Processos, Qualidade e Gestão de Software

v3 23 nov 2009

Indice

8.2 O RUP E SUAS CARACTERÍSTICAS ERROR! BOOKMARK NOT DEFINED.

8.3 VISÃO GERAL DO RUP ERROR! BOOKMARK NOT DEFINED.

8.2.1 CONCEPÇÃO ERROR! BOOKMARK NOT DEFINED.

8.2.2 ELABORAÇÃO ERROR! BOOKMARK NOT DEFINED.

8.2.3 CONSTRUÇÃO ERROR! BOOKMARK NOT DEFINED.

8.2.4 TRANSIÇÃO ERROR! BOOKMARK NOT DEFINED.

INTRODUÇÃO AO RUP:RATIONAL UNIFIED PROCESS-" PHILLIPPE KRUCHTEN" ERROR! BOOKMARK NOT DEFINED.

3.1. INTRODUÇÃO A PROCESSOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE 102

3.2. O MANIFESTO ÁGIL 103

3.3. PRINCIPAIS PROCESSOS ÁGEIS 104

3.4 EXTREME PROGRAMMING (XP) 106

3.4.1 VALORES, PRINCÍPIOS E PRÁTICAS DE XP 106
3.4.2 PAPÉIS DOS INTEGRANTES 108
3.4.3 CICLO DE VIDA 109

3.6. SCRUM 109

3.6.1 CARACTERÍSTICAS DO SCRUM 110
3.6.2 PAPÉIS DO SCRUM 110
3.6.3 PRÁTICAS DO SCRUM 111
3.6.4 CICLO DE VIDA DO SCRUM 111

3.7 FEATURE DRIVEN DEVELOPMENT 111

3.7.1 CARACTERÍSTICAS DO FDD 112
3.5.2 PAPÉIS DO FDD 112
3.5.3 PRÁTICAS DO FDD 114

9.7. CONSIDERAÇÕES FINAIS **117**

9.8. TÓPICOS DE PESQUISA **118**

9.9. SUGESTÕES DE LEITURA **118**

9.11. EXERCÍCIOS **118**

REFERÊNCIAS **118**

3.1 INTRODUÇÃO **121**

3.2 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE **121**

3.2.1 MOTIVAÇÕES PARA O DDS **122**
3.2.1 NÍVEIS DE DISPERSÃO **123**
3.2.2 MODELOS DE NEGÓCIO **124**
3.2.3 DESAFIOS **125**

- **ARQUITETURA DO *SOFTWARE*:** É UM DOS FATORES MAIS UTILIZADOS PARA A DIMINUIÇÃO DO ESFORÇO ENTRE AS EQUIPES. CONFORME KAROLAK [1998], UMA ARQUITETURA APROPRIADA PARA O DDS DEVE SE BASEAR NO PRINCÍPIO DA MODULARIDADE, POIS PERMITE ALOCAR TAREFAS COMPLEXAS DE FORMA DISTRIBUÍDA. COM ISSO HÁ UMA REDUÇÃO NA COMPLEXIDADE E É PERMITIDO UM DESENVOLVIMENTO EM PARALELO SIMPLIFICADO. **126**

- **ENGENHARIA DE REQUISITOS:** A ENGENHARIA DE REQUISITOS CONTÉM DIVERSAS TAREFAS QUE NECESSITAM DE ALTO NÍVEL DE COMUNICAÇÃO E COORDENAÇÃO. COM ISSO OS PROBLEMAS APRESENTADOS SÃO MAIS COMPLEXOS EM UM CONTEXTO DE DDS. **126**

- **GERÊNCIA DE CONFIGURAÇÃO:** O GERENCIAMENTO DE CONFIGURAÇÃO (CM) É A CHAVE PARA CONTROLAR AS MÚLTIPLAS PEÇAS EM UM PROJETO DISTRIBUÍDO. CONTROLAR MODIFICAÇÕES NOS ARTEFATOS EM CADA UMA DAS LOCALIDADES COM O PROCESSO DE DESENVOLVIMENTO DE TODO PRODUTO PODE SER COMPLEXO. APESAR DA UTILIZAÇÃO DE PRÁTICAS DE CM AUXILIAR NO CONTROLE DA DOCUMENTAÇÃO E DO SOFTWARE, A GERÊNCIA DE MODIFICAÇÕES SIMULTÂNEAS A PARTIR DE LOCAIS DIFERENTES É UM GRANDE DESAFIO. ALÉM DISSO, O USO DE FERRAMENTAS DE CM COMPARTILHADAS POR DUAS OU MAIS EQUIPES DE FORMA INADEQUADA GERA DIVERSOS RISCOS E PROBLEMAS EM PROJETOS DDS. **126**

- **PROCESSO DE DESENVOLVIMENTO:** EM PROJETOS DDS, O USO DE UMA METODOLOGIA QUE AUXILIA A SINCRONIZAÇÃO DAS ATIVIDADES É ESSENCIAL. COM ISSO TODOS OS MEMBROS UTILIZAM UMA NOMENCLATURA COMUM EM SUAS ATIVIDADES. **126**

3.3 PROCESSOS PARA DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE 126

SEGUNDO PRESSMAN [2001], PROCESSO É A CAMADA MAIS IMPORTANTE DA ENGENHARIA DE SOFTWARE, FAZENDO UMA LIGAÇÃO ENTRE FERRAMENTAS E MÉTODOS. UM PROCESSO DE SOFTWARE É UM CONJUNTO DE ATIVIDADES QUE DEFINEM A SEQÜÊNCIA EM QUE OS MÉTODOS SERÃO APLICADOS E COMO O PRODUTO SERÁ ENTREGUE, ALÉM DISSO, DEFINE OS CONTROLES QUE ASSEGURAM A QUALIDADE DO PRODUTO E COORDENAÇÃO DAS MUDANÇAS. **126**

EM UM AMBIENTE DE DESENVOLVIMENTO DISTRIBUÍDO, UM PROCESSO DE DESENVOLVIMENTO COMUM À EQUIPE É FUNDAMENTAL, TENDO EM VISTA QUE UMA METODOLOGIA AUXILIA DIRETAMENTE NA SINCRONIZAÇÃO, FORNECENDO AOS MEMBROS DA EQUIPE UMA NOMENCLATURA COMUM DE TAREFAS E ATIVIDADES, E UM CONJUNTO COMUM DE EXPECTATIVAS AOS ELEMENTOS ENVOLVIDOS NO PROCESSO [PRIKLADNICKI 2008]. **126**

A ENGENHARIA DE SOFTWARE (ES) SEMPRE ESTÁ APRESENTANDO GRANDES AVANÇOS E TRANSFORMAÇÕES RELACIONADAS ÀS TÉCNICAS, MODELOS E METODOLOGIAS. ESSES AVANÇOS SÃO DESTACADOS QUANDO SE TRABALHA COM PROCESSO DE DESENVOLVIMENTO DISTRIBUÍDO DE *SOFTWARE* (DDS), HAVENDO UMA NECESSIDADE DO USO DE PRÁTICAS QUE DÊ SUPORTE ÀS DIFICULDADES ENCONTRADAS NAS DEFINIÇÕES DE REQUISITOS QUE MUDAM DE FORMA DINÂMICA NO DECORRER DO TEMPO. ESTUDOS RELACIONADOS A PROCESSO PARA DDS AINDA É ESCASSO, SENDO ASSIM ESTE CAPÍTULO RELATA O USO DE PRATICAS DO DESENVOLVIMENTO TRADICIONAL QUE PODEM SER IMPLANTADAS EM UM AMBIENTE DISTRIBUÍDO E AS POSSÍVEIS ADAPTAÇÕES. **127**

3.4 PROCESSOS E ADAPTAÇÃO DAS PRÁTICAS EM PROJETOS DDS 127

A FORMA COMO UM PRODUTO DE *SOFTWARE* É CONCEBIDO, DESENVOLVIDO, TESTADO E ENTREGUE AO CLIENTE SOFRE GRANDE IMPACTO QUANDO O AMBIENTE DE DESENVOLVIMENTO É DISTRIBUÍDO [HERBSLEB 2001]. ASSIM, A ESTRUTURA NECESSÁRIA PARA O SUPORTE DESSE TIPO DE DESENVOLVIMENTO SE DIFERENCIA DA UTILIZADA EM AMBIENTES CENTRALIZADOS. DIFERENTES CARACTERÍSTICAS E TECNOLOGIAS SE FAZEM NECESSÁRIAS, CRESCENDO A IMPORTÂNCIA DE ALGUNS DETALHES ANTES NÃO PERCEBIDOS. **127**

3.4.1 MODELO DE KAROLAK [1998] 128

NESTE TRABALHO O AUTOR ABORDA O DDS SEGUINDO O CICLO DE VIDA TRADICIONAL DE UM PROJETO DE DESENVOLVIMENTO DE *SOFTWARE*. O AUTOR PROPÕE UM MODELO PARA DESENVOLVER PROJETOS DDS ABRANGENDO AS ATIVIDADES QUE DEVER OCORRER AO LONGO DO CICLO DE VIDA. A FIGURA ABAIXO ILUSTRA O MODELO PROPOSTO (FIGURA 3.2): **128**

128

FIGURA 3.2 – MODELO PARA PROJETOS DDS [KAROLAK 1998] **128**

3.4.2 USO DE PRÁTICAS ÁGEIS **130**
3.4.2.1 DXP – DISTRIBUTED EXTREME PROGRAMMING 131
3.4.2.2 ADOÇÃO DE *SCRUM* EM UM AMBIENTE DDS 133

3.5 OPORTUNIDADES DE PESQUISA **136**

3.6 CONSIDERAÇÕES FINAIS **137**

3.7 EXERCÍCIOS **137**

3.8 RECOMENDAÇÕES DE LEITURA **137**

REFERÊNCIAS **138**

HERBSLEB, J. D., MOITRA, D. "GLOBAL SOFTWARE DEVELOPMENT", IEEE SOFTWARE, MARCH/APRIL, EUA, 2001, P. 16-20. **138**

KAROLAK, D. W. "GLOBAL SOFTWARE DEVELOPMENT – MANAGING VIRTUAL TEAMS AND ENVIRONMENTS". LOS ALAMITOS, IEEE COMPUTER SOCIETY, EUA, 1998, 159P. **138**

KIEL, L. "EXPERIENCES IN DISTRIBUTED DEVELOPMENT: A CASE STUDY", IN: WORKSHOP ON GLOBAL SOFTWARE DEVELOPMENT AT ICSE, OREGON, EUA, 2003, 4P. **138**

KIRCHER, M., JAIN, P., LEVINE, A. "DISTRIBUTED EXTREME PROGRAMMING", IEEE, AGILE 2008. **138**

HERBSLEB, J.D., MOCKUS, A., FINHOLT, T.A. E GRINTER, R. E. "AN EMPIRICAL STUDY OF GLOBAL SOFTWARE DEVELOPMENT: DISTANCE AND SPEED", IN: ICSE 2001, TORONTO, CANADA. **138**

CARMEL, E. “GLOBAL SOFTWARE TEAMS – COLLABORATING ACROSS BORDERS AND TIME-ZONES” PRENTICE HALL, EUA, 1999, 269P. **138**

MARQUARDT, M. J., HORVATH, L. “GLOBAL TEAMS: HOW TOP MULTINATIONALS SPAN BOUNDARIESAND CULTURES WITH HIGH-SPEED TEAMWORK”. DAVIES-BLACK. PALO ALTO, EUA, 2001. **138**

YIN, ROBERT. “ESTUDO DE CASO: PLANEJAMENTO E MÉTODOS”. SP: BOOKMAN, 2001, 205P. **138**

YOUNG, C., TERASHIMA, H. “HOW DID WR ADAPT AGILE PROCESSES TO OUR DISTRIBUTED DEVELOPMENT?”, IEEE, AGILE 2008. **138**

PRIKLADNICKI, R., AUDY, J. L. N., EVARISTO, R. “GLOBAL SOFTWARE DEVELOPMENT IN PRACTICE: LESSONS LEARNED”, JOURNAL OF SOFTWARE PROCESS: PRACTICE AND IMPROVEMENT – SPECIAL ISSUE ON GLOBAL SOFTWARE DEVELOPMENT, 2004. **138**

PRIKLADNICKI, R. “MUNDDOS: UM MODELO DE REFERÊNCIA PARA DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE”. DISSERTAÇÃO DE MESTRADO, PPGCC – PUCRS, BRASIL, 2003.**138**

SOMMERVILLE, IAN. SOFTWARE ENGINNERING. 8.ED. [S.L] ADDISON WESLEY, 2007. **138**

J. L. N. PRIKLADINICKI, R.; AUDY. DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE. 2007. **138**

PERRELLI, HERMANO. VISÃO GERAL DO RUP. CENTRO DE INFORMÁTICA, UNIVERSIDADE FEDERAL DE PERNAMBUCO. DISPONÍVEL EM: HTTP://WWW.CIN.UFPE.BR/~IF717/SLIDES/3-VISAO-GERAL-DO-RUP.PDF. ACESSADO EM 20 MAIO 2009. **138**

PRESSMAN, ROGER S. SOFTWARE ENGINEERING: A PRACTITIONER’S APPROACH. EUA: MCGRAW HILL, 2001. 860 P. **139**

SUTHERLAND, J., “DISTRIBUTED SCRUM: AGILE PROJECT MANAGEMENT WITH OUTSOURCED DEVELOPMENT TEAMS”, HICSS, 2007. **139**

TELES, VINÍCIUS MANHÃES. EXTREME PROGRAMMING: APRENDA COMO ENCANTAR SEUS USUÁRIOS DESENVOLVENDO SOFTWARE COM AGILIDADE E ALTA QUALIDADE. 1. ED. SÃO PAULO: NOVATEC, 2004. 320 P. **139**

TRAVASSOS, G. H.,ABRANTES, J. F., “CARACTERIZAÇÃO DE MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE” **139**

10.1 INTRODUÇÃO ERROR! BOOKMARK NOT DEFINED.

10.2.1. CONCEITOS BÁSICOS ERROR! BOOKMARK NOT DEFINED.
10.2.2. PADRÕES OMG E A ARQUITETURA MDA ERROR! BOOKMARK NOT DEFINED.

10.3 ABORDAGENS MDD MODELOS ERROR! BOOKMARK NOT DEFINED.

10.3.1 OO-METHOD ERROR! BOOKMARK NOT DEFINED.
10.3.1.1. O PROCESSO BÁSICO DE TRANSFORMAÇÃO **ERROR! BOOKMARK NOT DEFINED.**
10.3.1.2. COMPARAÇÃO COM MDA **ERROR! BOOKMARK NOT DEFINED.**
10.3.1.3. O MODELO CONCEITUAL **ERROR! BOOKMARK NOT DEFINED.**
10.3.1.4. O COMPILADOR DE MODELOS **ERROR! BOOKMARK NOT DEFINED.**
10.3.1.5. OLIVANOVA **ERROR! BOOKMARK NOT DEFINED.**
10.3.2 . ANDROMDA ERROR! BOOKMARK NOT DEFINED.

10.4 PROBLEMAS E DESAFIOS DOS PROCESSOS MDD ERROR! BOOKMARK NOT DEFINED.

10.4.1. VISÃO GERAL ERROR! BOOKMARK NOT DEFINED.
10.4.2. **LIÇÕES APRENDIDAS NA ADOÇÃO DE SOLUÇÕES MDA** ERROR! BOOKMARK NOT DEFINED.
10.4.3. O PROGRAMA FASTSTART DA OMG ERROR! BOOKMARK NOT DEFINED.

10.5. TÓPICOS DE PESQUISA ERROR! BOOKMARK NOT DEFINED.

10.6 . SUGESTÕES DE LEITURA ERROR! BOOKMARK NOT DEFINED.

10.7 . EXERCÍCIOS ERROR! BOOKMARK NOT DEFINED.

REFERÊNCIAS ERROR! BOOKMARK NOT DEFINED.

1. INTRODUÇÃO ERROR! BOOKMARK NOT DEFINED.

1.1. O QUE É MODELAGEM DE PROCESSOS ERROR! BOOKMARK NOT DEFINED.
2.1. OBJETIVO DA MODELAGEM DE PROCESSOS SOFTWARE ERROR! BOOKMARK NOT DEFINED.
3.1 VANTAGENS E DESVANTAGENS DA UTILIZAÇÃO DE MODELAGEM DE PROCESSOS ERROR! BOOKMARK NOT DEFINED.
3.1.1 VANTAGENS **ERROR! BOOKMARK NOT DEFINED.**

- **BONS MODELOS DE PROCESSOS (CLAROS), SÃO A CHAVE PARA A BOA COMUNICAÇÃO.** ERROR! BOOKMARK NOT DEFINED.

- **SE O PROCESSO, É ALGUMA COISA NOVA QUE A EMPRESA ESTÁ PLANEJANDO EXECUTAR, O MODELO PODE AJUDAR A ASSEGURAR SUA EFICIÊNCIA DESDE O INÍCIO.** ERROR! BOOKMARK NOT DEFINED.

• **REVELAR ANOMALIAS, INCONSISTÊNCIAS, INEFICIÊNCIAS E OPORTUNIDADES DE MELHORIA, PERMITINDO À ORGANIZAÇÃO QUE SE COMPREENDA MELHOR E AUXILIANDO NA REENGENHARIA DESSES PROCESSOS.** ERROR! BOOKMARK NOT DEFINED.

• **FORNECER VISÃO CLARA E UNIFORMIZADA DAS ATIVIDADES, SUAS RAZÕES E FORMAS DE EXECUÇÃO.** ERROR! BOOKMARK NOT DEFINED.

• **UTILIZAR O MODELO COMO UM MEIO PARA DISTRIBUIÇÃO DE CONHECIMENTO DENTRO DA ORGANIZAÇÃO E TREINAR AS PESSOAS, AJUDANDO-AS A CONHECER MELHOR SEUS PAPÉIS E AS TAREFAS QUE EXECUTAM.** ERROR! BOOKMARK NOT DEFINED.

3.1.1 DESVANTAGENS **ERROR! BOOKMARK NOT DEFINED.**
4.1. LINGUAGEM DE MODELOS DE PROCESSOS ERROR! BOOKMARK NOT DEFINED.
4.1.1.BPM **ERROR! BOOKMARK NOT DEFINED.**
4.1.2.SPEM **ERROR! BOOKMARK NOT DEFINED.**

4.2.1 OMG(OBJECT MANAGEMENT GROUP) **ERROR! BOOKMARK NOT DEFINED.**
COMMON WAREHOUSE METAMODEL ™ **ERROR! BOOKMARK NOT DEFINED.**
COMMON OBJECT REQUEST BROKER ARCHITECTURE® **ERROR! BOOKMARK NOT DEFINED.**

6.1. FERRAMENTAS DE MODELAGEM *(COLOCAR EXEMPLO DIDATICO) ERROR! BOOKMARK NOT DEFINED.

6.1.1. COMPARAÇÃO ENTRA AS FERRAMENTAS ERROR! BOOKMARK NOT DEFINED.

7.1. SUGESTÕES DE LEITURA ERROR! BOOKMARK NOT DEFINED.

8.1. TÓPICOS DE PESQUISA ERROR! BOOKMARK NOT DEFINED.

EXERCÍCIO ERROR! BOOKMARK NOT DEFINED.

REFERENCIAS ERROR! BOOKMARK NOT DEFINED.

6.1. INTRODUÇÃO ERROR! BOOKMARK NOT DEFINED.

6.2. O QUE É QUALIDADE? ERROR! BOOKMARK NOT DEFINED.

6.3. COMPETITIVIDADE X PRODUTIVIDADE ERROR! BOOKMARK NOT DEFINED.

6.3.1. CONCEITO DE PRODUTIVIDADE ERROR! BOOKMARK NOT DEFINED.
6.3.2. CONCEITO DE COMPETITIVIDADE ERROR! BOOKMARK NOT DEFINED.

6.4. QUALIDADE TOTAL ERROR! BOOKMARK NOT DEFINED.

6.4.1. DEMING ERROR! BOOKMARK NOT DEFINED.

6.4.2. JURAN ERROR! BOOKMARK NOT DEFINED.

6.4.3. CROSBY ERROR! BOOKMARK NOT DEFINED.

6.4.4. FEIGENBAUN ERROR! BOOKMARK NOT DEFINED.

6.4.5. ISHIKAWA ERROR! BOOKMARK NOT DEFINED.

6.5. CONTROLE DA QUALIDADE TOTAL ERROR! BOOKMARK NOT DEFINED.

6.5.1. APRESENTAÇÃO DO CONTROLE DA QUALIDADE TOTAL ERROR! BOOKMARK NOT DEFINED.

6.5.2. SIGNIFICADO DO CONTROLE DA QUALIDADE TOTAL ERROR! BOOKMARK NOT DEFINED.

6.5.3. PRINCÍPIOS DA QUALIDADE TOTAL ERROR! BOOKMARK NOT DEFINED.

6.6. FERRAMENTAS DA QUALIDADE ERROR! BOOKMARK NOT DEFINED.

6.7. GESTÃO DA QUALIDADE TOTAL / ADMINISTRAÇÃO DA QUALIDADE ERROR! BOOKMARK NOT DEFINED.

1.7.1. GERENCIAMENTO POR PROCESSOS ERROR! BOOKMARK NOT DEFINED.
1.7.2. GERENCIAMENTO POR DIRETRIZES ERROR! BOOKMARK NOT DEFINED.
1.7.3. GERENCIAMENTO DA ROTINA ERROR! BOOKMARK NOT DEFINED.

1.8. GARANTIA DA QUALIDADE ERROR! BOOKMARK NOT DEFINED.

1.9. QUALIDADE NA INTERFACE COMPRAS/VENDAS ERROR! BOOKMARK NOT DEFINED.

1.9.1. QUALIDADE NAS VENDAS ERROR! BOOKMARK NOT DEFINED.
1.9.2. QUALIDADE NAS COMPRAS ERROR! BOOKMARK NOT DEFINED.

1.10. IMPLANTAÇÃO DO TQC ERROR! BOOKMARK NOT DEFINED.

6.9.1 FUNDAMENTOS ERROR! BOOKMARK NOT DEFINED.
6.9.2 ORGANIZAÇÃO PARA IMPLANTAÇÃO ERROR! BOOKMARK NOT DEFINED.
6.9.3 SISTEMA DE GERENCIAMENTO DA IMPLANTACÃO DO TQC ERROR! BOOKMARK NOT DEFINED.

6.10 TÓPICOS DE PESQUISA ERROR! BOOKMARK NOT DEFINED.

6.11. SUGESTÕES DE LEITURA ERROR! BOOKMARK NOT DEFINED.

6.12. EXERCÍCIOS ERROR! BOOKMARK NOT DEFINED.

6.13. REFERÊNCIAS ERROR! BOOKMARK NOT DEFINED.

9.2. ORGANISMOS NORMATIVOS ERROR! BOOKMARK NOT DEFINED.

9.2.1 ISO ERROR! BOOKMARK NOT DEFINED.

A PRIMEIRA VERSÃO ISO 9000:1987 SUBDIVIDIA-SE EM MODELOS PARA QUALIDADE, CLASSIFICADOS DA SEGUINTE FORMA [MATOS, 2009]: ERROR! BOOKMARK NOT DEFINED.

- **ISO 9001 : MODELO DE GARANTIA PARA QUALIDADE DE PROJETO, DESENVOLVIMENTO, PRODUÇÃO, MONTAGEM E FORNECEDORES APLICANDO-SE À ORGANIZAÇÕES CUJAS ATIVIDADES ERAM VOLTADAS PARA CRIAÇÃO DE NOVOS PRODUTOS.** ERROR! BOOKMARK NOT DEFINED.

9.3.4. NORMA ISO 9001 ERROR! BOOKMARK NOT DEFINED.

COM O LANÇAMENTO DA ISO 9000, VÁRIAS ORGANIZAÇÕES DESPERTARAM A TEMÁTICA DE QUE PRECISAVAM IMPOR, E PRINCIPALMENTE MANTER, PADRÕES DE QUALIDADE EM SEU FUNCIONAMENTO, SEJA NOS PROCESSOS, OU MESMO NAS PESSOAS QUE COLABORAM PARA O FUNCIONAMENTO DAS MESMAS. O PENSAMENTO COM UMA MELHOR VISÃO E AMBIÇÃO PARA O MERCADO DISPÕE DA REALIZAÇÃO DE INVESTIMENTOS QUE PRESTEM ALTERNATIVAS VIÁVEIS PARA O CRESCIMENTO E MELHORAMENTO DAS ATIVIDADES. ERROR! BOOKMARK NOT DEFINED.

MELLO, CARLOS HENRIQUE PEREIRA ET. AL.(2009) DESCREVE QUE AS NORMAS PARA SISTEMAS DE GESTÃO, PRINCIPALMENTE A ISO 9001, FORNECEM MODELOS BÁSICOS PARA QUE AS ORGANIZAÇÕES PREPAREM E OPEREM SEUS FLUXOS DE FUNCIONAMENTO SEGURAMENTE FORTIFICADOS. O AUTOR AINDA CITA QUE: ERROR! BOOKMARK NOT DEFINED.

"AS GRANDES ORGANIZAÇÕES, OU AQUELAS COM PROCESSOS COMPLEXOS, PODERIAM NÃO FUNCIONAR BEM SEM UM SISTEMA DE GESTÃO, APESAR DE ELE PODER TER SIDO CHAMADO POR ALGUM OUTRO NOME." ERROR! BOOKMARK NOT DEFINED.

A NORMA ISO 9001 FOI INSTITUÍDA COM ESSE PROPÓSITO. DESCREVER OS REQUISITOS PARA POSSIBILITAR A IMPLANTAÇÃO E ADMINISTRAÇÃO DE UM MODELO PARA GARANTIA DE QUALIDADE PARA PRODUTOS E SERVIÇOS ATRAVÉS DE UM SISTEMA DE GESTÃO DE QUALIDADE, COMO ESTRATÉGIA DE NEGÓCIOS PARA APRESENTAR UMA BASE SÓLIDA DE

SEGURANÇA E QUALIDADE NAS EMPRESAS, ESTA NORMA CONDIZ UM FATOR DE CERTIFICAÇÃO ATRAVÉS DE AUDITORIAS, INSPEÇÕES, DENTRE OUTRAS ATIVIDADES QUE CLASSIFIQUEM E GARANTAM BOA PROCEDÊNCIA PARA VERIFICAÇÃO E VALIDAÇÃO DE PROCESSOS E SERVIÇOS CONFORME AS TERMINOLOGIAS E VOCABULÁRIOS APRESENTADOS PELA ISO NA VERSÃO 9000. ERROR! BOOKMARK NOT DEFINED.

