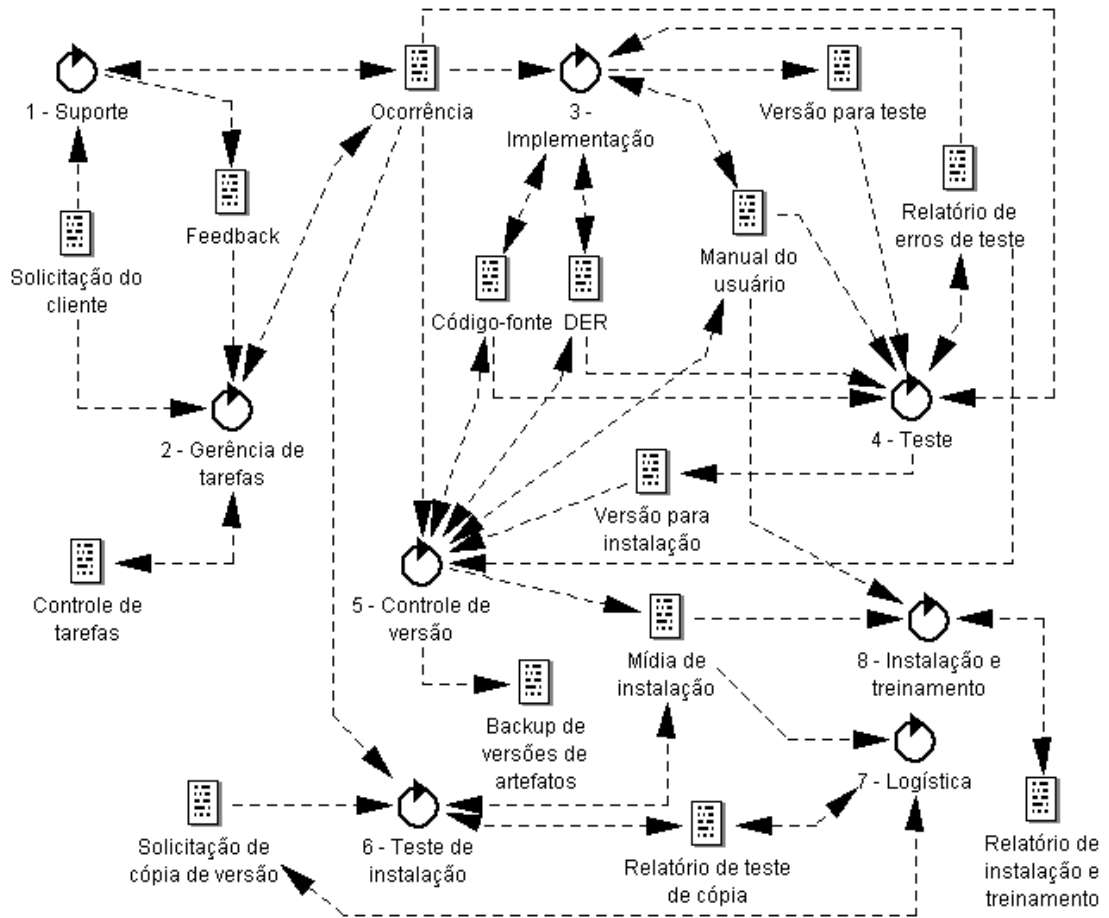


Figura 49: Papéis envolvidos no ciclo de vida

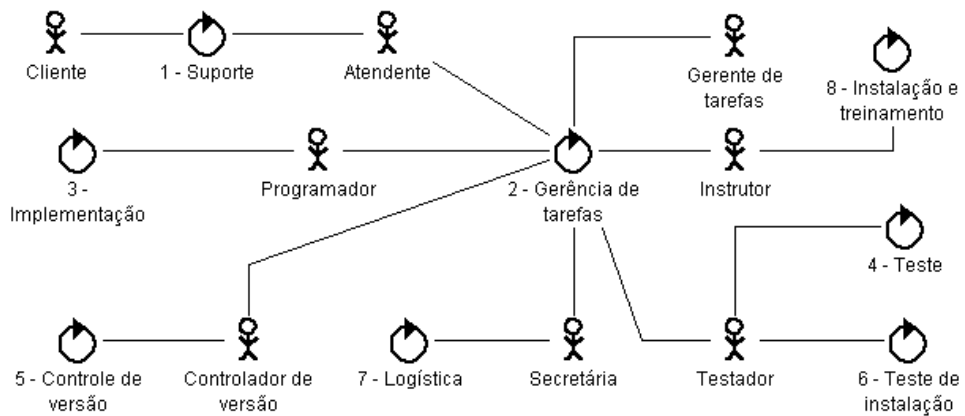


Figura 50: Ferramentas utilizadas no ciclo de vida

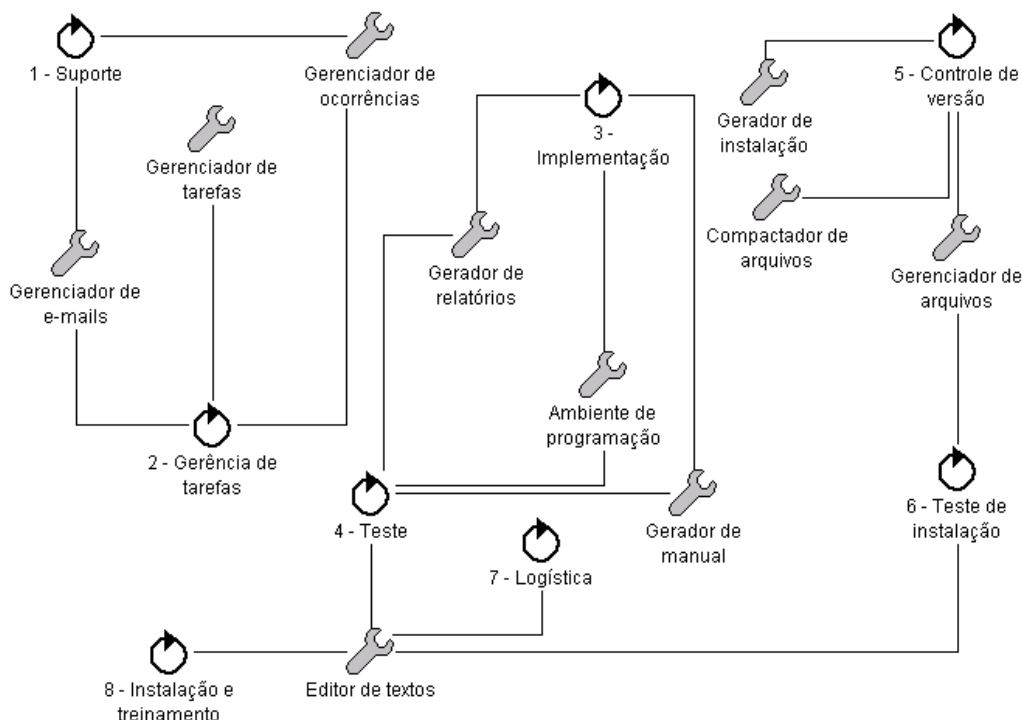


Figura 51: Refinamento do fluxo de controle da atividade Suporte

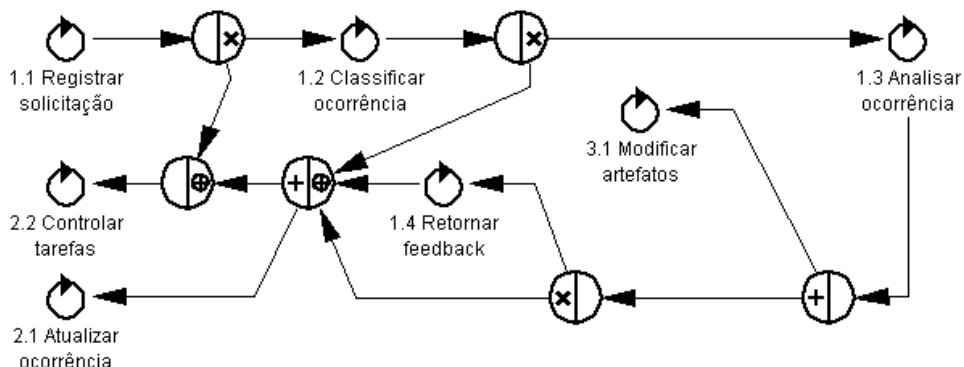


Figura 52: Fluxo de artefatos da atividade Suporte

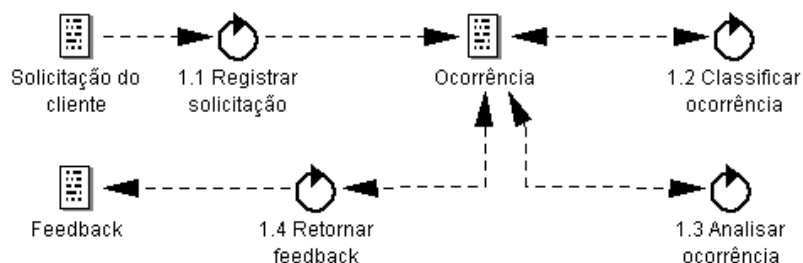


Figura 53: Refinamento do fluxo de controle da atividade Gerência de Tarefas

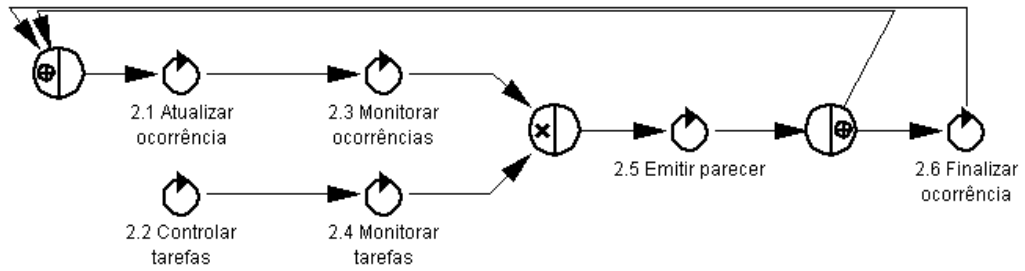


Figura 54: Fluxo de artefatos da atividade Gerência de Tarefas

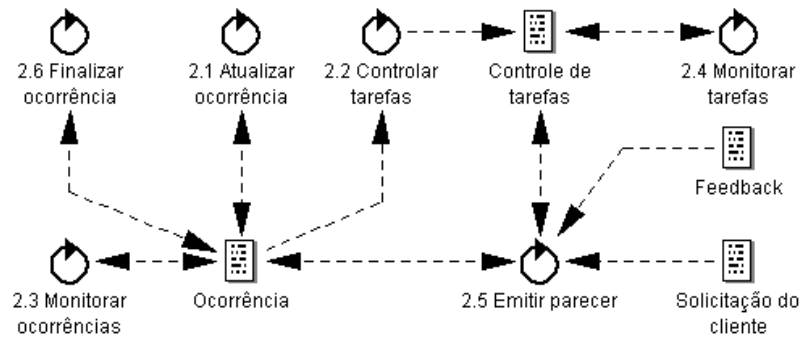


Figura 55: Refinamento do fluxo de controle da atividade Implementação

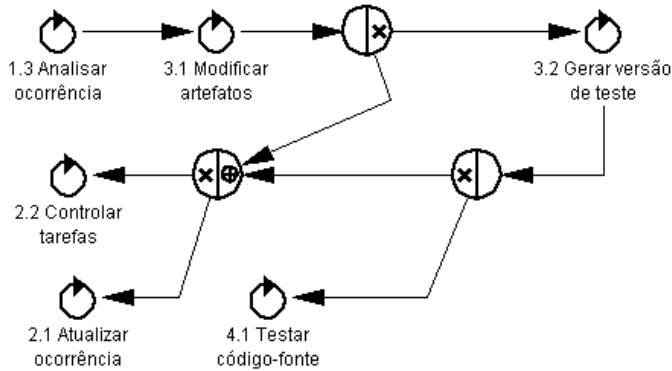


Figura 56: Fluxo de produtos da atividade Implementação

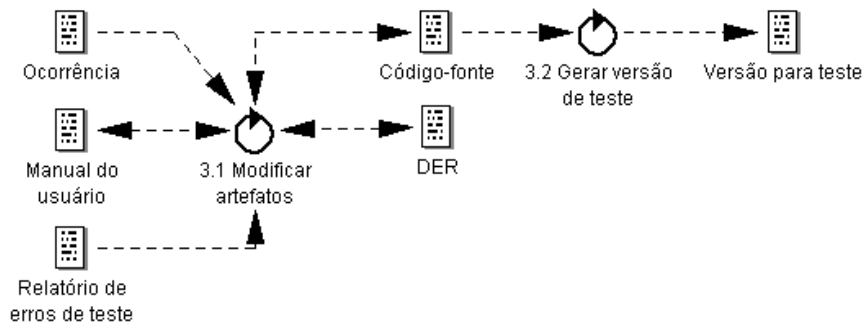


Figura 57: Refinamento da sub-atividade Modificar Artefatos

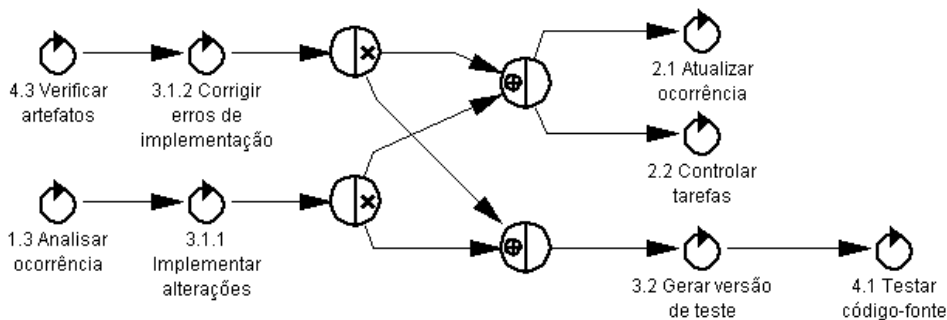


Figura 58: Fluxo de artefatos da sub-atividade Modificar Artefatos

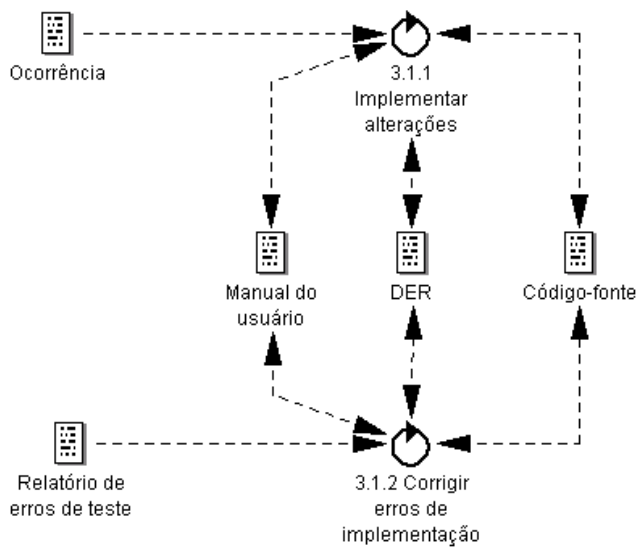


Figura 59: Refinamento do fluxo de controle da atividade Teste

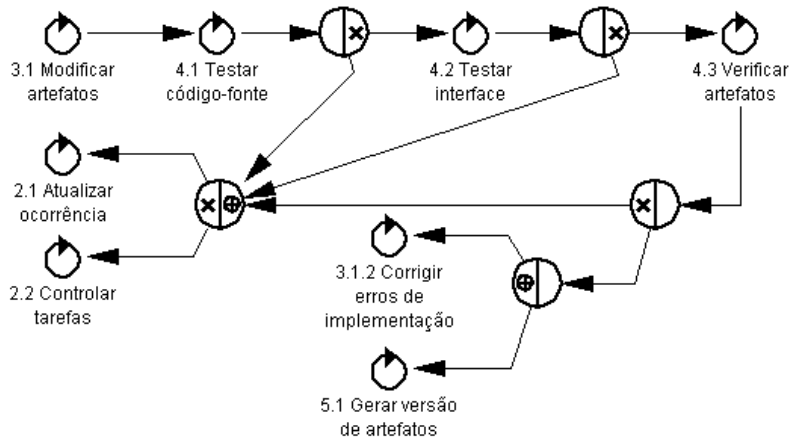


Figura 60: Fluxo de artefatos da atividade Teste

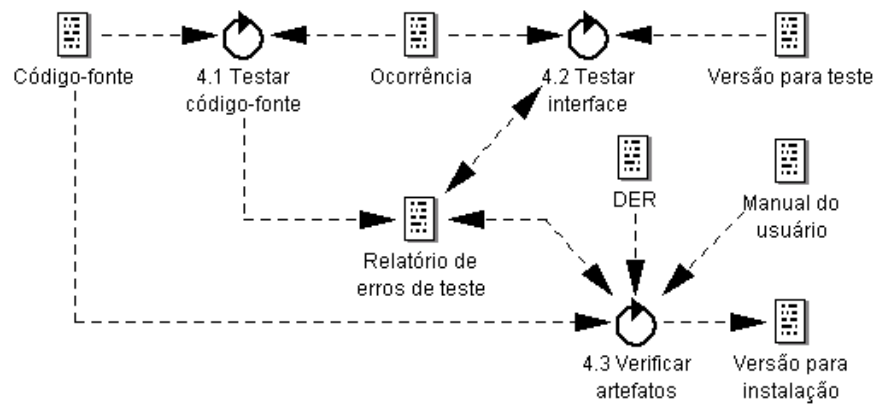


Figura 61: Refinamento do fluxo de controle da atividade Controle de Versão

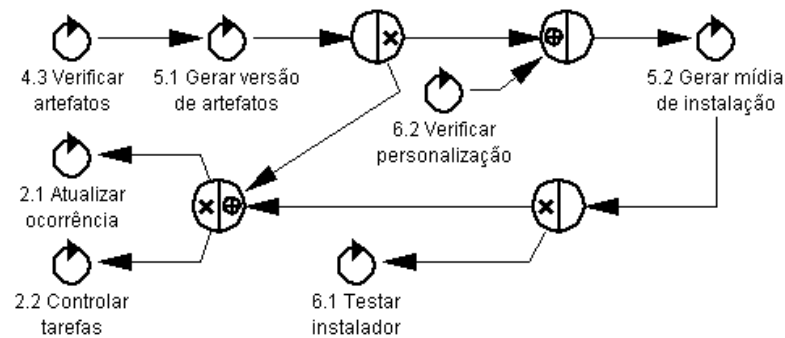


Figura 62: Fluxo de artefatos da atividade Controle de Versão

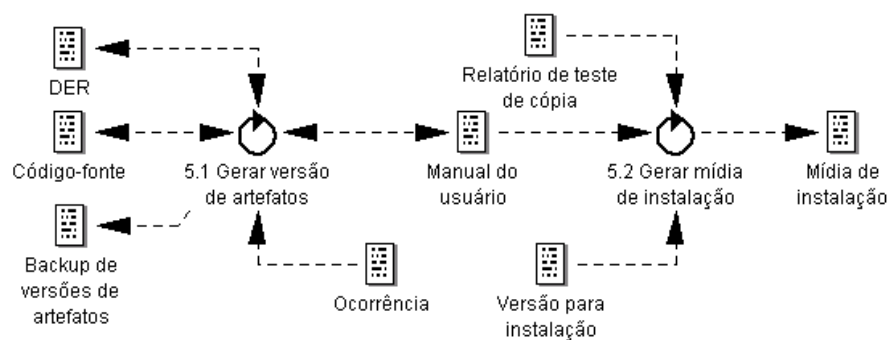


Figura 63: Refinamento do fluxo de controle da atividade Teste de Instalação

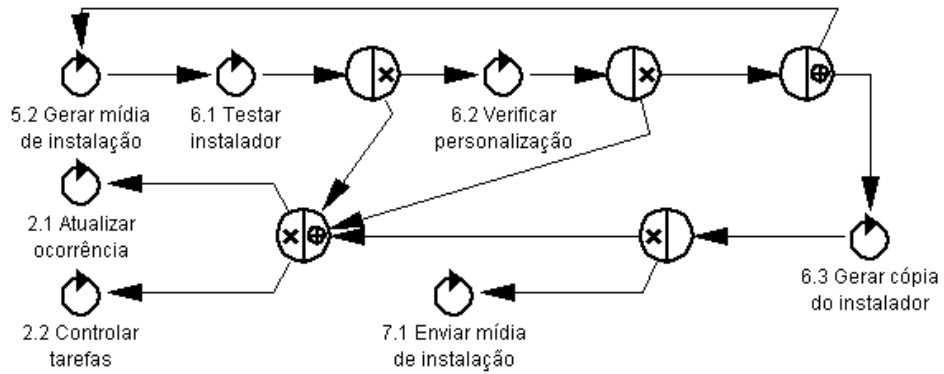


Figura 64: Fluxo de artefatos da atividade Teste de Instalação

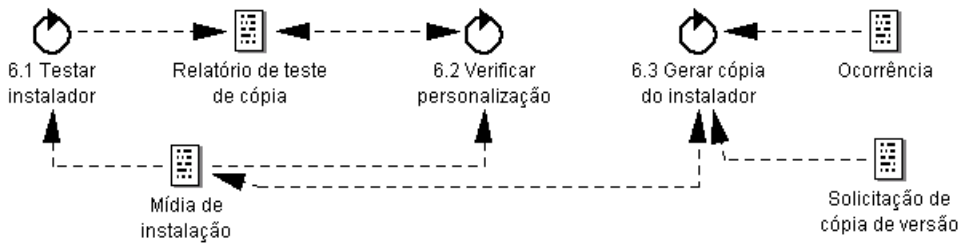


Figura 65: Refinamento do fluxo de controle da atividade Logística

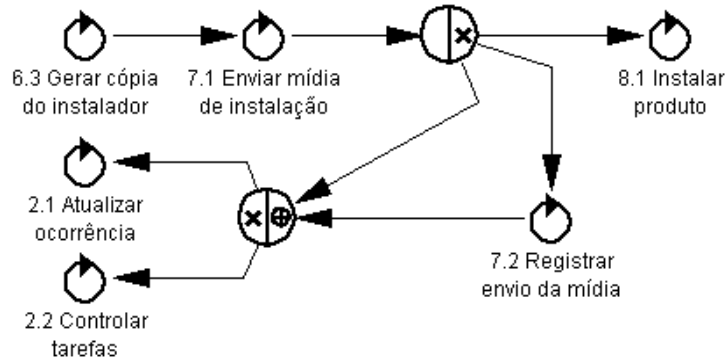


Figura 66: Fluxo de artefatos da atividade Logística

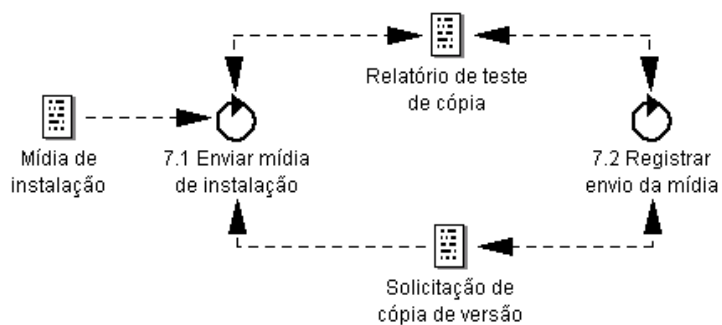


Figura 67: Refinamento do fluxo de controle da atividade Instalação e Treinamento

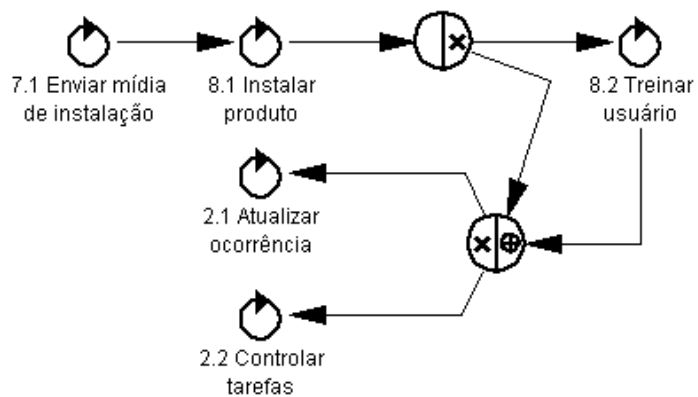


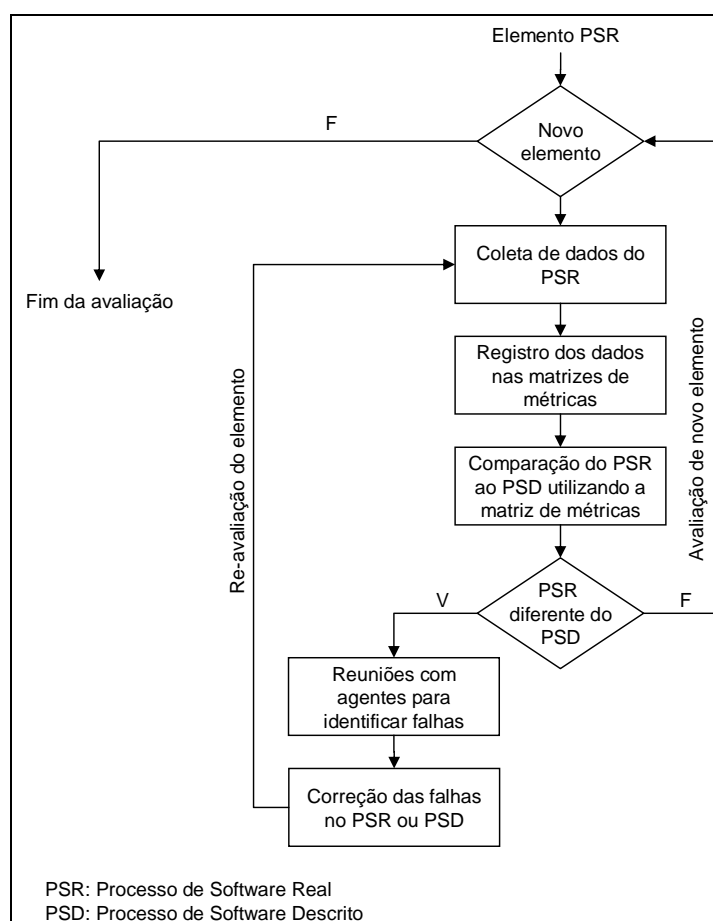
Figura 68: Fluxo de artefatos da atividade Instalação e Treinamento



4.4.11 Avaliação da abordagem

O propósito desta seção é apresentar o instrumento de avaliação da abordagem proposta no presente estudo. Para tanto, o resultado do estudo de caso foi avaliado considerando dois aspectos: a acurácia do Guia de Processo de Software (GPS) em relação ao processo de software real (vide Figura 69) e o cumprimento dos requisitos relacionados à modelagem de processos de software definidos pela microempresa na qual o estudo de caso foi conduzido.

Figura 69: Atividades realizadas na avaliação do GPS.



O grau de conformidade entre o GPS e o processo de software real foi aferido por meio da abordagem GQM. Um plano de mensuração foi elaborado e diversas métricas foram elaboradas com o intuito de investigar se a descrição do processo de software presente do GPS estava em conformidade com o processo de

software real praticado pela empresa. A aplicação das métricas presentes no plano de mensuração se deu por meio do acompanhamento da execução do processo de software real realizado pelos agentes e análise dos produtos resultantes da execução do processo. As expectativas da empresa em relação aos resultados esperados com a aplicação da abordagem na empresa foram avaliadas por meio de requisitos levantados antes da aplicação do estudo de caso. As próximas seções apresentarão em maiores detalhes os meios utilizados na condução da avaliação da abordagem.

4.5 Acurácia do Guia de Processo de Software

O produto de uma abordagem para modelar processos de software pode ser, entre outros, Guias de Processo de Software (GPS). Os GPS's podem ser utilizados como meio para representar as práticas vigentes em uma organização de desenvolvimento de software (descrevendo o processo de software atual) e como roteiros para auxiliar os agentes na execução de suas atribuições (KELLNER et al, 1998, p. 11). Para atender a esse objetivo, GPS's devem conter informações atualizadas que representem o processo de software real de uma organização (BECKER-KORNSTAEDT, 2001, p. 312). Sendo o GPS o produto resultante da modelagem de processo, espera-se que essas abordagens sejam capazes de oferecer meios que contribuam para elaborar GPS's que representem a realidade de processos de software reais. Portanto, uma forma de avaliar a eficiência da abordagem proposta nesse trabalho é avaliar o grau pelo qual o GPS reflete o processo de software real.

Devido à complexidade e diversidade de propósitos relacionados à modelagem de processos de software, o paradigma GQM (BASILI, 1992; BASILI, & WEISS, 1984) foi adotado em alguns estudos relacionados à modelagem de processos de software como meio eficiente para analisar e interpretar os dados pelo fato de adequar-se às particularidades do contexto no qual é aplicado (MADHAVJI et al, 1994; MACHADO et al, 2001).

Considerando as experiências reportadas na literatura e as características do presente trabalho, foi elaborado um plano de mensuração baseado no

paradigma GQM que permitiu definir questões que deveriam ser investigadas para atestar a acurácia do GPS. Para cada uma das questões, foram definidas métricas que possibilitaram coletar dados sobre a execução do processo de software real. O plano de mensuração desenvolvido apresentou quatro questões principais e vinte e duas métricas no total (Figura 70).

Figura 70: *Abstraction Sheet*: plano de mensuração da abordagem.

Objeto:	Abordagem Iterativa e Incremental para Modelagem de Processo de Software	
Objetivo	Avaliar o Guia de Processo de Software em relação ao processo de software real	
Ponto de vista	Guia de Processo de Software versus processo de software real	
Contexto	Processo de software real da microempresa	
Meta ▼		
Fatores de Qualidade (FQ)		Fatores de Variação (FV)
1. Identificar elementos do processo de software real não descritos no GPS.		• Produto mantido.
1.1. Número de atividades não descritas.		• Agente executor.
1.2. Número de artefatos não descritos.		
1.3. Número de ferramentas não descritas.		
1.4. Número de papéis não descritos.		
1.5. Número de fluxos entre atividades não descritos.		
1.6. Número de fluxos de produtos não descritos.		
2. Causas que motivaram inconsistências no GPS em relação ao processo de software real.		
2.1. Número de inconsistências provocadas por falhas na interpretação do processo de software real.		
2.2. Número de inconsistências provocadas por erros no preenchimento dos instrumentos de coleta de dados.		
2.3. Número de inconsistências provocadas por informações não coletadas.		
2.4. Número de inconsistências provocadas por outros motivos.		
3. Identificar elementos do processo de software descritos no GPS e não presentes no processo de software real executado.		
3.1. Número de atividades não executadas.		
3.2. Número de artefatos não manipulados.		
3.3. Número de ferramentas não utilizadas.		
3.4. Número de papéis não envolvidos.		
3.5. Número de fluxos entre atividades não executados.		
3.6. Número de fluxos de artefatos não executados.		
4. Causas que motivaram inconsistências ao executar o processo de software real em relação ao GPS.		
4.1. Número total de inconsistências provocadas pela interpretação textual (parte descritiva) do GPS.		
4.2. Número total de inconsistências provocadas por interpretação das representações gráficas do GPS.		
4.3. Número total de falhas provocadas por falta de informações no GPS.		
4.4. Número de inconsistências provocadas por omissão dos agentes.		
4.5. Número inconsistências provocadas por cronograma atrasado.		
4.6. Número total de inconsistências provocadas por outros motivos.		

O plano de mensuração foi elaborado considerando a relação bidirecional entre GPS e processo real, ou seja, métricas que investigassem o processo de software real em relação ao GPS e métricas que investigassem o GPS em relação ao processo de software real (doze métricas no total). Além disso, para cada uma das direções do relacionamento, foram identificadas métricas que revelassem possíveis causas que poderiam motivar inconsistências no GPS (dez métricas no total) e, conseqüentemente, subsidiar a análise de pontos fortes e fracos da abordagem proposta nesse trabalho.

O foco principal das questões foi avaliar os principais elementos do processo de software: atividades, artefatos, ferramentas, papéis, fluxo entre atividades e fluxo de artefatos. A ênfase nesses elementos deu-se pelo fato da microempresa priorizar em um primeiro momento os elementos de alto nível para sedimentar, progressivamente, a nova cultura que pretendia instituir ao quadro de empregados. As possíveis causas que poderiam motivar inconsistências na relação bidirecional entre GPS e processo de software real foram levantadas de lições aprendidas e requisitos mínimos de GPS's apresentados na literatura de modelagem de processo de software (MADHAVJI et al, 1994, p. 9; KELLNER et al, 1998, p.2; BECKER-KORNSTAEDT & BELAU, 2000, p. 12; BECKER-KORNSTAEDT, 2001, p. 322; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001, p. 74).

A aplicação das métricas ocorreu durante a execução do processo de software real. Essa estratégia, utilizada em diversos estudos de caso relacionados à modelagem de processos de software (MADHAVJI et al, 1994; BECKER-KORNSTAEDT & BELAU, 2000; BECKER-KORNSTAEDT, 2001; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001; MACHADO et al, 2001) possibilita que dados sobre a execução do processo de software sejam coletados e posteriormente comparados ao GPS. A coleta de dados foi dividida em duas etapas: (1ª) coleta de dados de uma semana (cinco dias úteis consecutivos) realizada após algumas semanas da institucionalização do processo de software e, (2ª) coleta de dados aleatória de dados de vários dias em diferentes datas a revelia do agente. A coleta de dados logo após a finalização do GPS final foi realizada devido a probabilidade de ocorrer maior número de inconsistências no período inicial da institucionalização do processo de software real

provocada pela mudança na cultura da organização. O objetivo da segunda etapa de coleta de dados foi averiguar se ocorriam inconsistências entre o GPS e processo de software real depois de vários dias de sua implantação. Ao todo, foram coletados dados de dez dias num período compreendido entre dois meses aproximadamente.