9.4.1 ESTRUTURA DA NORMA: PROCESSOS DE CICLO DE VIDA ERROR! BOOKMARK NOT DEFINED.

9.4.2 PROCESSOS PRIMÁRIOS ERROR! BOOKMARK NOT DEFINED.

9.4.3 PROCESSOS DE APOIO ERROR! BOOKMARK NOT DEFINED.

9.4.4 PROCESSOS ORGANIZACIONAIS ERROR! BOOKMARK NOT DEFINED.

9.5 ISO/IEC 15504 ERROR! BOOKMARK NOT DEFINED.

9.5.1 AVALIAÇÃO DE PROCESSOS ERROR! BOOKMARK NOT DEFINED.

9.5.2 PROJETO SPICE ERROR! BOOKMARK NOT DEFINED.

9.5.3 ESTRUTURA DA NORMA: REFERÊNCIA DE PROCESSOS ERROR! BOOKMARK NOT DEFINED.

9.5.4 DIMENSÃO DE PROCESSOS ERROR! BOOKMARK NOT DEFINED.

9.5.5 DIMENSÃO DE CAPACIDADE ERROR! BOOKMARK NOT DEFINED.

9.5.6 NÍVEIS DE CAPACIDADE ERROR! BOOKMARK NOT DEFINED.

9.6 CONCLUSÕES ERROR! BOOKMARK NOT DEFINED.

9.7 TÓPICOS DE PESQUISA ERROR! BOOKMARK NOT DEFINED.

9.8 SUGESTÕES DE LEIURA ERROR! BOOKMARK NOT DEFINED.

9.7 EXERCÍCIOS ERROR! BOOKMARK NOT DEFINED.

INTRODUÇÃO **125**

HISTÓRICO **126**

CMMI **127**

REPRESENTAÇÕES DO MODELO CMMI **129**
REPRESENTAÇÃO POR ESTÁGIOS 130
REPRESENTAÇÃO CONTÍNUA 131
REPRESENTAÇÃO POR ESTÁGIOS X CONTÍNUA 133
MÉTODO DE AVALIAÇÃO DO CMMI (SCAMPI) **134**
3.3.2.1. CONCEITO CENTRAL 134
3.3.2.2. PARÂMETROS OBSERVADOS NO SCAMPI 134
3.3.2.3. PRAZO E EXIGÊNCIA DE PESSOAL 135
3.3.2.4. CARACTERÍSTICAS ESSENCIAIS DO MÉTODO DE SCAMPI 135
3.3.2.5. MODOS DE USO 135
3.3.2.6. DESCRIÇÃO DO MÉTODO 136

MPS.BR **140**

3.4.1. REPRESENTAÇÃO DO MODELO MPS **141**
3.4.1.1. NÍVEL G – PARCIALMENTE GERENCIADO 142
3.4.1.2. NÍVEL F – GERENCIADO 142
3.4.1.3. NÍVEL E – PARCIALMENTE DEFINIDO 142
3.4.1.4. NÍVEL D – LARGAMENTE DEFINIDO 143
3.4.1.5. NÍVEL C – DEFINIDO 143
3.4.1.6. NÍVEL B – GERENCIADO QUANTITATIVAMENTE 144
3.4.1.7. NÍVEL A – EM OTIMIZAÇÃO 144
3.4.2. MÉTODO DE AVALIAÇÃO DO MPS.BR (MA-MPS) **144**
3.4.2.1. PRAZO E EXIGÊNCIA DE PESSOAL 145
3.4.2.2. DESCRIÇÃO DO MÉTODO 146

CMMI X MPS.BR **149**

EXERCÍCIOS **150**

SUGESTÕES DE LEITURA **151**

TÓPICOS DE PESQUISA **151**

REFERÊNCIAS **152**

INTRODUÇÃO A MODELOS PARA MELHORIA DE PROCESSOS DE SOFTWARE ERROR! BOOKMARK NOT DEFINED.
IDEAL ERROR! BOOKMARK NOT DEFINED.
- **FASES DO IDEAL** ERROR! BOOKMARK NOT DEFINED.
- **FASE INICIAL (INITIATING)** ERROR! BOOKMARK NOT DEFINED.
- **FASE DE DIAGNÓSTICO (DIAGNOSING)** ERROR! BOOKMARK NOT DEFINED.
- **FASE DE ESTABILIZAÇÃO (DIAGNOSING)** ERROR! BOOKMARK NOT DEFINED.

- **Fase de aproveitamento (Leveraging)** Error! Bookmark not defined.
- **Fase de gerenciamento do programa de melhoria do processo de software (Manage)** Error! Bookmark not defined.

PRO2PI Error! Bookmark not defined.

- **Engenharia de processo dirigida por perfis de capacidade e seus fundamentos** Error! Bookmark not defined.
- **O PRO2PI** Error! Bookmark not defined.
- **PRO2PI-PROP: Propriedades de PRO2PI** Error! Bookmark not defined.
- **PRO2PI-MODEL: Modelo de PRO2PI** Error! Bookmark not defined.
- **PRO2PI-MEAS: Medições para PRO2PI** Error! Bookmark not defined.
- **PRO2PI-CYCLE: Processo para ciclo de melhoria** Error! Bookmark not defined.

Seis Sigma Error! Bookmark not defined.

- **PDCA** Error! Bookmark not defined.
- **DMAIC** Error! Bookmark not defined.
- **Definir** Error! Bookmark not defined.
- **Medição** Error! Bookmark not defined.
- **Análise** Error! Bookmark not defined.
- **Melhoria** Error! Bookmark not defined.
- **Controle** Error! Bookmark not defined.

Considerações Finais Error! Bookmark not defined.
Exercícios Error! Bookmark not defined.
Sugestões de leitura Error! Bookmark not defined.

PARA ENTENDER MELHOR O QUE É MELHORIA DE PROCESSO DE SOFTWARE LEI A NORMA ISO/IEC 15504-4/2004. Error! Bookmark not defined.

PARA UM ESTUDO DETALHADO SOBRE O PRO2PI LEIA TESE DE DOUTORADO DE CLÊNIO SALVIANO, "UMA PROPOSTA ORIENTADA A PERFIS DE CAPACIDADE DE PROCESSO PARA EVOLUÇÃO DA MELHORIA DE PROCESSO DE SOFTWARE". DISPONÍVEL EM: HTTP://LIBDIGI.UNICAMP.BR/DOCUMENT/?CODE=VTLS000380495 Error! Bookmark not defined.

PARA UM ESTUDO DETALHADO SOBRE IDEAL LEIA O GUIA OFICIAL DE IMPLANTAÇÃO PRODUZIDO PELO SEI, "IDEAL - A USER'S GUIDE FOR SOFTWARE PROCESS IMPROVEMENT". DISPONÍVEL EM: HTTP://WWW.SEI.CMU.EDU/LIBRARY/ABSTRACTS/REPORTS/96HB001.CFM Error! Bookmark not defined.

Tópicos de pesquisa Error! Bookmark not defined.
Referências Error! Bookmark not defined.

1.1. MODELOS DE QUALIDADE DE PRODUTO 198

OS MODELOS DE QUALIDADE OBJETIVAM AVALIAR O PRODUTO DE SOFTWARE, SEGUNDO DIFERENTES ASPECTOS BASEADOS NA VISÃO DO USUÁRIO. PARA PADRONIZAR INTERNACIONALMENTE AS CARACTERÍSTICAS DE

IMPLEMENTAÇÃO DO SOFTWARE, FORAM CRIADAS ALGUMAS NORMAS QUE SERÃO VISTAS A SEGUIR. 198

1.1.1. ISO 9126 198

1.1.1.1. DIRETRIZES PARA USO DA NORMA NBR ISO/IEC 9126-1 198

1.1.1.2. CARACTERÍSTICAS E SUB-CARACTERÍSTICAS DE QUALIDADE DE SOFTWARE 199

1.1.2. ISO 12119 201

1.1.3. ISO 14598 203

É UM GUIA PARA AVALIAÇÃO DE PRODUTOS DE SOFTWARE, BASEADO NA UTILIZAÇÃO PRÁTICA DA NORMA ISO 9126, JÁ QUE ESTA DEFINE AS MÉTRICAS, CARACTERÍSTICAS E SUBCARACTERÍSTICAS DE QUALIDADE DE SOFTWARE [KOSCIANSKI & SOARES, 2007]. 203

1.1.4. PROJETO SQUARE 205

1.1.5. NORMA SQUARE 205

NA REORGANIZAÇÃO DAS ANTIGAS NORMAS 9126 E 14598, O PROJETO SQUARE ADOTOU UMA DIVISÃO DE ASSUNTOS EM CINCO TÓPICOS QUE APARECEM NA FIGURA 1.3: 205

1.2. TESTE DE SOFTWARE 207

1.2.1. ABORDAGENS DE TESTES 208

EXISTEM DUAS ABORDAGENS PRINCIPAIS DE TESTES: ABORDAGEM FUNCIONAL ("BLACK BOX" OU "CAIXA PRETA") E ABORDAGEM ESTRUTURAL ("WHITE BOX" OU "CAIXA BRANCA"). 208

- CAIXA PRETA: COMO O PRÓPRIO NOME JÁ SUGERE, NESTA ABORDAGEM O TESTADOR VISUALIZA O SOFTWARE COMO UMA CAIXA PRETA, OU SEJA, NÃO CONSIDERA A ESTRUTURA INTERNA DO PROGRAMA, DE QUE FORMA O CÓDIGO FOI IMPLEMENTADO OU QUE TECNOLOGIA FOI UTILIZADA, POR EXEMPLO. CONSIDERANDO OS DADOS DE ENTRADA, O OBJETIVO PRINCIPAL É OBSERVAR AS SAÍDAS GERADAS PELO SISTEMA E VERIFICAR SE ESTAS ESTÃO DE ACORDO COM O ESPERADO. A FIGURA 1.4 ILUSTRA ESTE TIPO DE ABORDAGEM. 208

- CAIXA BRANCA: DIFERENTEMENTE DA ABORDAGEM ANTERIOR, NESTE TIPO DE ABORDAGEM O TESTADOR ESTÁ INTERESSADO NO QUE ESTÁ

ACONTECENDO “DENTRO DA CAIXA”. É CARACTERIZADA POR AVALIAR AS FUNCIONALIDADES INTERNAS DOS COMPONENTES DO SOFTWARE, BASEANDO-SE NO CÓDIGO FONTE E PROCURANDO EXERCITAR ESTRUTURAS DE CONTROLE E DE DADOS DO PROGRAMA. SENDO ASSIM, FAZ-SE NECESSÁRIO QUE O ANALISTA DE TESTES TENHA BOA HABILIDADE EM PROGRAMAÇÃO DE MODO A ENTENDER TODOS OS CAMINHOS LÓGICOS POSSÍVEIS. A FIGURA 1.5 ILUSTRA A ABORDAGEM ESTRUTURAL. **208**

1.2.2. ESTÁGIOS DE TESTES **208**

OS TESTES DE SOFTWARE NORMALMENTE SÃO EXECUTADOS EM DIFERENTES ESTÁGIOS DURANTE O CICLO DE VIDA DO DESENVOLVIMENTO DO SOFTWARE. DEPENDENDO DO OBJETIVO PRINCIPAL DO TESTE, QUATRO ESTÁGIOS FORAM DEFINIDOS: **208**

- TESTE DE UNIDADE: REALIZA TESTES EM COMPONENTES INDIVIDUAIS (MÓDULOS, PROGRAMAS, OBJETOS, CLASSES, ETC) DE FORMA A DETERMINAR SE CADA UM DELES, SEPARADAMENTE, ESTÁ SENDO EXECUTADO DE MANEIRA CORRETA. NORMALMENTE ESTES TESTES SÃO TESTES DE CAIXA BRANCA REALIZADOS PELOS PRÓPRIOS DESENVOLVEDORES DO COMPONENTE. GERALMENTE UTILIZAM FERRAMENTAS QUE PROVÊEM UM SUPORTE ADICIONAL PARA CHECAR A CORRETUDE DO PROGRAMA, COMO FERRAMENTA DE *DEBBUGING* OU *FRAMEWORK* PARA TESTE UNITÁRIO, POR EXEMPLO. OS DEFEITOS ENCONTRADOS NESTE ESTÁGIO SÃO NORMALMENTE CORRIGIDOS DE IMEDIATO, SEM A NECESSIDADE DE DOCUMENTÁ-LOS FORMALMENTE, E ASSIM, REDUZINDO O CUSTO, POIS ANTECIPA A CORREÇÃO DE DEFEITOS. GERALMENTE É NECESSÁRIA A UTILIZAÇÃO DE *STUBS* (MÓDULOS QUE SUBSTITUEM OUTROS MÓDULOS SUBORDINADOS) E *DRIVERS* (UM MÓDULO QUE SUBSTITUI OUTRO MÓDULO QUE SEJA RESPONSÁVEL POR CONTROLAR A CHAMADA DE UM SISTEMA), PARA SEREM UTILIZADOS NO LUGAR DOS SOFTWARES QUE ESTEJAM EVENTUALMENTE FALTANDO E PARA SIMULAR A INTERFACE ENTRE OS COMPONENTES DE SOFTWARE. **208**

- TESTE DE INTEGRAÇÃO: NESTA ETAPA, AS UNIDADES QUE FORAM TESTADAS INDIVIDUALMENTE NO ESTÁGIO ANTERIOR SÃO TESTADAS DE FORMA INTEGRADA, BEM COMO AS INTERFACES ENTRE OS COMPONENTES. A INTEGRAÇÃO DEVE SER REALIZADA ADICIONANDO-SE OS COMPONENTES UM POR UM, E APÓS CADA PASSO UM TESTE É NECESSÁRIO (TESTE INCREMENTAL). ESTA TÉCNICA TEM A VANTAGEM DE ACHAR DEFEITOS O MAIS CEDO POSSÍVEL NO PROCESSO DE TESTES E CORRIGI-LOS MAIS RAPIDAMENTE, ENQUANTO É MAIS FÁCIL DETERMINAR AS CAUSAS DOS ERROS. POR OUTRO LADO, TEM A DESVANTAGEM DE SER UMA PRÁTICA BASTANTE CUSTOSA. SENDO ASSIM, A INTEGRAÇÃO PODE SER FEITA BASICAMENTE DE DUAS FORMAS: *TOP-DOWN* OU *BOTTOM-UP*. NA PRIMEIRA, OS TESTES SÃO REALIZADOS DE CIMA PARA BAIXO (COMEÇANDO DA GUI OU DO MENU PRINCIPAL); COMPONENTES OU SISTEMAS SÃO SUBSTITUÍDOS POR *STUBS*. NA SEGUNDA, OS TESTES COMEÇAM NA PARTE MAIS BÁSICA DO SISTEMA ATÉ O NÍVEL MAIS ALTO; COMPONENTES OU SISTEMAS SÃO SUBSTITUÍDOS POR *DRIVERS*. **209**

- TESTE DE SISTEMA: NESTE NÍVEL O PROPÓSITO DO TESTE ESTÁ EM VERIFICAR O FUNCIONAMENTO DE TODO O SISTEMA, JÁ INTEGRADO, E ANALISAR SE ELE ESTÁ DE ACORDO COM OS REQUISITOS QUE FORAM ESPECIFICADOS. NESTE MOMENTO NÃO SÓ É TESTADA A INTEGRAÇÃO DOS COMPONENTES DO SOFTWARE ENTRE SI, MAS TAMBÉM DESTES COM UM AMBIENTE DE TESTE CORRESPONDENTE À PRODUÇÃO FINAL (HARDWARE, SOFTWARE, OUTROS SISTEMAS), DE MODO A MINIMIZAR O RISCO DE QUE FALHAS RELACIONADAS COM O AMBIENTE OPERACIONAL DO PRODUTO NÃO SEJAM ENCONTRADAS. GERALMENTE A ESTRATÉGIA DE CAIXA PRETA É UTILIZADA NESTE ESTÁGIO, MAS TESTES DE CAIXA BRANCA TAMBÉM PODEM SER REALIZADOS. **209**

- TESTE DE ACEITAÇÃO: O TESTE DE ACEITAÇÃO CORRESPONDE AO TESTE REALIZADO PELO USUÁRIO DE FATO DO SISTEMA, NO MOMENTO EM QUE TODOS OU QUASE TODOS OS DEFEITOS ENCONTRADOS NAS ETAPAS ANTERIORES JÁ TENHAM SIDO CORRIGIDOS. O PROPÓSITO DESTE TESTE É ESTABELECER A CONFIANÇA DO SISTEMA; ELE ESTÁ MAIS RELACIONADO COM A VALIDAÇÃO DO SISTEMA, EM QUE ESTÁ SE TENTANDO DETERMINAR SE O SISTEMA FAZ AQUILO QUE É SUPOSTO FAZER. NORMALMENTE OS TESTES DE ACEITAÇÃO PODEM SER DE DUAS CATEGORIAS: TESTES *ALFA* E TESTES *BETA*. OS PRIMEIROS SÃO REALIZADOS NAS INSTALAÇÕES DO DESENVOLVEDOR, QUE FICA OBSERVANDO OS USUÁRIOS UTILIZAREM O SISTEMA, E ANOTAM OS PROBLEMAS IDENTIFICADOS. JÁ OS TESTES *BETA* SÃO REALIZADOS NO AMBIENTE REAL DE TRABALHO DO USUÁRIO, QUE INSTALA O SISTEMA E TESTA, SEM A PRESENÇA DO DESENVOLVEDOR. EM SEGUIDA, UM DOCUMENTO CONTENDO OS REGISTROS DOS PROBLEMAS ENCONTRADOS É ENVIADO À ORGANIZAÇÃO DESENVOLVEDORA. **209**

1.2.3. TIPOS DE TESTES **209**

CADA TIPO DE TESTE É FOCADO EM UM GRUPO DE ATIVIDADES COM UM DETERMINADO OBJETIVO DE TESTE. É NECESSÁRIO PENSAR EM DIFERENTES TIPOS DE TESTES UMA VEZ QUE TESTAR A FUNCIONALIDADE DE UM COMPONENTE OU SISTEMA PODE NÃO SER SUFICIENTE EM CADA UM DOS ESTÁGIOS ENVOLVIDOS PARA SE CHEGAR AOS OBJETIVOS DOS TESTES. UM TIPO DE TESTE É FOCADO NUM OBJETIVO PARTICULAR DE TESTE, QUE PODERIA SER UM TESTE DE UMA FUNÇÃO A SER EXECUTADA PELO COMPONENTE OU SISTEMA; ALGUMA CARACTERÍSTICA NÃO FUNCIONAL; A ESTRUTURA OU ARQUITETURA DO COMPONENTE OU SISTEMA, ETC. EXISTEM VÁRIOS TIPOS DE TESTES, DEPENDENDO DO OBJETIVO DE CADA PROJETO E DE CADA ORGANIZAÇÃO. ABAIXO SERÃO APRESENTADOS ALGUNS DOS MAIS COMUNS. **209**

- TESTE FUNCIONAL – ESTE TIPO DE TESTE ESTÁ FOCADO NAS REGRAS DE NEGÓCIO DO SISTEMA, OU SEJA, O FLUXO DE TRABALHO DO PROGRAMA É AVALIADO. **210**

- TESTE DE RECUPERAÇÃO DE FALHA – O SISTEMA É FORÇADO A FALHAR DE DIVERSAS MANEIRAS DE MODO A VERIFICAR SEU COMPORTAMENTO DIANTE DESTAS FALHAS, E REPARAR DE QUE FORMAS ELE SE RECUPERA. **210**

- TESTE DE INTEROPERABILIDADE – TESTA UM PRODUTO DE SOFTWARE DE MODO A DETERMINAR SUA CAPACIDADE DE INTERAGIR COM UM OU MAIS COMPONENTES OU SISTEMAS. **210**

- TESTE DE SEGURANÇA – VERIFICA SE O SISTEMA POSSUI ATRIBUTOS PARA PREVENIR ACESSOS NÃO AUTORIZADOS, ACIDENTAIS OU PROPOSITAIS, A PROGRAMAS E DADOS. **210**

- TESTE DE CARGA – UM TIPO DE TESTE PARA MEDIR O COMPORTAMENTO DO SISTEMA QUANDO ESTE É SUBMETIDO A NÍVEIS ALTOS DE CARGA, DIFERENTE DAS CONDIÇÕES NORMAIS. É IMPORTANTE DETERMINAR O QUANTO DE CARGA O SISTEMA CONSEGUE SUPORTAR SEM FALHAR. **210**

- TESTE DE PERFORMANCE – VERIFICA O RENDIMENTO DE UM SISTEMA, COMO O TEMPO DE RESPOSTA E PROCESSAMENTO, TAXA DE TRANSFERÊNCIA DE DADOS, PARA DIFERENTES CONDIÇÕES (CONFIGURAÇÕES, NUMERO DE USUÁRIOS, ETC) AS QUAIS O PROGRAMA É SUBMETIDO. **210**

- TESTE DE ESTRESSE – TESTE CONDUZIDO PARA AVALIAR O COMPORTAMENTO DO SISTEMA DIANTE DE CONDIÇÕES QUE ULTRAPASSEM O LIMITE ESPECIFICADO NOS REQUISITOS. **210**

- TESTE DE CONFIGURAÇÃO – TESTA O FUNCIONAMENTO DO SISTEMA EM DIFERENTES CONFIGURAÇÕES DE HARDWARE/SOFTWARE, TESTANDO COMPATIBILIDADE, CONFIGURAÇÃO DO SERVIDOR, TIPOS DE CONEXÕES COM A INTERNET, ETC. **210**

- TESTE DE USABILIDADE – TESTES PARA DETERMINAR SE UM PRODUTO É FACILMENTE ENTENDÍVEL, FÁCIL DE APRENDER, FÁCIL DE OPERAR E ATRATIVO AOS USUÁRIOS, OU SEJA, SE O PRODUTO TEM UMA INTERFACE AMIGÁVEL PARA OS QUE UTILIZARÃO O SISTEMA. **210**

- TESTE DE REGRESSÃO – TESTE DE REGRESSÃO É A ATIVIDADE DE TESTAR UMA NOVA VERSÃO DE UM SISTEMA PARA VALIDAR ESTA VERSÃO, DETECTANDO SE ERROS FORAM INTRODUZIDOS DEVIDO ÀS MUDANÇAS REALIZADAS NO SOFTWARE, E ENTÃO, GARANTIR A CORRETUDE DAS MODIFICAÇÕES. UMA VEZ QUE A RE-EXECUÇÃO DE TODOS OS TESTES É UMA ATIVIDADE BASTANTE CUSTOSA, UMA SELEÇÃO DE TESTES DE REGRESSÃO GERALMENTE É REALIZADA. **210**

1.2.4. PROCESSO DE TESTES **210**

SEGUNDO [GRAHAM ET. AL 2007], O PROCESSO DE TESTES PODE SER DIVIDIDO BASICAMENTE EM CINCO ETAPAS: *PLANEJAMENTO E CONTROLE, ANÁLISE E PROJETO, IMPLEMENTAÇÃO E EXECUÇÃO, AVALIAÇÃO DE CRITÉRIO DE SAÍDA E REPORTAGEM* E *ATIVIDADES DE ENCERRAMENTO DE TESTES*. ESTAS ATIVIDADES SÃO LOGICAMENTE SEQÜENCIAIS, PORÉM, EM UM PROJETO ESPECÍFICO, PODEM SE SOBREPOR, SEREM EXECUTADAS EM PARALELO OU ATÉ MESMO SEREM REPETIDAS. **210**

1.2.4.1. PLANEJAMENTO E CONTROLE **211**

DURANTE O PLANEJAMENTO DE TESTES DEVE-SE TER CERTEZA DE QUE OS OBJETIVOS DOS CLIENTES E *STAKEHOLDERS* FORAM ENTENDIDOS DE MANEIRA CORRETA. BASEADOS NESTE ENTENDIMENTO, OS PROPÓSITOS DA ATIVIDADE DE TESTES PROPRIAMENTE DITA SÃO ESTABELECIDOS, E ASSIM, UMA ABORDAGEM E PLANO PARA OS TESTES É OBTIDA INCLUINDO ESPECIFICAÇÃO DAS ATIVIDADES DE TESTE. O PLANEJAMENTO DE TESTES APRESENTA AS SEGUINTES ATIVIDADES PRINCIPAIS: **211**

- DETERMINAR O ESCOPO E RISCOS E IDENTIFICAR OS OBJETIVOS DE TESTE: SÃO DETERMINADOS OS SOFTWARES, COMPONENTES, SISTEMAS OU OUTROS PRODUTOS QUE DEVEM SER TESTADOS; OS RISCOS QUE DEVEM SER LEVADOS EM CONSIDERAÇÃO; E QUAL O PROPÓSITO DO TESTE (ENCONTRAR DEFEITOS, VERIFICAR SE ESTÁ DE ACORDO COM OS REQUISITOS OU DENTRO DOS PADRÕES DE QUALIDADE, ETC). **211**

- DETERMINAR A ESTRATÉGIA DE TESTE: AQUI SERÃO ESTABELECIDAS AS TÉCNICAS QUE SERÃO UTILIZADAS, O QUE PRECISA DE FATO SER TESTADO (SELECIONAR E PRIORIZAR OS REQUISITOS) E QUE NÍVEL DE COBERTURA É NECESSÁRIO. SERÃO TAMBÉM ANALISADAS QUAIS PESSOAS PRECISARÃO SE ENVOLVER E EM QUE MOMENTO (DESENVOLVEDORES, USUÁRIOS, ETC), INCLUINDO A DEFINIÇÃO DA EQUIPE DE TESTE. **211**

- DEFINIR RECURSOS: SÃO DEFINIDOS TODOS OS RECURSOS NECESSÁRIOS DURANTE O CICLO DE VIDA DE TESTES, TANTO RECURSOS MATERIAIS (PCS, SOFTWARE, FERRAMENTAS, ETC) COMO RECURSOS HUMANOS (PRINCIPAIS E DE APOIO). **211**

- FAZER UM CRONOGRAMA PARA ANÁLISE E PROJETO, IMPLEMENTAÇÃO, EXECUÇÃO E AVALIAÇÃO DE TESTE: DEVERÁ SER ELABORADO UM CRONOGRAMA DE TODAS AS TAREFAS E ATIVIDADES, PARA QUE SEJA POSSÍVEL TERMINAR A FASE DE TESTES A TEMPO. **211**

- ESTABELECER OS CRITÉRIOS DE SAÍDA: CRITÉRIOS DE SAÍDA, COMO CRITÉRIO DE COBERTURA, POR EXEMPLO, DEVERÃO SER ESTABELECIDOS DE MODO A DETERMINAR QUANDO A ETAPA DE TESTES CHEGOU AO FIM. **211**

APÓS PLANEJAR É NECESSÁRIA UMA MEDIDA DE CONTROLE PARA VERIFICAR SE TUDO ESTÁ INDO DE ACORDO COM O PLANEJADO. É PRECISO COMPARAR O ANDAMENTO REAL COM O QUE FOI ESTABELECIDO NO PLANO DE TESTES, E TOMAR MEDIDAS CORRETIVAS QUANDO NECESSÁRIO. **211**

1.2.4.2. ANÁLISE E PROJETO **211**

ESTA É A ATIVIDADE EM QUE OS OBJETIVOS GERAIS DE TESTES SÃO TRANSFORMADOS EM CONDIÇÕES E PROJETOS DE TESTE TANGÍVEIS. O PROPÓSITO PRINCIPAL É IDENTIFICAR E DESCREVER OS CASOS DE TESTE PARA CADA VERSÃO DE TESTE, E IDENTIFICAR E ESTRUTURAR OS PROCEDIMENTOS DE TESTE, ESPECIFICANDO COMO EXECUTAR OS CASOS DE TESTE. AS PRINCIPAIS TAREFAS DESTA ETAPA PODEM SER DESTACADAS EM: **211**

- REVISAR A BASE DE TESTES (COMO A ANÁLISE DE RISCO DO PRODUTO, REQUISITOS, ARQUITETURA, ESPECIFICAÇÃO DE PROJETO, E INTERFACES): A BASE DE TESTES É UTILIZADA PARA CRIAR OS TESTES. É POSSÍVEL COMEÇAR A PROJETAR OS TESTES DE CAIXA PRETA ANTES DA IMPLEMENTAÇO, UMA VEZ QUE A BASE DE TESTES PODE SER USADA PARA COMPREENDER O QUE O SISTEMA PRECISA FAZER. **211**

- IDENTIFICAR E DESCREVER CASOS DE TESTE: UM CASO DE TESTE É UM CENÁRIO ASSOCIADO A UM REQUISITO; É UM TEXTO CONTENDO: IDENTIFICADOR, OBJETIVO, PRÉ-CONDIÇÕES DE EXECUÇÃO, ENTRADAS, PASSOS ESPECÍFICOS DO TESTE A SER EXECUTADO E RESULTADOS ESPERADOS E/OU PÓS-CONDIÇÕES DE EXECUÇÃO. UM CASO DE TESTE BEM PROJETADO TEM MUITA CHANCE DE ENCONTRAR UM ERRO AINDA NÃO CONHECIDO. **212**

- ESTRUTURAR PROCEDIMENTOS DE TESTE: O PASSO A PASSO QUE DESCREVE COMO OS CASOS DE TESTE DEVEM SER EXECUTADOS. INCLUI O ESTADO INICIAL DA APLICAÇÃO; CONDIÇÕES DE FUNCIONAMENTO; COMO E QUANDO FORNECER OS DADOS DE ENTRADA E OBTER OS RESULTADOS; A FORMA DE AVALIAR ESTES RESULTADOS, DENTRE OUTROS. **212**

- AVALIAR A CAPACIDADE DE TESTAR OS REQUISITOS: A ESPECIFICAÇÃO DE REQUISITOS DEVE SER COMPLETAMENTE CLARA, INFORMANDO AS CONDIÇÕES NECESSÁRIAS PARA SE DEFINIR OS TESTES. POR EXEMPLO, SE A PERFORMANCE DO SOFTWARE É ALGO CRÍTICO, DEVE SER CLARAMENTE ESPECIFICADO O TEMPO DE RESPOSTA MÍNIMO EM QUE O SISTEMA DEVE RESPONDER. **212**

1.2.4.3. IMPLEMENTAÇÃO E EXECUÇÃO **212**

UMA VEZ QUE OS CASOS E PROCEDIMENTOS DE TESTE FORAM ESPECIFICADOS EM ALTO NÍVEL NA ETAPA ANTERIOR, ESTE É O MOMENTO EM QUE O AMBIENTE SERÁ PREPARADO PARA QUE ELES SEJAM EXECUTADOS E COMPARADOS COM OS RESULTADOS DESEJADOS. ALÉM DISSO, É A ETAPA EM QUE OS COMPONENTES NECESSÁRIOS SÃO IMPLEMENTADOS PARA QUE OS

TESTES SEJAM EXECUTADOS. AS PRINCIPAIS TAREFAS DESTAS DUAS FASES SERÃO DESTACADAS A SEGUIR. 212

- IMPLEMENTAÇÃO: 212

○ IMPLEMENTAR COMPONENTES: EFETUAR A IMPLEMENTAÇÃO DE NOVOS COMPONENTES DE APOIO NECESSÁRIOS À APLICAÇÃO DOS TESTES, OU MODIFICAÇÃO DE COMPONENTES JÁ EXISTENTES. FERRAMENTAS DE AUTOMAÇÃO PODEM SER UTILIZADAS OU OS COMPONENTES PODEM SER DESENVOLVIDOS EXPLICITAMENTE. 212

○ CRIAR SUÍTES DE TESTE: BASEADO NOS CASOS DE TESTE, UM CONJUNTO DE TESTES QUE NATURALMENTE TRABALHAM JUNTOS, FORMA UMA SUÍTE DE TESTE E SÃO UTILIZADOS PARA UMA EXECUÇÃO DE TESTE EFICIENTE. 212

○ IMPLEMENTAR E VERIFICAR O AMBIENTE: PREPARAR E VERIFICAR SE O AMBIENTE DE TESTE ESTÁ FUNCIONANDO CORRETAMENTE. 212

- EXECUÇÃO: 212

○ EXECUTAR AS SUÍTES DE TESTE E CASOS DE TESTE INDIVIDUAIS, DE ACORDO COM OS PROCEDIMENTOS DE TESTE. PODE SER FEITO MANUALMENTE OU COM O AUXÍLIO DE FERRAMENTAS DE EXECUÇÃO DE TESTES. 212

○ SEGUIR AS ESTRATÉGIAS DE TESTE DEFINIDAS NA ETAPA DE PLANEJAMENTO. 212

○ CRIAR UM *LOG* COM AS SAÍDAS DA EXECUÇÃO DOS TESTES E REGISTRAR OS IDENTIFICADORES E VERSÕES DO SOFTWARE QUE ESTÁ SENDO TESTADO. 212

○ FAZER A COMPARAÇÃO DOS RESULTADOS ESPERADOS E DOS RESULTADOS OBTIDOS. 212

○ QUANDO HOUVER DIFERENÇAS ENTRE OS RESULTADOS ESPERADOS E OS RESULTADOS OBTIDOS, REGISTRAR OS DEFEITOS EM UM REPOSITÓRIO CENTRALIZADO. NÃO SE DEVE REGISTRÁ-LOS DE FORMA ALEATÓRIA. 212

○ REALIZAÇÃO DE TESTES DE REGRESSÃO PARA CONFIRMAR QUE UMA FALHA ANTERIORMENTE REGISTRADA FOI DE FATO CONSERTADA. 213

1.2.4.4. AVALIAÇÃO DO CRITÉRIO DE SAÍDA E RELATÓRIO 213

ESTA É A FASE EM QUE SE DESEJA OBSERVAR SE JÁ FORAM EXECUTADOS TESTES SUFICIENTES PARA GARANTIR A QUALIDADE DESEJADA DO PRODUTO

SENDO ASSIM, CRITÉRIOS DE SAÍDA SÃO DEFINIDOS COM ESTA FINALIDADE. ESTES CRITÉRIOS INFORMAM SE UMA DADA ATIVIDADE DE TESTES PODE SER CONSIDERADA COMPLETA. AS PRINCIPAIS ATIVIDADES SÃO: **213**

- CHECAR SE OS *LOGS* DE TESTES BATEM COM OS CRITÉRIOS DE SAÍDA ESPECIFICADOS NO PLANO DE TESTES: PROCURA-SE PELOS TESTES QUE TENHAM SIDO EXECUTADOS E AVALIADOS, E SE DEFEITOS FORAM ENCONTRADOS, CONSERTADOS OU RE-TESTADOS. **213**

- VERIFICAR SE SERÁ NECESSÁRIA A INCLUSÃO DE MAIS TESTES OU SE OS CRITÉRIOS DE SAÍDA ESPECIFICADOS DEVEM SER MUDADOS: MAIS CASOS DE TESTES PODEM PRECISAR SER EXECUTADOS, SE POR ACASO ESTES NÃO TIVEREM SIDO TODOS EXECUTADOS CONFORME ESPERADO, OU SE FOR DETECTADO QUE A COBERTURA DE REQUISITOS NECESSÁRIA AINDA NÃO FOI ATINGIDA, OU ATÉ MESMO SE AUMENTARAM OS RISCOS DO PROJETO. **213**

- ESCREVER UM RELATÓRIO DE RESUMO DE TESTES PARA OS *STAKEHOLDERS*: TODOS OS *STAKEHOLDERS* DEVEM SABER QUAIS TESTES FORAM EXECUTADOS E QUAIS OS RESULTADOS DESTES TESTES, DE MODO A PERCEBER QUE DECISÕES PRECISAM AINDA SER TOMADAS VISANDO A MELHORIA DA QUALIDADE DO SOFTWARE. **213**

1.2.4.5. ATIVIDADES DE ENCERRAMENTO DE TESTE 213

A ATIVIDADE DE ENCERRAMENTO DE TESTE PODE SER DADA ATRAVÉS DE DIVERSOS FATORES, COMO POR EXEMPLO, AS INFORMAÇÕES NECESSÁRIAS DO PROCESSO DE TESTES JÁ FORAM ATINGIDAS; O PROJETO É CANCELADO; QUANDO UM MARCO PARTICULAR É ALCANÇADO; OU QUANDO UMA VERSÃO DE MANUTENÇÃO OU ATUALIZAÇÃO ESTÁ CONCLUÍDA. AS ATIVIDADES PRINCIPAIS SÃO: **213**

- CHECAR SE AS ENTREGAS QUE FORAM PROGRAMADAS FORAM DE FATO ENTREGUES E GARANTIR QUE TODOS OS PROBLEMAS REPORTADOS FORAM REALMENTE RESOLVIDOS. PARA OS QUE PERMANECERAM EM ABERTO DEVEM-SE REQUISITAR MUDANÇAS EM UMA FUTURA VERSÃO. **213**

- FINALIZAR E ARQUIVAR OS ARTEFATOS PRODUZIDOS DURANTE O PROCESSO NECESSÁRIO PARA PLANEJAR, PROJETAR E EXECUTAR TESTES, COMO POR EXEMPLO, DOCUMENTAÇÃO, SCRIPTS, ENTRADAS, RESULTADOS ESPERADOS, ETC. É IMPORTANTE REUTILIZAR TUDO QUE FOR POSSÍVEL DESTES ARTEFATOS, POIS ASSIM SE CONSEGUE ECONOMIZAR TEMPO E ESFORÇO DO PROJETO. **213**

- REPASSAR OS ARTEFATOS ANTERIORMENTE CITADOS PARA A EQUIPE DE MANUTENÇÃO, QUE IRÁ PROVER SUPORTE AOS USUÁRIOS DO SISTEMA E RESOLVER QUALQUER PROBLEMA ENCONTRADO DEPOIS DE SUA ENTREGA. **213**

• AVALIAR COMO SE DEU O PROCESSO DE TESTES E ANALISAR AS LIÇÕES APRENDIDAS, QUE SERÃO DE GRANDE UTILIDADE PARA FUTURAS VERSÕES DOS PROJETOS. ESTE PASSO PODE PERMITIR NÃO SÓ MELHORIAS NO PROCESSO DE TESTES, COMO TAMBÉM MELHORIAS NO PROCESSO DE DESENVOLVIMENTO DO SOFTWARE COMO UM TODO. 213

1.2.5. TESTES AO LONGO DO CICLO DE VIDA DE SOFTWARE 213

AS ATIVIDADES DE TESTE NÃO SÃO ATIVIDADES QUE SÃO REALIZADAS SOZINHAS, MAS SIM EM PARALELO COM O CICLO DE VIDA DE DESENVOLVIMENTO DO SOFTWARE. DESSA FORMA, A ESCOLHA DO CICLO DE VIDA DO PROJETO IRÁ AFETAR DIRETAMENTE AS ATIVIDADES DE TESTE. O PROCESSO DE DESENVOLVIMENTO ADOTADO DEPENDE MUITO DOS OBJETIVOS E PROPÓSITOS DO PROJETO. PORTANTO, O MODO COMO AS ATIVIDADES DE TESTE SÃO ESTRUTURADAS DEVE SE AJUSTAR AO MODELO DE DESENVOLVIMENTO DE SOFTWARE, OU DO CONTRARIO, NÃO CONSEGUIRÁ OBTER O SUCESSO DESEJADO. 213

UM MODELO DE DESENVOLVIMENTO DE SOFTWARE BASTANTE CONHECIDO É O MODELO EM CASCATA, QUE COMO O PRÓPRIO NOME JÁ SUGERE, TEM SUA BASE VOLTADA A UM DESENVOLVIMENTO SEQÜENCIAL DAS ATIVIDADES. AS PRIMEIRAS ATIVIDADES COMEÇAM NO TOPO DA CASCATA, E ENTÃO VÃO SEGUINDO SEQÜENCIALMENTE ATRAVÉS DAS VÁRIAS ATIVIDADES DE CONCEPÇÃO DO PROJETO, E FINALMENTE TERMINANDO COM A ETAPA DE IMPLEMENTAÇÃO. APÓS ISSO, É QUE AS ATIVIDADES DE TESTE SÃO INTRODUZIDAS, E DESSA FORMA OS DEFEITOS SÓ PODEM SER DETECTADOS BEM PERTO DA FASE DE IMPLEMENTAÇÃO. A FIGURA 1.6 ILUSTRA O MODELO EM CASCATA. 214

COM O OBJETIVO DE TENTAR CONTORNAR OS PROBLEMAS DO MODELO EM CASCATA, FOI DESENVOLVIDO O MODELO V, QUE FOCA NOS TESTES DO PRODUTO DURANTE TODO O CICLO DE DESENVOLVIMENTO PARA CONSEGUIR UMA DETECÇÃO ADIANTADA DE DEFEITOS. A IDÉIA É QUE AS ATIVIDADES DE TESTES NÃO SÃO SIMPLESMENTE UMA FASE ÚNICA, MAS PELO CONTRÁRIO, COMO JÁ FOI VISTO NA SESSÃO ANTERIOR, SE FAZ NECESSÁRIA TODA UMA PREPARAÇÃO, PASSANDO POR ETAPAS DE PLANEJAMENTO, ANÁLISE, PROJETO, ETC, QUE DEVEM SER EXECUTADAS EM PARALELO COM AS ATIVIDADES DE DESENVOLVIMENTO. 215

TODOS OS ARTEFATOS GERADOS PELOS DESENVOLVEDORES E ANALISTAS DE NEGÓCIO DURANTE O DESENVOLVIMENTO, PROVÊEM A BASE DE TESTES EM UM OU MAIS NÍVEIS. PROMOVENDO AS ATIVIDADES DE TESTE MAIS CEDO, DEFEITOS PODEM SER GERALMENTE ENCONTRADOS NOS DOCUMENTOS DA BASE DE TESTES. O MODELO V DEMONSTRA COMO AS ATIVIDADES DE VERIFICAÇÃO E VALIDAÇÃO PODEM SER EXECUTADAS EM CONJUNTO COM CADA FASE DO CICLO DE VIDA DE DESENVOLVIMENTO DO SOFTWARE. 215

1.3 INSPEÇÃO DE SOFTWARE 216

COMO EXPLICADO NA SESSÃO ANTERIOR, A INSPEÇÃO DE SOFTWARE É UMA TÉCNICA ESTÁTICA DO PROCESSO DE V & V, EM QUE SÃO EFETUADAS REVISÕES NO SISTEMA COM O OBJETIVO DE ENCONTRAR DEFEITOS E ENTÃO, CORRIGI-LOS. O OBJETIVO PRINCIPAL DAS INSPEÇÕES É GARANTIR QUE DEFEITOS SEJAM REPARADOS O MAIS CEDO POSSÍVEL NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE, UMA VEZ QUE QUANTO MAIS TARDE, MAIS DIFÍCIL DE SE ENCONTRAR OS ERROS E MAIS CUSTOSO AINDA CONSERTÁ-LOS. QUALQUER ARTEFATO PRODUZIDO NO DESENVOLVIMENTO DO SOFTWARE PODE SER UTILIZADO NO PROCESSO DE INSPEÇÃO, COMO REQUISITOS, MODELO DE PROJETO OU CÓDIGO. **216**

O MODELO CMMI EXIGE A REALIZAÇÃO DE REVISÕES COMO UMA PRÁTICA ESPECÍFICA DO PROCESSO DE VERIFICAÇÃO, DEMONSTRANDO ASSIM SUA IMPORTÂNCIA NA GARANTIA DA QUALIDADE DO PRODUTO. SEGUNDO FAGAN, [FAGAN 1986] A UTILIZAÇÃO DE INSPEÇÕES INFORMAIS DE SOFTWARE CAPTURAM EM TORNO DE 60% DOS ERROS EM UM PROGRAMA. MILLS ET AL. [MILLS ET AL. 1987] SUGERE QUE UMA APLICAÇÃO MAIS FORMAL DE INSPEÇÃO DE SOFTWARE PODE DETECTAR ATÉ MAIS DE 90% DOS ERROS DE UM PROGRAMA. SELBY E BASILI [SELBY ET AL. 1987] COMPARAM EMPIRICAMENTE A EFETIVIDADE DE INSPEÇÕES E TESTES. ELES PERCEBERAM QUE A REVISÃO DE CÓDIGO ESTÁTICA SE MOSTRAVA MAIS EFETIVA E MENOS CARA DO QUE A PROCURA POR ERROS UTILIZANDO TESTES. **216**

1.3.1. A EQUIPE DE INSPEÇÃO (PARTICIPANTES) 217

A EQUIPE DE INSPEÇÃO É COMPOSTA POR UM PEQUENO GRUPO DE PESSOAS QUE POSSUAM INTERESSE E CONHECIMENTO DO PRODUTO. GERALMENTE O TAMANHO DA EQUIPE VARIA DE QUATRO A SETE PARTICIPANTES, E O NÚMERO MÍNIMO É DE TRÊS PESSOAS. EQUIPES MAIORES SÃO NORMALMENTE UTILIZADAS PARA ANALISAR DOCUMENTOS DE MAIS ALTO NÍVEL DO PRODUTO, ENQUANTO QUE TIMES MENORES SÃO PREFERÍVEIS AO SE INSPECIONAR DETALHES MAIS TÉCNICOS. UMA BOA VARIEDADE DE INSPETORES, REPRESENTANDO DIFERENTES ÁREAS DE CONHECIMENTO, É INTERESSANTE PARA O PROCESSO DE INSPEÇÃO. O PAPEL DE CADA PARTICIPANTE SERÁ EXPLICADO ABAIXO. **217**

- AUTOR – É O CRIADOR (DESENVOLVEDOR) DO ARTEFATO QUE SERÁ INSPECIONADO. SUAS PRINCIPAIS RESPONSABILIDADES SÃO: CORRIGIR OS PROBLEMAS DETECTADOS DURANTE O PROCESSO DE INSPEÇÃO, PROVER UMA VISÃO GERAL DO PRODUTO AOS DEMAIS PARTICIPANTES E TIRAR QUAISQUER DÚVIDAS QUE SURGIREM COM RELAÇÃO AO ARTEFATO DESENVOLVIDO. **217**

- INSPETOR – EXAMINA O PRODUTO ANTES DA REUNIÃO DE INSPEÇÃO (FASE DE PREPARAÇÃO) E DURANTE DE MODO A TENTAR ENCONTRAR DEFEITOS. PODE TAMBÉM IDENTIFICAR PROBLEMAS AMPLOS QUE ESTÃO FORA DO ESCOPO DA EQUIPE DE INSPEÇÃO, COMO TAMBÉM SUGERIR MELHORIAS. **217**

- LEITOR – PESSOA RESPONSÁVEL POR APRESENTAR O ARTEFATO AOS DEMAIS PARTICIPANTES DO PROCESSO DE INSPEÇÃO DURANTE A REUNIÃO. UMA PESSOA QUE USARÁ O PRODUTO NUMA PRÓXIMA ETAPA DO SEU CICLO DE VIDA É UM CANDIDATO FORTE PARE ESTA TAREFA, UMA VEZ QUE A ATIVIDADE DE LER SOBRE O PRODUTO IRÁ PERMITIR A ESTE POTENCIAL USUÁRIO SE TORNAR BASTANTE FAMILIAR COM O PRODUTO. **217**

- ESCRITOR – TEM O PAPEL DE REGISTRAR AS INFORMAÇÕES SOBRE CADA DEFEITO ENCONTRADO DURANTE A REUNIÃO, QUE INCLUEM: A LOCALIZAÇÃO DO DEFEITO, UM RESUMO DO PROBLEMA, SUA CLASSIFICAÇÃO E UMA IDENTIFICAÇÃO DO INSPETOR QUE O ENCONTROU. TODAS AS DECISÕES E RECOMENDAÇÕES FEITAS TAMBÉM SÃO REGISTRADAS. **217**

- MODERADOR – O MODERADOR TEM O PAPEL MAIS CRÍTICO NO PROCESSO DE INSPEÇÃO E POR ESTE MOTIVO FAZ-SE NECESSÁRIO UM TREINAMENTO MAIS APROFUNDADO DO QUE OS OUTROS MEMBROS DA EQUIPE. ELE É A PESSOA QUE LIDERA TODA A EQUIPE E PARTICIPA ATIVAMENTE DE TODAS AS ETAPAS. DENTRE SUAS PRINCIPAIS RESPONSABILIDADES PODEMOS DESTACAR: SELECIONAR E LIDERAR A EQUIPE DE INSPEÇÃO, DISTRIBUIR O MATERIAL A SER INSPECIONADO, AGENDAR AS REUNIÕES, ATUAR COMO MODERADOR NOS ENCONTROS, SUPERVISIONAR A CORREÇÃO DOS DEFEITOS, E EMITIR RELATÓRIO DE INSPEÇÃO. UMA OUTRA RESPONSABILIDADE MUITO IMPORTANTE DO MODERADOR É GARANTIR QUE O FOCO DA REUNIÃO SE MANTENHA EM ENCONTRAR FALHAS NO PRODUTO, E NÃO EM ACUSAR O AUTOR DOS PROBLEMAS ENCONTRADOS. **217**

1.3.2. O PROCESSO DE INSPEÇÃO DE SOFTWARE (ETAPAS) 217

O PROCESSO TRADICIONAL DE INSPEÇÃO DE SOFTWARE [FAGAN 1976] É DEFINIDO POR SEIS ESTÁGIOS, CADA UM REPRESENTADO POR SEU PRINCIPAL RESPONSÁVEL. A FIGURA 1.8 ILUSTRA ESTA SEQÜÊNCIA DE ETAPAS E EM SEGUIDA CADA UMA DAS ETAPAS SERÁ EXPLICADA DETALHADAMENTE. **217**

- PLANEJAMENTO: O MODERADOR É A PESSOA RESPONSÁVEL POR ESTA ETAPA. O PLANEJAMENTO ENVOLVE SELECIONAR A EQUIPE, CHECAR SE O PRODUTO ESTÁ PRONTO PARA INSPEÇÃO, ORGANIZAR A REUNIÃO, DELEGAR AS ATIVIDADES DE CADA MEMBRO E GARANTIR A COMPLETUDE DOS MATERIAIS A SEREM INSPECIONADOS. NESTA ETAPA O MODERADOR TAMBÉM DEVE VERIFICAR SE O MATERIAL A SER INSPECIONADO POSSUI UM TAMANHO ADEQUADO PARA UMA ÚNICA REUNIÃO. CASO CONTRARIO, O MATERIAL DEVERÁ SER DIVIDIDO EM TAMANHOS MENORES, COM INSPEÇÕES A SEREM REALIZADAS PARA CADA UMA DESTAS PARTES. **219**

- VISÃO GERAL: NESTA ETAPA O AUTOR APRESENTA O PRODUTO AOS DEMAIS MEMBROS DA EQUIPE, DESCREVENDO O QUE O PROGRAMA É SUPOSTO FAZER. O MODERADOR É RESPONSÁVEL POR DECIDIR SE ESTA ETAPA SE FAZ REALMENTE NECESSÁRIA, POIS SE A EQUIPE JÁ FOR BEM FAMILIARIZADA COM

O MATERIAL A SER INSPECIONADO OU NOVAS TÉCNICAS NÃO ESTEJAM SENDO APLICADAS, ESTE ESTÁGIO É DISPENSÁVEL. **219**

- PREPARAÇÃO: ESTE É O MOMENTO EM QUE CADA MEMBRO DO TIME DE INSPEÇÃO ESTUDA INDIVIDUALMENTE A ESPECIFICAÇÃO E O PROGRAMA A SER INSPECIONADO, E PROCURA POR DEFEITOS NO MATERIAL. TODOS OS POSSÍVEIS DEFEITOS DEVEM SER REGISTRADOS NUM *LOG* DE PREPARAÇÃO, ASSIM COMO O TEMPO QUE FOI GASTO NA PREPARAÇÃO. O MODERADOR É ENCARREGADO DE ANALISAR OS *LOGS* ANTES DA REUNIÃO DE INSPEÇÃO PARA DETERMINAR SE A EQUIPE ESTÁ PREPARADA PARA SUAS TAREFAS, E CASO CONTRARIO, ELE PODE REMARCAR A REUNIÃO. **219**

- REUNIÃO: NESTA ETAPA, O PASSO A PASSO PRINCIPAL CONSISTE NA LEITURA E INTERPRETAÇÃO DO PRODUTO, PELO LEITOR; EM SEGUIDA O AUTOR TIRA QUAISQUER DÚVIDAS QUE EVENTUALMENTE SURGIREM COM RELAÇÃO AO MATERIAL, E A EQUIPE DE INSPETORES ENTÃO IDENTIFICAM OS POSSÍVEIS DEFEITOS. ESTA REUNIÃO DEVE SER CURTA, NÃO PODENDO PASSAR MAIS DO QUE DUAS HORAS, E DEVE SER FOCADA NA DETECÇÃO DE DEFEITOS, CONFORMIDADE COM O PADRÃO E PROGRAMAÇÃO DE MÁ QUALIDADE. O TIME DE INSPEÇÃO NÃO DEVE DISCUTIR COMO ESTES DEFEITOS PODERIAM SER CORRIGIDOS E NEM SUGERIR MUDANÇAS EM OUTROS COMPONENTES. **219**

- RE-TRABALHO: O PROPÓSITO DO RE-TRABALHO É CORRIGIR OS DEFEITOS IDENTIFICADOS DURANTE A REUNIÃO DE INSPEÇÃO. O AUTOR É A PESSOA RESPONSÁVEL POR ESSAS CORREÇÕES, DEVENDO CORRIGIR EM PRIMEIRO LUGAR OS DEFEITOS CONSIDERADOS MAIS RELEVANTES E GRAVES, E CORRIGINDO OS DE MENOR IMPORTÂNCIA APENAS SE O TEMPO PERMITIR. **219**

- ACOMPANHAMENTO: AQUI O MODERADOR DEVE DECIDIR SE UMA NOVA INSPEÇÃO É NECESSÁRIA OU NÃO. ELE DEVE ANALISAR O MATERIAL CORRIGIDO PELOS AUTORES E VERIFICAR SE OS DEFEITOS FORAM CORRIGIDOS COM SUCESSO. O MODERADOR PODE INCLUIR REVISORES ADICIONAIS NESTA ETAPA SE FOREM NECESSÁRIOS CONHECIMENTOS TÉCNICOS EXTRAS. SE TODOS OS PROBLEMAS MAIS RELEVANTES FOREM RESOLVIDOS, TODOS OS PROBLEMAS EM ABERTO SOLUCIONADOS, E O PRODUTO SATISFIZER AOS CRITÉRIOS DE SAÍDA, O MODERADOR APROVA O *RELEASE* DO PRODUTO. SE AS CONDIÇÕES NÃO FORAM ATINGIDAS, AINDA SERÁ NECESSÁRIO MAIS UM TEMPO NA ETAPA DE RE-TRABALHO. **219**

1.3.3. FERRAMENTAS DE APOIO AO PROCESSO DE INSPEÇÃO 219

BASEADO NA CLASSIFICAÇÃO DE *GROUPWARE* (SOFTWARES VOLTADOS PARA O APOIO A ATIVIDADES DE TRABALHO EM GRUPO) E NAS CONSTANTES MUDANÇAS TECNOLÓGICAS, [HEDBERG 2004] IDENTIFICOU QUATRO GERAÇÕES DE FERRAMENTAS DE INSPEÇÃO DE SOFTWARE: **220**

1. PRIMEIRAS FERRAMENTAS *(EARLY TOOLS*) **220**

2. FERRAMENTAS DISTRIBUÍDAS (*DISTRIBUTED TOOLS*) **220**

3. FERRAMENTAS ASSÍNCRONAS (ASYNCHROUNOUS TOOLS) **220**

4. FERRAMENTAS BASEADAS EM WEB (WEB-BSED TOOLS) **220**

NOTA-SE QUE AS PRIMEIRAS FERRAMENTAS A SURGIREM FORAM CLASSIFICADAS COMO PRIMEIRAS FERRAMENTAS, NO INÍCIO DA DÉCADA DE 90 E LOGO EM SEGUIDA VIERAM AS FERRAMENTAS DISTRIBUÍDAS. NO FIM DA DÉCADA DE 90 SURGIRAM AS FERRAMENTAS PARA INTERNET. **220**

AS FERRAMENTAS DA PRIMEIRA GERAÇÃO SÃO AQUELAS QUE APENAS PERMITEM O TRABALHO DE TODA A EQUIPE NO MESMO AMBIENTE E AO MESMO TEMPO (INSPEÇÕES SÍNCRONAS). A SEGUNDA JÁ PERMITE QUE A EQUIPE POSSA TRABALHAR DE FORMA DISTRIBUÍDA, OU SEJA, EM LUGARES DIFERENTES, PORÉM AINDA É PRECISO QUE SEJA AO MESMO TEMPO (INSPEÇÕES DISTRIBUÍDAS). A TOTAL INDEPENDÊNCIA DE TEMPO E LUGAR FOI INTRODUZIDA NA TERCEIRA GERAÇÃO, COM AS FERRAMENTAS ASSÍNCRONAS. AS FERRAMENTAS DA QUARTA GERAÇÃO TAMBÉM SÃO ASSÍNCRONAS, DIFERENCIANDO-SE DAS DEMAIS DEVIDO A SUA BASE TECNOLÓGICA. **220**

A SEGUIR SERÁ APRESENTADA UMA FERRAMENTA REPRESENTANTE DE CADA GERAÇÃO INTRODUZIDA ANTERIORMENTE. A FERRAMENTA ICICLE REPRESENTARÁ A GERAÇÃO DE PRIMEIRAS FERRAMENTAS. EM SEGUIDA A FERRAMENTA SCRUTINY EXEMPLIFICARÁ AS FERRAMENTAS DISTRIBUÍDAS. ASSIST ILUSTRARÁ AS FERRAMENTAS ASSÍNCRONAS, E FINALMENTE, IBIS SERÁ A REPRESENTANTE DAS FERRAMENTAS BASEADAS EM WEB. **220**