As principais fontes de dados utilizadas na coleta dados foram os artefatos, principalmente os artefatos Ocorrência e Controle de Tarefas. O artefato Ocorrência registra, entre outros dados, todas as decisões executadas pelos agentes (histórico) para cumprir suas atribuições no processo de software da microempresa. O artefato Controle de Tarefas registra informações relacionadas ao tempo de execução (número da ocorrência, data, hora início, hora término e tarefa executada) de cada tarefa de uma determinada Ocorrência. Esses dois artefatos permitiram rastrear a execução do processo de software identificando atividades executadas, fluxos entre atividades, fluxo de artefatos, ferramentas utilizadas e os papéis desempenhados. Foram coletados dados de 48 ocorrências, sendo 23 Ocorrências dos cinco dias da primeira etapa da coleta de dados e 25 ocorrências consultadas na segunda etapa da coleta de dados. Quando uma inconsistência era identificada, o agente responsável deveria justificar a causa de tal inconsistência. Para subsidiar as métricas do plano de mensuração, foram coletados os seguintes dados: data da coleta, número da ocorrência, agente, inconsistência, motivo e justificativa do agente. A inconsistência e justificativa eram analisados e subsidiavam as métricas do plano de mensuração da avaliação da abordagem.

A coleta de dados apresentou 16 inconsistências, sendo 12 inconsistências identificadas na primeira etapa da coleta de dados e 4 na segunda etapa. Embora todos os elementos presentes no processo de software real estivessem descritos no GPS, foram constatadas inconsistências nas descrições de alguns elementos (vide Quadro 5). Não foram encontrados elementos do processo de software real que não tinham sido descritos no GPS no período avaliado, considerando os quatro produtos da empresa.

Quadro 5: Resultado da coleta de dados da etapa de avaliação da abordagem.

Questão	Causas				Total		
	Interpretação do processo	Erros no preenchimento	Informações não coletadas	Outros motivos			
1. Número total de elementos do processo de software real não descritos no GPS.							
1.1 Número de atividades não descritas.					0		
1.2 Número de artefatos não descritos.					0		
1.3 Número de ferramentas não descritas.					0		
1.4 Número de papéis não descritos.					0		
1.5 Número de fluxos entre atividades não descritos.					0		
1.6 Número de fluxos de produtos não descritos.					0		
Total de inconsistências por causa	0	0	0	0	0		
3. Número total de elementos do processo de software descritos no GPS e não presentes no processo de software real executado.	Interpretação textual	Interpretação gráfica	Falta de informações	Omissão dos agentes	Cronograma atrasado	Outros motivos	
							Total
3.1 Número de atividades não executadas.			1		1		2
3.2 Número de artefatos não manipulados.		1	1	6	1		9
3.3 Número de ferramentas não utilizadas.							0
3.4 Número de atribuições não aplicadas pelo papel.							0
3.5 Número de fluxos entre atividades não executados.							0
3.6 Número de fluxos de artefatos não executados.			1	3	1		5
Total de inconsistências por causa	0	1	3	9	3	0	16

A inexistência de inconsistências e, conseqüentemente, falhas relacionadas à métrica 1 nos dados coletados, demonstrou também que o processo de software descrito no GPS estava adequado para os diferentes produtos da empresa (um determinado produto poderia exigir o acréscimo ou decréscimo de elementos do processo de software para atender às suas necessidades). Entretanto, a inexistência de inconsistências reforçou a uniformização do processo de software descrito na microempresa um único modelo descritivo. Provavelmente, a similaridade entre os processos de software aplicados aos diferentes produtos esteja relacionada ao fato da empresa desenvolver produtos para um mesmo ramo de atividade.

5 AVALIAÇÃO DA ABORDAGEM

A avaliação da abordagem ocorreu logo após o término de sua aplicação e enfatizou dois aspectos: a acurácia da descrição do processo de software (Guia de Processo de Software) em relação ao processo de software real e se a abordagem havia atendido às necessidades (requisitos) da microempresa em relação à modelagem de processos de software. A primeira avaliação foi realizada por meio de 22 métricas elaboradas pela aplicação do paradigma GQM. De forma geral, as inconsistências constatadas entre o Guia de Processo de Software (GPS) e processo de software real, foram classificadas em dois tipos: inconsistências provocadas pelos agentes e inconsistências provocadas por falhas na abordagem. As inconsistências motivadas pelos agentes foram relacionadas à resistência em mudar a cultura de desenvolvimento de software da empresa. Na superação dessa dificuldade, foi imprescindível o envolvimento efetivo da Gerência de Pessoal reforçando a importância da adoção das novas práticas e cobrando a implementação das mesmas. As inconsistências motivadas pela abordagem se referiram ao fato da abordagem não considerar que o processo de software é dinâmico e, portanto, deveria prover mecanismos para verificar e atualizar o GPS. O segundo aspecto avaliado na abordagem ocorreu pela comparação entre os requisitos definidos inicialmente pela empresa na qual o estudo foi realizado e o atendimento da abordagem a esses requisitos. Nesse item, a abordagem contemplou satisfatoriamente às necessidades determinadas pela empresa.

O estudo de caso foi realizado em aproximadamente nove meses. Entretanto, dois funcionários se ausentaram da empresa (férias) adiando a descrição da atividade de Logística por aproximadamente um mês. A soma das horas destinadas ao estudo de caso totalizou 722h23min (sendo 686h23min para descrever a versão inicial do GPS e 36h para reformular e validar o GPS final – vide detalhes na Tabela 2). As atividades Suporte, Gerência de Tarefas, Implementação, Teste e Teste de Instalação somaram maior número de horas devido a reaplicação das Etapas de Descrição dos Elementos do Processo de Software e Verificação & Validação pelo fato de ocorrerem inconsistências entre as métricas e representações presentes no Guia de Processo de Software.

Tabela 2: Total de horas empregas na descrição do processo de software

Etapas	Horas
Planejamento inicial	125:10:00
Descrição da atividade Suporte	94:31:00
Descrição da atividade Gerência	81:37:00
Descrição da atividade Implementação	79:47:00
Descrição da atividade Teste	78:56:00
Descrição da atividade Controle Versão	51:36:00
Descrição da atividade Teste Instalação	78:05:00
Descrição da atividade Logística	45:21:00
Descrição da atividade Instalação e Treinamento	51:20:00
Reformulação do GPS, validação e verificação	36:00:00
Total de horas	722:23:00

A distribuição das horas entre as principais atividades desenvolvidas no estudo de caso (vide detalhes na Tabela 3) concentrou-se principalmente na coleta de dados por intermédio dos instrumentos de coleta de dados (76%) seguida da representação gráfica dos componentes do processo de software por meio da ferramenta Spearmint (10,5%). As demais atividades, reuniões com os agentes (aprovação do guia em cada etapa e discussão de inconsistências), reformulação e validação do GPS, entrevistas e treinamento (abordando os principais tópicos relacionados à modelagem de processos de software) ocuparam, respectivamente, 5,5%, 5%, 2% e 1% do total das horas empregadas no estudo de caso. O Guia de Processo de Software apresentou na sua versão final: oito atividades de alto nível, vinte e seis sub-atividades, quinze artefatos, oito papéis e dez ferramentas totalizando sessenta e sete componentes descritos.

Tabela 3: Distribuição de horas entre as principais atividades do estudo de caso

Atividades	Total horas
Coleta e análise de dados	549:01:00
Entrevistas	14:52:00
Representação gráfica dos elementos	75:53:00
Reuniões com agentes	39:37:00
Reformulação e validação do GPS	36:00:00
Treinamento	7:00:00
Total de horas do estudo de caso	722:23:00

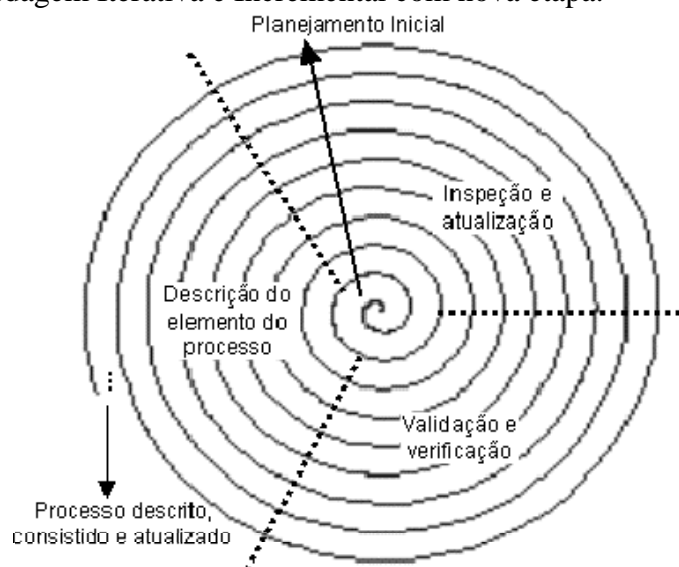
As dezesseis inconsistências coletadas por intermédio da métrica 3 (vide Quadro 5) foram enquadradas em quatro das seis causas previstas: interpretação gráfica (1), falta de informação (3), omissão do agente (9) e cronograma atrasado (3). A maior parte das inconsistências foi provocada por falhas do agente (tempo e omissão) ao executar o processo de software em desconformidade com o GPS (12) sendo oito inconsistências observadas na etapa 1 da coleta de dados. O número de inconsistências

observadas na primeira semana de coleta de dados, após a institucionalização do processo de software da microempresa, provocou algumas discussões entre o Gerente de Tarefas e agentes. As discussões abordaram a importância da execução do processo de software de acordo com o GPS e possíveis consequências caso não fosse cumprido (advertências verbais e escritas). O número de inconsistências provocadas pelos agentes observados nos demais dias de coleta de dados diminuiu significativamente indicando a importância da gerência na institucionalização do processo de software.

Embora o GPS apresentasse todos os elementos do processo de software, algumas inconsistências foram constatadas quanto ao seu conteúdo. Duas causas foram identificadas pelas métricas: falta de informações no GPS para executar o processo real (3) e erro na interpretação dos gráficos (1). Essa constatação apontou deficiências na abordagem por não considerar a dinâmica do processo de software real. Uma abordagem deve não somente definir “como” modelar o processo de software, mas também dispor de mecanismos que possibilitem o acompanhamento do processo de software institucionalizado para, conseqüentemente, corrigir deficiências (por exemplo, descrições incompletas ou ambíguas, contradição etc) e/ou atualizar o GPS quando mudanças ocorrerem no processo de software real (por exemplo, redefinição de critérios, mudança nos padrões, acréscimos de novos elementos do processo de software etc).

As inconsistências motivadas tanto pelos agentes como por falhas da abordagem sugeriram a necessidade, em estudos futuros, de modificações na estrutura da abordagem que permitam inspecionar e atualizar o processo de software descrito. A abordagem poderia ser adequada por meio do acréscimo de uma nova etapa (vide sugestão na Figura 71) a qual seria responsável tanto pela inspeção (garantir que o processo de software real está sendo executado conforme descrito no GPS) quanto pela atualização do GPS (observar e incorporar mudanças do contexto do processo de software real ao GPS). No entanto, a proposta de adequação da abordagem pela adição de uma nova etapa deve ser objeto de novos estudos que avaliem sua eficiência.

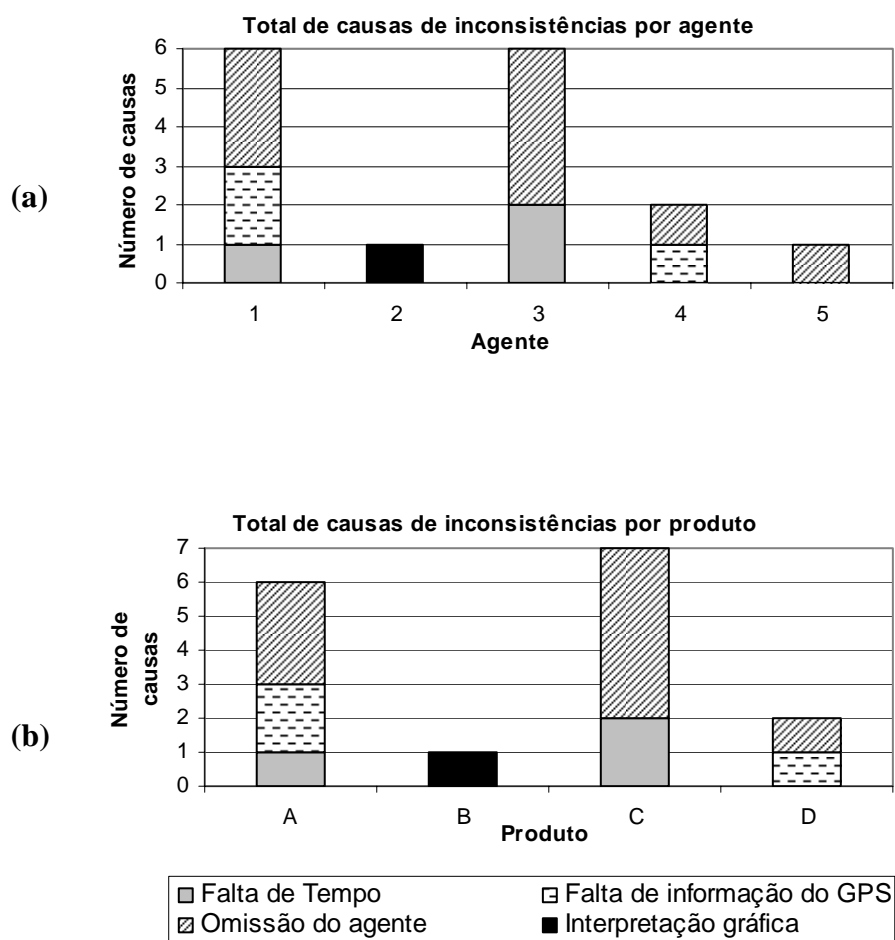
Figura 71: Abordagem Iterativa e Incremental com nova etapa.



Considerando as variáveis agentes e produtos, constatou-se que o maior número de causas de inconsistências foi provocado por omissão do agente (“esquecimento”, “não achou relevante aplicar o GPS em um determinado momento”, “adoção de outro meio não especificado no GPS”). Falta de tempo e falta de informação no GPS apresentaram número igual de causas de inconsistências e, em menor número, a interpretação gráfica (vide Figura 72 – ‘a’ e ‘b’).

Os dados demonstraram que as causas estavam distribuídas e não se concentraram somente em um agente ou produto (maior incidência). Portanto, os dados sob o ponto de vista das variáveis agente e produto, aparentemente, não indicaram necessidades específicas em relação a essas variáveis. Por exemplo, se um determinado produto apresentasse maior número de uma das causas, poderia haver a indicação de que esse produto, provavelmente, exigiria uma atenção maior em relação a essa causa. No entanto, é necessário que a coleta de dados seja realizada por um maior período de tempo para que essa especulação seja verificada com maior precisão.

Figura 72: Número de causas de inconsistências provocadas por agentes e produtos



5.1 Avaliação da abordagem em relação aos requisitos da microempresa

O segundo aspecto avaliado na abordagem foi sua conformidade aos requisitos definidos pela microempresa, ou seja, como a abordagem se portou diante das necessidades da microempresa em relação à modelagem de processos de software. Os requisitos foram definidos com base na pesquisa realizada na etapa 1 da metodologia e restrições específicas apresentadas pela microempresa antes da aplicação da abordagem. Para análise dos dados foi considerada a conformidade entre os requisitos previamente definidos e meios utilizados pela abordagem para atender a essas especificações (vide Quadro 6).

Quadro 6: Avaliação da conformidade aos requisitos da empresa

Requisitos da empresa	Meio utilizado na abordagem
1. Exigir pouco tempo dos agentes em entrevistas e reuniões. <u>Parâmetros</u> : uma entrevista por semana com duração máxima de 2h por agente (período da tarde), reuniões com duração máxima de 4h sendo realizada somente na segunda-feira no período da manhã (periodicidade quinzenal preferencialmente).	<ul style="list-style-type: none"> - Utilização da estratégia iterativa e incremental para dividir o processo de software em partes menores. - Utilização de formulários estruturados para orientar, conduzir e agilizar as entrevistas e coleta de dados. - Utilização de representações gráficas para facilitar a comunicação com os agentes entrevistados.
2. Modelar o processo de software de acordo com as prioridades e características da empresa.	- A abordagem considerou as características da empresa ao incluir em sua estrutura na Fase de Reconhecimento da Organização os passos 1 (Definir os objetivos da descrição do processo de software) e 2 (Levantar os produtos mantidos pela empresa).
3. Priorizar determinados elementos do processo de software.	<ul style="list-style-type: none"> - Utilização da estratégia <i>top-down</i> para descrever o processo de software em profundidade. - Foram utilizados os dados específicos da empresa coletados na pesquisa do Perfil de Microempresas (anexo A) conduzida na etapa 1 da metodologia.
4. Não exigir conhecimentos prévios em modelagem de processo de software.	- A abordagem procurou atender esse requisito por meio da adição da Fase de Treinamento.

A utilização das estratégias iterativa e incremental associada à *top-down*, possibilitou dividir o processo de software em partes menores que, por sua vez, dividiram a complexidade do processo de software real. Partes menores do processo de software possibilitaram manipular um número menor de informações e, conseqüentemente, exigiu pouco tempo para a realização da coleta de dados junto aos agentes. Além disso, a utilização de formulários estruturados agilizou a coleta de dados nas entrevistas definindo e organizando as informações que deveriam ser coletadas. Em resumo, a estratégia *top-down* apresentou diversas vantagens frente às dificuldades apontadas pela empresa em relação à modelagem de processos de software:

- a) Permitiu que experiências (em modelagem de processo de software), que até então eram inexistentes, fossem adquiridas pelos participantes à medida que os elementos do processo de software (atividades, sub-atividades, artefatos, papéis, ferramentas etc) eram descritos e reaproveitadas na descrição dos demais elementos do processo de software;
- b) A descrição gradual do processo de software possibilitou que fossem realizadas verificações e validações durante toda a aplicação da abordagem, evitando que inconsistências na descrição fossem propagadas e/ou descobertas somente ao término da modelagem do processo de software;
- c) Dividir a complexidade inerente à atividade de descrição do processo de software demandou menor tempo em atividades relacionadas à modelagem de processos de software e, conseqüentemente, permitiu a participação imprescindível das pessoas diretamente envolvidas no processo de software da empresa;
- d) Possibilitou à empresa selecionar e determinar os elementos do processo de software considerados críticos e que, portanto, deveriam ser modelados em um primeiro momento;

Concretamente, foram registradas 14h52m (2% do total do estudo de caso) em entrevistas sendo que duas entrevistas (2h13m e 2h31m) de um total de 10 ultrapassam o tempo limite de duas horas definido pela empresa. Entretanto, não foram somadas às horas de entrevistas, o tempo de perguntas realizadas aos agentes durante as observações do processo de software em execução. Todavia, esse tempo foi considerado no total de horas da coleta de dados.

As reuniões foram realizadas às segundas-feiras (com exceção de uma que foi realizada em uma terça-feira) no período da manhã. O tempo total das reuniões somou 39h37m. Duas reuniões ultrapassaram o limite de 4h estabelecido pela empresa, entretanto, nos dois casos houve a anuência da empresa visto tratarem de tópicos que exigiram maior tempo para sua discussão: definição do objetivo da modelagem de processo de software (4h15m) e seminário de apresentação e aprovação do GPS (7h23m – realizado em um sábado). O treinamento ministrado no início da aplicação da abordagem contribuiu para introduzir os conceitos básicos relacionados à modelagem de processo de software e apresentar a abordagem proposta no presente trabalho. Em linhas gerais, a abordagem atendeu aos requisitos relacionados ao tempo determinados pela empresa e o treinamento proporcionou informações básicas sobre modelagem de processo de software e aplicação da abordagem.

Quanto ao requisito 2, o levantamento conduzido na Fase de Reconhecimento da Organização disponibilizou dados que permitiram adequar a modelagem de processo de software ao perfil da empresa. O levantamento apontou como deveria ser procedida a modelagem do processo de software (iniciando pela atividade de Suporte e prosseguindo pelas demais atividades sequencialmente). Além disso, a caracterização dos produtos indicou a modelagem de um único processo de software à empresa.

Os dados coletados na Pesquisa Perfil de Microempresas (Anexo A) determinaram os principais elementos do processo de software a serem descritos (atividades, sub-atividades, artefatos, ferramentas, papéis, fluxo de controle entre atividades e fluxo de artefatos) atendendo ao requisito 3. Devido à exigência da empresa em modelar progressivamente o processo de software, foi adotada a estratégia *top-down* em profundidade para atender a esse critério.

A estratégia *top-down* foi aplicada em profundidade (vide Figura 23) pelo fato da empresa priorizar inicialmente a descrição de todos os elementos do processo de software da atividade de Suporte, considerada a mais crítica. Os dados coletados na pesquisa e adoção da estratégia *top-down* permitiram à abordagem proposta nesse trabalho, descrever o processo de software de acordo com a necessidade

prescrita pela empresa. Os dados coletados por meio das métricas (vide Quadro 5) evidenciam, dentro do período de coleta de dados, o cumprimento ao requisito 3.

Entretanto, a aplicação da estratégia *top-down* em profundidade apresentou algumas dificuldades. Em diversas situações ocorreu a necessidade de relacionar um elemento do processo de software descrito a outros elementos ainda não descritos. Por exemplo, ao descrever a sub-atividade “Analisar Ocorrência”, percebeu-se que a mesma deveria se relacionar com a sub-atividade “Modificar Artefatos” ainda não descrita e, portanto, a relação não poderia existir nesse momento. Posteriormente, ao descrever a sub-atividade “Modificar Artefatos”, a relação entre essa e a sub-atividade “Analisar Ocorrência” deveria ser recuperada. Essa dificuldade foi observada na relação entre elementos de processo de software, mais especificamente na relação entre atividades, na relação entre artefatos e na relação entre atividades e artefatos. Provavelmente, se fosse adotado a estratégia *top-down* em largura (vide Figura 24) tal dificuldade não ocorresse. Portanto, essa constatação deve ser considerada ao escolher a estratégia *top-down*.

6 DISCUSSÃO

Nos últimos anos, a modelagem de processos de software tem sido considerada como uma área fundamental no alcance da qualidade e produtividade do processo de software. Diante da relevância da área conferida pela indústria e academia, diversas pesquisas surgiram para aprimorar e propor novas tecnologias. KAISER (1990), NGUYEN, WANG & CONRADI (1997) e SILVA (2001) apresentam ambientes de desenvolvimento de software. Outros estudos (BECKER-KORNSTAEDT & WEBBY, 1998; COCKBURN, 2000) apresentam diretrizes que norteiam modelos de processo de software. Pesquisas como as de BECKER, HAMANN & VERLAGE (1997), BECKER-KORNSTAEDT & BELAU (2000) abordam a modelagem descritiva de processos de software. Aspectos conceituais, terminológicos e taxonômicos relacionados à modelagem de processo de software são discutidos em CURTIS, KELLNER & OVER (1992), LONCHAMP (1993), HUFF (1996) e SCACCHI (2001). Ainda há pesquisas que apresentam linguagens capazes de representar os modelos de processo por meio de uma sintaxe precisa e semântica bem definida (CONRADI et al 1992; BIRK et al, 1997; SUTTON & OSTERWEIL, 1997).

Entretanto, poucos estudos abordam “como” modelar processos de software. Em resposta a essa necessidade, alguns estudos (MADHAVJI et al, 1994; HÖLTJE et al, 1994; BRIAND et al, 1998; BECKER-KORNSTAEDT, 2001; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001; MACHADO et al, 2001) propõem abordagens que sistematizam a modelagem de processo de software. As abordagens propostas procuram definir um conjunto de passos que sistematizam a captura de conhecimentos sobre o processo de software em um documento descritivo. Muitos aspectos são comuns entre essas abordagens e podem ser utilizados como referencial à elaboração de novas abordagens.

A comparação dos estudos apresentados na literatura sobre descrição processo de software (vide seção 3.4) e a abordagem apresentada no presente trabalho, demonstra que existem poucas diferenças considerando as etapas necessárias para

modelar o processo de software, mesmo sendo realizados em diferentes organizações que desenvolvem produtos distintos (vide Tabela 4).

Tabela 4: Similaridade entre as etapas dos estudos de casos

Estudo ▶ Etapas genéricas ▼	GSFC	LG	Elicit	Daimler Chrysler	Thales	Abordagem Proposta
1. Compreensão da organização	✓	✓	✓			✓
2. Objetivo da modelagem	✓	✓	✓	✓	✓	✓
3. Definição da estratégia		✓	✓			✓
4. Treinamento e Esclarecimento		✓	✓	✓	✓	✓
5. Descrição do modelo em alto nível		✓		✓	✓	✓
6. Descrição detalhada do modelo	✓	✓	✓	✓	✓	✓
7. Revisão do modelo descrito	✓	✓	✓	✓	✓	✓
8. Validação do modelo descrito	✓		✓	✓	✓	✓

Considerando ainda as particularidades do presente estudo, alguns aspectos da abordagem proposta se diferenciaram dos estudos relatados (vide Tabela 5). A maioria dos estudos aplica as etapas das abordagens linearmente, ou seja, a execução das etapas é realizada seqüencialmente (MADHAVJI et al, 1994; BRIAND et al, 1998; BECKER-KORNSTAEDT, 2001; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001). O estudo conduzido por MACHADO et al (2001) embora possa ser caracterizado como uma abordagem linear, faz referências a estratégia incremental de descrição do processo de software em uma única etapa da abordagem. Diferentemente dos estudos citados, no presente trabalho, todas as etapas foram acondicionadas à estratégia iterativa e incremental para permitir a descrição e validação progressiva do processo de software real. Embora não haja relatos na literatura especializada sobre o uso da estratégia iterativa e incremental associada à *top-down*, a aplicação dessa estratégia possibilitou a modelagem do processo de software de uma microempresa atendendo às necessidades e expectativas previamente definidas pela mesma.

Outro aspecto observado nas experiências e abordagens para descrever processos de software (com exceção do estudo conduzido na LG, o qual não apresenta nenhuma referência ao meio utilizado para representar o modelo de processo de software) diz respeito à utilização de representações gráficas e descrições textuais

(linguagem natural). Os estudos conduzidos pelo IESE (DaimlerChrysler e Thales) enfatizam que a descrição de processo de software utilizando esses dois meios torna o modelo de processo descrito mais compreensível aos seus usuários. A representação gráfica do modelo de processo, na maioria dos estudos, utiliza ferramentas automatizadas e, no caso das descrições textuais, normalmente é utilizado texto estruturado. A utilização de texto estruturado é justificada pelo fato de contribuir com a diminuição de possíveis redundâncias comuns ao utilizar linguagem natural. O estudo apresentado por HÖLTJE et al (1994) propõe ainda um conjunto de atributos que norteiam a descrição dos elementos de processo de software.

Tabela 5: Principais características dos estudos ao descrever o processo

Características ▼	Estudo ►						Abordagem Proposta
		GSFC	LG	Elicit	Daimler Chrysler	Thales	
1. Utiliza notação gráfica para descrever o modelo		Modelo A-D	*	Statemate	Spearmint	Spearmint	Spearmint
2. Utiliza ferramenta automatizada para representação gráfica		*	*	Statemate	Spearmint	Spearmint	Spearmint
3. Utiliza descrições textuais complementares		Texto estruturado	*	Texto estruturado	Texto estruturado	Texto estruturado	Texto estruturado
4. Meio utilizado para estruturar as descrições textuais		Formulários	*	Formulários	Formulários	Formulários	Formulários
5. Abordagem utilizada para avaliar o modelo descrito		Não avalia	*	GQM	GQM	GQM	GQM
6. Estratégia utilizada para descrever o processo real		linear	linear	linear	linear	linear	Iterativa Incremental
7. Porte da organização		G	M	*	G	G	MC
8. Coleta de dados	Entrevistas	Sim	Sim	Sim	Sim	Sim	Sim
	Questionários	*	*	Sim	Sim	Sim	Sim
	Observações	Sim	Sim	Sim	Sim	Sim	Sim
	Documentos	Sim	Sim	Sim	Sim	Sim	Sim
Legenda:	*: Não consta no estudo			M: Média / G: Grande / MC Micro			

Quanto à avaliação da aplicação das abordagens, MACHADO et (2001) não apresenta em seu estudo a forma de avaliação da abordagem proposta. BRIAND et al (1998) foca os defeitos gerados durante o processo de software e não considera a própria abordagem na sua avaliação. Outros estudos (MADHAVJI et al, 1994; BECKER-KORNSTAEDT, 2001; BECKER-KORNSTAEDT, NEU & HIRCHE, 2001) aplicam o paradigma GQM, no entanto, não fica claro como e quais métricas

foram elaboradas e aplicadas para avaliar as abordagens. MADHAVJI et al (1994) exemplifica apenas algumas métricas utilizadas na avaliação do Método Elicit. O presente trabalho se propôs a explicitar “como” e “o que” avaliar com relação à aplicação da abordagem proposta por intermédio das metas e plano de mensuração baseado no paradigma GQM.

A aplicação das métricas nos três estudos que utilizam o paradigma GQM ocorreu somente após a descrição total do processo de software. Diferentemente dessas abordagens, devido a descrição iterativa e incremental adotada no presente estudo, a coleta e avaliação do modelo descrito por meio do paradigma GQM ocorreu a cada iteração do modelo descritivo. Desta forma, é possível avaliar o modelo descrito em fases iniciais e evitar que inconsistências sejam somente descobertas no final da modelagem do processo.

Finalmente, considerando que abordagens descritivas para modelar processos de software são relativamente recentes e pouco exploradas, que a estratégia iterativa e incremental associada à *top-down* e contexto de microempresa não tenham sido discutidas nos estudos consultados, que as avaliações da aplicação das abordagens descritivas presentes na literatura consultada não apresentam claramente o que foi avaliado, torna-se difícil realizar comparações pontuais entre as pesquisas. É necessário evidenciar, portanto, que não se trata de comparações que questionem a efetividade das abordagens, mas trata-se da focalização dos aspectos necessários, buscando expandir a compreensão sobre o desenvolvimento e aplicação de abordagens descritivas para modelagem de processo de software submetendo-as a contextos diversificados de forma que contribuam para essa área de pesquisa.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A modelagem descritiva de processo de software por meio de abordagens que sistematizam “como” realizá-la é uma proposta discutida em vários estudos. Abordagens descritivas são consideradas um meio para alcançar melhorias no processo de desenvolvimento e manutenção de software. O presente trabalho propôs, aplicou e avaliou uma abordagem para descrever o processo de software de uma microempresa de desenvolvimento de software. A abordagem foi elaborada com base em estudos apresentados na literatura e requisitos determinados pelo contexto na qual deveria ser aplicada. A abordagem é composta de três etapas, sendo que cada etapa possui fases contendo passos para sua execução.

Por meio da abordagem sugerida no presente trabalho, foi possível modelar o processo de software real de uma empresa de pequeno porte atendendo a um dos seus principais requisitos: modelar o processo de software real de forma progressiva. Essa particularidade (modelagem iterativa e incremental *top-down*) presente na abordagem permitiu: a) eleger, por ordem de prioridade, os elementos do processo de software considerados críticos pela empresa e que deveriam ser modelados em um primeiro momento; b) Dividir a complexidade inerente à atividade de descrição de processo de software; c) Avaliar e validar o modelo progressivamente e, d) acumular experiência na medida em que o processo de software real era modelado. Ao todo, foram identificados e descritos no modelo de processo de software da empresa sessenta e sete componentes do processo de software, sendo: oito atividades de alto-nível, vinte e seis sub-atividades, quinze artefatos, oito papéis e dez ferramentas.

A avaliação do modelo de processo de software ocorreu por meio do paradigma GQM que permitiu identificar diversas inconsistências na medida em que o processo de software real era descrito. A principal causa das inconsistências foi a omissão dos agentes (desconformidade entre sua prática e o GPS), reforçando, portanto, a questão cultural, ou seja, a existência de um modelo de processo formal não garante

sua institucionalização na empresa se não houver também uma mudança na atitude dos agentes envolvidos.

Em trabalhos futuros é necessário aperfeiçoar a abordagem proposta no presente trabalho incluindo mecanismos que possibilitem inspecionar o modelo de processo de software descrito (Guia de Processo de Software) para corrigir omissões que por ventura não foram percebidas durante sua elaboração. E, ainda, atualizar o modelo de processo de software descrito (GPS) para acomodar mudanças provocadas, por exemplo, por evoluções do processo de software real devido a sua dinâmica. Conforme dados apresentados no presente trabalho, a modelagem de processo é uma tarefa trabalhosa que exige muitas horas para ser concluída, então, o desenvolvimento de uma ferramenta automatizada poderia contribuir no sentido de diminuir o tempo necessário para modelar um processo de software, principalmente em atividades relacionadas à coleta de dados.

Além disso, devido aos poucos estudos relatados na literatura direcionados à modelagem de processos de software, deve haver uma preocupação por parte de pesquisas que se proponham abordar a descrição de processos de software, em explicitar os critérios, parâmetros, meios, métodos, técnicas etc ao relatar suas pesquisas. Quando esses dados são apresentados, é possível reaplicar, comparar, questionar, validar e/ou derivar novos estudos que pretendam abordar tal tema como objeto de pesquisa.

Finalmente, por tratar-se de um estudo aplicado somente a uma microempresa, obviamente, não é possível inferir, que a mesma seja adequada para empresas de igual ou maior porte. Além disso, as características de uma abordagem descritiva pressupõem que as particularidades do contexto na qual será aplicada devam ser consideradas. Entretanto, se considerados os limites apresentados no presente trabalho e as lições aprendidas, a abordagem proposta no presente trabalho pode contribuir fornecendo diversos subsídios para estudos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

- ALVES, Evandro Plese; FALBO, Ricardo de Almeida. Implantando um programa de melhoria de processo: uma experiência prática. WQS'2001 – Workshop de Qualidade de Software, 2001, p. 161-168.
- AMBRIOLA, Vincenzo; CONRADI, Reidar; FUGGETTA, Alfonso. Assessing process-centered software engineering environments. In: ACM Transactions on Software Engineering and Methodology, v. 6, n. 3, Jul. 1997, p. 283-328.
- ARMITAGE, James W.; KELLNER, Marc I. A conceptual schema for process definitions and models. In Dewayne E. Perry, editor, Proceedings of the Third International Conference on the Software Process. IEEE Computer Society Press, october 1994, p. 153-165.
- BANDINELLI, Sergio; FUGGETTA, Alfonso; GHEZZI, Carlo. Software process model evolution in the SPADE environment. GOODSTEP, Technical Report n. 014, December, 1993.
- BASILI, R. Victor; TURNER, Albert J. Iterative enhancement: a practical technique for software development. In: IEEE Transactions on Software Engineering, v. 1, n. 4, December 1975, p. 390-396.
- BASILI, Victor R.; WEISS, David M. A methodology for collecting valid software engineering data. In IEEE Transactions on Software Engineering, v. SE-10, N. 6, November 1984.
- BASILI, Victor; BRIAND, Lionel; CONDON, Steven; KIM, Yong-Mi; MELO, Walcélio L.; VALETT, Jon D. Understanding and predicting the process of software maintenance releases. Proceedings of the 18th International Conference on Software Engineering (ICSE'96), 1996, p. 464-474.