- ICICLE – O ICICLE (*INTELLIGENT CODE INSPECTION ENVIRONMENT IN A C LANGUAGE ENVIRONMENT*) É O PRIMEIRO SOFTWARE DE REVISÃO PUBLICADO E VISA APOIAR O PROCESSO TRADICIONAL DE INSPEÇÃO DE SOFTWARE. COMO O PRÓPRIO NOME JÁ SUGERE, ELE FOI DESENVOLVIDO PARA O CONTEXTO ESPECÍFICO DE INSPEÇÃO DE CÓDIGO C E C++, PODENDO SER USADO PARA O AUXÍLIO DA INSPEÇÃO DO CÓDIGO, TANTO NAS FASES DE PREPARAÇÃO INDIVIDUAL COMO NAS REUNIÕES EM GRUPO. A REUNIÃO DE INSPEÇÃO EM GRUPO DEVE SER REALIZADA NO MESMO LOCAL E A INSPEÇÃO INDIVIDUAL PERMITE ENTRAR COM COMENTÁRIOS EM CADA LINHA DE CÓDIGO. A FERRAMENTA NÃO SE APLICA A INSPEÇÕES MAIS GENÉRICAS, LIMITANDO O TIPO DE ARTEFATO A SER INSPECIONADO E A TÉCNICA DE DETECÇÃO DE DEFEITOS, MAS PODE, ENTRETANTO, SER UTILIZADA PARA INSPECIONAR LINHAS DE TEXTO NUMA ANÁLISE INICIAL. UM DOS PRINCIPAIS OBJETIVOS DESTA FERRAMENTA É O DE AJUDAR OS INSPETORES DE CÓDIGO A ENCONTRAREM DEFEITOS ÓBVIOS. **220**

- SCRUTINY – O SCRUTINY É UMA FERRAMENTA COLABORATIVA *ONLINE*, SENDO A PRIMEIRA A PERMITIR QUE OS MEMBROS DO TIME DE INSPEÇÃO SE ENCONTRASSEM DISPERSOS GEOGRAFICAMENTE, PODENDO SER USADA TANTO

OUTRAS FERRAMENTAS E CUSTOMIZADA PARA APOIAR DIFERENTES PROCESSOS DE DESENVOLVIMENTO. ATUALMENTE APENAS SUPORTA INSPEÇÕES DE TEXTOS. A FERRAMENTA É BASEADA NUM PROCESSO DE INSPEÇÃO DIVIDIDO EM QUATRO ETAPAS. NO PRIMEIRO ESTÁGIO, DE INICIAÇÃO, O MODERADOR DISPONIBILIZA O DOCUMENTO A SER INSPECIONADO NA FERRAMENTA. NO PRÓXIMO ESTÁGIO, PREPARAÇÃO, OS INSPETORES INSEREM SEUS COMENTÁRIOS A SEREM DISCUTIDOS NA REUNIÃO. DEPOIS, NA FASE DE RESOLUÇÃO, O MODERADOR GUIA OS INSPETORES ATRAVÉS DOS DOCUMENTOS E DOS DEFEITOS COLETADOS. FINALMENTE, NO ESTÁGIO DE FINALIZAÇÃO, APÓS AS DISCUSSÕES E ACORDOS REFERENTES AOS DEFEITOS LEVANTADOS, A FERRAMENTA FORNECE UM RESUMO DOS DEFEITOS QUE FORAM DISCUTIDOS. **220**

- ASSIST – *ASYNCHRONOUS/ SYNCHRONOUS SOFTWARE INSPECTION SUPPORT TOOL* FOI DESENVOLVIDA PARA PROVER INSPEÇÕES INDIVIDUAIS E EM GRUPO. COMO O NOME SUGERE, PERMITE INSPEÇÕES SÍNCRONAS E ASSÍNCRONAS, COM REUNIÕES TANTO EM LOCAIS DIFERENTES COMO NO MESMO AMBIENTE. UTILIZA UMA LINGUAGEM DE DEFINIÇÃO DE PROCESSO DE INSPEÇÃO (IPDL) E UM SISTEMA FLEXÍVEL PARA O TIPO DE DOCUMENTO INSPECIONADO, PERMITINDO O SUPORTE A QUALQUER TIPO DE PROCESSO DE INSPEÇÃO DE SOFTWARE. INSPEÇÃO DE CÓDIGO, COLETAS DE DADOS PARA MÉTRICAS E CÁLCULOS PARA APOIO AS INSPEÇÕES TAMBÉM ESTÃO PRESENTES NESTA FERRAMENTA. É BASEADA NUMA ARQUITETURA CLIENTE/SERVIDOR, EM QUE O SERVIDOR É USADO COMO UM REPOSITÓRIO CENTRAL DE DOCUMENTOS E OUTROS TIPOS DE DADO. UM *BROWSER C++* PODE AUTOMATICAMENTE APRESENTAR ITENS RELEVANTES DE *CHECKLIST* PARA A SESSÃO DE CÓDIGO INSPECIONADO. **221**

- IBIS – *INTERNET-BASED INSPECTION SYSTEM* É UMA FERRAMENTA BASEADA EM WEB COM NOTIFICAÇÕES POR *EMAIL* QUE AUXILIA NO PROCESSO DE INSPEÇÃO DESENVOLVIDO POR FAGAN. PERMITE QUE AS INSPEÇÕES SEJAM REALIZADAS ENTRE PESSOAS GEOGRAFICAMENTE DISTRIBUÍDAS E POSSUI UMA INTERFACE BASTANTE LEVE E AMIGÁVEL, TENDO TODA SUA ESTRUTURA E DADOS ARMAZENADOS EM ARQUIVOS XML. ELA NÃO LIMITA O TIPO DE ARTEFATO A SER INSPECIONADO E PROVÊ SUPORTE A DECISÕES, APOIO A ANOTAÇÕES E *CHECKLISTS*. AS PRINCIPAIS VANTAGENS DESTA FERRAMENTA SÃO: POR SER BASEADA EM WEB, PERMITE QUE OS INSPETORES ACESSEM A APLICAÇÃO DE SEUS PRÓPRIOS COMPUTADORES; PERMITE QUE A INSPEÇÃO SEJA REALIZADA COM INTEGRANTES DA EQUIPE DISTRIBUÍDOS EM LOCAIS DIFERENTES, ATÉ MESMO EM PAÍSES DIFERENTES; PERMITE QUE ESPECIALISTAS DIFERENTES PARTICIPEM DA REUNIÃO, PODENDO SER ESPECIALISTAS DE OUTRO DEPARTAMENTO OU MESMO FORA NA ORGANIZAÇÃO. **221**

1.4. MODELOS DE MATURIDADE DE TESTES DE SOFTWARE 221

PARA SE CONSTRUIR SOFTWARE COM QUALIDADE, É NECESSÁRIO QUE SE TENHA UM PROCESSO DE TESTES BEM DEFINIDO E QUE ELE ESTEJA ALINHADO AO PROCESSO DE DESENVOLVIMENTO. NESTA SEÇÃO SERÃO VISTOS TRÊS

MODELOS DE MATURIDADE DE TESTE DE SOFTWARE, OS QUAIS INDICAM COMO CRIAR E/OU MELHORAR O PROCESSO DE TESTES. 221

1.4.1. PROCESSO DE MELHORIA DE TESTES – TPI 221

1.4.1.1. ESCOPO DO TPI 222

1.4.1.2. ÁREAS CHAVE 223

1.4.1.3 PASSOS PARA IMPLANTAR A MELHORIA 224

1.4.2. TMM – TEST MATURITY MODEL 225

1.4.2.1. NÍVEIS DE MATURIDADE DO TMM 226

1.4.3. TIM – TEST IMPROVEMENT MODEL 227

1.4.3.1. MODELO DE MATURIDADE 227

1.4.3.2. ÁREAS CHAVE 229

CONSIDERAÇÕES FINAIS 232

O DESENVOLVIMENTO DE SOFTWARE ENGLOBA UM MERCADO DE EXTREMA COMPETITIVIDADE. TENDO EM VISTA QUE OS SISTEMAS QUE APRESENTAM MELHOR QUALIDADE GARANTEM SEU ESPAÇO NO MERCADO, AS EMPRESAS QUE OS DESENVOLVEM TÊM INVESTIDO BASTANTE ESFORÇO PARA ASSEGURAR A QUALIDADE DE SEUS PRODUTOS E GARANTIR A SATISFAÇÃO DOS CLIENTES. A QUALIDADE DE UM PRODUTO PODE SER DEFINIDA COMO SUA CAPACIDADE DE CUMPRIR OS REQUISITOS INICIALMENTE ESTIPULADOS PELOS CLIENTES, E SENDO ASSIM, ESTÁ DIRETAMENTE RELACIONADA À QUALIDADE DO PROCESSO DE DESENVOLVIMENTO. POR ESTE MOTIVO, TEM SURGIDO UMA GRANDE DEMANDA AO INCENTIVO DE PESQUISAS QUE LEVEM EM CONSIDERAÇÃO À PROCURA POR FORMAS DE MELHORIA DA QUALIDADE DOS PRODUTOS. 232

ESTE CAPÍTULO PROCUROU INTRODUZIR AO LEITOR BOAS PRÁTICAS NO QUE DIZ RESPEITO À QUALIDADE DOS PRODUTOS, APRESENTANDO UM CONJUNTO DE NORMAS QUE REPRESENTAM A PADRONIZAÇÃO MUNDIAL PARA AVALIAÇÃO DA QUALIDADE DE PRODUTOS DE SOFTWARE. AS ATIVIDADES DE TESTE E INSPEÇÃO TAMBÉM FORAM DESTACADAS COMO FORMA DE ENCONTRAR DEFEITOS NO SOFTWARE E CORRIGI-LOS A TEMPO, ANTES DE ENTREGAR O PRODUTO A SEUS CLIENTES, E ANALISAR SE O SISTEMA FAZ O QUE É SUPOSTO FAZER. FINALMENTE, MODELOS DE MATURIDADE DE TESTES FORAM APRESENTADOS COMO MAIS UMA TENTATIVA DE ATINGIR OS OBJETIVOS DESEJADOS, BUSCANDO MELHORIAS NA QUALIDADE DO PROCESSO

DE TESTE DE SOFTWARE, QUE AFETA DIRETAMENTE A QUALIDADE DO PRODUTO. 232

EXERCÍCIOS 233

1. QUAIS SÃO AS DIRETRIZES PARA USO DA NORMA NBR ISO/IEC 9126-1? 233

2. A QUE SE PROPÕE A NORMA ISO 12119? 233

3. QUE SUBDIVISÕES DA NORMA ISO 14598 ESTABELECEM ITENS NECESSÁRIOS PARA O SUPORTE À AVALIAÇÃO? 233

4. QUAIS SÃO OS COMPONENTES DO PROJETO SQUARE? DEFINA-OS. 233

5. QUAL A DIFERENÇA ENTRE TESTES E INSPEÇÕES DE SOFTWARE? 233

6. CITE 5 TIPOS DE TESTES E EXPLIQUE CADA UM DELES. 233

7. QUAIS OS ESTÁGIOS DE TESTES POSSÍVEIS E QUAIS AS CARACTERÍSTICAS DE CADA UM DELES? 233

8. O QUE SÃO TESTES BETA? 233

9. O QUE SÃO TESTES DE REGRESSÃO? 233

10. QUAL A DIFERENÇA ENTRA A ABORDAGEM DE CAIXA PRETA E A ABORDAGEM DE CAIXA BRANCA? 233

11. QUAIS SÃO OS PAPÉIS EXISTENTES NA EQUIPE DE INSPEÇÃO DE SOFTWARE E QUAIS SUAS RESPONSABILIDADES? 233

12. QUAIS SÃO AS ETAPAS DO PROCESSO DE INSPEÇÃO DE SOFTWARE? EXPLIQUE CADA UMA DELAS. 233

13. EXPLIQUE COMO É FEITA A IMPLANTAÇÃO DA MELHORIA NO TPI. 233

14. DEFINA OS NÍVEIS DE MATURIDADE DO TMM. 233

15. NO ASPECTO ORGANIZAÇÃO, COMO SÃO CARACTERIZADOS OS NÍVEIS DE MATURIDADE DO TIM? 233

SUGESTÕES DE LEITURA 234

PARA CONHECER MAIS SOBRE NORMAS DE QUALIDADE DE PRODUTO DE SOFTWARE, É RECOMENDADA A LEITURA DO LIVRO TECNOLOGIA DA INFORMAÇÃO: QUALIDADE DE PRODUTO DE SOFTWARE, GUERRA & COLOMBO 2009. **234**

PARA AMPLIAR O ENTENDIMENTO SOBRE O ASSUNTO DE TESTE DE SOFTWARE É RECOMENDADA A LEITURA DO LIVRO *FOUNDATIONS OF SOFTWARE TESTING*, GRAHAM, D., VEENENDAAL, E. V., EVANS, I. AND BLACK, R., 2007. ESTE LIVRO É UTILIZADO POR PESSOAS QUE DESEJAM TIRAR O CERTIFICADO ISTQB (*INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD)*, PORTANTO, É MUITO INTERESSANTE PARA ADQUIRIR MELHORES CONHECIMENTOS SOBRE ESTE CONTEÚDO. **234**

PARA UM MELHOR CONHECIMENTO SOBRE OS CONCEITOS E O PROCESSO DE INSPEÇÃO DE SOFTWARE É SUGERIDA A LEITURA DE *DESIGN AND CODE INSPECTION TO REDUCE ERRORS IN PROGRAM DEVELOPMENT*, FAGAN, M.E.,1976. **234**

PARA SE APROFUNDAR MAIS SOBRE AS FERRAMENTAS DE INSPEÇÃO DE SOFTWARE É RECOMENDADA A LEITURA DE *MODERN SOFTWARE REVIEW TECHNIQUES AND TECHNOLOGIES*, WONG, Y. K., 2006. **234**

PARA MELHOR CONHECIMENTO SOBRE O TPI (*TEST PROCESS IMPROVEMENT*) É RECOMENDADA A LEITURA DO LIVRO *TEST PROCESS IMPROVEMENT A PRACTICAL STEP-BY-STEP GUIDE TO STRUCTURED TESTING*, KOOMEN & POL, 1999. **234**

PARA APROFUNDAR A LEITURA SOBRE TMM (*TEST MATURITY MODEL*), É SUGERIDA A LEITURA DO LIVRO *A MODEL TO ASSESS TESTING PROCESS MATURITY*, BURNSTEIN & GROM, 1998.
TÓPICOS DE PESQUISA **234**

TÓPICOS DE PESQUISA **235**

O TMM (*TESTING MATURITY MODE*), COMO VISTO ANTERIORMENTE, FOI DESENVOLVIDO PELO *ILLINOIS INSTITUTY OF TECNOLOGY* COMO UM GUIA PARA MELHORIA DE PROCESSOS DE TESTES. EXISTE UMA ORGANIZAÇÃO SEM FINS LUCRATIVOS, EM DUBLIN – IRLANDA, QUE FOI FUNDADA PARA TENTAR TRANSFORMAR O MODELO TMM EM UMA NORMA E, CONSEQUENTEMENTE, PROMOVER A SUA ACEITAÇÃO COMO UM PADRÃO DA INDÚSTRIA INTERNACIONAL DE AVALIAÇÃO E DE ORGANIZAÇÕES DE TESTE DE SOFTWARE. **235**

A FUNDAÇÃO TMMI TEM OS SEGUINTES OBJETIVOS: **235**

- DEFINIÇÃO DE UM NÚCLEO TMMI PARA A PADRONIZAÇÃO INTERNACIONAL **235**
- CRIAÇÃO E GESTÃO DE UMA ORGANIZAÇÃO INDEPENDENTE, IMPARCIAL COM REPOSITÓRIO CENTRAL DE DADOS E PRESTAÇÃO DE SERVIÇOS **235**
- MÉTODOS DE AVALIAÇÃO COM BASE NO MODELO PADRÃO **235**

- PRESTAÇÃO DE UM MECANISMO INDEPENDENTE PARA FACILITAR A VERIFICAÇÃO FORMAL DE AVALIAÇÕES TMMI **235**

- DEFINIÇÃO E MANUTENÇÃO DE AVALIADOR INDEPENDENTE **235**

- PRESTAÇÃO DE UM FÓRUM PÚBLICO DAS PARTES INTERESSADAS PARA FACILITAR A LIVRE TROCA DE INFORMAÇÃO, EDUCAÇÃO, IDÉIAS E USO DA NORMA PÚBLICA. **235**

O SITE OFICIAL DA FUNDAÇÃO É **HTTP://WWW.TMMIFOUNDATION.ORG**. **235**

EXISTEM VÁRIOS ESTUDOS ATUALMENTE NA ACADEMIA NO QUE DIZ RESPEITO À SELEÇÃO DE TESTES DE REGRESSÃO, UMA VEZ QUE EXECUTAR TODOS OS CASOS DE TESTE NOVAMENTE SEMPRE QUE UMA NOVA VERSÃO DO SISTEMA FOR LIBERADA É UMA PRÁTICA INVIÁVEL. DESSA FORMA, VÁRIAS PESQUISAS E PROPOSTAS DE SOLUÇÕES E TÉCNICAS PARA REALIZAR UMA QUANTIDADE SUFICIENTE DE TESTES QUE ATINJA A COBERTURA NECESSÁRIA PARA GARANTIR A CORRETUDE DO SOFTWARE PODEM SER ENCONTRADAS NA LITERATURA. **235**

OUTRA ÁREA DE PESQUISA BASTANTE DESAFIADORA NA ÁREA DE TESTE DE SOFTWARE É A GERAÇÃO AUTOMÁTICA DE CASOS DE TESTE, CONSIDERANDO QUE A ELABORAÇÃO DE CASOS TESTES MANUALMENTE É UM PROCESSO QUE CONSOME MUITO TEMPO E ESFORÇO. SENDO ASSIM, DIVERSAS PROPOSTAS SÃO ELABORADAS DIA APÓS DIA COM O OBJETIVO DE TORNAR O PROCESSO DE TESTE MAIS ÁGIL, MENOS SUSCEPTÍVEL A ERROS E DEPENDENTE DA INTERAÇÃO HUMANA. **235**

NA ÁREA DE INSPEÇÃO DE SOFTWARE, GRANDES DESAFIOS PODEM SER OBSERVADOS COM O OBJETIVO DE ENCONTRAR ESTRATÉGIAS PARA DIMINUIR A QUANTIDADE DE DEFEITOS DE UM SOFTWARE. NA LITERATURA, PODEM SER ENCONTRADAS PESQUISAS E ARTIGOS COM ESTUDOS FOCADOS NESTE OBJETIVO. **235**

REFERÊNCIAS **236**

BOEHM, B. W. AND BASILI, V.R. (2001) "SOFTWARE DEFECT REDUCTION TOP 10 LIST.", IEEE COMPUTER 34 (1), P. 135-137. **236**

GUERRA, A., C AND COLOMBO, R., M., T. (2009) "TECNOLOGIA DA INFORMAÇÃO: QUALIDADE DE PRODUTO DE SOFTWARE", PBQP SOFTWARE. **236**

FAGAN, M.E. (1976) "DESIGN AND CODE INSPECTION TO REDUCE ERRORS IN PROGRAM DEVELOPMENT", IBM SYSTEMS JOURNAL, VOL. 15, NO. 3, P. 182-211.**236**

HEDBERG, H. (2004) "INTRODUCING THE NEXT GENERATION OF SOFTWARE INSPECTION TOOLS", IN: INTERNATIONAL CONFERENCE OF PRODUCT FOCUSED SOFTWARE PROCESS IMPROVEMENT, 5, KANSAI. LECTURE NOTES IN COMPUTER

GRAHAM, D., VEENENDAAL, E. V., EVANS, I. AND BLACK, R. “FOUNDATIONS OF SOFTWARE TESTING”, ISTQB CERTIFICATION, THOMSON LEARNING, 2007. **236**

WONG, Y. K. “MODERN SOFTWARE REVIEW TECHNIQUES AND TECHNOLOGIES”, IRM PRESS, 2006. **236**

SELBY, R. W. AND BASILI, V. R., ET AL. (1987). “CLEANROOM SOFTWARE DEVELOPMENT: AN EMPIRICAL EVALUATION”, IEEE TRANS ON SOFTWARE ENGINEERING , SE-13(9), 102-37. (CHS. 4,22) **236**

MILLS, H. D. AND DYER, M., ET AL. (1987). “CLEANROOM SOFTWARE ENGINEERING”, IEEE SOFTWARE, 4(5), 19-25. (CHS. 3,4,22) **236**

11.2.2. ÁREAS DE CONHECIMENTO **240**
11.2.2.1. REQUISITOS DE SOFTWARE **241**
11.2.2.5. MANUTENÇÃO DE SOFTWARE 245
11.2.2.8. PROCESSO DE ENGENHARIA DE SOFTWARE 250
11.2.2.9. MÉTODOS E FERRAMENTAS DE ENGENHARIA 252
11.2.2.10. QUALIDADE DE SOFTWARE 254

REFERÊNCIAS **259**

VISÃO GERAL DA COMUNICAÇÃO **296**

14.1. PROCESSO DA COMUNICAÇÃO **297**

14.1.1. A COMUNICAÇÃO **297**

ATRAVÉS DAS HIERARQUIAS DE AUTORIDADE E ORIENTAÇÕES FORMAIS. **297**

INTEGRAÇÃO SOCIAL DENTRO DE GRUPOS SATISFAZENDO AS NECESSIDADES SOCIAIS. **297**

FORNECE SUBSÍDIOS PARA FACILITAR A TOMADA DE DECISÃO **298**

14.1.4. A COMUNICAÇÃO EM ORGANIZAÇÕES **300**

ATUALMENTE, O AMBIENTE ORGANIZACIONAL É CARACTERIZADO POR MUDANÇAS CONTÍNUAS, ASSIM, SURGINDO A NECESSIDADE DE MUDANÇA NOS MODELOS TRADICIONAIS DAS PRÁTICAS DA COMUNICAÇÃO ORGANIZACIONAL PARA MANTER A COMPETITIVIDADE EMPRESARIAL. **300**

14.1.5. COMUNICAÇÃO EM PROJETOS **300**

14.1.6. A COMUNICAÇÃO COMO DESAFIO PARA O GERENTE DE PROJETOS 302

14.2. GERENCIAMENTO DE COMUNICAÇÃO EM PROJETOS 303

14.2.1.1. ENTRADAS PARA O PLANEJAMENTO DAS COMUNICAÇÕES: 305

14.2.1.2. FERRAMENTAS E TÉCNICAS PARA O PLANEJAMENTO DAS COMUNICAÇÕES: 308

14.2.1.3. SAÍDAS DO PLANEJAMENTO DAS COMUNICAÇÕES: 309

***1. PLANO DE GERENCIAMENTO DAS COMUNICAÇÕES* 309**

TEMPLATE DO PLANO DE COMUNICAÇÃO 309

1. INTRODUÇÃO 309

2. NECESSIDADES DE INFORMAÇÃO 309

3. TIPOS DE INFORMAÇÃO 310

4. FORMATOS (TEMPLATES DE RELATÓRIOS) 310

5. GLOSSÁRIO 310

14.2.2. DISTRIBUIÇÃO DAS INFORMAÇÕES 310

14.2.3. RELATÓRIO DE DESEMPENHO 314

14.2.4. GERENCIAR AS PARTES INTERESSADAS 319

EM PROJETOS DISTRIBUÍDOS, A COMUNICAÇÃO É A BASE PARA DEFINIR COMO SERÃO REPASSADAS AS INFORMAÇÕES PARA AS PARTES INTERESSADAS ENVOLVIDAS NO PROJETO. NÃO EXISTE UMA REGRA PARA GERENCIAR PROJETOS DISTRIBUÍDOS, MAS EXISTEM BOAS PRÁTICAS QUE SÃO PONTOS RELEVANTES E QUE AJUDAM OS PROJETOS A CHEGAREM A SEU OBJETIVO FUNDAMENTAL: SUA CONCLUSÃO NO PRAZO, DENTRO DO CUSTO E COM QUALIDADE. NA LITERATURA, PODEM SER ENCONTRADAS PESQUISAS E ARTIGOS COM ESTUDOS FOCADOS NESTE ASSUNTO. **323**

REFERÊNCIAS 324

ALVES, A. A COMUNICAÇÃO NA GERÊNCIA DO PROJETO. REVISTA: TECHOJE: UMA REVISTA DE OPINIÃO. DISPONÍVEL EM: HTTP://WWW.IETEC.COM.BR/SITE/TECHOJE/CATEGORIA/ DETALHE_ARTIGO/101. ACESSADO EM: SET. 2009. **324**

ARCANJO, C. (2008). CONTEXTO DA COMUNICAÇÃO NAS ORGANIZAÇÕES. DISPONÍVEL EM: HTTP://WWW.WEBARTIGOS.COM/ARTICLES/5381/1/CONTEXTO-DA-COMUNICACAO-NA-GESTAO-DAS-ORGANIZACOES/PAGINA1.HTML. ACESSADO EM: OUT. 2009. **324**

BARBOSA, L. O DESAFIO DA COMUNICAÇÃO EFICAZ NO GERENCIAMENTO DE PROJETOS. REVISTA: TECHOJE: UMA REVISTA DE OPINIÃO. DISPONÍVEL EM: HTTP://WWW.IETEC.COM.BR/SITE/TECHOJE/CATEGORIA/DETALHE_ARTIGO/61. ACESSO EM: SET. 2009. **324**

CARVALHO, M.; MIRANDOLA, D. A COMUNICAÇÃO EM PROJETOS DE TI: UMA ANÁLISE COMPARATIVA DAS EQUIPES DE SISTEMAS E DE NEGÓCIOS, V.17 N.2, SÃO PAULO MAIO/AGO. 2007. DISPONÍVEL:HTTP://WWW.SCIELO.BR/SCIELO.PHP?SCRIPT=SCI_ARTTEXT&PID=S0103-6513200700 0200009&LNG=PT&NRM=ISO&TLNG=PT. ACESSADO EM: OUT. 2009. 324

CASTELO, L. GERÊNCIA PARTICIPATIVA: A COMUNICAÇÃO E O GERENTE. DISPONÍVEL EM: HTTP://WWW.GERANEGOCIO.COM.BR/HTML/GERAL/GP4.HTML. ACESSADO EM: SET. 2009. **325**

JACOB, M. IMPORTÂNCIA DA COMUNICAÇÃO NA GERÊNCIA DE PROJETOS.REVISTA: TECHOJE: UMA REVISTA DE OPINIÃO. DISPONÍVEL EM: HTTP://WWW.IETEC.COM.BR/SITE/ TECHOJE/CATEGORIA/DETALHE_ARTIGO/100. ACESSADO EM: SET. 2009. **325**

PIMENTA, J. A COMUNICAÇÃO NAS EMPRESAS E EM PROJETOS. REVISTA: TECHOJE: UMA REVISTA DE OPINIÃO. DISPONÍVEL EM: HTTP://WWW.IETEC.COM.BR/SITE/TECHOJE/CATEGORIA/ DETALHE_ARTIGO/691. ACESSADO EM: OUT. 2009. **326**

PMI (PROJECT MANAGEMENT INSTITUTE) A GUIDE TO THE PROJECT MANAGEMENT BODY OF KNOWLEDGE – GUIA PMBOK® 4. ED. UPPER DARBY, 2008. **326**

RIVAS, M. PLANEJAMENTO & COMUNICAÇÃO PARA ESTABELECER UM DIFERENCIAL COMPETITIVO. REVISTA: TECHOJE: UMA REVISTA DE OPINIÃO. DISPONÍVEL EM: HTTP://WWW.IETEC.COM.BR/SITE/ TECHOJE/CATEGORIA/DETALHE_ARTIGO/379. ACESSADO EM: SET. 2009. **3**

27

SCHNEIDER, G. (2008) O GERENTE DE PROJETOS TAMBÉM CUIDA DA COMUNICAÇÃO. WEBINSIDER. DISPONÍVEL EM: HTTP://WEBINSIDER.UOL.COM.BR/INDEX.PHP/2008/11/05/O-GERENTE-DE-PROJETOS-TAMBEM-CUIDA-DA-COMUNICACAO/. ACESSADO EM: SET. 2009. **327**

15.1. IMPORTÂNCIA DA MEDIÇÃO ERROR! BOOKMARK NOT DEFINED.

15.2. O QUE SÃO MÉTRICAS ERROR! BOOKMARK NOT DEFINED.

REFERÊNCIAS ERROR! BOOKMARK NOT DEFINED.

16 GESTÃO DE PROGRAMAS ERROR! BOOKMARK NOT DEFINED.

PROGRAMAS ERROR! BOOKMARK NOT DEFINED.

GERENCIAMENTO DE PROGRAMAS **ERROR! BOOKMARK NOT DEFINED.**
RELAÇÃO ENTRE GERENCIAMENTO DO PROGRAMA E GERENCIAMENTO DO PROJETO **ERROR! BOOKMARK NOT DEFINED.**
TEMAS DO GERENCIAMENTO DE PROGRAMA **ERROR! BOOKMARK NOT DEFINED.**
GERENCIAMENTO DE BENEFÍCIOS **ERROR! BOOKMARK NOT DEFINED.**
GERENCIAMENTO DE *STAKEHOLDERS* **ERROR! BOOKMARK NOT DEFINED.**
GOVERNANÇA **ERROR! BOOKMARK NOT DEFINED.**
CICLO DE VIDA DO PROGRAMA **ERROR! BOOKMARK NOT DEFINED.**
FASE 1: SET UP PRÉ-PROGRAMA **ERROR! BOOKMARK NOT DEFINED.**
FASE 2: SET UP PROGRAMA **ERROR! BOOKMARK NOT DEFINED.**
FASE 3: ESTABELECER ESTRUTURA DE GESTÃO DO PROGRAMA **ERROR! BOOKMARK NOT DEFINED.**
FASE 4: BENEFÍCIOS INCREMENTAIS **ERROR! BOOKMARK NOT DEFINED.**
FASE 5: ENCERRAMENTO **ERROR! BOOKMARK NOT DEFINED.**
PROCESSOS DO GERENCIAMENTO DE PROGRAMA **ERROR! BOOKMARK NOT DEFINED.**
GRUPO PROCESSOS DE INICIAÇÃO **ERROR! BOOKMARK NOT DEFINED.**
GRUPO PROCESSOS DE PLANEJAMENTO **ERROR! BOOKMARK NOT DEFINED.**
GRUPO PROCESSOS DE EXECUÇÃO **ERROR! BOOKMARK NOT DEFINED.**
GRUPO PROCESSOS DE MONITORAMENTO E CONTROLE **ERROR! BOOKMARK NOT DEFINED.**
GRUPO PROCESSOS DE ENCERRAMENTO **ERROR! BOOKMARK NOT DEFINED.**

TÓPICOS DE PESQUISA ERROR! BOOKMARK NOT DEFINED.

SUGESTÕES DE LEITURA ERROR! BOOKMARK NOT DEFINED.

EXERCÍCIOS ERROR! BOOKMARK NOT DEFINED.

REFERÊNCIAS ERROR! BOOKMARK NOT DEFINED.

GESTÃO DE PORTFÓLIO DE PROJETOS ERROR! BOOKMARK NOT DEFINED.

INTRODUÇÃO ERROR! BOOKMARK NOT DEFINED.
DEFINIÇÃO DE PORTFÓLIO ERROR! BOOKMARK NOT DEFINED.
ESTRATÉGIA CORPORATIVA E GESTÃO DE PORTFÓLIO ERROR! BOOKMARK NOT DEFINED.
GESTÃO DE PORTFÓLIO VERSUS GESTÃO DE MÚLTIPLOS PROJETOS ERROR! BOOKMARK NOT DEFINED.
RELAÇÃO ENTRE A GESTÃO DE PORTFÓLIO E A GESTÃO DE PROJETOS/PROGRAMAS ERROR! BOOKMARK NOT DEFINED.
MÉTRICAS EM GESTÃO DE PORTFÓLIO ERROR! BOOKMARK NOT DEFINED.
GERENTE DE PORTFÓLIO ERROR! BOOKMARK NOT DEFINED.
MODELOS E PADRÕES DE GESTÃO DE PORTFÓLIO ERROR! BOOKMARK NOT DEFINED.
PADRÃO DE GESTÃO DE PORTFÓLIO [PMI 2006] **ERROR! BOOKMARK NOT DEFINED.**
PROCESSO STAGE-GATE [COOPER ET AL 2001] **ERROR! BOOKMARK NOT DEFINED.**
PROCESSO INTEGRADO DE SELEÇÃO E PRIORIZAÇÃO DE PROJETOS [ARCHER AND GHASEMZADEH 1999] **ERROR! BOOKMARK NOT DEFINED.**
ESTUDO DE CASO: GESTÃO DE PORTFÓLIO DE PROJETOS NO SERPRO ERROR! BOOKMARK NOT DEFINED.
SUGESTÕES DE LEITURA ERROR! BOOKMARK NOT DEFINED.
TÓPICOS DE PESQUISA (TRABALHOS FUTUROS E CORRENTES) ERROR! BOOKMARK NOT DEFINED.

O IMPACTO DA GESTÃO DE PORTFÓLIO DE PROJETOS EM PROJETOS DE TECNOLOGIA DA INFORMAÇÃO [REYCK ET AL 2005] **ERROR! BOOKMARK NOT DEFINED.**
PORTFOLIUS: UM MODELO DE GESTÃO DE PORTFÓLIO DE PROJETOS DE SOFTWARE [CORREIA 2005] **ERROR! BOOKMARK NOT DEFINED.**
SELEÇÃO DE PROJETOS EM UM PORTFÓLIO PARA APOIO A TOMADA DE DECISÃO [GHASEMZADEH AND ARCHER 2000] **ERROR! BOOKMARK NOT DEFINED.**
UM PROCESSO INTEGRADO PARA SELEÇÃO DE PROJETOS EM UM PORTFÓLIO [ARCHER AND GHASEMZADEH 1999] **ERROR! BOOKMARK NOT DEFINED.**
EXERCÍCIOS ERROR! BOOKMARK NOT DEFINED.
REFERÊNCIAS BIBLIOGRÁFICAS ERROR! BOOKMARK NOT DEFINED.

– **INTRODUÇÃO** ERROR! BOOKMARK NOT DEFINED.

– **PAPÉIS E FUNÇÕES** ERROR! BOOKMARK NOT DEFINED.

– **OBJETIVOS DE UM PMO** ERROR! BOOKMARK NOT DEFINED.

– **CLASSIFICAÇÕES DOS PMOS** ERROR! BOOKMARK NOT DEFINED.

– **BOAS PRÁTICAS NA IMPLANTAÇÃO DE PMOS** ERROR! BOOKMARK NOT DEFINED.

– **CASO DE SUCESSO NA IMPLANTAÇÃO DE UM PMO** ERROR! BOOKMARK NOT DEFINED.

– **O SERVIÇO FEDERAL DE PROCESSAMENTO DE DADOS – SERPRO** ERROR! BOOKMARK NOT DEFINED.

– **MOTIVAÇÃO** ERROR! BOOKMARK NOT DEFINED.

– **IMPLANTAÇÃO** ERROR! BOOKMARK NOT DEFINED.

– **ESTRATÉGIA DA IMPLANTAÇÃO** ERROR! BOOKMARK NOT DEFINED.

– **FASES DA IMPLANTAÇÃO** ERROR! BOOKMARK NOT DEFINED.

– **BENEFÍCIOS ALCANÇADOS** ERROR! BOOKMARK NOT DEFINED.

– **MELHORIA CONTÍNUA** ERROR! BOOKMARK NOT DEFINED.

– **TÓPICOS DE PESQUISA** ERROR! BOOKMARK NOT DEFINED.

– **SUGESTÕES DE LEITURA** ERROR! BOOKMARK NOT DEFINED.

DURANTE A CONSTRUÇÃO DESTE CAPÍTULO FORAM IDENTIFICADOS ALGUMAS SUGESTÕES DE LEITURA INTERESSANTES QUE PODEM AJUDAR O LEITOR A MELHOR COMPREENDER O CONTEXTO DE ESCRITÓRIO DE PROJETOS. ESTAS LEITURAS SÃO LISTADAS A SEGUIR: ERROR! BOOKMARK NOT DEFINED.

– **EXERCÍCIOS** ERROR! BOOKMARK NOT DEFINED.

– **QUAL DAS OPÇÕES ABAIXO NÃO FAZ PARTE DAS TÍPICAS FUNÇÕES DE UM PMO?** ERROR! BOOKMARK NOT DEFINED.

– **REPORTAR STATUS DOS PROJETOS PARA GERENTES SUPERIORES.** ERROR! BOOKMARK NOT DEFINED.

– **DESENVOLVER E PADRONIZAR UMA METODOLOGIA PADRONIZADA.** ERROR! BOOKMARK NOT DEFINED.

– **MONITORAR E CONTROLAR O DESEMPENHO DOS PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **DESENVOLVER E MANTER UM PAINEL DE CONTROLE DE PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **AUTORIZAR INVESTIMENTOS PARA A ORGANIZAÇÃO.** ERROR! BOOKMARK NOT DEFINED.

– **QUAL DOS GRUPOS DE FUNÇÕES ABAIXO NÃO FAZ PARTE DO GRUPO DEFINIDO NA PESQUISA DE HOBBS E AUBRY?** ERROR! BOOKMARK NOT DEFINED.

– **MONITORAMENTO E CONTROLE DO DESEMPENHO DO PROJETO.** ERROR! BOOKMARK NOT DEFINED.

– **FINANCIAMENTO DE RECURSOS.** ERROR! BOOKMARK NOT DEFINED.

– **DESENVOLVIMENTO DAS COMPETÊNCIAS E METODOLOGIAS EM GERENCIAMENTO DE PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **GERENCIAMENTO DE MÚLTIPLOS PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **GERENCIAMENTO ESTRATÉGICO.** ERROR! BOOKMARK NOT DEFINED.

– **QUAIS DOS OBJETIVOS APRESENTADOS ABAIXO NÃO PODEM SER CITADOS COMO OBJETIVOS DE UM PMO?** ERROR! BOOKMARK NOT DEFINED.

– **CONTRATAÇÃO DE PESSOAL.** ERROR! BOOKMARK NOT DEFINED.

– **O APOIO NA COMUNICAÇÃO EFICIENTE ENTRE OS GERENTES DE PROJETO E A ALTA ADMINISTRAÇÃO.** ERROR! BOOKMARK NOT DEFINED.

– **MELHORA NA EFICIÊNCIA DO PLANEJAMENTO E CONDUÇÃO DOS PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **ORIENTAR E DAR SUPORTE AOS GERENTES DE PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **UNIFORMIZAR PROCESSOS, PRÁTICAS E OPERAÇÕES DE GERENCIAMENTO DE PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **DE ACORDO COM KERZNER, OS TRÊS TIPOS DE PMO'S SÃO:** ERROR! BOOKMARK NOT DEFINED.

– **ESCRITÓRIO DE PROJETOS FUNCIONAL, ESCRITÓRIO DE PROJETOS DE GRUPO DE CLIENTES E ESCRITÓRIO DE PROJETOS CORPORATIVO.** ERROR! BOOKMARK NOT DEFINED.

– **ESCRITÓRIO DE PROJETOS NÍVEL 1, ESCRITÓRIO DE PROJETOS NÍVEL 2 E DE ESCRITÓRIO DE PROJETOS NÍVEL 3.** ERROR! BOOKMARK NOT DEFINED.

– **ESCRITÓRIO DE PROJETOS LOCAL, ESCRITÓRIO DE PROJETOS REGIONAL E CORPORATIVO.** ERROR! BOOKMARK NOT DEFINED.

– **EQUIPE DE PROJETO AUTÔNOMA, ESCRITÓRIO DE SUPORTE DE PROJETOS E CENTRO DE EXCELÊNCIA EM GESTÃO DE PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

– **CENTRO DE EXCELÊNCIA EM GESTÃO DE PROJETOS, ESCRITÓRIO DE GERÊNCIA DE PROGRAMAS E *CHIEF PROJECT OFFICER*.** ERROR! BOOKMARK NOT DEFINED.

– **SEGUNDO DINSMORE CINCO SÃO OS TIPOS DE PMO, SENDO ELES:** ERROR! BOOKMARK NOT DEFINED.

– **ESCRITÓRIO DE PROJETO MASTER, ESCRITÓRIO DE SUPORTE DE PROJETOS, CENTRO DE EXCELÊNCIA EM GESTÃO DE PROJETOS, ESCRITÓRIO DE GERÊNCIA DE PROGRAMAS E *CHIEF PROJECT OFFICER* .** ERROR! BOOKMARK NOT DEFINED.

– **EQUIPE DE PROJETO AUTÔNOMA, ESCRITÓRIO DE SUPORTE DE PROJETOS, CENTRO DE EXCELÊNCIA EM GESTÃO DE PROJETOS, ESCRITÓRIO DE GERÊNCIA DE PROGRAMAS E *CHIEF PROJECT OFFICER*.** ERROR! BOOKMARK NOT DEFINED.

– **EQUIPE DE PROJETO AUTÔNOMA, ESCRITÓRIO DE SUPORTE DE PROJETOS, CENTRO DE EXCELÊNCIA EM GESTÃO DE PROJETOS, ESCRITÓRIO DE GERÊNCIA DE PROGRAMAS E CENTRO DE PROJETOS.** ERROR! BOOKMARK NOT DEFINED.

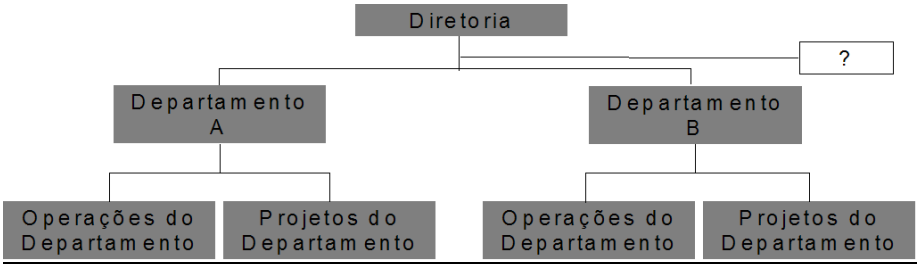
– **EQUIPE DE PROJETO AUTÔNOMA, ESCRITÓRIO DE SUPORTE DE PROJETOS, CENTRO DE EXCELÊNCIA EM GESTÃO DE PROJETOS, ESCRITÓRIO DE CONTROLE DE PROJETOS E *CHIEF PROJECT OFFICER*.** ERROR! BOOKMARK NOT DEFINED.

– **EQUIPE DE PROJETO AUTÔNOMA, ESCRITÓRIO DE SUPORTE DE PROJETOS, ESCRITÓRIO ESTRATÉGICO DE PROJETOS, ESCRITÓRIO DE GERÊNCIA DE PROGRAMAS E *CHIEF PROJECT OFFICER*.** ERROR! BOOKMARK NOT DEFINED.

– **DE ACORDO COM CRAWFORD PODEMOS CLASSIFICAR OS PMO'S EM 3 (TRÊS) NÍVEIS DENOMINADOS:** ERROR! BOOKMARK NOT DEFINED.

– **ANALISANDO A FIGURA ABAIXO, A QUAL REPRESENTA UM ORGANOGRAMA ORGANIZACIONAL DE UMA EMPRESA FICTÍCIA, QUE TIPO ESCRITÓRIO DE PROJETOS MELHOR SE ENQUADRARIA SEGUNDO A CAIXA DESTACADA EM CINZA?** ERROR! BOOKMARK NOT DEFINED.

– **ANALISANDO A FIGURA ABAIXO, A QUAL REPRESENTA UM OUTRO ORGANOGRAMA ORGANIZACIONAL DE UMA SEGUNDA EMPRESA FICTÍCIA, QUE TIPO ESCRITÓRIO DE PROJETOS MELHOR SE ENQUADRARIA SEGUNDO A CAIXA DESTACADA EM CINZA?** ERROR! BOOKMARK NOT DEFINED.



ERROR! BOOKMARK NOT DEFINED.

REFERÊNCIAS ERROR! BOOKMARK NOT DEFINED.

14.1. INTRODUÇÃO A MATURIDADE EM GESTÃO DE PROJETOS ERROR! BOOKMARK NOT DEFINED.

14.2. MODELOS DE MATURIDADE EM GESTÃO DE PROJETOS ERROR! BOOKMARK NOT DEFINED.

14.2.1. ORGANIZATIONAL PROJECT MANAGEMENT MATURITY MODEL - PMI ERROR! BOOKMARK NOT DEFINED.
14.2.2. PROJECT MANAGEMENT MATURITY MODEL – PM SOLUTIONS ERROR! BOOKMARK NOT DEFINED.
14.2.3. MODELO DE MATURIDADE EM GERENCIAMENTO DE PROJETOS – DARCI PRADO ERROR! BOOKMARK NOT DEFINED.
14.2.4. PORTFOLIO, PROGRAMME AND PROJECT MANAGEMENT MATURITY MODEL – OGC ERROR! BOOKMARK NOT DEFINED.
14.2.5. KERZNER PROJECT MANAGEMENT MATURITY MODEL – HAROLD KERZNER ERROR! BOOKMARK NOT DEFINED.

14.3. OPM3 ERROR! BOOKMARK NOT DEFINED.

14.3.1. ESTRUTURA DO MODELO ERROR! BOOKMARK NOT DEFINED.
14.3.2. AVALIAÇÃO DA MATURIDADE ERROR! BOOKMARK NOT DEFINED.
14.3.3. IMPLANTAÇÃO DO MODELO ERROR! BOOKMARK NOT DEFINED.

14.4 MMGP ERROR! BOOKMARK NOT DEFINED.

14.4.1. ESTRUTURA DO MODELO ERROR! BOOKMARK NOT DEFINED.
14.4.2. AVALIAÇÃO DA MATURIDADE ERROR! BOOKMARK NOT DEFINED.
14.4.3. IMPLANTAÇÃO DO MODELO ERROR! BOOKMARK NOT DEFINED.

14.5. KPMMM ERROR! BOOKMARK NOT DEFINED.

14.5.1. ESTRUTURA DO MODELO ERROR! BOOKMARK NOT DEFINED.
14.5.2. AVALIAÇÃO DA MATURIDADE ERROR! BOOKMARK NOT DEFINED.
14.5.3. IMPLANTAÇÃO DO MODELO ERROR! BOOKMARK NOT DEFINED.

14.6. UM ESTUDO DE CASO ERROR! BOOKMARK NOT DEFINED.

14.6.1. METODOLOGIA ERROR! BOOKMARK NOT DEFINED.
14.6.2. RESULTADOS COLETADOS ERROR! BOOKMARK NOT DEFINED.
14.6.3. PERFIL DOS PARTICIPANTES ERROR! BOOKMARK NOT DEFINED.
14.6.4. SEGMENTAÇÃO POR NÍVEL DE MATURIDADE ERROR! BOOKMARK NOT DEFINED.
14.6.5. SEGMENTAÇÃO POR PERCENTUAL DE ADERÊNCIA AOS NÍVEIS DE MATURIDADE ERROR! BOOKMARK NOT DEFINED.
14.6.6. CONCLUSÃO ERROR! BOOKMARK NOT DEFINED.

14.7. ANÁLISE COMPARATIVA ERROR! BOOKMARK NOT DEFINED.

14.8. SUGESTÕES DE LEITURA ERROR! BOOKMARK NOT DEFINED.

14.9. TÓPICOS DE PESQUISA ERROR! BOOKMARK NOT DEFINED.

14.10. EXERCÍCIOS ERROR! BOOKMARK NOT DEFINED.

REFERÊNCIAS ERROR! BOOKMARK NOT DEFINED.

GOVERNANÇA EM TIC **139**

○ **GESTÃO EM TIC** **139**

RELEVÂNCIA E EVOLUÇÃO DO PAPEL DA TIC NAS ORGANIZAÇÕES 141
DA GESTÃO À GOVERNANÇA EM TIC 144

○ **MODELOS DE GESTÃO EM TIC** **147**

COBIT 147
ITIL 148
BSC 148
IT FLEX 148
COSO 149
ISO/IEC 20000 150
VAL IT 151
CMMI SOB A PERSPECTIVA DE GOVERNANÇA DE TI 152

○ **ITIL** **152**

DEFINIÇÃO 152
HISTÓRICO 152
REGULAMENTAÇÃO DO ITIL 153
- CERTIFICAÇÕES / TREINAMENTOS 153
- DIREITOS AUTORAIS 154
- PUBLICAÇÃO DE CONTEÚDOS OFICIAIS 154
- FÓRUM DE FOMENTO (ITSMF) 155

ESTRUTURA DO ITIL 156
- *SERVICE STRATEGY* (ESTRATÉGIA DE SERVIÇOS) 156
- *SERVICE DESIGN* (PLANEJAMENTO DE SERVIÇOS) 156
- *SERVICE TRANSITION* (TRANSIÇÃO DE SERVIÇOS) 156
- *SERVICE OPERATION* (OPERAÇÃO DE SERVIÇOS) 156
- *CONTINUAL SERVICE IMPROVEMENT* (APRIMORAMENTO CONTÍNUO DE SERVIÇOS) 156

O QUE NÃO É ITIL 158
FRONTEIRAS COM OUTROS MODELOS E LIMITAÇÕES 159
PONTO DE PARTIDA 160
COMENTÁRIOS SOBRE PRÁTICAS DE SUCESSO 161
PÚBLICO ALVO 162
UTILIZAÇÃO DO ITIL 163

DEFINIÇÃO 164
HISTÓRICO 165
REGULAMENTAÇÃO DO COBIT 166
- CERTIFICAÇÕES / TREINAMENTOS 166
- DIREITOS AUTORAIS 167
- PUBLICAÇÃO DE CONTEÚDOS OFICIAIS 167
- FÓRUM DE FOMENTO (ISACA) 168

ESTRUTURA DO COBIT 168
- PRIMEIRA DIMENSÃO DO CUBO – PROCESSOS DE TI 169
- SEGUNDA DIMENSÃO DO CUBO – CRITÉRIOS DE INFORMAÇÃO 171
- TERCEIRA DIMENSÃO DO CUBO – RECURSOS DE TI 172

NÃO É COBIT 173
FRONTEIRAS COM OUTROS MODELOS 174
PONTO DE PARTIDA 175
COMENTÁRIOS SOBRE PRÁTICAS DE SUCESSO 176
PÚBLICO ALVO 176
UTILIZAÇÃO DO COBIT 177

- **INICIATIVAS DE INTEGRAÇÃO DOS PRINCIPAIS MODELOS 178**
- **IMPLANTAÇÃO DE MODELOS DE GESTÃO 179**
- **TÓPICOS DE PESQUISA 182**
- **SUGESTÕES DE LEITURA 183**
- **EXERCÍCIOS 185**
- **REFERÊNCIAS 187**

Parte 1

PROCESSOS

Índice do Capítulo

3.1. INTRODUÇÃO A PROCESSOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE102
3.2. O MANIFESTO ÁGIL103
3.3. PRINCIPAIS PROCESSOS ÁGEIS104
3.4 EXTREME PROGRAMMING (XP)106
3.4.1 VALORES, PRINCÍPIOS E PRÁTICAS DE XP106
3.4.2 PAPÉIS DOS INTEGRANTES108
3.4.3 CICLO DE VIDA109
3.6. SCRUM109
3.6.1 CARACTERÍSTICAS DO SCRUM110
3.6.2 PAPÉIS DO SCRUM110
3.6.3 PRÁTICAS DO SCRUM111
3.6.4 CICLO DE VIDA DO SCRUM111
3.7 FEATURE DRIVEN DEVELOPMENT111
3.7.1 CARACTERÍSTICAS DO FDD112
3.5.2 PAPÉIS DO FDD112
3.5.3 PRÁTICAS DO FDD114
3.5.4 CICLO DE VIDA DO FDD115
9.7. CONSIDERAÇÕES FINAIS117
9.8. TÓPICOS DE PESQUISA118
9.9. SUGESTÕES DE LEITURA118
9.11. EXERCÍCIOS118
REFERÊNCIAS118

Comment [wis1]: Formatar o índice

Capítulo 3

Processos Ágeis de Desenvolvimento de Software

Márcio Amorim de Medeiros, Milton Moura Campos Neto

Este capítulo apresenta uma nova abordagem de desenvolvimento de software, os Processos Ágeis, que surgiram para reduzir os problemas e custos em comparação aos Processos Tradicionais. Neste capítulo são citadas as principais características das Metodologias Ágeis em geral, com ênfase na Extreme Programming (XP), Scrum e Feature Driven Development (FDD).

3.1. Introdução a Processos Ágeis de Desenvolvimento de Software

Os processos tradicionais de desenvolvimento de software geralmente não se adéquam à realidade de algumas organizações, em especial, as pequenas e médias fábricas de software que não possuem recursos para seguirem processo algum. Os processos ágeis surgiram como uma nova tendência de desenvolvimento para melhorar a qualidade dos sistemas e reduzir a quantidade de projetos fracassados, eliminando gastos com documentação excessiva, enfatizando a comunicação, mais flexível à mudança e privilegiando as atividades que agregam valor ao negócio.

Tanto os métodos pesados, tanto os ágeis possuem o mesmo objetivo, satisfazer as necessidades dos usuários construindo sistemas de qualidade. A diferença entre eles está nos princípios utilizados por cada um. [SATO 2007] Os princípios relacionados aos processos tradicionais são citados no **Capítulo 1**, já os ágeis são detalhados na seção **O Manifesto Ágil** deste capítulo.

Comment [wis2]: Acho que não cabe aqui.Melhor mudar para Quanto.

Comment [wis3]: Ficou Vago.

Atualmente, mudança é algo bastante comum na vida de um software, a fim de garantir adaptação do sistema às novas necessidades do cliente, instituições ou do mercado. Os processos tradicionais tendem a tentar planejar grande parte do software por um longo período antes de iniciar a implementação. Com isso, o software demora a ser disponibilizado ao cliente. Durante esse tempo pode surgir novos padrões, políticas e tecnologias que afetam os requisitos do software, o cliente pode perceber que alguma funcionalidade não está conforme solicitado ou precisa de outras. Esses fatores implicam em mudança no sistema, que não é bem-vinda nos métodos tradicionais, pois

Outro fator comum no desenvolvimento tradicional, é a implementação de funcionalidades que não agregará valor ao cliente, ou seja, o sistema vai disponibilizar funcionalidades aos usuários que serão pouca ou nunca utilizada, enquanto outras funções mais prioritárias ainda não foram implementadas.

Comment [wis4]: Começar no mesmo parágrafo

As metodologias ágeis surgiram com a finalidade de desburocratizar o processo de desenvolvimento. Elas tentam se adaptar e fortalecer com as mudanças, até mesmo ao ponto de se auto-modificarem. Os clientes têm, em curto espaço de tempo, versões de software executável, onde são priorizadas as funcionalidades que agregam mais valor ao seu negócio. Com isso, ele já pode sugerir novas funcionalidades e correções.

Outro fator determinante da agilidade é o fato de "não documentar, apenas por documentar". Só é documentado aquilo que for necessário em outro momento e que justifique o esforço e recursos gastos na documentação.

Comment [wis5]: Fala no mesmo assunto

Vale ressaltar que os Processos ágeis são orientados a pessoas ao invés dos tradicionais que são orientados a processos. Metodologias ágeis afirmam que nenhum processo jamais será semelhante à habilidade da equipe de desenvolvimento. Em vista disso, o papel do processo é dar suporte à equipe e seu trabalho.

Nas próximas seções será detalhado desde o Manifesto Ágil, onde serão citados os princípios comuns aos métodos ágeis existentes, e citadas e caracterizadas as principais metodologias.

3.2. O Manifesto Ágil

No início de 2001, 17 especialistas em software reuniram-se para propor um conjunto de princípios e valores para agilizar o desenvolvimento dos seus sistemas tendo como base suas elevadas experiências em programação. Foram motivados pela conclusão de que os processos de desenvolvimento estavam tornando-se cada vez mais longos, atolando as equipes de construção de softwares. Esse movimento, marco inicial do desenvolvimento ágil de software, foi descrito abaixo:

Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar [AGILE MANIFESTO 2009]:

Comment [wis6]: Texto incompleto.