- BASIL, Victor “Software Modeling and Measurement: The Goal/Question/Metric Paradigm,” University of Maryland, CS-TR-2956, UMIACS-TR-92-96, September 1992.
- BAXTER, Ira D. Design maintenance systems. Communications of the ACM, v.35, n. 4, April, 1992, p. 73-89.
- BECKER, Ulrike; HAMANN, Dirk; VERLAGE, Martin. Descriptive modeling of software process. IESE-Report No. 045.97/E version 1, Kaiserslautern, Germany, 1997.
- BECKER-KORNSTAEDT, Ulrike; WEBBY, Richard. Towards a comprehensive schema integrating software process modeling and software measurement. (IESE-Report No. 021.97/E version 1.1), Kaiserslautern, Germany, 1998.
- BECKER-KORNSTAEDT, Ulrike; BELAU, Wolfgang. Descriptive process modeling in an industrial environment: experience and guidelines. Proceedings of 7th European Workshop on Software Process Technology, Kaprun, Austria, February, 2000.
- BECKER-KORNSTAEDT, Ulrike; NEU, Holger; HIRCHE, Gunter. Software process technology transfer: using a formal process notation to capture a software process in industry. In: AMBRIOLA, Vincenzo (Ed.), Proceedings of the eighth european workshop on software process technology, lecture notes in Computer Science, Witten, Germany, p. 63-76, 2001.
- BECKER-KORNSTAEDT, Ulrike. Towards systematic knowledge elicitation for descriptive software process modeling. In: Frank Bomarius and Seija Komi-Servis (Editors), Proceedings of the Third International Conference on Product-Focused Software Processes Improvement (PROFES), Lecture Notes in Computer Science 2188, p. 312-325, Kaiserslautern, September, 2001.
- BENNETT, K. H.; RAJLICH, V. T. Software maintenance and evolution: a roadmap. Proceedings of the conference on the future of software engineering, Limerick, Ireland 2000.

- BIRK, Andreas et al. Process enactment. (IESE-Report No. 061.97/E version 1.0), Kaiserslautern, Germany, december, 1997a.
- BOEHM, Barry. Spiral development: experience, principles, and refinements. (CMU/SEI-2000-SR-008), Spiral Development Workshop February, 2000.
- BRIAND, Leonel C.; DIFFERDING, Christiane M.; ROMBACH, H. D. Practical Guidelines for Measurement-Based Process Improvement. Software Process Improvement and Practice, vol. 2, 1997.
- BRIAND, Leonel C.; KIM, Yong-Mi; MELO, Walcélio; SEAMAN, Carolyn; BASILI, Victor R. Q-mopp: qualitative evaluation of maintenance organizations, process and products. Journal Software maintenance: research and practice. John Wiley & Sons Ltd., n. 10, 1998, p. 249-278.
- CARR, Mahil; VERNER, June. Prototyping and software development approaches. Department of Information Systems, City University of Hong Kong, 1997. Artigo disponível na url <http://www.is.cityu.edu.hk/Research/Publication/paper/9704.pdf>. Consultado em 03/01/2002.
- COCKBURN, Alistair. Selection a project's methodology. IEEE Software Computer Soc. Press, Los Alamitos, California, July/August, 2000, p. 64-71.
- CONRADI, Reidar; FERNSTRÖM, Christer; FUGGETTA, Alfonso. Concepts for evolving software process. In PROMOTER book: Anthony Finkelstein, Jeff Kramer and Bashar A. Nuseibeh (Eds.): Software Process Modelling and Technology, 1993, p. 9-32.
- CONRADI, Reidar; JACCHERI, M. Letizia; MAZZI, Cristina; NGUYEN, Minh Ngoc; AARSTEN, Amund. Design, use, and implementation of SPELL: a Language for software process modeling and evolution. In J.-C. Derniame (ed.): Proc. from 2nd European Workshop on Software Process Technology (EWSPT'92), Trondheim, Norway, Sept. 1992.

- CONRADI, Reidar; FUGGETTA, Alfonso. Improving software process improvement. IEEE Software, July-August, 2001.
- CURTIS, Bill; KELLNER, Marc I.; OVER, Jim. Process modeling. Communications of the ACM, v. 35, n. 9, September 1992, p. 75-90.
- DURSKI, Gislene Regina. Diagnóstico da produtividade sistêmica do setor de software no Brasil. In: WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti da; NASCIMENTO, Célia Joseli. Qualidade e produtividade em software. São Paulo: Makron Books, 2001.
- ESTUBLIER, Jacky. Software configuration management: a roadmap. In: Anthony Finkelstein (Ed.), The future of software engineering, ACM Press, 2000.
- FEILER, Peter H. Configuration management models in commercial environments. Technical Report: CMU/SEI-91-TR-7. Pittsburgh: Carnegie Mellon University, 1991.
- FEILER, Peter H.; HUMPHREY, Watts S. Software process development and enactment: concepts and definitions. Proceedings of the Second International Conference on the Software Process (Held at Berlin, Germany, February 25-26, 1993), IEEE Computer Society Press, 1993, p. 28-40.
- HÖLTJE, Dirk; MADHAVJI, Nazim H.; BRUCKHAUS, Tilmann; HONG, Wonkook. Eliciting formal models of software engineering process. In: IBM Canada Ltd and The National Research Council of Canada, Proceedings CAS Conference (CASCON'94), Toronto, Ontario, Canada, 1994.
- HUFF, Karen E. Software process modeling. New York: John Wiley & Sons, 1996.
- HUFF, Karen E.; LESSER, Vitor R. A plan-based intelligent assistant that supports the software development process. Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, 1988, Boston, Massachusetts, United States, p. 97-106.

- HUMPHREY, Watts S.; KELLNER, Marc I. Software process modeling: principles of entity process models. (SEI-CMU-89-TR-2), Pittsburgh, PA.: Software Engineering Institute, Carnegie Mellon University, February, 1989.
- HUMPHREY, Watts S. Tendências segundo Watts Humphrey. In WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti da; NASCIMENTOS, Célia Joseli do (org.). Qualidade e produtividade em software, 4ª edição. São Paulo: Makron Books, 2001.
- IEEE Std 828-1998. IEEE standard for software configuration management plans. In: IEEE Software Engineering Standard Collection, 1998.
- IEEE Std 1219-1998. IEEE standard for software maintenance. In: IEEE Computer Society, Software Engineering Standards Committee, October, 1998.
- KAISER, Gail E. Experience with MARVEL. Proceedings of the 5th International Software Process Workshop on Experience with Software Process Models, 1990, Kennebunkport, Maine, United States, p. 82-84.
- KELLNER, Marc I.; BECKER-KORNSTAEDT, Ulrike; RIDDLE, William E.; TOMAL, Jennifer; VERLAGE, Martin. Process guides: effective guidance for process participants. Proc. 5th International Conf. on the Software Process: Computer Supported Organizational Work. Lisle, Illinois, USA, 14-17 June 1998. New Jersey: The International Software Process Association Press, 1998, p. 11-25.
- LIENTZ, Bennet P.; SWANSON, E. Burton. Problems in application software maintenance. Communications the ACM, n. I, v. 24, November, 1981.
- LINDVALL, Mikael; RUS, Ioana. Process diversity in software development. IEEE Software Computer Soc. Press, Los Alamitos, California, July/August, 2000, p. 14-17.
- LONCHAMP, Jacques. A structured conceptual and terminological framework for software process engineering. IEEE Software Computer Soc. Press, 1993.

- MACHADO, L. F. D. C. et al. Experiência na definição e implantação de processos de software. WQS'2001 Workshop de Qualidade de Software, Rio de Janeiro, 2001, p. 147-153.
- MADHAVJI, Nazim H.; GRUHN, Volker; DEITERS, Wolfgang; SCHÄFER, Wilhelm. Prism: methodology + process-oriented environment. Proceedings of the Twelfth International conference on Software Engineering, 1990, Nice, France.
- MADHAVJI, Nazim H.; HÖLTJE, Dirk; HONG, Wonkook; BRUCKHAUS, Tilmann. Elitict: a method for eliciting process models. In: IEEE Computer Society Press, Proceedings 3rd International Conference on Software Process, 1994.
- MATTSSON, Mira Kajko. A conceptual model of software maintenance. Proceedings of the 1998 International Conference on Software Engineering, Kyoto, Japan, 1998.
- McCONNEL, Steve. Rapid development. Redmond, WA: Microsoft Press, 1996.
- MCT/SEPIN. Qualidade e produtividade no setor de software brasileiro. Brasília, Ministério da Ciência e Tecnologia, Secretaria de Política de Informática, 2002.
- NASCIMENTO, Célia Joseli do; MARINHO, Diva da Silva. Diagnósticos da qualidade e produtividade em software in WEBER, Kival Chaves; ROCHA, Ana regina Cavalcanti da; NASCIMENTOS, Célia Joseli do (org.). Qualidade e produtividade em software, 4^a edição. São Paulo: Makron Books, 2001.
- NBR ISO/IEC 12207:1998. Tecnologia da informação – Processos de ciclo de vida de software.
- NGUYEN, Minh N.; WANG, Alf Inge; CONRADI, Reidar. Total software process model evolution in EPOS: experience report. Proceedings of the 1997 international conference on Software engineering, 1997, Boston, Massachusetts, United States, p. 390-399.
- OSTERWEIL, Leon J. Software processes are software too. Revisited: An Invited Talk on the Most Influential Paper of ICSE 9. Proceedings of the 1997 international

- conference on Software engineering, 1997, Boston, Massachusetts, United States, p. 539-548.
- PFLEEGER, Shari Lawrence. Maturity, models and goals: how to build a metrics plan. J. Systems Software, New York: Elsevier Science Inc., 1995, p. 143-155.
- PRESSMAN, Roger S. Software engineering: a practitioner's approach. New York: McGraw-Hill, 5th ed., 2001.
- ROYCE, W. W. Managing the development of large software systems: concept and techniques. Proceedings WESCON, AIEE, 1970, p. 1-9.
- SCACCHI, Walter. Process models in software engineering. J.J. Marciniak (ed.), Encyclopedia of Software Engineering, 2nd Edition, John Wiley and Sons, Inc, New York, February 2001.
- SHAW, Mary. Software engineering education: a roadmap. In FINKELSTEIN, Anthony (Ed.), The future of software engineering. New York: ACM Press, 2000.
- SILVA, Fábio Augusto das Dores. Um modelo de simulação de processos de software baseado em conhecimento para o ambiente PROSOFT. Porto Alegre, BR – RS, 2001. Dissertação (Mestrado em Computação) – Universidade Federal do Rio Grande do Sul.
- SOMMERVILLE, Ian. Software engineering. Addison-Wesley, 5th ed., 1996.
- SUTTON, Stanley M. Jr.; OSTERWEIL, Leon J. The design of a next-generation process language. Proceedings of the 6th European conference held jointly with the 5th ACM SIGSOFT symposium on Software engineering, 1997, Zurich, Switzerland, p. 142-158.
- SUTTON, Stanley M. The role of process in a software start-up. IEEE Software, Computer Soc. Press, Los Alamitos, California, July/August, 2000, p. 33-39.
- TURPIN, Russell. A progressive software development lifecycle. Proceedings of the 2nd International Conference on Engineering of Complex Systems (ICECCS '96), IEEE Computer Soc. Press, Los Alamitos, California, 1996, p. 208-211.

ANEXO A: RESULTADO DA PESQUISA DO PERFIL DE MICROEMPRESAS

Índice

<u>INTRODUÇÃO.....</u>	<u>1</u>
1.1 OBJETIVO DO LEVANTAMENTO DE DADOS	1
1.2 MÉTODO UTILIZADO PARA O LEVANTAMENTO DE DADOS	1
1.2.1 PROCEDIMENTO	2
<u>2 ANÁLISE E INTERPRETAÇÃO DOS DADOS.....</u>	<u>6</u>
2.1 PERFIL DAS EMPRESAS, PRODUTOS E DOMÍNIO DE APLICAÇÃO.....	6
2.2 CARACTERÍSTICAS E PARTICIPAÇÃO DO QUADRO DE FUNCIONÁRIOS	8
2.3 CARACTERÍSTICAS DOS PROCESSOS DE SOFTWARE DAS EMPRESAS.....	9
2.4 FORMALISMO APLICADO AOS ELEMENTOS DO PROCESSO DE SOFTWARE.....	10
2.5 DADOS SOBRE O PROCESSO.....	11
2.6 REQUISITOS PARA MODELOS DE PROCESSO	14
<u>3 CONSIDERAÇÕES SOBRE ASPECTOS METODOLÓGICOS DA PESQUISA</u>	<u>17</u>
<u>4 DISCUSSÃO E CONSIDERAÇÕES FINAIS</u>	<u>19</u>
<u>5 QUESTIONÁRIO MODELO.....</u>	<u>21</u>

Índice de Gráficos

<i>Gráfico 1: Tipos de software desenvolvido pelas empresas.</i>	7
<i>Gráfico 2: Mercado de atuação das empresas.</i>	7
<i>Gráfico 3: Tempo médio de desenvolvimento dos projetos.</i>	8
<i>Gráfico 4: Porcentagem de funcionários envolvidos em cada etapa do projeto de software.</i>	9
<i>Gráfico 5: Formalismo aplicado aos componentes do processo de software</i>	11
<i>Gráfico 6: Coleta de dados sobre o processo de desenvolvimento</i>	12
<i>Gráfico 7: Dificuldades na definição/manutenção do Ciclo de Vida.</i>	13
<i>Gráfico 8: Requisitos para modelos de processo (segundo as empresas).</i>	15

Índice de Tabelas

<i>Tabela 1: Caracterização das Empresas</i>	6
<i>Tabela 2: Características sobre o Ciclo de Vida (total: 6 empresas)</i>	9
<i>Tabela 3: Formas de utilização de Ferramentas no Ciclo de Vida</i>	14

INTRODUÇÃO

A literatura especializada apresenta diversos requisitos para modelos e modelagem de processos em empresas de desenvolvimento de software. No entanto, é necessário adequar os requisitos e diretrizes ao contexto ao qual serão aplicados (CONRADI, FERSTRÖM & FUGGETTA, 1993, p. 1). Considerando as propostas apresentadas na literatura e a necessidade de adequar as propostas ao perfil de microempresas, o presente trabalho realizou um levantamento sobre vários aspectos relacionados à modelagem de processos de software.

1.1 Objetivo do levantamento de dados

A existência de poucas pesquisas direcionadas à modelagem de processo de software de microempresas dificulta o delineamento das necessidades dessas empresas em relação ao seu processo de desenvolvimento de software. Portanto, é necessário obter informações mais detalhadas sobre aspectos que influenciam a definição de modelos de processo de software direcionados especificamente para esse perfil de empresa. O levantamento de dados teve como objetivo principal compreender a situação atual em relação ao uso de modelos de processos de software em microempresas.

1.2 Método utilizado para o levantamento de dados

A população-alvo da coleta de dados foi constituída de organizações que se enquadrem como microempresa segundo o porte, considerando a força de trabalho efetiva das organizações, que inclui sócios, dirigentes e empregados efetivos (MCT/SEPIN, 2002, p. 16).

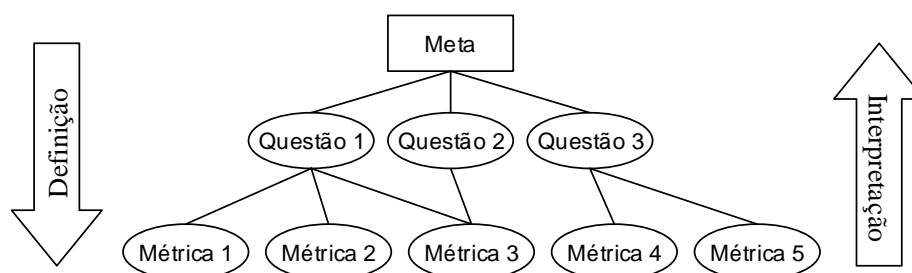
1.2.1 Procedimento

A pesquisa foi dividida em quatro etapas: elaboração do instrumento de coleta de dados, seleção das empresas, aplicação do instrumento de coleta de dados e análise e interpretação dos dados coletados.

Etapa 1: Elaboração do instrumento de coleta de dados

O instrumento de coleta aplicado é do tipo formulário estruturado composto por questões fechadas de opção única, questões fechadas de opção múltipla e algumas questões abertas e a aplicação do mesmo foi realizada por um único entrevistador. O questionário modelo pode ser consultado ao final do presente anexo. A estrutura do questionário foi baseada na abordagem GQM (*Goal/Question/Metric Paradigm*) o qual permitiu definir aspectos importantes que deveriam ser considerados e mensurados em microempresas de software no que se refere a modelos e modelagem de processos de software adequados a esse perfil de empresa. O paradigma GQM é uma abordagem orientada a metas que tem sido aplicada com muito sucesso em diversas organizações: NASA/SEL, TRW, Allianz, Robert Bosch, Schlumberger, Digital, Siemens e Ericsson (BIRK et al, 1997b). Um Plano GQM consiste na definição de metas, questões e métricas (vide Figura 1). As questões operacionalizam as metas e métricas quantificam as questões. Como pode ser observado na Figura 1, a definição do plano GQM ocorre de forma *top-down* e a interpretação dos dados ocorre de forma *bottom-up*. Maiores detalhes sobre o paradigma GQM podem ser encontrados em BIRK et al (1997b).