Indivíduos e interação entre eles mais que processos e ferramentas

Software em funcionamento mais que documentação abrangente

Colaboração com o cliente mais que negociação de contratos

Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Assinaram esse manifesto: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith,

Os envolvidos se denominaram de Aliança Ágil. Esta abordagem tentava manter a qualidade dos projetos de software permitindo aos mesmos que mudanças fossem inseridas em seus desenvolvimentos, mas que reduzisse seus impactos, esta flexibilidade foi traduzida nos quatro valores vistos acima e em doze princípios que estão mostrados a seguir:

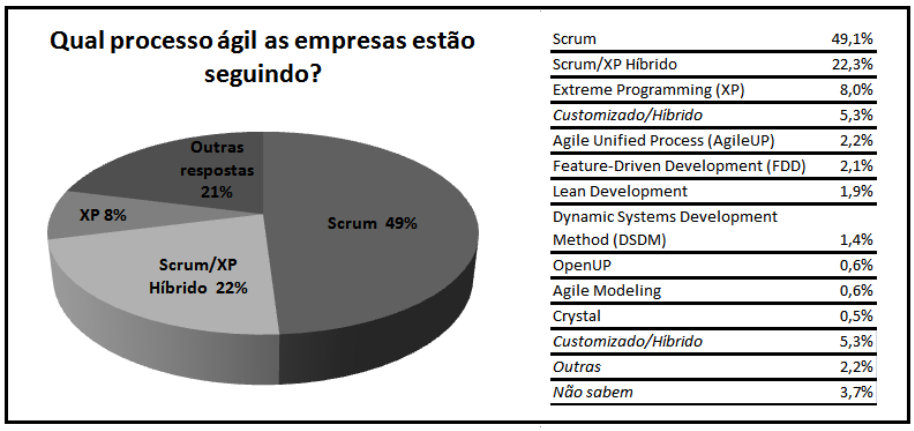
- Nossa maior prioridade é satisfazer o cliente através de entregas antecipadas e contínuas de software de valor ao cliente;
- Mudanças de requisitos são bem vindas, mesmo que tardiamente no desenvolvimento. Processos ágeis se aproveitam da mudança para vantagem competitiva do cliente;
- Entrega freqüente de software funcionando, de duas semanas de trabalho até dois meses, com preferência à escala de tempo mais curta;
- Pessoas de software e negócios devem trabalhar juntas diariamente durante o desenvolvimento do projeto;
- Construa projetos em torno de indivíduos motivados. Dê-los o ambiente e o suporte necessário e acredite neles para fazer o trabalho;
- O Método mais eficiente e efetivo de repassar a informação dentro de uma equipe de desenvolvimento é a conversa face a face
- Software funcionando é a primeira medida de progresso;
- Processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um passo sustentável indefinidamente;
- Atenção contínua a excelência técnica e um bom projeto melhoram a agilidade;
- Simplicidade: a arte de maximizar a quantidade de trabalho não feito, é essencial;
- As melhores arquiteturas, requisitos, e projetos emergem de times auto-organizáveis;
- Em intervalos regulares, o time deve refletir sobre como se tornar mais efetivo, então melhora e ajusta seu comportamento de acordo com a reflexão.

3.3. Principais Processos Ágeis

As metodologias que seguem os princípios do Manifesto Ágil – como entrega freqüente, flexão a mudança, foco em pessoas e simplicidade – são consideradas Processos Ágeis de Desenvolvimento de Software.

Comment [wis7]: Falta complement.

A 3ª pesquisa sobre o estado do Desenvolvimento Ágil promovido pela VersionOne **[VERSIONONE 2008]** e aplicada a mais de três mil entrevistados de 80 países, revela entre diversos indicadores, quais as metodologias ágeis mais utilizadas nas empresas, conforme Figura 3.1.



Scrum	49,1%
Scrum/XP Híbrido	22,3%
Extreme Programming (XP)	8,0%
Customizado/Híbrido	5,3%
Agile Unified Process (AgileUP)	2,2%
Feature-Driven Development (FDD)	2,1%
Lean Development	1,9%
Dynamic Systems Development Method (DSDM)	1,4%
OpenUP	0,6%
Agile Modeling	0,6%
Crystal	0,5%
Customizado/Híbrido	5,3%
Outras	2,2%
Não sabem	3,7%

Figura 3.1. Metodologias Ágeis mais utilizadas nas empresas de acordo com a 3° Pesquisa Anual sobre o Estado do Desenvolvimento Ágil – Ano 2008 Adaptado de [VERSIONONE 2008].

A metodologia *Agile Unified Process* (AUP), também conhecida como AgileUP e desenvolvida por Ambler [AMBLER 2002], é uma versão simplificada do IBM Rational Unified Process (RUP), que estar detalhado no Capítulo 1. Ela descreve como desenvolver sistemas utilizando técnicas ágeis, como *Test Driven Development* (TDD), *Agile Modeling* e *Refactoring* no banco de dados para melhorar a produtividade, mes mo assim mantendo-se fiel ao RUP.

A *Lean Development* tem suas raízes na indústria automotiva, ele é uma adaptação para software do *Lean Manufacturing* do revolucionário Sistema Toyota de Produção. Inicialmente proposto por Bob Charette, tem como principais objetivos tentar reduzir em um terço o prazo, o custo e o nível de defeito no desenvolvimento de software e para isso exige um grande comprometimento da alta administração com predisposição inclusive para mudanças radicais.

Baseado no Desenvolvimento Rápido de Aplicações (RAD) e no modelo iterativo e incremental, a metodologia *Dynamic Systems Development Method* (DSDM) se tornou o framework para RAD. A idéia central do DSDM baseia-se no seguinte: ao invés de fixar o escopo de funcionalidades do produto e a partir daí estimar tempo e recursos para alcançar o escopo definido, é preferível fixar tempo e recursos e ajustar o escopo de acordo com estas limitações [COHEN et al., 2003].

Comment [wis8]: Acho que não coube aqui.Faltou colocar uma frase antes para preparar o leitor sobre o que iria falar.

O Crystal não é apenas uma única metodologia, e sim, uma família de métodos denominada portanto de Família Crystal, proposto por Alistair Cockburn. É organizada por cores de acordo com o número de pessoas envolvidas, o que se traduz em diferentes níveis de ênfase na comunicação. A versão mais ágil e mais documentada é a Crystal Clear que pode ser usada em projetos de até 8 pessoas, seguida pela Crystal Yellow (para times de 8 a 20 pessoas), Crystal Orange (para times de 20 até 50 pessoas), Crystal Red (para times de 50 até 100 pessoas), etc.

Comment [wis9]:

Nas próximas seções deste capítulo serão tratadas com maior detalhamento as

grandes projetos e para empresas que possuem dificuldades para migrar para um ambiente completamente ágil.

3.4 EXTREME PROGRAMMING (XP)

Extreme Programming é considerada uma metodologia leve de desenvolvimento de software, esta metodologia é chamada de leve por que sempre que há mudanças o projeto está leve de documentação para ser modificada. Esta metodologia nasceu com Kent Beck e Ward Cunningham a partir da observação das necessidades decorrentes do surgimento de uma outra opção além daquelas oferecidas pela engenharia de software corrente que não traziam bons resultados.

O termo "Extreme" indica que a metodologia emprega ao extremo as boas práticas de engenharia de software. Esta metodologia tem algumas práticas mais peculiares do que aquelas observadas na metodologia ágil como programação em pares, forte cultura de testes que são automatizados e executados várias vezes ao dia e propriedade de código coletivo, detalharemos esta metodologia a seguir.

3.4.1 Valores, princípios e práticas de XP

Extreme programming é definida através de valores, princípios e práticas. Os valores descrevem os objetivos de longo prazo de quem aplica XP e definem critérios para se obter sucesso. Os quatro valores de XP são:

- **Comunicação –** Parte do insucesso de alguns projetos de software é atribuído a falta de comunicação. Por isto, a metodologia emprega algumas práticas que forçam uma maior comunicação por parte da equipe;
- **Simplicidade –** O mais simples possível que seja funcional. Para isto, deve-se desenvolver pensando apenas no presente. Pois nem sempre o software será modificado;
- **Retorno –** Assim como o valor comunicação, XP estabelece várias práticas para que o retorno freqüente. Retorno do Cliente, andamento do projeto, resultados dos testes tudo é devidamente medido e repassado a toda equipe para que haja uma melhor resposta;
- **Coragem –** A probabilidade de um código ser jogado fora por uma insatisfação do cliente, ou por mudanças de requisitos é muito grande, também é possível ter várias opções de implementação e realizar um pouco de cada uma para escolher a melhor.

Para sustentar estes valores a metodologia define princípios que devem ser seguidos por todos os praticantes:

- **Retorno rápido** – Além de obter retorno este deve ser rápido;
- **Assumir simplicidade** – Os problemas devem ser tratados da forma mais simples possível. XP prega que o tempo gasto pensando em reuso ou resolvendo possíveis problemas futuros pode ser maior do que ter o retrabalho de fazer *refactoring*;
- **Mudança incremental** – Mudanças devem ser feitas pouco a pouco, ou seja, de forma incremental e contínua;

Comment [wis10]: Onde estão as pontuações?

Comment [wis11]: Poderia está no mesmo parágrafo.

Comment [wis12]: Pelo que percebi o pois é uma conjunção coordenada que pode ser conclusiva ou explicativa. Serve como conector entre as duas orações.

Comment [wis13]: Falta complemento.

Comment [wis14]: Acho que está muito sintetizado.

- **Trabalho de qualidade** – Deve-se sempre buscar produzir um trabalho de qualidade, de outra forma a equipe perderá a motivação necessária ao projeto;
- **Ensinar aprendendo** – XP faz com que o praticante aprenda suas próprias medidas daquilo que deve projetar, do quanto se deve testar e de como fazer *refactoring*;
- **Investimento inicialmente pequeno** – Um projeto deve iniciar com poucos recursos e no decorrer, com sucesso do projeto e retorno do cliente, ir aumentando estes recursos. Um projeto pode ser cancelado ou ter seu escopo diminuído, nestes casos é melhor encerrá-lo com um investimento ainda pequeno;
- **Jogar para vencer** – A equipe deve ser vencedora, ou seja obter sucesso ao fim do projeto;
- **Experimentos concretos** – Decisões abstratas devem ser testadas em forma de experimentos;
- **Comunicação aberta e honesta** – Problemas como atrasos, má qualidade de código e insegurança devem ser tratados abertamente pelo grupo de projeto;
- **Trabalhar a favor dos instintos das pessoas** – Visto que os instintos são baseados em seus interesses de curto prazo, como vontade de vencer e aprender a interagir com outras pessoas, deve-se respeitá-lo e utilizá-lo a seu favor;
- **Aceitar responsabilidades** – Responsabilidades devem ser aceitas e não impostas. Isto leva as pessoas a um maior comprometimento com o projeto;
- **Adaptação ao local** – XP deve ser adaptada ao local de trabalho em que será empregada;
- **Viajar leve** – A equipe deve gerar poucos e valiosos artefatos. Como os projetos de XP têm que se adaptar às várias mudanças necessárias em um projeto, a equipe deve estar "leve" de artefatos, documentações e especificações. De outra forma será empregado muito esforço para atualizá-los;
- **Medidas honestas** – As métricas a serem utilizadas devem ser o mais honestas possíveis e refletir as necessidades do projeto.

Comment [wis15]: Acho que deveria mudar o nome da prática.

Extreme Programming define também práticas que tornam a metodologia viável e possível de seguir seus valores e princípios. Abaixo estão listadas todas as práticas que são:

- **O jogo do planejamento** - Deve ser elaborado de forma simples, fácil e rápido o plano para o próximo *release*, que consiste em determinar o escopo baseado nas prioridades do negócio, nas possibilidades e estimativas técnicas;
- ***Releases* pequenos** – Os *releases* devem ser de pouca duração e devem sempre conter software de valor;
- **Metáforas** – Guiam o desenvolvimento do projeto através de uma estória que explique como o sistema funciona;
- **Projeto de software simples** – O sistema deve ser projetado da forma mais simples que possa solucionar o software.
- **Teste** – Há dois estágios de testes definidos: testes unitários e testes de aceitação. Ambos devem ser automatizados. Os testes unitários são feitos pelo programador durante o desenvolvimento. Os testes de aceitação são especificados pelo cliente ditando que deve funcionar para que o software seja

- ***Refactoring*** – Reestruturação de parte do código, sem alterar seu comportamento;
- **Programação em pares** – Todo o código é produzido por pares de programadores, cada par trabalhando na mesma máquina;
- **Propriedade coletiva** – Todos os integrantes da equipe são donos e responsáveis pelo código produzido. Assim, todos podem alterar qualquer parte do código a qualquer momento;
- **Integração contínua** – O código é integrado, o *build* do software é gerado e os testes são executados várias vezes ao dia;
- **40 horas semanais** – Os membros da equipe não devem trabalhar mais que quarenta horas semanais;
- **Cliente no local –** O cliente é parte integrante da equipe e deverá estar presente no local de desenvolvimento sempre que necessário para tirar alguma dúvida ou priorizar alguma estória;
- **Padrão de codificação** – O código é utilizado como fonte de informação para comunicação e discussão entre os membros da equipe. Assim, ele deve estar bem escrito e estruturado, o que requer a conformidade a um padrão de codificação estabelecido para todo projeto.

Comment [wis16]: Colocar em português.

3.4.2 Papéis dos integrantes

XP relaciona papéis, com responsabilidades explícitas, a serem desempenhados em um projeto. Os mais importantes são:

Comment [wis17]: Precisa de mais um incentive ao leitor. Colocar um texto antes.

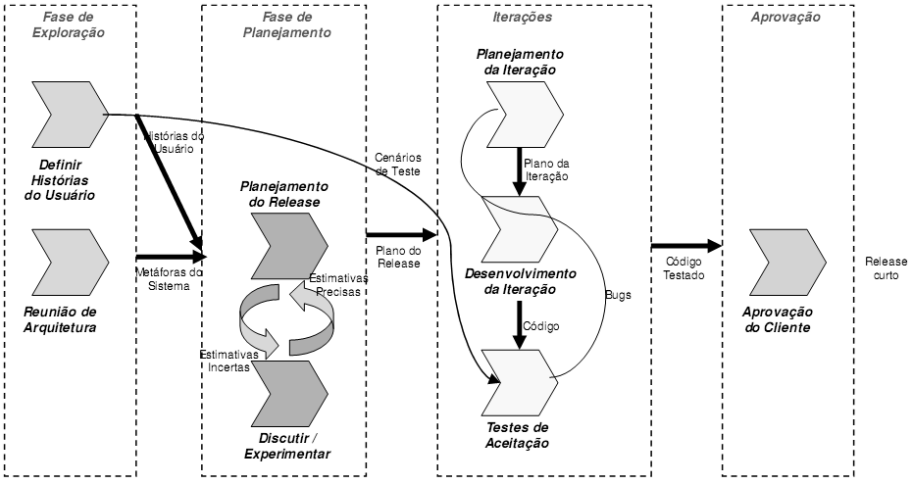
- **Programador** – É o coração de XP. Responsável por projetar o software, codificar, implementar testes e estimar suas tarefas. Todos da equipe assumem este papel no desenvolvimento do projeto;
- **Cliente** – Representante do cliente, responsável por estabelecer prioridades e escopo, escrever estórias, escrever os testes funcionais. Deve estar disponível no local do desenvolvimento sempre que necessário para esclarecer duvidas;
- **Testador –** O testador é responsável por apoiar o cliente a escolher e escrever os testes funcionais, além de assegurar a execução e reportagem dos problemas identificados nestes testes;
- **Rastreador (*tracker*) –** É a consciência da equipe XP. Responsável por reportar as métricas do projeto promovendo visibilidade sobre a acurácia das estimativas e progresso do projeto;
- **Técnico –** Praticamente um gerente de projeto. Responsável por identificar problemas e resolvê-los para que a equipe possa trabalhar da melhor forma. Não requer conhecimentos técnicos profundos, mas, não se deve negar que conhecimento técnico ajuda no papel.
- **Consultor –** Papel que é opcional em uma equipe em XP. Ele age ajudando os outros a resolverem seus próprios problemas. O Consultor é um papel de profundo conhecimento técnico que aparece quando um membro da equipe tem um problema a resolver. Ele ajuda o membro a resolver o problema, joga a

- **Chefe –** O Big Boss tem um papel de gerente sênior, é ele quem toma decisões de mais alto nível e se reporta ao cliente, esse papel não exige conhecimento apesar de se o chefe o tiver é muito valioso. O Chefe precisa de coragem, confiança no time e muita insistência.

Existe também a possibilidade, caso seja desejo da equipe, que os membros possam "trocar de boné" dentro da organização. Ou seja, pessoas diferentes troquem de papel dentro da mesma organização.

3.4.3 Ciclo de Vida

Comment [wis18]: Falta Escrever o ciclo de vida



3.6. Scrum

A metodologia ágil Scrum foi criada em 1996 por Ken Schwaber e Jeff Sutherland e destaca-se das demais metodologias ágeis pela maior ênfase dada ao gerenciamento do projeto. Reúne atividades de monitoramento e feedback, em geral, reuniões rápidas e diárias com toda a equipe, visando a identificação e correção de quaisquer deficiências e/ou impedimentos no processo de desenvolvimento. [SCHWABER 2008]

Scrum vem sendo largamente utilizando em organizações ao redor do mundo. Ele permite manter o foco na entrega do maior valor de negócio, no menor tempo possível

equipes se auto-organizam para definir a melhor maneira de entregar as funcionalidades de maior prioridade. Portanto, entre cada duas a quatro semanas todos podem ver o software real em produção, decidindo se o mesmo deve ser liberado ou continuar a ser aprimorado.

3.6.1 Características do Scrum

Trata-se de uma abordagem empírica focada nas pessoas para ambientes em que os requisitos surgem e mudam rapidamente, resultando em uma abordagem que reintroduz as idéias de flexibilidade, adaptabilidade e produtividade [BEEDLE e SCHAWABER 2002].

Segundo [BEEDLE e SCHAWABER 2002], a metodologia baseia-se em princípios como: equipes pequenas de, no máximo, 10 pessoas, requisitos que são pouco estáveis ou desconhecidos e iterações curtas. Divide o desenvolvimento em intervalos de tempos de, no máximo 30 dias, também chamadas de Sprints. Esta metodologia não requer ou fornece qualquer técnica ou método específico para a fase de desenvolvimento de software, apenas estabelece conjuntos de regras e práticas gerenciais que devem ser adotadas para o sucesso de um projeto.

3.6.2 Papéis do Scrum

O Scrum define para sua estrutura iterativa e incremental três papéis principais para diferentes tarefas, propósitos do processo e suas práticas: Scrum Master, Product Owner, Team (SCHWABER 2008).

• **Scrum Master:** Gerencia o processo, ensinando o Scrum a todos os envolvidos no projeto e implementando o Scrum de modo que esteja adequado à cultura da organização; deve garantir que todos sigam as regras e práticas do Scrum; é responsável por remover os impedimentos do projeto (SCHWABER 2008). Ele é o líder e facilitador para o Team e Product Owner, responsável por (BIRD e SCHAWABER 2008):

1. Resolver barreiras entre o Team e o Product Owner;

2. Ensinar o cliente a maximizar o retorno sobre o investimento (ROI);

3. Garantir que o processo seja seguido;

4. Melhorar a vida da equipe de desenvolvimento, facilitando a criatividade e a capacitação;

5. Melhorar a produtividade da equipe de desenvolvimento de qualquer forma possível;

6. Melhorar as práticas de engenharia e prover ferramentas de modo que cada nova funcionalidade seja potencialmente realizada;

7. Manter as informações sobre os progressos da equipe até a data atual e proporcionar esta visibilidade a todas as partes envolvidas no projeto.

• **Product Owner:** É o responsável pelo retorno sobre o investimento (ROI), ou seja, seu foco é na parte comercial do produto. Ele é o representante de todos os stakeholders

Comment [wis19]: Pontuação.

Comment [wis20]: Colocar no mesmo parágrafo.

Comment [wis21]: Colocar em português.

estar à disposição da equipe em qualquer momento, mas, sobretudo durante o Sprint Planning Meeting e Sprint Review Meeting (SZALVAY 2007).

Desafios de um Product Owner:

1. Não gerenciar a equipe. Isto é especialmente desafiador se alguns membros da equipe requisitam sua intervenção para questões que devem se resolver por si;

2. Não acrescentar mais funcionalidades após a Sprint já estar em andamento, somente em casos devidamente justificados;

3. Equilibrar os interesses dos Stakeholders.

• **Team:** É um grupo de pessoas com diferentes habilidades necessárias para transformar requisitos em um incremento potencialmente entregável. Para tanto, geralmente são uma mescla de analistas, designer, gerente de qualidade, desenvolvedor etc. A equipe tem a autoridade de decidir, quando necessário, as ações que serão realizadas e priorizá-las organizando-as em Sprints. Effort estimation, Sprint Backlog, revisões de product Backlog List e sugestões de impedimentos para serem removidos do projeto também são atividades do time (SZALVAY 2007).

3.6.3 Práticas do Scrum

Comment [wis22]: Falta escrever

3.6.4 Ciclo de Vida do Scrum

Comment [wis23]: Falta escreever

3.7 FEATURE DRIVEN DEVELOPMENT

O *Feature Driven Development* (FDD) é um processo ágil para gerenciamento e desenvolvimento de software, criado em 1977 em um grande projeto em Java para o *Unided Overseas Bank*, em Singapura. Nasceu a partir da experiência de análise e modelagem orientadas por objetos de Peter Coad e de gerenciamento de projetos de Jeff De Luca, frente a uma necessidade da referida instituição [SLIGER 2008].

A FDD é uma metodologia voltada para o cliente e orientada a modelagem, combinando algumas das melhores práticas do gerenciamento ágil de projetos com uma abordagem completa para Engenharia de Software orientada por objetos [COAD 1999]. Compõe de um arcabouço particular, através de seus princípios e práticas, que proporciona um equilíbrio entre as filosofias tradicionais e as ágeis. O referido equilíbrio, segundo Michele Sliger [2008] proporciona uma transição mais suave para organizações mais conservadoras, e a retomada da responsabilidade para as organizações que se desiludiram com as propostas mais radicais.

Comment [wis24]: Colocar no mesmo parágrafo.

Stephen Palmer [PALMER 2002] coloca a FDD como sendo algo que será incorporado a sua empresa com fácil adaptação e que possibilitará resultados frequentes, tangíveis e funcionais.

3.7.1 Características do FDD

As metodologias ágeis devem seguir o todo ou parte do que foi acordado no Manifesto Ágil, seção introdutória desse capítulo, mesmo surgidas antes do referido manifesto e por isso tendem a possuir características comuns [BECK, FOWLER 2001].

Comment [wis26]: Acho que precisa de mais texto.

Algumas características intrínsecas nos processos ágeis são levantadas por Pekka Abrahamsson [2003], a saber:

- Cliente presente;
- Iterações curtas;
- Equipes pequenas (<12 aproximadamente);
- Entregas frequentes do produto;
- Adaptativos as mudanças;
- Flexibilidade; e
- Simplicidade.

Porém Craig Larman [LARMAN 2003] destaca que dentre as características comuns aos processos ágeis sempre existirá particularidades que as diferenciem.

Seguindo essa linha das características ágeis, temos alguns pontos que particularizam o FDD [PALMER 2002]:

- Resultados úteis a cada duas semanas ou menos;
- Blocos bem pequenos de funcionalidade valorizada pelo cliente, chamados "features";
- Planejamento detalhado e guia para medição;
- Rastreabilidade e relatórios com incrível precisão;
- Monitoramento detalhado dentro do projeto, com resumos de alto nível para clientes e gerentes, tudo em termos de negócio; e
- Fornece uma forma de saber, dentro dos primeiros 10% de um projeto, se o plano e a estimativa são sólidos.

Essas características poderão ser observadas e, também, outras novas identificadas ao longo das seções que abordam sobre o FDD.

O FDD apresenta em seu escopo a definição de papeis para que se possa ter uma maior organização e visão na hora de se pensar/iniciar um projeto. Nesse contexto, o FDD estrutura seu time [PALMER 2002] em:

- Gestor do Projeto: trata das questões financeiras e administrativas do projeto. É o membro que decide sobre o escopo, objetivos, o time e prazos, no que se refere à decisão final. É, também, atribuição sua prezar por ótimas condições de trabalho e manter o time focado, com vistas a maximizar os resultados;

- Chefe de Design: responsável por toda a arquitetura do projeto, bem como das sessões de design, nas quais apresenta seus entendimentos ao time;

- Gestor de Desenvolvimento: acompanha as atividades de desenvolvimento do código diariamente, bem como a incumbência de fazer com que problemas não cheguem ao time ou que o mesmo seja resolvido o mais rapidamente. Desempenha suas funções afinado com o gestor de projeto;

- Programador Chefe: é responsável por uma equipe pequena no que se refere a divisão e atribuição de trabalho entre seus membros. Recomenda-se que seja um programador experiente, pois fará parte de suas atribuições a escolha das features a serem implementadas em cada iteração, bem como o relatório de atividades do time. Deve permitir um canal aberto de comunicação com o chefe de design e com o programador chefe;

- Dono de Classe: responsável pela arquitetura, implementação, teste e documentação de uma determinada classe e fará parte das equipes cujas features sejam envolvidas a sua classe;

- Especialista da Área: membro conhecedor do assunto sobre o qual a aplicação atuará. Trabalha em conjunto com o gestor de projeto em algumas questões macro que sua área lhe habilita, bem como ao lado dos desenvolvedores com suporte de conhecimento necessários a construção da fature.

Por se tratar de uma metodologia ágil, na qual a flexibilidade e adaptabilidade são presentes em sua essência, um membro pode assumir mais de um papel, simultaneamente, e um mesmo papel pode ser assumido por vários membros. Isso acontece a partir das características de cada projeto.

Como a proposta do FDD foi utilizada inicialmente em um time com aproximadamente 50 pessoas, pode fazer necessário o surgimento de outros papéis para compor o time dentro da característica do projeto proposto.

O FDD, inicialmente, recomenda uma composição de equipe de até 20 membros, mas existem casos conhecidos na literatura e na prática de indústrias de software, o processo atendendo a times bem maiores.

Alguns métodos ágeis, inclusive o FDD, afirmam se aplicar a qualquer projeto

Na seção seguinte será abordado a respeito das práticas recomendadas pelo FDD.

3.5.3 Práticas do FDD

O FDD possui em seu arcabouço um conjunto de boas práticas baseadas nas que identificamos e/ou vivemos na engenharia de software. As práticas do FDD focam em atender as necessidades do cliente a produção do sistema com qualidade. Abaixo serão descritas algumas dessas práticas [PALMER 2002]:

- Modelagem de objeto do domínio: é construída, inicialmente, uma modelagem genérica com suas funcionalidades, dentro da perspectiva da orientação a objetos. Essa modelagem possibilita um maior entendimento/visibilidade do problema a ser resolvido;

- Desenvolvimento por funcionalidade: Qualquer atividade a ser desenvolvida deve ser analisada para verificar se esta pode ser quebrada em atividades menores, ou funcionalidades atômicas (indivisíveis). Essa prática torna o código inteiro mais seguro contra erros. Além do mais, garante flexibilidade e escalabilidade ao código;

- Posse individual do código: os desenvolvedores possuem individualmente posse de código. O proprietário do código torna-se automaticamente responsável pela performance, consistência, corretude e integração da classe. O proprietário é aconselhado a utilizar os melhores padrões da engenharia de software;

- Equipes de funcionalidades: são equipes pequenas responsáveis por desenvolver uma pequena atividade. Dessa forma, a equipe decide de forma conjunta como será feito o design de determinada funcionalidade. Com mais de uma pessoa pensando no mesmo problema, soluções diferentes podem surgir. O que se torna muito eficiente, visto que no final essas soluções podem ser fundidas em uma solução final da equipe;

- Inspeções: inspeção de código é uma ótima prática de engenharia, pois além de fazer verificações contra erros, incentiva uma melhor modelagem do sistema, junto com atributos como legibilidade e alta coesão.

- Gerência de configuração: essa prática tem como objetivo manter um controle sobre o código fonte das funcionalidades implementadas, mapeando as funcionalidades aos seus respectivos códigos-fontes e aos seus proprietários. Além disso, mantém as datas de modificação do código, guardando um histórico de alterações;

- Build constante: deve existir sempre uma versão do sistema rodando numa máquina. Isso garante que a equipe possui pelo menos uma versão do sistema que funciona. Dessa forma, sempre que for necessário apresentar alguma funcionalidade para o cliente, existirá uma versão do sistema que pode ser utilizada para isso; e

- Visibilidade do progresso: Como sempre é mantido um relatório de progresso das atividades do projeto, todos na equipe e fora dela sabem exatamente como estão em termos de produtividade. No entanto, essa atividade deve ser feita com muita precisão e frequência. Caso contrário, os dados extraídos do relatório serão enganosos e poderão levar a decisões desastrosas para o cliente.