Figura 1: Abordagem GQM



Plano GQM – *abstraction sheet*

As perguntas constantes no questionário foram elaboradas com base nas questões presentes no *abstraction sheet* (Quadro 1). As questões foram divididas em dois grupos: 1º.) fatores de qualidade os quais abordam vários aspectos relevantes relacionados a modelos e modelagem de processos em microempresa de software e, 2º) fatores de variação os quais influenciam a definição e utilização dos modelos de processos no desenvolvimento de software. A disposição dos fatores de qualidade e fatores de variação nas respectivas colunas do Quadro 1 não indicam que estejam relacionadas, ou seja, o FQ1 não está relacionado ao FV1, pelo fato de se encontrarem na mesma linha.

Quadro 1: *Abstraction Sheet*

Metas ▼	Objeto Processo de software	Objetivo Caracterização	Enfoque de qualidade Eficiência	Ponto de vista Equipe de desenv.	Contexto Microempresas
Fatores de Qualidade (FQ)			Fatores de Variação (FV)		
FQ 1. Levantar Qual(is) ciclo(s) de vida a empresa adota FQ 1.1. Modelo de ciclo de vida utilizado FQ 1.2. Variação na utilização de ciclo de vida pela empresa FQ 2. Determinar o grau de formalismo aplicado na definição do ciclo de vida FQ 2.1. Adoção de formalismo no ciclo de vida FQ 2.2. Nível de formalismo aplicado ao ciclo de vida FQ 2.3. Forma como o ciclo de vida é documentado FQ 2.4. Freqüência na adoção de formalismo FQ 3. Coleta de dados sobre aplicação de processos de software FQ 4. Averiguar se são adotados modelos de melhoria de processos e guias FQ 5. Diagnosticar as principais dificuldades na implantação/manutenção de um ciclo de vida FQ 5.1. Custo na formalização do ciclo de vida FQ 5.2. Tempo despendido na formalização do ciclo de vida FQ 5.3. Ferramentas de apoio FQ 5.4. Métodos disponíveis no mercado FQ 5.5. Informação sobre os ciclos de vida FQ 5.6. Flexibilidade dos ciclos de vida FQ 5.7. Treinamento de pessoal FQ 5.8. Adequação do ciclo de vida ao perfil da empresa			FV 1. Localização da empresa (cidade/UF) FV 2. Porte da empresa (força de trabalho) FV 3. Domínio de produtos da empresa FV 3.1. Tempo de atuação no domínio da aplicação FV 3.2. Quantidade de produtos de software produzidos/mantidos pela empresa FV 3.3. Atividade da empresa no tratamento de software (tipo de software) FV 3.4. Setor de atividade da empresa (mercado de atuação) FV 4. Projeto FV 4.1. Duração típica de um projeto de software FV 4.2. Número habitual de pessoas envolvidas em um projeto de software FV 5. Tempo de atuação da empresa no mercado de desenvolvimento FV 6. Quadro funcional. FV 6.1. Experiência do quadro funcional no desenvolvimento de software FV 6.2. Política de qualificação profissional FV 6.3. Composição do quadro funcional FV 6.4. Rotatividade do quadro funcional FV 7. Investimento na melhoria do processo de software		

Etapa 2: Seleção das empresas

As empresas participantes da coleta de dados deveriam atender ao perfil de microempresa conforme o critério porte considerando a força de trabalho efetiva das organizações, que inclui sócios, dirigentes e empregados efetivos (MCT/SEPIN, 2002). As empresas foram indicadas em Florianópolis - SC pelo Projeto Geness e, em Londrina - PR pelo Núcleo Softex. Foram selecionadas seis empresas sendo uma empresa em Florianópolis e cinco empresas em Londrina.

Etapa 3: Aplicação do instrumento de coleta de dados

O instrumento de coleta de dados foi aplicado aos representantes das empresas por meio de entrevistas. Em todas as seis empresas, os entrevistados eram sócio-proprietários e participavam efetivamente dos processos de software da empresa. A aplicação do instrumento de dados durou, em média, uma hora e sua aplicação ocorreu nas próprias empresas com exceção da empresa localizada em Florianópolis-SC que ocorreu na sala do Projeto Geness localizada na Universidade Federal de Santa Catarina.

Etapa 4: Análise e interpretação dos dados

A análise e interpretação dos dados ocorreram por meio da contagem do número de ocorrências de respostas para cada questão respondida presente no questionário excetuando-se as questões abertas que foram analisadas qualitativamente. Comparações foram realizadas entre as respostas de cada questão com o objetivo de evidenciar as proposições concorrentes.

2 ANÁLISE E INTERPRETAÇÃO DOS DADOS

2.1 Perfil das empresas, produtos e domínio de aplicação

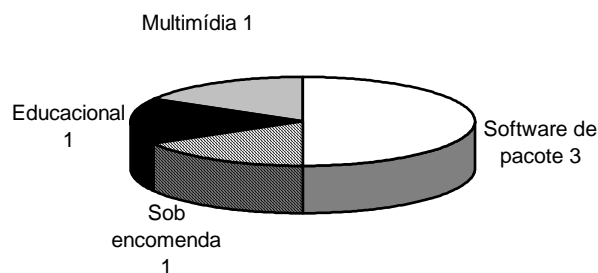
O instrumento de coleta foi aplicado em seis microempresas empresas de desenvolvimento de software sendo uma situada em Florianópolis (SC) e as demais em Londrina (PR). A Tabela 1 apresenta a caracterização das empresas.

Tabela 1: Caracterização das Empresas

Empresa	Data da Fundação	Quantidade de Funcionários	Quantidade de produtos	Tempo de experiência no domínio da aplicação
1	1991	Entre 1 - 9	1	mais de 3
2	2001	Entre 1 - 9	5	2 - 3
3	2001	Entre 1 - 9	2	mais de 3
4	N/R	Entre 1 - 9	4	1 - 2
5	1994	Entre 1 - 9	4	mais de 3
6	N/R	Entre 1 - 9	1	mais de 3

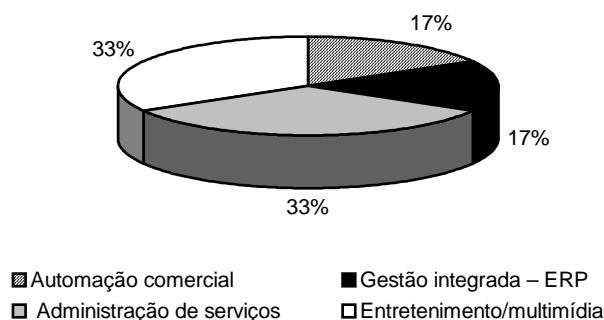
Como pode ser observado, as empresas enquadram-se no critério de microempresa por terem um quadro funcional entre 1 e 9 funcionários. A quantidade de produtos atualmente mantidos pelas empresas varia entre 1 e 5. Apesar do número de produtos variar entre 1 e 5 (em quatro empresas), esses produtos são desenvolvidos para o mesmo domínio de aplicação, não se caracterizando como produtos distintos. O tempo de experiência no domínio da aplicação, para a maioria das empresas, é superior a 2 anos. O Gráfico 1 apresenta os tipos de software produzidos pelas empresas. Observa-se que o tipo de software mais produzido é o de pacote (49%). Os demais tipos, sob encomenda, educacional e multimídia, são menos produzidos apresentando a mesma porcentagem de produção (17%).

Gráfico 1: Tipos de software desenvolvido pelas empresas.



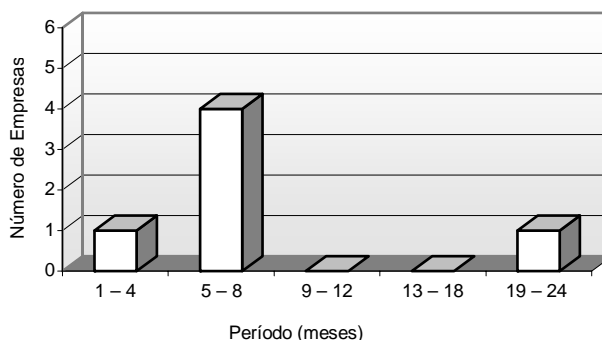
O Gráfico 2 apresenta o mercado de atuação das empresas. Quanto a este aspecto constatou-se que a atuação no mercado de administração em serviços e entretenimento/multimídia apresentaram maior percentagem (33%) e os mercados de Automação Comercial e Gestão Integrada menor percentagem (17%)

Gráfico 2: Mercado de atuação das empresas.



O tempo médio de desenvolvimento dos projetos pode ser observado no Gráfico 3. Verificou-se que quatro das seis organizações demoram em média de 5 a 8 meses na execução do projeto. O tempo de execução das outras duas empresas variou, sendo que em uma delas o tempo para execução do projeto varia entre 1 a 4 meses e na outra entre 19 a 24 meses.

Gráfico 3: Tempo médio de desenvolvimento dos projetos.

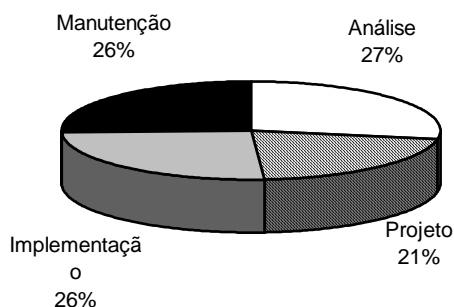


2.2 Características e participação do quadro de funcionários

Quanto à experiência profissional dos funcionários e a política de capacitação percebeu-se que a maioria dos funcionários possui experiência profissional acima de um ano e a política de capacitação, geralmente atinge todos os funcionários. No entanto, a capacitação dos funcionários não é uma prática sistemática na maioria das empresas (83,3%) sendo realizadas principalmente sob demanda. Observou-se também que a qualificação dos funcionários é predominantemente de graduados na área de informática, seguido por especialistas. A minoria dos funcionários apresenta-se sem formação ou com curso técnico. Todas as empresas consideraram a rotatividade de pessoal baixa ou inexistente.

O Gráfico 4 apresenta a média, em porcentagem, do número de funcionários envolvidos em cada etapa do projeto de software. Observa-se que não há uma diferença expressiva entre as etapas, sendo que etapa de análise envolvia 27% dos funcionários seguida pelas etapas de implementação e manutenção (26%) e etapa de projeto (21%). A distribuição equilibrada entre as principais etapas de desenvolvimento de software dá-se pelo fato de normalmente os mesmos funcionários atuarem em todas as etapas do desenvolvimento de software e, principalmente, pelo fato dos processos de desenvolvimento centrarem-se na manutenção de software. Processos de desenvolvimento de novos projetos de software, considerando que as empresas não possuem processo de desenvolvimento formalizado, tendem a destinar mais empregados à etapa de manutenção.

Gráfico 4: Porcentagem de funcionários envolvidos em cada etapa do projeto de software.



2.3 Características dos processos de software das empresas

Constatou-se em quatro organizações (66,6%) que o ciclo de vida utilizado é definido pela própria empresa, duas organizações (33,3%) utilizam ciclos de vida adaptados - uma utiliza o tipo Cascata e a outra o tipo Prototipação Evolutiva (vide Tabela 2). Esses dados confirmam que não existe um modelo de processo consensual que atenda às necessidades desse perfil de empresa o que provavelmente conduz as empresas a definirem seus próprios modelos de processo. Mesmos os modelos existentes, quando utilizados, são adaptados para atender as necessidades da empresa.

Tabela 2: Características sobre o Ciclo de Vida (total: 6 empresas)

Questões	Opções	Porcentagem
Ciclo de Vida utilizado	Definido pela Empresa	66,6
	Adaptado	33,3
Formalização do Ciclo de Vida	Parcial	66,6
	Não documentado	33,3
Variação do Ciclo de Vida em Projetos	Eventualmente	16,6
	Não varia	83,3
Documentação do Ciclo de Vida	Todos os projetos	33,3
	Maioria dos projetos	16,6
	Não é documentado	50,0

Nenhuma das empresas formaliza totalmente o ciclo de vida utilizado para desenvolver e manter seus produtos. Em quatro empresas (66,6%) a formalização

do Ciclo de Vida é parcial e, em duas (33,3%) o Ciclo de Vida não é documentado. A documentação parcial do Ciclo de Vida ocorre principalmente na etapa de projeto o que demonstra uma menor preocupação das empresas com as demais etapas do Ciclo de Vida. A utilização do mesmo Ciclo de Vida entre diferentes projetos, em cinco das seis organizações (83,3%) mostra-se invariável. Apenas uma empresa eventualmente varia o Ciclo de Vida.

Metade das empresas não documenta os processos do Ciclo de Vida considerando todos os projetos da empresa. A documentação de todos os projetos é realizada por duas empresas (33,3%). Apenas uma empresa (16,6%) documenta a maioria dos projetos. É importante observar que as empresas que documentam do ciclo de vida em diferentes projetos o fazem parcialmente, conforme dados descritos anteriormente, uma vez que nenhuma empresa documenta totalmente o Ciclo de Vida. A documentação do Ciclo de Vida, ainda que parcial, não é uma prática comum nessas organizações. Embora não tenha sido abordada no questionário, em conversas informais foi possível averiguar que o motivo da pouca atenção ao documentar o Ciclo de Vida é fundamentado na crença da perda de tempo. Essa crença da perda de tempo ao documentar o Ciclo de Vida, pode estar relacionada à forma como os Ciclos de Vida são documentados.

No caso das organizações que documentam o Ciclo de Vida observou-se que o meio mais utilizado (duas empresas – 33,3%) é o Editor de Texto ou similar. Apenas uma organização utiliza a ferramenta Case como um meio para documentação. Esses dados demonstram que nenhuma empresa utiliza técnicas ou ferramentas apropriadas para documentar o Ciclo de Vida. Isso pode contribuir para a crença de perda de tempo na documentação do Ciclo de Vida.

2.4 Formalismo Aplicado aos Elementos do Processo de Software

A formalização dos elementos do processo de software mostrou-se bastante variável (Gráfico 5). Na formalização das fases, os dados mostram que 17%

das empresas formaliza todas as fases e 83% não formaliza nenhuma das fases. Deve-se destacar que a única empresa a formalizar todas as fases apenas formaliza alguns componentes da fase como as atividades, critérios e papéis. Quanto à formalização das atividades constatou-se que 66% das empresas formaliza algumas atividades, 17% não formaliza qualquer uma das atividades e 17% formaliza todas as atividades. As técnicas e os métodos não são formalizados por nenhuma das empresas. Os critérios de entrada e saída não são formalizados por 83% das empresas. No caso da formalização dos papéis e responsabilidades 50% das empresas formalizam todos e as demais não os formalizam.

Gráfico 5: Formalismo aplicado aos componentes do processo de software

Todas formalizadas
 Algumas formalizadas
 Nenhuma formalizada

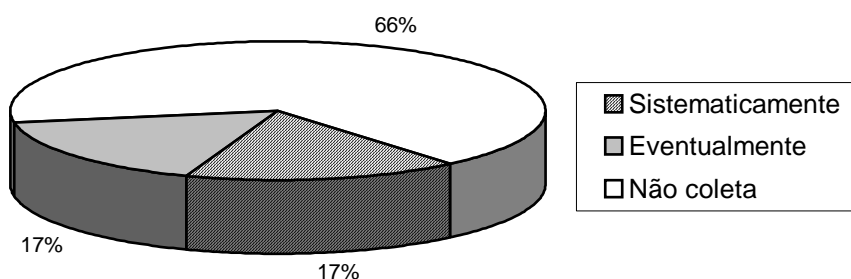
Os dados sobre a formalização dos componentes do processo de software demonstram que, embora as empresas utilizem modelos de processo (próprios ou adaptados), a maioria dos componentes não é formalizada ou parcialmente formalizada (minoria). Considerando os seis componentes do modelo de processo abordado na pesquisa, quatro deles (fases, critérios de entrada e saída, técnicas e métodos) apresentaram percentuais superiores a 80% de não formalização pelas empresas. O formalismo dos componentes ocorre totalmente em apenas três componentes (fases, atividades e papéis e responsabilidades), porém, o percentual de empresas que os formalizam é baixo. Os componentes fases e atividades apresentaram o mesmo percentual (17%) e o componente papéis e responsabilidades apresentaram percentuais iguais nas empresas (50%).