A seção seguinte detalha o ciclo do sistema a luz do FDD.

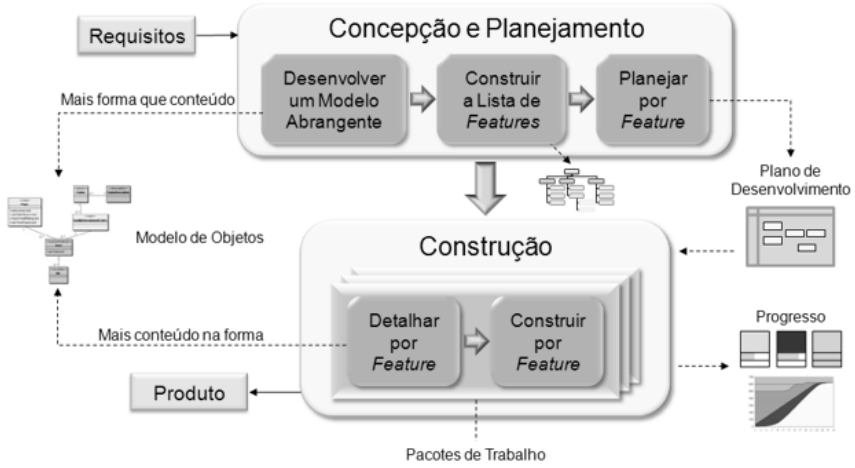
3.5.4 Ciclo de Vida do FDD

O FDD possui uma estrutura muito objetiva. A sua composição apresenta-se em duas fazes (Concepção/Planejamento e Construção) e possui cinco processos (Desenvolver um modelo abrangente, Construir a lista de features, Planejar por features, Detalhar por features e Construir por feature). **A figura X** abaixo ilustrará a descrição dos processos que será posteriormente feita.

Comment [wis27]: Em baixo tem modelo geral.

Comment [wis28]: Colocar o mesmo nome nos texto abiaxo.

Comment [wis29]: Coloca r embaixo da figura(Fig X….)



Desenvolver um modelo geral: o projeto começa com uma análise superficial do escopo do sistema e seu contexto. Assim, são estudados os domínios de negócio do sistema e criado um modelo geral baseado nestes. Depois é criada uma modelagem superficial para cada área de domínio existente. Cada um desses modelos é revisado por um grupo aleatório de membros do projeto e melhorias são discutidas. É escolhido

domínio são fundidos para gerar um modelo do domínio como um todo (ou domínio principal) do sistema.

Sub-atividades:

- Formar equipe de modelagem;
- Estudar o domínio de negócio;
- Estudar os documentos;
- Formar várias equipes pequenas para sugerir uma solução de modelo;
- Desenvolver o modelo escolhido;
- Refinar o modelo geral gerado;
- Escrever notas explicativas sobre o modelo final.

Gerar uma lista de funcionalidades: o conhecimento obtido na fase de desenvolvimento do modelo geral é essencial para esta fase. Nesta, será elaborada uma lista de funcionalidades do sistema decompondo as áreas de domínio obtidas. Cada funcionalidade é uma pequena tarefa a ser implementada que gere valor para o cliente. Devem seguir o formato <ação> <resultado> <objeto>, por exemplo: "Gerar relatório de vendas" ou "Validar a senha do usuário". Essas funcionalidades são muito importantes para o processo como um todo. A não identificação delas causa um impacto enorme sobre projeto, pois significa que a primeira fase não foi feita com sucesso e consequentemente o efeitos aparecem em cascata nas próximas fases.

Sub-atividades:

- Escolher uma equipe para gerar uma lista de funcionalidades;
- Gerar a lista de funcionalidades.

Planejar por funcionalidade: é realizado o planejamento de desenvolvimento de cada funcionalidade da lista obtida da fase anterior. São designadas classes ou código específico para os programadores-chefe tomarem conta. Daí em diante, os programadores-chefe serão responsáveis pelo código produzido nessas classes.

Sub-atividades:

- Formar uma equipe de planejamento
- Determinar a sequência de desenvolvimento das funcionalidades;
- Designar atividades de negócio para os programadores-chefe;
- Designar classes para os desenvolvedores.

Modelar por funcionalidade: os programadores-chefe escolhem algumas funcionalidades para que, junto com os proprietários de código, sejam feitos os

funcionalidade em questão. Ou seja, nesta etapa pensa-se nas classes, métodos e atributos que irão existir. Ao final da modelagem, é realizada uma inspeção do modelo pela equipe que a fez ou por outra equipe designada.

Sub-atividades:

- Formar uma equipe para a funcionalidade em questão;
- Estudar a funcionalidade como parte inserida no modelo de domínio;
- Estudar documentos relacionados á funcionalidade;
- Desenvolver diagrama de sequência;
- Refinar objeto modelo;
- Escrever as classes e as assinaturas dos métodos (tipo de retorno, parâmetros e exceções lançadas);
- Realizar inspeção da modelagem.

Construir por funcionalidade: após a modelagem, o código é finalmente implementado e os testes finalmente escritos. Assim, o código fonte é produzido e as funcionalidades ganham vida. O programador-chefe designa um programador para implementar uma certa funcionalidade. Logo, este último será o proprietário do código escrito. Após uma inspeção no código escrito, a funcionalidade é terminada.

Sub-atividades:

- Implementar as regras de negócio das classes;
- Inspecionar código;
- Conduzir testes unitários;
- Release da funcionalidade.

Os três primeiros processos citados, que compõem a primeira fase, acontecem de forma sequenciada, lembrando os métodos tradicionais, porém os processos estão em uma mesma fase. Já os dois últimos, que estão na segunda fase do FDD, possuem uma dinâmica interativa e incremental. Isso reforça a afirmação feita por Michele Sliger no início da abordagem sobre FDD.

9.7. Considerações Finais

<PENDENTE>

9.8. Tópicos de Pesquisa

<PENDENTE>

9.9. Sugestões de Leitura

<PENDENTE>

9.11. Exercícios

<PENDENTE>

Referências

SATO, D. (2007) Uso eficaz de métricas em desenvolvimento de software. 155 p. Dissertação (Mestrado em Ciência da Computação). Instituto de Matemática e Estatística – Universidade de São Paulo, São Paulo, 2007.

AGILE MANIFESTO (2001). http://www.agilemanifesto.org/ - Último acesso em 18/10/2009.

VERSIONONE (2008). 3º Annual Survey: The State Of Agile Development. Disponível em: http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf

AMBLER, S. (2002) Agile Modeling. New York: Wiley Computer Publishing.

SCHWABER, K. (2008) *Agile Project Management With Scrum.* Redmond: Microsoft Press.

SCHWABER, K., BEEDLE, M. (2001) Agile Software Development with Scrum. City: Prentice Hall.

KOSCIANSKI, A., SOARES, M. (2006) Qualidade de Software. 2. ed. São Paulo: Novatec.

ABRAHAMSSON, P., WARSTA, J., SIPONEN, M.T., & RONKAINEN, J. New Directions on Agile Methods: A Comparative Analysis. In: ICSE 2003, USA

ANDERSON, D.J.. Agile Management for Software Engineering: Using the Theory of Constraints for Business Results. 2003

BECK, Kent; FOWLER, Martin. Planning Extreme Programming. 1. ed. Boston: Addison-Wesley, 2001.

COAD, Peter; LEFEBVRE, Eric; LUCA, Jeff. Java Modeling In Color With UML: Enterprise Components and Process. Upper Saddle River, N.J.: Prentice Hall, 1999.

FOWLER, M.. The New Methodology. Disponível em: http://martinfowler.com/articles/newMethodology.html. Acesso em: 15 Set 2009.

HIGHSMITH, J. Agile software development ecosystems. Boston, MA., Pearson Education, 2002.

LARMAN, Craig. Agile and iterative development : a manager's guide. Addison-Wesley, 2003.

PALMER S. R., FELSING J. M. A Practical Guide to Feature-Driven Development (The Coad Series) , Prentice Hall PTR, USA, 2002.

SLIGER, Michele; BRODERICK, Stacia. The Software Project Manager's Bridge to Agility. Addison Wesley Professional, 2008.

Comment [wis30]: Formatar a referência

Índice

3.1 INTRODUÇÃO..... 121

3.2 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE..... 121

3.2.1 MOTIVAÇÕES PARA O DDS..... 122
3.2.1 NÍVEIS DE DISPERSÃO..... 123
3.2.2 MODELOS DE NEGÓCIO..... 124
3.2.3 DESAFIOS..... 125

3.3 PROCESSOS PARA DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE..... 126

3.4 PROCESSOS E ADAPTAÇÃO DAS PRÁTICAS EM PROJETOS DDS..... 126

3.4.1 MODELO DE KAROLAK [1998]..... 127
3.4.2 USO DE PRÁTICAS ÁGEIS..... 129

3.4.2.2 ADOÇÃO DE *SCRUM* EM UM AMBIENTE DDS.....132

3.5 TÓPICOS DE PESQUISA.....136

3.6 CONSIDERAÇÕES FINAIS.....136

3.7 EXERCÍCIOS.....137

3.8 SUGESTÕES DE LEITURA.....137

REFERÊNCIAS.....138

Capítulo

3

Processos para desenvolvimento distribuído de Software

Camila Cunha Borges

O objetivo do capítulo é apresentar como os modelos de processos e praticas de desenvolvimento de software podem ser aplicados em um ambiente de desenvolvimento distribuído de software.

3.1 Introdução

Nas últimas quatro décadas podemos observar que o software passou a ser o elemento-chave da evolução de sistemas e produtos baseados em computador [PRESSMAN 2007], onde é necessário desenvolver software com rapidez e qualidade. É importante observar que, à medida que o tempo passa a forma de se desenvolver um software vem passando por mudanças e os problemas relativos ao desenvolvimento continuam semelhantes: usuários insatisfeitos, longo tempo de desenvolvimento, nível de qualidade, dificuldade de comunicação, etc.

Com a globalização dos negócios, surgem grandes desafios para o processo desenvolvimento de software, que está cada vez mais distribuído e global [AUDY 2008]. Sendo assim, observamos a necessidade de implantação de métodos e ferramentas para a melhoria do processo de desenvolvimento distribuído de software.

3.2 Desenvolvimento Distribuído de Software

Grandes investimentos têm permitido uma movimentação do mercado local para o global, assim são criadas novas formas de colaboração e competição na área de Engenharia de Software [DAMIAN 2006]. Neste ambiente, muitas organizações

obter sucesso nos negócios. Segundo AUDY e PRIKLADNICKI, [2008], o DDS ganha cada vez mais força, motivado por três fatores ligados ao ambiente de negócios: (1) a globalização, (2) o crescimento da importância dos sistemas de informação nas empresas e (3) os processos de terceirização que geram um ambiente propício a esse cenário de desenvolvimento.

CARMEL [1999] afirma que as principais características que diferenciam o desenvolvimento co-localizado do desenvolvimento distribuído são: distância, diferenças de fuso horário e diferenças culturais. Distância refere-se à distribuição geográfica dos desenvolvedores e clientes finais. Diferenças culturais idioma, tradições, costumes, comportamentos e normas locais.

De acordo com PRIKLADNICKI [2003], o desenvolvimento distribuído criou uma nova classe de problemas a serem resolvidos pelos pesquisadores na área de desenvolvimento de software. Estas mudanças impactam não apenas no mercado propriamente dito, mas também na maneira como os produtos são criados, modelados, construídos, testados e entregues aos clientes.

3.2.1 Motivações para o DDS

O desenvolvimento de Software era realizado por pessoas com alto grau de especialização, trabalhando em centros de processamento de dados (CPD) em países avançados. Atualmente, o desenvolvimento de *software* vem ocorrendo de uma forma cada vez mais distribuída. [PRIKLADINIKI 2003].

As organizações visam obter vantagens competitivas associadas ao custo, qualidade e flexibilidade no desenvolvimento de *software*. Na maioria dos casos esse processo ocorre no mesmo país ou em regiões com incentivos fiscais. Algumas empresas buscam soluções em outros países (soluções globais), assim obtendo maiores vantagens competitivas.

Segundo a *International Data Group* (IDC) [2006] pode-se ter uma economia entre 25% e 50% em termos de custo quando grandes projetos são transferidos para operações *offshore* (em outro país). Além da redução de custo, é observada a disponibilidade de profissionais habilitados para trabalhar em outro idioma e incentivos de governos locais contribuem para essa nova forma de desenvolver software.

Existem diversas razões para a aplicação do DDS, em seguida são listadas

- **Demanda e custo:** Com o aumento na demanda por profissionais de software, o custo da mão-de-obra sofreu um aumento conforme as organizações competiam por suas contratações [KAROLAK 1998]. Assim a disponibilidade de recursos equivalentes em outras localidades tornou-se um grande atrativo.
- **Rapidez de resposta ao mercado:** A possibilidade de um desenvolvimento *follow-the-sun*, onde existem 24h contínuas, é um grande atrativo para empresas que visam reduzir o *time-to-market* (tempo para colocar o produto no mercado).
- **Mercado e presença global:** À medida de os custos são reduzidos e há um aumento no poder computacional, o mercado global de informática cresce. Assim o DDS é uma opção para atender a demanda do mercado global.

3.2.1 Níveis de Dispersão

O nível de dispersão dos atores envolvidos em um processo é uma característica importante do DDS. O entendimento dos níveis de dispersão auxilia na identificação de possíveis dificuldades. Se existe uma distancia de 30 metros ou mais entre os colaboradores, a freqüência da comunicação diminui na mesma proporção aos colaboradores distribuídos a milhares de metros [HERBSLEB 2001]. É importante entender o nível de distancias e suas implicações para as equipes. A seguir apresentamos tipos de distância física (Figura 3.1) e principais características:

- **Mesma Localização Física:** A empresa possui todos os atores em um mesmo lugar. Nesta situação, não existem dificuldades de reuniões e há uma interação presente entre membros das equipes. Além disso, não há diferença de fuso-horário e/ou diferenças culturais. Neste cenário as dificuldades são as já existentes no desenvolvimento centralizado.
- **Distancia Nacional:** Os grupos de atores estão localizados em um mesmo país, podendo reunir-se em curtos intervalos de tempo. Em alguns países podem ocorrer diferenças no fuso-horário.
- **Distancia Continental:** As equipes de atores estão localizadas em países diferentes, necessariamente no mesmo continente. As reuniões são mais difíceis de acontecerem face a face. Além disso, o fuso-horário dificulta as

- **Distancia Global**: Os grupos de atores estão em países e continentes diferentes, formando distribuição global. A comunicação e diferenças culturais neste cenário podem ser barreiras para o andamento do projeto e as reuniões face a face geralmente acontecem no inicio do projeto.

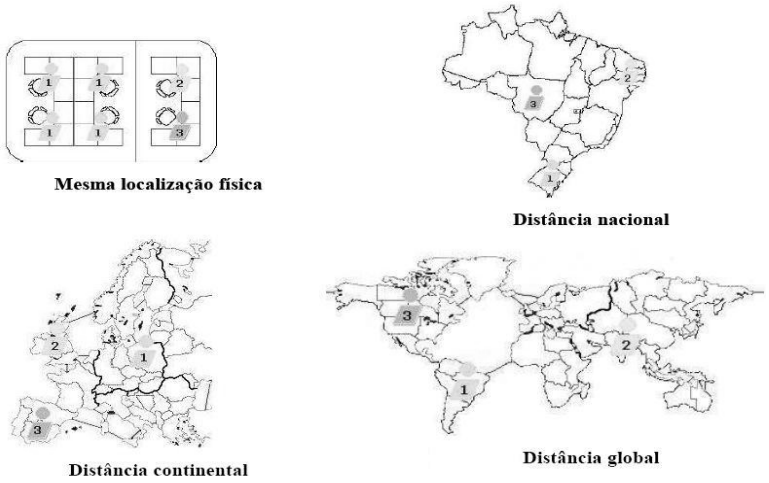


Figura 3.1 – Tipos de Distância Física [PRIKLADINIKI 2007]

3.2.2 Modelos de Negócio

De acordo com PRIKLADINIKI [2007], entre as relações existentes entre clientes e provedores de serviço, podemos caracterizar *Outsourcing* (terceirização) e *Insourcing* (subsidiárias da mesma empresa) como as principais relações existentes. Quanto à dimensão relacionada com a distância geográfica, a distribuição pode ser *Onshore* (em um país diferente) ou *Offshore* (no mesmo país). A seguir, os modelos são definidos:

- ***Onshore Insourcing:*** Existe um departamento dentro da própria empresa ou uma subsidiária da empresa no mesmo país (*onshore*) que provê serviços de desenvolvimento de software através de projetos internos (*insourcing*).
- ***Onshore Outsourcing* ou *Outsourcing:*** É a contratação de uma empresa terceirizada (*Outsourcing*) para o desenvolvimento de determinados serviços ou produtos de software. A empresa terceirizada está localizada no mesmo país da empresa contratante (*onshore*).

- ***Offshore Outsourcing* ou *Offshoring:*** É a contratação de uma empresa terceirizada (*Outsourcing*) para o desenvolvimento de determinados serviços ou produtos de software, sendo que ela está localizada em um país diferente da contratante (*offshoring*).
- ***Offshore Insourcing* ou *Internal Offshoring:*** É a criação de uma subsidiária da empresa para prover serviços de desenvolvimento de software (*Insourcing*) em um país diferente da empresa contratante (*offshore*).

É importante que seja observado que, além de outras formas de relacionamento entre empresas, também podem surgir outros tipos de distribuição geográfica, resultando em outros tipos de modelos de negócio. Neste livro serão abordados apenas os modelos de negócio já citados na seção 3.2.2.

3.2.3 Desafios

Conforme apresentado na seção 3.2.2, o DDS apresenta níveis de dispersão física, distância temporal e diferenças culturais, com isso alguns desafios foram acrescentados ao processo. O ambiente global apresenta grande impacto na forma como os produtos são concebidos, desenvolvidos, testados e entregue aos clientes [AUDY 2008]. Diferentes tecnologias e características são necessárias para o suporte ao DDS. Entre muitos desafios relacionados ao DDS, nesta seção vamos detalhar desafios focados no processo de desenvolvimento.

- **Arquitetura do Software:** É um dos fatores mais utilizados para a diminuição do esforço entre as equipes. Conforme KAROLAK [1998], uma arquitetura apropriada para o DDS deve se basear no princípio da modularidade, pois permite alocar tarefas complexas de forma distribuída. Com isso há uma redução na complexidade e é permitido um desenvolvimento em paralelo simplificado.

- **Engenharia de Requisitos:** A engenharia de requisitos contém diversas tarefas que necessitam de alto nível de comunicação e coordenação. Com isso os problemas apresentados são mais complexos em um contexto de DDS.

- **Gerência de Configuração:** O gerenciamento de configuração (CM) é a chave

modificações nos artefatos em cada uma das localidades com o processo de desenvolvimento de todo produto pode ser complexo. Apesar da utilização de práticas de CM auxiliar no controle da documentação e do software, a gerência de modificações simultâneas a partir de locais diferentes é um grande desafio. Além disso, o uso de ferramentas de CM compartilhadas por duas ou mais equipes de forma inadequada gera diversos riscos e problemas em projetos DDS.

- **Processo de Desenvolvimento:** Em projetos DDS, o uso de uma metodologia que auxilia a sincronização das atividades é essencial. Com isso todos os membros utilizam uma nomenclatura comum em suas atividades.

3.3 Processos para Desenvolvimento Distribuído de Software

Em um ambiente de desenvolvimento distribuído, um processo de desenvolvimento comum à equipe é fundamental, tendo em vista que uma metodologia auxilia diretamente na sincronização, fornecendo aos membros da equipe uma nomenclatura comum de tarefas e atividades, e um conjunto comum de expectativas aos elementos envolvidos no processo [PRIKLADNICKI 2008].

A engenharia de software (ES) sempre está apresentando grandes avanços e transformações relacionadas às técnicas, modelos e metodologias. Esses avanços são destacados quando se trabalha com processo de Desenvolvimento Distribuído de Software (DDS), havendo uma necessidade do uso de práticas que dê suporte às dificuldades encontradas nas definições de requisitos que mudam de forma dinâmica no decorrer do tempo. Estudos relacionados a processo para DDS ainda é escasso, sendo assim este capítulo relata o uso de praticas do desenvolvimento tradicional que podem ser implantadas em um ambiente distribuído e as possíveis adaptações.

3.4 Processos e adaptação das Práticas em projetos DDS

A forma como um produto de software é concebido, desenvolvido, testado e entregue ao cliente sofre grande impacto quando o ambiente de desenvolvimento é distribuído [HERBSLEB 2001]. Assim, a estrutura necessária para o suporte desse tipo de desenvolvimento se diferencia da utilizada em ambientes centralizados. Diferentes

características e tecnologias se fazem necessárias, crescendo a importância de alguns detalhes antes não percebidos.

Estratégias, soluções e práticas para tornar esta abordagem um sucesso tornam-se imperativas. O desenvolvimento de ambientes, modelos e ferramentas para gerenciar processos de software neste contexto tornam-se cada vez mais importantes. A seguir apresenta-se uma abordagem relacionada ao processo de desenvolvimento.

3.4.1 Modelo de Karolak:

Karolak [1998] aborda o DDS seguindo o ciclo de vida tradicional de um projeto de desenvolvimento de software. O autor propõe um modelo para desenvolver projetos DDS abrangendo as atividades que dever ocorrer ao longo do ciclo de vida. A figura abaixo ilustra o modelo proposto (Figura 3.2):

Figura 3.2 – Modelo para projetos DDS [KAROLAK 1998]

A seguir apresentam-se as atividades do modelo proposto

- **Alinhar o negócio:** Primeira atividade necessária para desenvolver projetos DDS, pois será identificado o tipo de estrutura que será utilizada. Nesta atividade é definido se existirão interações com outras empresas ou se serão criadas unidades da empresa em outras localidades.
- **Identificar a equipe distribuída:** Nesta atividade são definidos os integrantes da equipe, seus respectivos papéis e responsabilidades. A formação da equipe deve considerar os seguintes aspectos: aquisição de confiança, diferenças

- **Identificar as tecnologias:** Devido à grande demanda de comunicação em projetos DDS, há a necessidade de um apoio tecnológico considerável. Nesta atividade é identificada a infra-estrutura disponível para os membros das equipes se comunicarem, considerando o nível de dispersão da equipe.
- **Definir o contrato:** Um contrato é um documento que define o escopo do que deve ser feito. Quando o projeto é distribuído esta atividade se torna mais complexa.
- **Dividir o trabalho:** Após a identificação da equipe, tecnologia e definição do contrato, é proposta a divisão do esforço de trabalho entre os membros de uma equipe. Deve ser levado em consideração o nível de experiência e a modularidade do projeto.
- **Identificar ferramentas e métodos:** Identificação dos recursos técnicos que serão utilizados na modelagem e implementação do projeto. Deve-se considerar o nível de dispersão da equipe e o processo de desenvolvimento.
- **Estabelecer responsável por SCM:** A gerência de configuração de software (SCM – Software *Configuration Management*) tem como objetivo controlar modificações nos artefatos, dando suporte ao controle de versões. O autor sugere a existência de um grupo responsável pelo controle de configuração e versões do sistema. Por este motivo, esta atividade visa identificar os membros deste grupo, bem como as ferramentas que eles utilizarão e a freqüência necessária de reuniões para discutir o andamento do trabalho.
- **Identificar e gerenciar riscos:** Esta atividade faz parte de qualquer projeto. De acordo com o autor, os riscos em projetos DDS tendem a ser mais centrados em aspectos não tão visíveis. Esta atividade deve acontecer em todas as fases do desenvolvimento, exceto entrega e manutenção. Em projetos DDS podem existir três categorias de risco: organizacional, técnico e de comunicação.
- **Controlar a documentação:** É conhecida a resistência em documentar por partes de equipes de desenvolvimento. Em projetos DDS, uma documentação pobre pode causar ineficiência na colaboração. Uma boa documentação pode

- **Desenvolver plano e casos de teste:** KAROLAK [1998] menciona que um projeto distribuído necessita de pelo menos dois artefatos de teste. O plano de teste com as estratégias, métodos e ambiente documentados e o Caso de teste com as funcionalidades que serão testadas.
- **Criar matriz de rastreabilidade:** uma matriz de rastreabilidade é um artefato que identifica as funcionalidades do projeto e os módulos que as implementam. Este artefato mostra a ligação entre os requisitos e como um requisito influencia em outro.
- **Criar matriz de versão de módulos:** uma matriz de versão de módulos é um artefato que identifica qual versão de um módulo foi utilizada na compilação do código de um projeto. Este artefato é essencial principalmente para a coordenação das atividades e divisão do trabalho entre os membros da equipe do projeto.
- **Criar grupo de manutenção:** O modelo sugere a criação de um grupo responsável por revisar solicitações de alterações após o produto ser entregue ao cliente.
- **Controlar a qualidade do software:** Devem existir atividades que melhoram a qualidade do software a ser desenvolvido, tais como revisões de modelagem, inspeções de código e teste.
- **Gerenciar a propriedade intelectual**: O autor prevê uma atividade onde se busca a devida proteção, levando-se em consideração leis e restrições do local onde o projeto foi desenvolvido (alguns locais fisicamente dispersos podem ter leis muitas vezes desconhecidas pelas organizações).

3.4.2 Uso de Práticas Ágeis

A partir do ano 2000 surgiu uma tendência para o desenvolvimento ágil de aplicações devido a um ritmo acelerado de mudanças e inovações na tecnologia da informação, em organizações e no ambiente de negócios. Desde então vários métodos ágeis foram surgindo, entre eles: *Adaptive Software Development, Crystal, Dynamic Systems Development, eXtreme Programming (XP), Feature Driven Development (FDD)* e

De acordo com TRAVASSOS [2005], os métodos ágeis são projetados para (1) produzir a primeira entrega em semanas e alcançar feedback rápido e mais cedo; (2) criar soluções mais simples de modo que se houverem mudanças que haja mais facilidade e menor volume de alterações a serem feitas; (3) melhorar continuamente a qualidade do projeto, fazendo com que a iteração seguinte tenha menor custo de implementação; (4) testar constantemente, para detectar defeitos mais cedo e removê-los com menor custo.

Quando o ambiente é distribuído o uso de em práticas ágeis parece ser incompatível. Práticas ágeis necessitam de comunicação face a face constantemente e a comunicação é um grande desafio em ambientes DDS. Apesar disso, o uso de metodologias ágeis de desenvolvimento de *software* tem se tornado uma demanda em equipes distribuídas de software devido ao aumento na velocidade de desenvolvimento, alinhamento dos objetivos individuais com os organizacionais e melhoria no desenvolvimento [SUTHERLAND 2007].

3.4.2.1 DXP – *Distributed Extreme Programming*

Conforme abordado no capítulo anterior, a metodologia de desenvolvimento XP (*Extreme Programming*) requer uma comunicação forte e eficaz entre os membros de uma equipe de desenvolvimento de software. Para isso a metodologia enfatiza a necessidade de ter os membros da equipe fisicamente próximos uns dos outros. No entanto, nem sempre os membros da equipe de um projeto estão fisicamente próximos uns dos outros. Nesta seção será apresentada uma adaptação do uso do XP em ambientes DDS "*Distributed Extreme Programming*" (DXP). Estudos mostram que a aplicação do DXP pode ser eficaz e gratificante em projetos cujas equipes estão geograficamente dispersas.

Segundo YOUNG [2008], o DXP aplica princípios XP em um ambiente distribuído, onde os membros das equipes também podem ser altamente móveis. A figura abaixo (Figura 3.3) resume alguns dos aspectos que são relevantes para DXP e alguns que não são referentes ao fato da distribuição ou não das equipes.

	localizado?
Jogo do Planejamento **Programação em Par** **Integração Contínua** **Cliente local**	**Sim. Estes fatores dependem de uma aproximação entre o negócio, cliente e pessoal técnico.**
***Releases* Pequenos** **Metáforas** **Projeto de Software Simples** **Teste** **Refatoração** **Propriedade Coletiva** **40 horas semanais** **Padrão de codificação**	**Não. Independem se a equipe é Co-localizada ou não.**

Figura 3.3 – Adaptação do XP em DDS [Adaptada de KIRCHER 2000]

Conforme apresentado na figura acima, podemos observar que para a utilização do DXP de forma eficaz é necessário que o *Planning Game, Pair Programming, Continous Integration e On-site Custumer* sejam abordadas em uma equipe distribuída. Na figura acima o autor considera que a prática de *Refactoring* não exige um ambiente co-localizado apesar de esta prática exigir o uso da prática *Pair Programming.* Kircher [2000] afirma que estas duas práticas podem iniciar separadamente.