2.5 Dados sobre o processo

Quanto a coleta de dados sobre o processo de desenvolvimento de software (Gráfico 1), das seis empresas apenas uma (17%) a faz sistematicamente e outra eventualmente (17%). As demais empresas (66%) não coletam. Conseqüentemente, as empresas tendem a deter pouco conhecimento sobre os processos de software empregados no desenvolvimento de seus produtos. O baixo percentual de

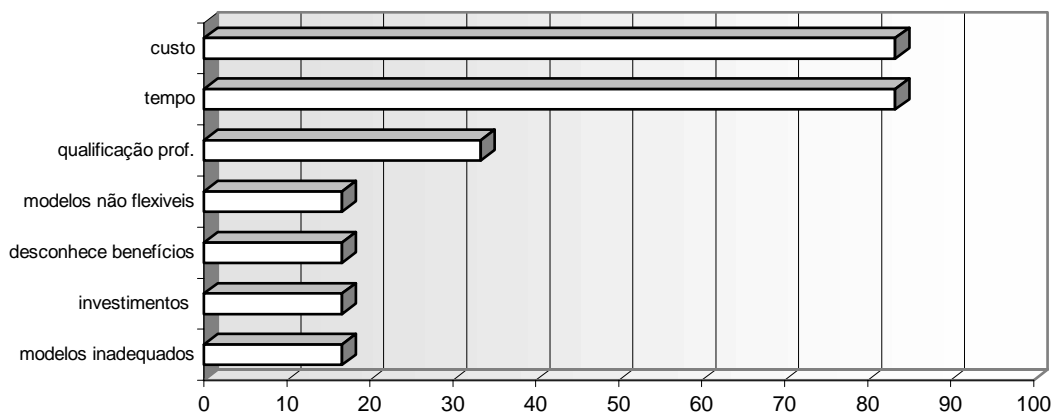
coleta de dados sobre o processo pode estar relacionado ao baixo percentual de formalismo aplicado aos componentes do processo de software uma vez que a definição dos processos constitui a base para a sua medição (PFLEEGER, 1995).

Gráfico 6: Coleta de dados sobre o processo de desenvolvimento



Quanto as dificuldades apresentadas pelas empresas para definição/manutenção do Ciclo de Vida (Gráfico 7), custo e tempo foram as dificuldades mais apontadas (83% das empresas). Uma outra dificuldade apontada que se destacou entre as demais foi a qualificação profissional (33%). As demais dificuldades tiveram o mesmo percentual (17%). As duas principais dificuldades (tempo e custo) estão estritamente relacionadas ao esforço necessário pelas empresas na definição de seus processos. A falta de profissionais qualificados para a definição de processos, pode estar relacionada a cultura estabelecida pelas empresas ao iniciarem suas atividades no desenvolvimento de software, ou seja, a maioria das empresas estabelecem uma prática de desenvolvimento que, após algum tempo é difícil de ser revertida.

Gráfico 7: Dificuldades na definição/manutenção do Ciclo de Vida.



Na Tabela 3 são apresentados os dados referentes ao emprego de ferramentas de apoio no Ciclo de Vida. Os resultados indicam que 50% das organizações fazem uso sistemático de ferramentas para o desenvolvimento de software e as demais organizações (50%) a utilizam eventualmente. A Aplicação das ferramentas nas fases do Ciclo de Vida, ocorre predominantemente na fase de Projeto (83%), nas demais fases, Análise (50%), Manutenção (33%) e Implementação (16%) a aplicação ocorre em menor percentual. Quanto a utilização das mesmas ferramentas por todos os funcionários constatou-se que na maioria das empresas são utilizadas sistematicamente ou eventualmente pelos mesmos funcionários. No caso de duas organizações, o uso da ferramenta varia conforme o funcionário. A maioria das organizações (66%) a utilização de ferramentas não varia conforme o projeto e em 44% dos casos as ferramentas variam eventualmente conforme o projeto. Como pode ser observado nos dados, a utilização de ferramentas de apoio ao desenvolvimento concentra-se a etapa de projeto. Informalmente, constatou-se que, na maioria dos casos, não são utilizados todos os recursos disponibilizados pelas ferramentas e seu uso restringe-se à documentação das tarefas executadas. Outro dado informal constatado nas empresas refere-se ao meio como as ferramentas foram incorporadas ao processo de desenvolvimento. Na maioria dos casos, as ferramentas são incorporadas por iniciativa individual e não por iniciativa da empresa, ou seja, normalmente não existe uma política definida pela empresa quanto a pesquisa e seleção de ferramentas apropriadas ao desenvolvimento de software.

Tabela 3: Formas de utilização de Ferramentas no Ciclo de Vida

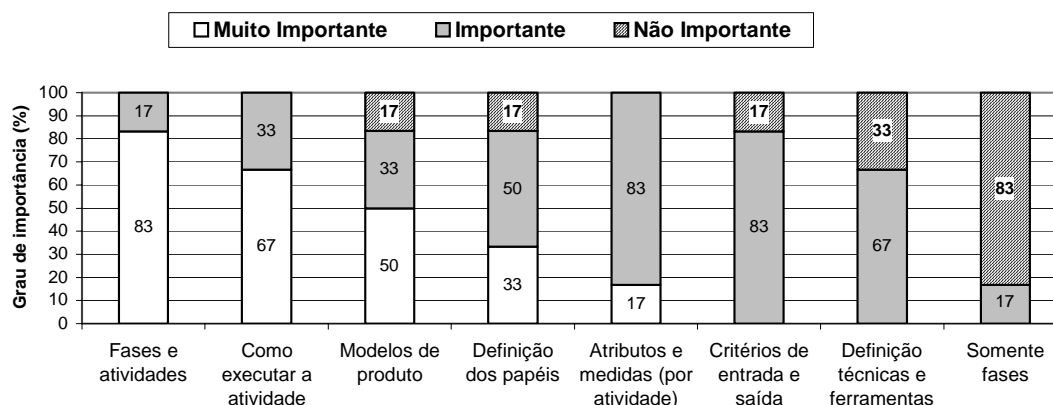
Questões	Opções	%
Utilização das Ferramentas	Sistematicamente	50
	Eventualmente	50
	Não utiliza	0
Fases em que a Ferramenta é utilizada	Análise	50
	Projeto	83
	Implementação	17
Padronização das ferramentas	Manutenção	33
	Sistematicamente	50
	Eventualmente	17
Variação das ferramentas conforme o projeto	Variam	33
	Não variam	67
	Eventualmente variam	33
	Variam	0

2.6 Requisitos para modelos de processo

O Gráfico 8 apresenta, segundo as empresas, o grau de importância dos requisitos para modelos de processos. O grau de importância constitui-se de Muito Importante, Importante e Não Importante. Observou-se que, em relação a modelagem de fases e atividades, 83% das empresas consideraram muito importante, 17% consideraram importante e, nenhuma empresa considera este aspecto não importante. Quanto ao requisito descrição de como executar a atividade: 67% consideraram muito importante, 33% importante e nenhuma considerou não importante. Em relação ao requisito modelo de produtos a serem desenvolvidos, 50% das empresas consideraram muito importante, 33% importante e 17% não importante. Quanto ao requisito definição de papéis e responsabilidade, 33% consideraram muito importante, 50% importante e 17% não importante. O requisito atributo e medidas por atividade obtiveram 17% muito importante, 83% importante e nenhuma empresa considerou esse requisito não importante. Em relação aos critérios de entrada e saída, nenhuma empresa considerou muito importante, 83% consideraram importante e 17% não importante. Quanto a definição de técnicas e ferramentas a serem utilizadas, nenhuma empresa considerou muito importante, 67% considerou importante e 33% avaliaram como não importante. Finalmente, em relação ao requisito modelagem dos processos considerando apenas

fases, nenhuma empresa considerou muito importante, 17% consideram importante e 83% consideram esse requisito não importante.

Gráfico 8: Requisitos para modelos de processo (segundo as empresas).



Com exceção do requisito somente fases o qual obteve o maior grau de não importante (83%), os demais (sete primeiros requisitos) atingiram índices acima de 66% considerando os graus de muito importante e importante demonstrando que as empresas entendem que tais requisitos devem ser considerados em um modelo de processo de software. No entanto, como pode ser observado, é perceptível que alguns requisitos (entre os sete primeiros requisitos) são mais priorizados que outros. Relevando esse aspecto, o presente trabalho entende que esses requisitos poderiam ser incorporados ao modelo de processo das empresas de forma gradual, por exemplo em três etapas: curto, médio e longo prazo ou por intermédio de estágios (semelhante a estratégia utilizada pelo Modelo CMM).

Dentre as etapas genéricas do Ciclo de Vida (análise, projeto, implementação, teste e manutenção) todas as empresas consideraram a fase de manutenção como o processo no qual a empresa encontra maior dificuldade. A dificuldade na manutenção normalmente ocorre pelo fato das atividades iniciais do desenvolvimento de software não serem adequadamente executadas, ou seja, as empresas concentram seu esforço na implementação do software. Outro aspecto que pode estar relacionado à manutenção se refere ao fato das empresas adaptarem seu

produto para clientes diferentes como relatado em conversas informais.

Além disso, informalmente, foi possível constatar que geralmente as empresas iniciam suas atividades desenvolvendo diversos produtos (domínios distintos) e à medida que um produto tem maior aceitação no mercado, essas empresas concentram seu desenvolvimento nesse produto. A diversidade inicial de produto ocorre pela necessidade das empresas formarem uma carteira de clientes que sustentem a empresa financeiramente em um primeiro momento. No entanto, à medida que o número de produtos (domínios distintos) aumenta, torna-se mais difícil o gerenciamento dessa diversidade o que força as empresas a dirigirem sua atuação para um domínio específico.

Então, a diversidade inicial de domínios distintos aos quais essas empresas se submeteram, pode ter sido o motivo inicial que impediu a definição dos processos nessas empresas, ou seja, a diversidade de domínios exige muito esforço e cada domínio distinto pode exigir modelos de processos de software distintos o que exigiria mais esforço da empresa. Assim, mesmo após a definição de um domínio, é difícil para a empresa mudar a cultura estabelecida.

3 CONSIDERAÇÕES SOBRE ASPECTOS METODOLÓGICOS DA PESQUISA

Diversos aspectos devem ser considerados com relação a presente pesquisa. Inicialmente, deve-se salientar que o universo de empresas pesquisadas (seis no total), estatisticamente, não permite que sejam realizadas inferências com o objetivo de atribuir, a todas as empresas que se enquadrem no perfil de microempresas, os mesmos fenômenos observados. Portanto, qualquer generalização deve ser cuidadosamente analisada com o objetivo de não atribuir a empresas não participantes o mesmo resultado apresentado nessa pesquisa. No entanto, diversos dados coletados corroboram com dados coletados na pesquisa realizada pelo MCT/SEPIN (MCT/SEPIN, 2002).

Diversas variáveis podem interferir e provocar variações que poderiam conduzir a resultados bem diferentes aos apresentados aqui. No Quadro 1 do presente anexo, diversos fatores de variação foram levantados e podem ser considerados como referência para possíveis distorções. Assim, algumas considerações, sem o objetivo de cercar todas as possíveis possibilidades, serão tecidas com o objetivo de alertar, caso a presente pesquisa seja utilizada como referência para outros trabalhos relacionados, e até mesmo consideradas nos resultados do presente trabalho.

O tempo de atuação da empresa no desenvolvimento de software pode conduzir a resultados diferenciados. Empresas de desenvolvimento de software que atuam no mercado a mais tempo tendem, por exemplo, a utilizar métodos, técnicas e ferramentas em maior proporção se comparadas a empresas que atuam a menor tempo no mercado. A localização geográfica das empresas pode ser outro fator que poderia interferir nos resultados e provocar diversas leituras. Por exemplo, em determinadas regiões há mais oferta de mão-de-obra qualificada e, por conseguinte, as práticas de Engenharia de Software podem ser empregadas em maior ou menor número conforme a qualificação do quadro de empregados. Sob o ponto de vista da localização geográfica, a pesquisa do MCT/SEPIN aponta, por exemplo, que em determinadas regiões

programas de qualidade total, sistemas da qualidade ou similares são mais utilizados do que em outras regiões (MCT/SEPIN, 2002). Esse fator de variação pode aumentar, por exemplo, o percentual de microempresas que definem total ou parcialmente, seus processos de software.

A quantidade de produtos distintos (de diferentes tipos e domínios) mantidos por uma organização também pode interferir nos resultados. Teoricamente, empresas que possuem maior número de produtos tendem a ter dificuldades no estabelecimento de seus processos de software pelo fato de diferentes produtos de software poderem exigir variação nos processos de software e, conseqüentemente, poderiam tornar a definição dos processos uma tarefa mais complexa ainda, de alto custo e esforço. Esse contexto poderia contribuir com o aumento do índice de processos indefinidos.

O alto índice de rotatividade do quadro de empregados em uma empresa também poderia propagar diversas variações na leitura dos dados. Empresas que possuem alto índice de rotatividade do quadro de empregados estão propensas a maiores dificuldades no estabelecimento de seus processos de software. Portanto, os dados apresentados nesta pesquisa apontam tendências relacionadas a situação da modelagem de processos em microempresas. O contexto da aplicação é restrito e estatisticamente insignificante. Alguns indicadores apresentados nessa pesquisa convergem com indicadores semelhantes apresentados na pesquisa do MCT/SEPIN 2002. A intenção da pesquisa foi coletar dados sobre indicadores que não foram contemplados na pesquisa do MCT/SEPIN com o objetivo de subsidiar a proposta apresentada no presente trabalho.

4 DISCUSSÃO E CONSIDERAÇÕES FINAIS

Os indicadores da situação atual de microempresas em relação à modelagem de processos de software apresentados nessa pesquisa mostram que existe a necessidade de se definir políticas específicas para essa categoria de empresas. A informalidade dos processos de software está presente em todas as empresas pesquisadas e, quando há a formalização de alguns elementos do processo, essa ocorre para atender a situações críticas e não como uma prática sistemática.

Praticamente todo o esforço de trabalho das organizações é direcionado para atividades do processo de desenvolvimento e/ou manutenção e, atividades importantes como definição dos processos de software, gerência, programas de mensuração, programas de qualidade, políticas preventivas de prevenção de defeitos entre outras, normalmente são relegadas para segundo plano. No entanto, existe entre as microempresas pesquisadas o consenso de que esse contexto é extremamente prejudicial e que medidas devem ser tomadas para reverter essa situação. A predisposição das empresas se mostrou pela aceitação imediata em participar de projetos que as auxiliem na implantação de programas da qualidade e produtividade.

No entanto, projetos que visem melhorar a qualidade de processos e produtos em microempresas devem considerar diversas restrições. Existe, entre outras restrições, limitação orçamentária, pouca disponibilidade de força de trabalho, inexperiência do quadro de empregados em atividades relacionadas à qualidade. Essas restrições dificultam, por exemplo, destinar tempo dos empregados para estudos e avaliações de novos métodos, técnicas e ferramentas ou, ainda, adquirir tecnologias de Engenharia de Software que poderiam contribuir à melhoria da qualidade e produtividade dos processos de software devido ao custo ou ainda que qualquer iniciativa dessas empresas voltada para qualidade e produtividade seja dificultada pela falta de experiência dos empregados em determinar o que, como, quem e quando fazer.

Portanto, é necessário que políticas da qualidade e produtividade apropriadas para microempresas sejam pesquisadas e desenvolvidas. Iniciativas poderiam iniciar com parcerias entre microempresas e instituições de ensino, aliás, algumas parcerias nesses moldes já ocorrem com sucesso, por exemplo o projeto Geness da Universidade Federal de Santa Catarina e tem resultado em inúmeros benefícios às microempresas participantes.

5 QUESTIONÁRIO MODELO

Entrevistado: _____

Cargo do entrevistado: _____

Data da coleta: ____/____/____

Caracterização da empresa (Fatores de Variação)

1. Cidade de atuação:

- Curitiba/Pr
- Londrina/Pr
- Florianópolis/SC
- Outra:

2. Data de fundação da empresa: ____/____/____

3. Quantidade de funcionários:

- 01 - 09 (micro)
- 0 - 50 (pequena)

4. Qual a quantidade de produtos de software desenvolvidos/mantidos atualmente pela empresa ? _____ (* Caso a resposta seja igual a 1 responder a questão 5)

5. Quanto tempo de experiência a empresa possui no domínio da aplicação?

- Menos de um ano
- 1 – 2 anos
- 2 – 3 anos
- mais de 3 anos

6. Atividade da empresa no tratamento de software (tipo de software):

- | | |
|---|--|
| <input type="checkbox"/> Software de pacote | <input type="checkbox"/> Software para uso próprio |
| <input type="checkbox"/> Sob encomenda | <input type="checkbox"/> Editoração de software de terceiros |
| <input type="checkbox"/> Software embarcado | <input type="checkbox"/> Outro: _____ |
| <input type="checkbox"/> Software educacional | <input type="checkbox"/> |
| <input type="checkbox"/> Software para Internet | |

7. Setor de atividade da empresa (mercado de atuação) :

- | | |
|--|--|
| <input type="checkbox"/> Financeiro | <input type="checkbox"/> Gestão integrada – ERP |
| <input type="checkbox"/> Administração geral | <input type="checkbox"/> Administração pública |
| <input type="checkbox"/> Automação comercial | <input type="checkbox"/> Administração de serviços |
| <input type="checkbox"/> Contabilidade | <input type="checkbox"/> Automação de escritórios |
| <input type="checkbox"/> Administração de RH | <input type="checkbox"/> Automação industrial |
| <input type="checkbox"/> Páginas Web | |

Outro: _____**8. Tempo médio de desenvolvimento dos projetos da empresa:**

- 1 – 4 meses
- 5 – 8 meses
- 9 – 12 meses
- 13 – 18 meses
- 19 – 24 meses
- Acima de 24 meses

**9. Número habitual de pessoas envolvidas em um projeto de software: _____
pessoas****10. Qual a distribuição da força de trabalho no projeto de software**

- Análise
- Projeto
- Implementação
- Manutenção

11. Os projetos de software são mantidos (acompanhados) pelas mesmas pessoas que iniciaram o projeto?

- Sistemáticamente
- Eventualmente
- Os projetos são mantidos por pessoas diferentes

12. Tempo de atuação da empresa no mercado de software:

- 1 ano
- 2 anos
- 3 a 5 anos
- 6 a 10 anos
- acima de 10 anos

13. Experiência profissional dos funcionários (média em número de funcionários):

- ____ até 1 ano
____ 1 – 3 anos
____ 3 – 5 anos
____ acima de 5 anos

14. A política de capacitação profissional atinge:

- Todos os funcionários
 Parte dos funcionários. Quais: _____
 Não possui (ir para questão 16)

15. O investimento em capacitação profissional é:

- Sistemático
 Eventual
 Sob demanda

16. Quantidade de profissionais quanto a sua qualificação (em número de funcionários qualificados na área de informática):

- ____ sem formação
____ curso técnico
____ graduado
____ especialista
____ mestre
____ doutor

17. Composição do quadro funcional (em número de funcionários):

- ____ Estagiários
____ Contratados
____ Terceirizados/Parceiros
____ Sócios – proprietários

18. A rotatividade do quadro funcional pode ser enquadrada como:

- Inexistente
 Baixa
 Alta

Processos de software (Fatores de Qualidade)

19. O modelo de ciclo de vida utilizado foi:

- definido pela própria empresa (**Responder questão 21**)
- a aplicação integral de um modelo existente
- adaptado a partir de um modelo existente

20. Qual o ciclo de vida é utilizado/adaptado?

- Cascata
- Espiral
- Prototipação
- Outro: _____

21. O Modelo de Ciclo de Vida é formalizado

- Totalmente
- Parcialmente
- Não é documentado (**responder questão 32**)

22. O ciclo de vida adotado varia em diferentes projetos

- Frequentemente
- Eventualmente
- Não varia

23. O ciclo de vida é documentado:

- Em todos os projetos
- Na maioria dos projetos
- Em poucos projetos
- Não é documentando, porque não?

24. Qual o meio utilizado na documentação do ciclo de vida

- Editor de texto ou similar
- Em papel
- Via ferramenta Case, qual:
- Usado pelo um sistema workflow
- Outro: _____

25. As fases são formalizadas

- Todas
- Algumas. Quais: () Análise () Projeto () Implementação () Manutenção
- Nenhuma

26. As atividades são formalizadas

- Todas
- Algumas
- Nenhuma

27. Os critérios de entrada e saídas são formalizados

- Todos
- Alguns.
- Nenhum

28. Os papéis e responsabilidades são definidos

- Todos
- Alguns
- Nenhum

29. Os métodos são formalizados

- Todos
- Alguns. Quais.
- Nenhum

30. Técnicas são formalizadas

- Todas

31. E empresa coleta dados sobre o processo de desenvolvimento de software?

- Algumas
- Nenhuma
- Sistemáticamente
- Eventualmente
- Não coleta

32. Quais são as principais dificuldades na implantação/manutenção do ciclo de vida?

- os modelos existentes não apresentam-se adequados ao perfil da empresa.
- custos
- tempo
- exige investimentos em ferramentas de apoio.
- exige investimentos em treinamento de funcionários.
- não são claros os benefícios da formalização de um ciclo de vida.
- não possui pessoal qualificado para formalizar o ciclo de vida.
- os modelos não são flexíveis para aplicação em domínios distintos (o processo de desenvolvimento da empresa não é constante).
- Outros: _____

33. A empresa utiliza ferramentas?

- Sistemáticamente
- Eventualmente
- Não utiliza

34. Em que fases do ciclo de vida as ferramentas são utilizadas?

- Análise
- Projeto
- Implementação
- Manutenção

35. As mesmas ferramentas são adotadas por todos os funcionários?

- Sistemáticamente
- Eventualmente
- As ferramentas variam conforme o funcionário

36. As ferramentas variam conforme o projeto?

- Não variam
- Eventualmente variam
- Variam

37. A empresa adota alguma forma de melhoria de qualidade?

- Sistemática
- Esporádico
- Já adotou
- Não (responder questão 39)

38. Qual modelo de melhoria a empresa utiliza/utilizou?

Modelo	Utiliza	Utilizou
<input type="checkbox"/> ISO 9000.		
<input type="checkbox"/> CMM.		
<input type="checkbox"/> Gestão de Qualidade Total.		
<input type="checkbox"/> SPICE.		
<input type="checkbox"/> Outro:		

39. A empresa investe em técnicas e ferramentas de melhoria de qualidade?

- Não.
 Pretende. Quando (ano): _____ Quanto: _____ % faturamento (aprox.).
 Já investe. Desde (ano): _____ Quanto: _____ % faturamento (aprox.).

40. Geralmente, na aprovação do projeto pelo cliente, o termo qualidade:

- não é mencionado.
 é mencionado esporadicamente.
 é mencionado regularmente.

41. Independente de ter ou não modelo de processo definido. Na modelagem explícita do processo, qual é a maior necessidade percebida atualmente na empresa em relação a melhoria da modelagem de um processo.

42. Quais elementos seriam mais necessários para serem incluídos num modelo de processo. Classifique:

- 1 – muito importante
 2 – importante
 3 – não importante

- Só fases
 fases e atividades
 modelos dos produtos a serem desenvolvidos
 critérios de entrada e saída
 definição dos papéis e responsabilidades
 atributos e medidas para cada atividade
 descrição de como executar a atividade
 definição das técnicas e ferramentas que devem ser utilizadas

43. Em qual(is) processo(s) a empresa tem maior dificuldade?

ANEXO B: INSTRUMENTOS DE COLETA DE DADOS

Instrumento 1: Caracterização dos produtos.

	Produtos	Produto 1	Produto 2	Produto...
Identificação do produto				
1. Descrição do sistema				
2. Data da 1 ^o versão				
3. N ^o . da versão atual				
4. N ^o . de clientes				
5. N ^o . de módulos				
6. N ^o . de arquivos (BD)				
Caracterização do processo		Sim/Não		
7. Está relacionado aos outros produtos da empresa? Por exemplo, utiliza a mesma ou parte da base de dados, a mesma interface gráfica etc.				
8. Os agentes responsáveis pela manutenção do produto são mantidos? Por exemplo: qualquer agente pode alterar o produto ou existe um agente responsável?				
9. Há diferenças na quantidade de artefatos consumidos/produzidos em relação aos outros produtos da empresa?				
10. Há diferenças no formato dos artefatos gerados em relação aos outros produtos da empresa?				
11. Há diferenças na quantidade de atividades executadas em relação aos outros produtos?				
12. Há diferenças nas características das atividades executadas em relação a outros produtos? Por exemplo: procedimentos adotados.				
13. Há diferenças na quantidade de ferramentas de desenvolvimento utilizadas em relação aos demais produtos?				
14. Há diferença nas atribuições e responsabilidade dos mesmos papéis em relação aos outros produtos da empresa?				
15. Há diferenças em quantidade de papéis necessários para manter o produto em relação aos outros produtos da empresa?				

Instrumento 2: Exemplo de instrumento de coleta de dados de agente x produto.

Nome do produto selecionado:	
Nome(s) do(s) agente(s) envolvido(s)	Envolvimento com o produto
1.	[] integral [] parcial
2.	[] integral [] parcial
...	[] integral [] parcial
Instrução: identificar os agentes e seu envolvimento no processo de software de cada produto da empresa.	

Instrumento 3: Atividades executadas pelos agentes envolvidos no processo de software do produto.

Nome do agente:	
Produto:	
Atividades de alto nível	
1.	
2.	
...	
Instrução: Para cada agente, independente de seu envolvimento no processo de software (integral ou parcial), identificar as atividades (alto nível) executadas ao manter o produto de software.	

Instrumento 4: Fluxo de execução do ciclo de vida executado por um determinado agente ao manter um determinado produto.

Nome do agente:	
Produto:	
Representação do ciclo de vida executado pelo agente	
<p>Exemplo</p> <pre> graph LR A((Engenharia de requisitos)) --> B((Projeto)) B --> C((Implementação)) C --> D((Teste)) D --> E((Instalação)) E --> F((Treinamento)) F --> A </pre>	
Instrução: Representar graficamente a seqüência entre as atividades executadas pelo agente, ou seja, descrever o ciclo de vida executado pelo agente para cada produto. Utilizar a notação definida na fase de Definição de Padrões.	

Instrumento 5: Relação entre atividades executadas ao manter cada produto da empresa.

		Produtos ►	Produto 1	Produto 2	Produto...
Atividades ▼	Freqüência ►		S/E/N	S/E/N	S/E/N
1.					
2.					
...					
Legenda:	S: Sistemático. A atividade ocorre independente de qualquer situação.				
	E: Eventual. A atividade pode ocorrer ou não, dependendo da situação.				
	N: Não Ocorre. A atividade não ocorre no ciclo de vida desse produto.				
	Obs: Somente uma opção deve ser assinalada para cada ocorrência da atividade do produto, ou seja, ou "S", ou "E" ou "N".				
Instrução: utilizar os dados levantados no Erro! Resultado não válido para índice. na elaboração do presente instrumento.					

Instrumento 6: Relação entre as atividades e agentes participantes no processo do produto de software da empresa.

Produto:				
	Agente ►	Agente 1	Agente 2	Agente...
Atividade ▼				
1.				
2.				
...				
Legenda:	S: Sistemático. A atividade é executada independente de qualquer situação.			
	E: Eventual. A atividade pode ou não ser executada, dependendo da situação.			
	N: Não Ocorre. A atividade não é executada pelo agente no ciclo de vida desse produto.			
	Obs: Somente uma opção deve ser assinalada para cada ocorrência da atividade do produto, ou seja, ou “S”, ou “E” ou “N”.			
Instrução: utilizar os dados levantados no Erro! Resultado não válido para índice. na elaboração do presente instrumento.				

Instrumento 7: Coleta de dados de atividades.

FONTE: AGENTE FULANO		HORA INÍCIO: ____:____	
DATA COLETA: DD/MM/AAAA		HORA TÉRMINO: ____:____	
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)			
Identificador:			
Hierarquia:			
Descrição:			
Classificação:	<input type="checkbox"/> Atividade	<input type="checkbox"/> Sub-Atividade	
Tipo:	<input type="checkbox"/> Atividade Executável	<input type="checkbox"/> Atividade Não executável	
Procedimento(s):			
Sub-atividades:			
Meta(s):			
Propriedades Estáticas (Perspectiva Funcional e Organizacional)			
	Artefato	Finalidade	Origem
Artefatos entrada:			
	Artefato	Finalidade	Destino
Artefatos de saída:			
Papéis:			
Fluxo de entrada entre atividades:			
Fluxo de saída entre atividades:			
Ferramentas:			
Propriedades Dinâmicas (Perspectiva Comportamental)			
Critérios de entrada:			
Critérios de saída:			
Estados possíveis da atividade:			

Instrumento 8: Coleta de dados de artefatos.

FONTE: AGENTE FULANO DATA COLETA: DD/MM/AAAA		HORA INÍCIO: ____:____ HORA TÉRMINO: ____:____	
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)			
Identificador:			
Descrição:			
Caminho:			
Modelo:			
Tipo:	<input type="checkbox"/> Arquivo-Fonte	<input type="checkbox"/> Componente	<input type="checkbox"/> Documentação
Propriedades Estáticas (Perspectiva Funcional e Organizacional)			
Sub-Artefatos:			
Atividades produtoras:			
Atividades consumidoras:			
Atividades modificadoras:			
Ferramentas produtoras:			
Propriedades Dinâmicas (Perspectiva Comportamental)			
Estados:			

Instrumento 9: Coleta de dados de ferramentas.

FONTE: AGENTE FULANO DATA COLETA: DD/MM/AAAA		HORA INÍCIO: ____:____ HORA TÉRMINO: ____:____	
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)			
Identificador:			
Descrição:			
Tipo:	<input type="checkbox"/> <i>Implementação</i>	<input checked="" type="checkbox"/> <i>Documentação</i>	
Propriedades Estáticas (Perspectiva Funcional e Organizacional)			
Artefatos gerados:			
Utilizada nas Atividades:			
Propriedades Dinâmicas (Perspectiva Comportamental)			
CrITÉrios para uso:			

Instrumento 10: Coleta de dados de papéis.

FONTE: AGENTE FULANO		HORA INÍCIO: ____:____	
DATA COLETA: DD/MM/AAAA		HORA TÉRMINO: ____:____	
PROPRIEDADES DESCRITIVAS (“PERSPECTIVA INFORMACIONAL”)			
Identificador:	<i>Programador.</i>		
Descrição:	<i>Papel responsável pela manipulação de ambientes de desenvolvimento de software e edição de artefatos relacionados aos produtos da empresa</i>		
Hierarquia:	<i>Subordinado diretamente ao gerente de projeto</i>		
Atribuições:	<ol style="list-style-type: none"> 1. Implementar arquivos-fonte de acordo com o documento de requisitos; 2. Modificar arquivos-fonte de acordo com o documento de requisitos; 3. Aplicar normas e padrões relacionados a cada atividade executada; 4. Documentar as atividades realizadas de acordo com os padrões. 		
Qualificações:	<ol style="list-style-type: none"> 1. Lógica de programação; 2. Domínio do ambiente de programação Delphi 7; 3. Domínio do banco de dados Oracle 8i; 4. Domínio dos sistemas operacionais Windows e Linux; 5. Domínio do gerador de instalador Install Shield; 6. Domínio do gerador de ajuda contextual Help & Manual. 		
Propriedades Estáticas (Perspectiva Funcional e Organizacional)			
Artefatos manipulados:	<ol style="list-style-type: none"> 1. Código-Fonte; 2. Projeto de Dados; 3. Documento de Requisitos. 		
Executa as Atividades:	<ol style="list-style-type: none"> 1. Implementação. 		
Utiliza as ferramentas:	<ol style="list-style-type: none"> 1. Delphi 7; 2. SGBD Oracle 8i; 3. Help & Manual; 4. Microsoft WinWord; 4. Install Shield. 		

Instrumento 1: Exemplo de instrumento de coleta de dados para relacionamentos.

DATA: DD/MM/AAAA					HORA INÍCIO: ____:____											
FONTE: _____					HORA TÉRMINO: ____:____											
ATIVIDADE: RELACIONAMENTOS																
	Atividade	Atividade 1	Atividade 2	Atividade...	Artefatos	Artefato 1	Artefato 2	Artefato...	Papéis	Papel 1	Papel 2	Papel...	Ferramentas	Ferramenta 1	Ferramenta 2	Ferramenta 3
Atividade 1			CT	CT		PD	PD	CS		US		US		US		
Atividade 2				SA		CS	MD				US	US		US	US	US
Atividade...			EN					PD		US						US
CT: Contém EN: Entrada SA: Saída CS: Consome MD: Modifica PD: Produz US: Usa																
ARTEFATO: RELACIONAMENTOS																
	Atividade	Atividade 1	Atividade 2	Atividade...	Artefatos	Artefato 1	Artefato 2	Artefato...	Papéis	Papel 1	Papel 2	Papel...	Ferramentas	Ferramenta 1	Ferramenta 2	Ferramenta 3
Artefato 1																
Artefato 2																
Artefato...																
CT: Contém EN: Entrada SA: Saída CS: Consome MD: Modifica PD: Produz US: Usa																
PAPEL: RELACIONAMENTOS																
	Atividade	Atividade 1	Atividade 2	Atividade...	Artefatos	Artefato 1	Artefato 2	Artefato...	Papéis	Papel 1	Papel 2	Papel...	Ferramentas	Ferramenta 1	Ferramenta 2	Ferramenta 3
Papel 1																
Papel 2																
Papel...																
CT: Contém EN: Entrada SA: Saída CS: Consome MD: Modifica PD: Produz US: Usa																
FERRAMENTA: RELACIONAMENTOS																
	Atividade	Atividade 1	Atividade 2	Atividade...	Artefatos	Artefato 1	Artefato 2	Artefato...	Papéis	Papel 1	Papel 2	Papel...	Ferramentas	Ferramenta 1	Ferramenta 2	Ferramenta 3
Ferramenta 1																
Ferramenta 2																
Ferramenta...																
CT: Contém EN: Entrada SA: Saída CS: Consome MD: Modifica PD: Produz US: Usa																

ANEXO C: AUTORIZAÇÃO PARA DIVULGAÇÃO DOS DADOS DA PESQUISA

Londrina, 01 de setembro de 2006.

AUTORIZAÇÃO PARA DIVULGAÇÃO DOS DADOS DA PESQUISA

Eu, Edson dos Santos Cordeiro, RG 5.195.531-5 SSP-PR, declaro, para os devidos fins, que a empresa na qual o estudo de caso relativo à dissertação de mestrado intitulada "Modelagem descritiva iterativa e incremental de processo de software: uma experiência em uma microempresa de desenvolvimento de software", autorizou-me a divulgar os dados relativos a essa pesquisa em eventos e publicações científicas desde que seja mantido em anonimato a identificação dessa instituição.

Londrina - Pr, 01 de setembro de 2006.



Edson dos Santos Cordeiro