O DXP assume a existência de algumas condições para que seja eficaz, tais como a disponibilidade de diversas ferramentas e tecnologias. Além das práticas do XP, o DXP assume:

- **Conectividade:** Alguma forma de conectividade precisa existir entre os membros da equipe. Para longas distâncias a *Internet* é utilizada como meio de comunicação.
- ***E-Mail*:** É um meio de troca de informação muito utilizado no DXP.
- **Gerenciamento de Comunicação:** Para uma gestão eficaz dos artefatos de programação é necessário que seja utilizada uma ferramenta de gerenciamento de configuração.
- **Compartilhamento de Aplicação:** Aplicações ou *Softwares* de compartilhamento de *desktop* devem estar disponíveis para as equipes distribuídas.
- **Uso de Vídeo Conferência:** O uso de vídeo conferência em equipes distribuídas é

cliente neste meio de comunicação, pois o mesmo não tem disponibilidade de estar no local da reunião.

- **Integração entre os membros de uma equipe móvel:** Caso necessitem se deslocar, podem utilizar equipamentos móveis para participar das atividades de desenvolvimento.

De acordo com YOUNG [2008], o DXP pode integrar membros de equipes remotas e móveis processo de desenvolvimento e, portanto, uma extensão valiosa para o XP tradicional. Além disso, permite um envolvimento maior com o cliente quando comparado ao XP, principalmente em situações que é necessário ter o cliente *on-site.* O autor enfatiza também a necessidade de atentar para os problemas já existentes em ambientes DDS, tais como comunicação, disponibilidade dos membros das equipes, coordenação, infra-estrutura e gestão.

3.4.2.2 Adoção de *Scrum* em um ambiente DDS

Inserido neste contexto de desenvolvimento distribuído de software, esta seção apresenta a aplicação da metodologia *Scrum*, abordada no capítulo anterior, no processo de desenvolvimento de uma fábrica de software em um ambiente de desenvolvimento distribuído.

A experiência que será descrita nesta seção foi parte da disciplina de Engenharia de ***Software*** [2009] com um estudo em fábricas de software, fazendo uso de DDS e metodologias ágeis para realizar projetos reais. Os alunos tiveram o período da disciplina (primeiro semestre de 2009) para desenvolver o produto conforme definido no inicio do curso. O projeto relatado, denominado *FireScrum*, é uma ferramenta de gerenciamento de projetos que utiliza a metodologia *Scrum*, cujo objetivo é o de facilitar o uso da referida metodologia em ambientes distribuídos.

O desenvolvimento foi dividido em seis módulos: *Core, TaskBoard, Planning Poker, Test Module, BugTracking* e *Desktop Agent.* A fábrica era composta por sessenta alunos distribuídos em seis times, na qual cada time era responsável por um dos módulos citados. Todos os componentes de todos os times realizaram suas atividades de forma distribuída. O módulo que será relatado é o *Bugtracking*, composto por nove

Para o desenvolvimento do módulo *Bugtracking,* o time realizou um estudo entre ferramentas *open source Mantis* e *Bugzilla.* Assim, foi possível identificar as vantagens e desvantagens de cada uma para que o módulo fosse desenvolvido de forma diferenciada e inovadora, prezando pela simplicidade e usabilidade. O *Firescrum* foi desenvolvido utilizando a ferramenta *Adobe Flex,* o banco de dados utilizado foi o *Postgree SQL* e para o controle de versão foi utilizado o SVN.

O processo de desenvolvimento seguiu a metodologia *Scrum.* As *Sprints* tinham duração de 15 dias. A *Sprint Planning* 1, reunião para definir os itens de backlog que seriam atendidos na *sprint*, acontecia de forma presencial semanalmente após a aula. A *Sprint Planning* 2, reunião na qual são definidas as tarefas necessárias à implementação das funcionalidades definidas na *Sprint Planning* 1, acontecia de forma remota utilizando os seguintes recursos: *skype, MSN* e a planilha de gerenciamento criada no *Google Docs*. As reuniões diárias (*Daily Scrum Meeting*), com o objetivo de acompanhar a realização das tarefas, inicialmente acontecia com o auxilio do *Skype, MSN* e posteriormente foi adotado um grupo de *email*, pois os horários dos membros da equipe eram incompatíveis e nem sempre todos poderiam participar das reuniões no horário marcado. Para os participantes que residiam na mesma cidade acontecia encontros em duplas (uso da prática de programação em pares do XP) para discutir a sobre o desenvolvimento, em seguida as dúvidas e conclusões eram postadas no grupo de *email*.

Ao longo do desenvolvimento, o time manteve sempre evidente e aplicada à filosofia de que cada membro era seu próprio gerente e responsável pelos resultados do projeto. Práticas foram acordadas para que os resultados necessários à conclusão da *sprint* fossem alcançados, aprendizado e bom convívio entre os membros. A principal delas foi que questionamentos viriam após a conclusão de qualquer tarefa/, ou seja, cada membro estava focado em concluir tarefas e manter os meios de gerenciamento atualizados.

Os membros acompanhavam a evolução de três artefatos: a planilha de tarefas no *Google Docs*, o *burndown* e o grupo de *email.* Isso possibilitou que o time mantivesse durante todo o processo de desenvolvimento um autogerenciamento

das cinco *sprints* do projeto de desenvolvimento e a planilha de gerenciamento do *Google docs* (Figura 3.5).

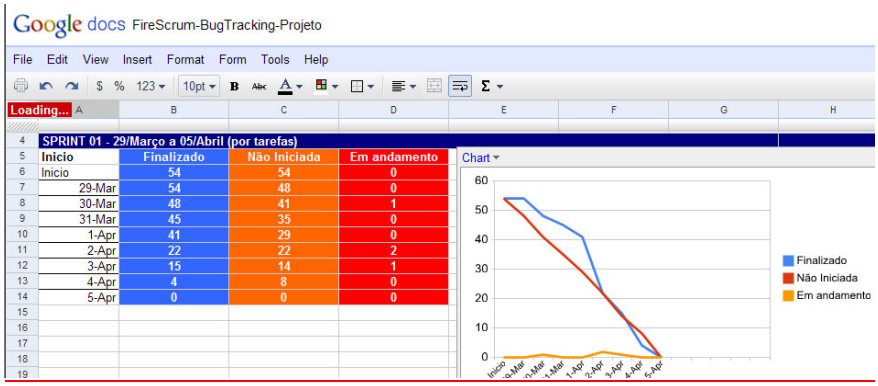


Figura 3.4 – Gráfico *Burndown*

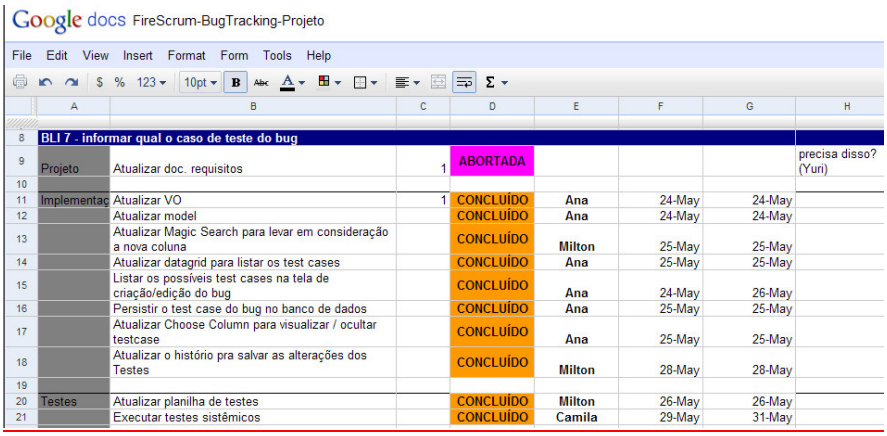


Figura 3.5 – Planilha de Gerenciamento

Adaptando a metodologia para solução dos problemas

A grande dificuldade foi à realização da *Daily Scrum Meeting* (DSM), pois os integrantes do time possuíam atividades paralelas, tais como outras disciplinas da pós-graduação e alguns até trabalhavam, assim os horários disponíveis eram incompatíveis.

De forma geral, estão listados abaixo problemas encontrados e adaptações na metodologia para um melhor resultado.

- Foi determinado pelo time que a DSM seria realizada a cada dois dias, assim evitando o risco de algum membro não ter nada a informar do que foi feito no dia, pois nem todos os membros do time estavam disponíveis todos os dias para executar tarefas relacionadas ao projeto.
- Para realizar a DSM, primeiramente utilizamos o *Skype e MSN* e foi acordado o horário das 20h30min para a reunião remota. Esta foi uma tentativa que não deu certo, pois a reunião tornava-se cansativa a freqüentes eram as perdas na conexão.
- A demora na construção de frases claras é característica de perda de foco nas reuniões, assim as reuniões se prolongavam mais que o necessário e acabássemos entrando em peculiaridades dos problemas.
- Foi criado um grupo de email, e foi acordado que a cada DSM postaríamos até 23h as respostas para as perguntas: O que foi feito até hoje?; O que será feito até a próxima DSM?; e Quais os impedimentos?.
- Foi criada uma planilha no *Google Docs*, na qual constava o *Product Backlog,* o objetivo de cada *sprint* e suas respectivas tarefas, o *burndown* e os impedimentos. Assim era possível acompanhar a dinâmica do time.

Lições aprendidas com o uso do *Scrum* em um ambiente DDS

- Integrantes do time auto-gerenciáveis: A utilização do *Scrum* mostrou que para o sucesso do projeto é indispensável que os participantes sejam auto-gerenciáveis. E apesar da dispersão entre os participantes do time foi possível obter um bom resultado.
- Implementação: A experiência adquirida pela equipe com a utilização do *Scrum* em um ambiente DDS revela que a programação realizada a distancia é possível de ser aplicada com o suporte de ferramentas disponíveis, tais como *emails* ou *MSN.*

- Comunicação: Várias ferramentas foram utilizadas e algumas não atingiam o objetivo do time por não suportar vários usuários conectados ao mesmo tempo, por exemplo.

3.5 Considerações finais

O desenvolvimento de software sempre se apresentou de forma complexa. Existe uma série de problemas e desafios inerentes ao processo. Assim como o processo de desenvolvimento de software tem se tornado cada vez mais complexo, a distribuição das equipes no tempo e no espaço tem tornado os projetos distribuídos cada vez mais comuns. O DDS, ao acrescentar fatores como dispersão geográfica, dispersão temporal e diferenças culturais, acentuaram alguns dos desafios existentes e acrescentou novos desafios ao processo de desenvolvimento. O trabalho em ambientes de DDS é mais complexo do que em ambientes centralizados e não existem métodos ou práticas específicas para o ambiente distribuído.

3.6 Tópicos de Pesquisa

Mecanismos de coordenação em ambientes DDS não existem ou são falhos [Herbsleb 2001]. A distância afeta a colaboração entre as equipes, pois há uma menor freqüência de comunicação, comunicação ineficiente, falta de percepção, além da incompatibilidade de processos, ferramentas e práticas de trabalho.

Nos últimos anos muitos trabalhos apresentam propostas de solução para as dificuldades e desafios existentes em projetos DDS. As pesquisas podem se concentrar em questões como o investimento em um modelo de negócio de DDS, a estrutura da operação, a relação com outras empresas ou unidades da própria empresa, projetos para equipes distribuídas e outros. Como este capítulo aborda processos para DDS, a seguir apresentamos uma lista de tópicos de pesquisa.

- **Processo de desenvolvimento em um ambiente DDS:** A definição de um processo que considere o contexto de uma equipe distribuída. Os modelos de qualidade de software reconhecidos internacionalmente (CMMI) e nacionalmente (MPS-BR) orientam as organizações no desenvolvimento de processos, mas não propõem modelos para distribuição e distância.

- **Uso de práticas em ambientes DDS:** O uso de uma prática pode ser aplicado em diferentes modelos de negócio de um ambiente DDS? Podemos comparar o modelo *Outsourcing* e o *Insourcing.*
- **Ferramentas de colaboração:** Atualmente existem muitas ferramentas que oferecem suporte as atividades do ciclo de vida do desenvolvimento de um *software*. Estas ferramentas são adaptadas para o cenário distribuído. Neste contexto, observa-se a necessidade de ferramentas que oferecem suporte ao *awareness* de atividade (quem está fazendo o quê), de processo (quem deve fazer o quê) e de disponibilidade (quem está disponível quando).

3.7 Sugestões de Leitura

- Desenvolvimento Distribuído de Software [Prikladiniki 2008];
- Global Software Teams [Carmel 1999];
- MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software [Prikladiniki 2003];
- Distributed Scrum: Agile Project Management with Outsourced Development Teams [Sutherland 2007].

3.8 Exercícios

1. Defina o que é Desenvolvimento Distribuído de Software.
2. Quais as vantagens que uma organização tem ao utilizar um processo DDS?
3. Quais são os níveis de dispersão em um ambiente DDS? Exemplifique.
4. Quais os modelos de negócio em um ambiente DDS? Exemplifique.
5. Quais as principais dificuldades ao realizar um projeto DDS?

Referências

Herbsleb, J. D., Moitra, D. “Global Software Development”, IEEE Software, March/April, EUA, 2001, p. 16-20.

Karolak, D. W. “Global Software Development – Managing Virtual Teams and Environments”. Los Alamitos, IEEE Computer Society, EUA, 1998, 159p.

Kiel, L. “Experiences in Distributed Development: A Case Study”, In: Workshop on Global Software Development at ICSE, Oregon, EUA, 2003, 4p.

Kircher, M., Jain, P., Levine, A. “Distributed Extreme Programming”, IEEE, Agile 2008.

Herbsleb, J.D., Mockus, A., Finholt, T.A. e Grinter, R. E. “An empirical study of global software development: distance and speed”, In: ICSE 2001, Toronto, Canada.

Carmel, E. “Global Software Teams – Collaborating Across Borders and Time-Zones” Prentice Hall, EUA, 1999, 269p.

Marquardt, M. J., Horvath, L. “Global Teams: how top multinationals span boundariesand cultures with high-speed teamwork”. Davies-Black. Palo Alto, EUA, 2001.

Young, C., Terashima, H. “How Did Wr Adapt Agile Processes to Our Distributed Development?”, IEEE, Agile 2008.

Prikladnicki, R., Audy, J. L. N., Evaristo, R. “Global Software Development in Practice: Lessons Learned”, Journal of Software Process: Practice and Improvement – Special Issue on Global Software Development, 2004.

Prikladnicki, R. “MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software”. Dissertação de Mestrado, PPGCC – PUCRS, Brasil, 2003.

J. L. N. Prikladinicki, R.; Audy. Desenvolvimento Distribuído de Software. 2007.

Perrelli, Hermano. Visão Geral do RUP. Centro de Informática, Universidade Federal de Pernambuco. Disponível em: http://www.cin.ufpe.br/~if717/slides/3-visao-geral-do-rup.pdf. Acessado em 20 Maio 2009.

PRESSMAN, Roger S. Software Engineering: a practitioner’s approach. EUA: McGraw Hill, 2001. 860 p.

Sutherland, J., “Distributed Scrum: Agile Project Management with Outsourced Development Teams”, HICSS, 2007.

Teles, Vinícius Manhães. Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. 1. ed. São paulo: Novatec, 2004. 320 p.

Travassos, G. H.,Abrantes, J. F., “Caracterização de Métodos Ágeis de Desenvolvimento de Software”

Capítulo

10

Desenvolvimento de Software Dirigido a Modelos

Almir Buarque

O objetivo geral deste capítulo é apresentar o processo desenvolvimento de software dirigido a modelos (MDD), padronizado pela Arquitetura Dirigida a Modelos (MDA) do grupo OMG, sua relevância para elevação da qualidade do processo de engenharia de software e, consequentemente, do produto. Duas abordagens MDD serão descritas: OO-Method e AndroMDA. O capítulo mencionará ainda os problemas e desafios atuais do processo de desenvolvimento dirigido por modelos. Será apresentada mais detalhadamente, a abordagem OO-Method por ser uma referência na literatura MDD, ter precisão e definição semântica baseada na linguagem formal orientada a objeto chamada OASIS e por ser totalmente suportado pelo ambiente OLIVANOVA.

10.1 INTRODUÇÃO 142

10.2 ARQUITETURA DIRIGIDA A MODELOS 144

10.2.1. CONCEITOS BÁSICOS 144
10.2.2. PADRÕES OMG E A ARQUITETURA MDA 152

10.3 ABORDAGENS MDD MODELOS 153

10.3.1 OO-METHOD 154
10.3.1.1. O PROCESSO BÁSICO DE TRANSFORMAÇÃO 154
10.3.1.2. COMPARAÇÃO COM MDA 155
10.3.1.3. O MODELO CONCEITUAL 156
10.3.1.4. O COMPILADOR DE MODELOS 159
10.3.1.5. OLIVANOVA 159
10.3.2 . ANDROMDA 160

10.4 PROBLEMAS E DESAFIOS DOS PROCESSOS MDD 161

10.4.1. VISÃO GERAL 161
10.4.2. LIÇÕES APRENDIDAS NA ADOÇÃO DE SOLUÇÕES MDA 162
10.4.3. O PROGRAMA FASTSTART DA OMG 162

10.5. TÓPICOS DE PESQUISA 163

10.6 . SUGESTÕES DE LEITURA 163

10.7 . EXERCÍCIOS 164

REFERÊNCIAS 167

Lista de Figuras

Figura 10.1. Transformações em MDA 147
Figura 10.2 Metamodelo MDA 148
Figura 10.3. Transformações de mapeamentos por metamodelos 149
Figura 10.4. Transformações com UML Profile 151
Figura 10.5. UML Profiles da OMG 151
Figura 10.6. Padrões MDA 152
Figura 10.7. Abordagem OO-Method 155
Figura 10.8 Diagrama de Classes 166
Figura 10.9 Diagrama de Atividades 166

Lista de Tabelas

10.1 Introdução

Dentro do contexto de que modelar é uma atividade essencial da engenharia de software, desenvolvimento de software dirigido a modelos "Model Driven Software Development", cujo acrônimo em inglês é MDD, vem representando atualmente um papel central no processo de engenharia de software. Convém lembrar que essa idéia não é nova. Desde a década de 1970, que os métodos formais difundiram o desenvolvimento de software a partir de modelos formais matemáticos e suas transformações até se obter código executável. A partir de um desenvolvimento formal é possível elevar a qualidade do software com técnicas formais de validação e verificação. Com o amadurecimento das linguagens de modelagem de software e a complexidade da conjuntura atual da indústria de software, cada vez mais, essa idéia tem se consolidado através de abordagens que adotam MDD como um padrão de desenvolvimento. Em 2001, quando o grupo OMG especifica a Arquitetura Dirigida a Modelos-MDA (Model Driven Architecture), ele cria uma nova instância de processo de desenvolvimento de software dirigido a modelos (MDD) que já existia há anos, renomeando-a de MDA.

Os principais argumentos para a utilização de um processo de desenvolvimento dirigido a modelos são os seguintes: maior produtividade, portabilidade, interoperabilidade, menor custo, mais facilidade na evolução do software, enfim, maior qualidade do produto. Esses benefícios são evidenciados, por exemplo, num estudo [MDA 2003] que comparou uma produção de software usando-se a tecnologia MDD com o mesmo software fabricado com tecnologia OO tradicional. Isso ocorre principalmente pelas seguintes razões: Primeiramente porque a principal idéia em MDD é a transformação de modelos de maiores níveis de abstração (domínio do problema) em modelos mais concretos (domínio solução) até se obter, por fim, o código do sistema. Depois, o paradigma MDD preconiza que o desenvolvimento inicial e modificações futuras da aplicação sejam efetuados apenas no modelo mais abstrato.

Em processos MDD automatizado, esse modelo abstrato do sistema deve representar com precisão o código, ou seja, ele deve ser executável e ter uma equivalência funcional com todos os outros modelos mais concretos. Dessa forma, as modificações no modelo de mais alto nível de abstração são refletidas automaticamente nos modelos de mais baixo nível, tornando a atividade de modelar no nível mais abstrato o centro de todo processo de desenvolvimento do software e dispensando completamente, nos melhores ambientes MDD, atividades manuais nos modelos de mais baixos níveis de abstração (projeto e implementação).

Entretanto, a indústria de software tem potencializado e exagerado esses benefícios, transmitindo a falsa idéia aos desenvolvedores de que em MDD, apenas com um click ou passo de mágica, obtém-se todas as transformações e o produto de software final. Além disso, passa-se a idéia de que gerar código é o principal objetivo MDD quando é transformar modelos, confundindo os desenvolvedores com várias ferramentas (ambientes) CASE que apenas geram códigos a partir de técnicas diversas e que, na verdade, não transformam modelos. Ademais, existem ainda problemas semânticos, complexidades e imprecisões (ambigüidades) inerentes aos modelos atuais que tornam esse processo de transformação e mapeamentos de modelos uma tarefa árdua e propensa

Por outro lado, não há um consenso na comunidade acadêmica sobre qual modelo de maior nível de abstração é mais adequado (necessário e suficiente) para se modelar um sistema, dificultando-se padronizações, interoperabilidade e produzindo-se ambientes MDD que não são integrados com modelos a nível de requisitos que são essenciais para todo o processo de Engenharia de Software. Este capítulo do livro abordará todos esses tópicos referentes ao processo de desenvolvimento de software dirigido a modelos, mas precisamente sobre a arquitetura MDA.

10.2 Arquitetura Dirigida a Modelos

Esta seção objetiva descrever uma visão geral dos padrões OMG, arquitetura MDA e seus conceitos básicos.

10.2.1. Conceitos Básicos

Para um melhor entendimento da arquitetura dirigida a modelos, o padrão MDA do OMG [OMG 2003] define os seguintes conceitos:

- **Modelo**

  Um modelo de um sistema é a sua representação (especificação) funcional, estrutural e comportamental. Uma especificação é dita como formal quando é baseada em uma linguagem que tem uma sintaxe e semântica bem definidas e, possivelmente, tem também regras de análise, inferência ou prova de seus elementos. Essa sintaxe pode ser gráfica (visual) ou textual. A semântica pode ser mais ou menos formal.

- **Dirigido a Modelos**

  MDA é uma abordagem de desenvolvimento de sistema que usa o poder dos modelos. É dirigida a modelos porque provê meios de usar modelos para direcionar o curso de entendimento, projeto, construção, distribuição, operação, manutenção e modificação.

- **Arquitetura**

  Arquitetura de um sistema é a especificação de suas partes e conectores, além das regras de interação dessas partes usando os conectores.

- **Ponto de vista (Viewpoint)**

  Um ponto de vista de um sistema é uma técnica de abstração, usando um conjunto selecionado de conceitos arquiteturais e regras de estruturação que visa focar ou representar um aspecto (característica) dentro desse sistema. O termo abstração está sendo usado para significar o processo de suprimir (esconder) um detalhe selecionado para estabelecer um modelo simplificado.

- **Plataforma**

  Uma plataforma é um conjunto de subsistemas e tecnologias que provê um conjunto coerente de funcionalidade através de interfaces e padrões de uso especificados, que qualquer aplicação (sistema) suportada por essa plataforma pode usar, sem ter que saber os detalhes de como essa funcionalidade provida pela plataforma é implementada.

- **Pontos de Vistas (Modelos) MDA**

  - **Modelo Independente de Computação (CIM)**

    É uma visão do sistema a partir de um ponto de vista (viewpoint) independente de computação. O CIM não mostra detalhes da estrutura dos sistemas, sendo usualmente chamado de modelo de domínio ou modelo de négocio e utiliza, em sua especificação, um vocabulário familiar aos usuários do domínio (problema) em questão. Os usuários do CIM geralmente não têm conhecimento sobre modelos ou artefatos usados para realizar as funcionalidades definidas através dos requisitos. Esse modelo foca no ambiente do sistema e nos seus requisitos, deixando os detalhes da estrutura e processamento (computação) do sistema escondidos aos usuários (stakeholders) ou, mesmo, esses detalhes são indeterminados.

    Dessa forma o CIM tem um importante papel de fazer a ponte (reduzir a lacuna "gap") entre aqueles que são especialistas no domínio do problema e seus requisitos, e aqueles que são especialistas em projeto (arquitetura) e construção dos artefatos que juntos vão satisfazer aos requisitos do domínio, elicitados pelos usuários.

    O CIM é obtido no processo de documentação e especificação dos requisitos, ou seja, ao se especificar um modelo de requisitos para o sistema. Outra forma também de definição do CIM é o modelo de negócios do sistema.

    - **Modelo Independente de Plataforma (PIM)**

      O PIM foca na operação do sistema (modelo computacional), mas escondendo os detalhes necessários para implantar esse modelo numa plataforma específica. O PIM é único para o sistema e não muda quando se varia de uma plataforma para outra. Esse ponto de vista independente de plataforma pode ser especificado usando-se uma linguagem de modelagem de propósito geral (UML) ou uma linguagem específica (OO-Method) como será visto na seção 10.3.1. O PIM é um modelo conceitual do sistema.

    - **Modelo Específico de Plataforma (PSM)**

      Este modelo é uma visão do sistema que agrega

tecnologia utilizada na aplicação como a linguagem de programação, os componentes de middleware, a arquitetura de hardware e de software. Para que isso seja possível é necessário o suporte de ferramentas que façam o mapeamento adequado de uma especificação abstrata (PIM) para uma determinada plataforma. O PSM, por sua vez, passa por processo(s) de refinamento(s) para obtenção do nível de especificação desejado. A obtenção desse nível torna possível a transformação do mesmo no código (implementação) da aplicação. O modelo PSM é o responsável por lidar com toda heterogeneidade e complexidade dos diversos tipos de plataformas existentes.

- **Transformações (Mapeamentos)**

A força motriz do padrão MDA é a transformação de modelos, que pode ser realizada entre modelos de um mesmo ponto de vista ou entre pontos de vistas diferentes, tanto num sentido direto quando inverso (reverso). Em qualquer caso, sempre um modelo é usado como parâmetro de entrada para ser transformado em outro modelo .

A figura 10.1 mostra o ciclo (sentido) mais natural MDA, partindo do CIM (modelo de requisitos) até o nível mais baixo de código (implementação)

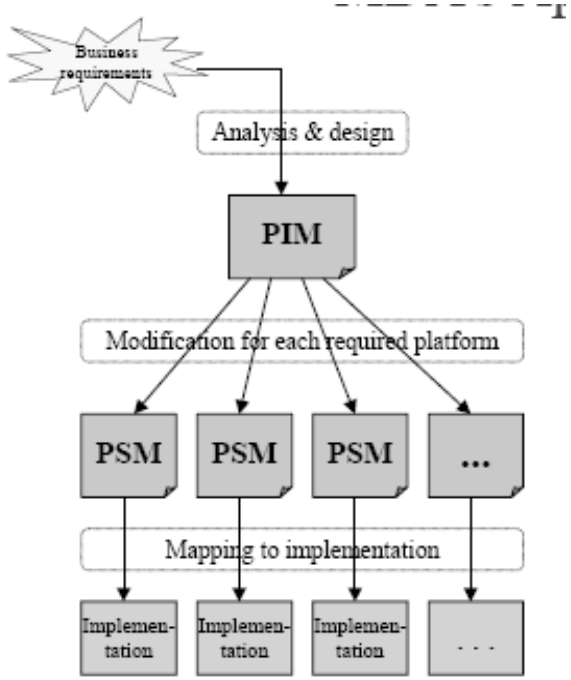


Figura 10.1. Transformações em MDA

Entretanto, é possível também as seguintes transformações (mapeamentos):

- PSM => PIM (Engenharia Reversa)
- PIM => PIM, PSM => PSM (Modelos de mesmo nível
- Implementação => PSM (Engenharia Reversa)
- PIM => Implementação

- **Transformações e Mapeamentos em MDA**

Existe uma quantidade enorme de ferramentas para suportar transformação de modelos. Transformações podem utilizar diferentes técnicas que vão desde uma transformação manual, semi-automática e automatizada. Por exemplo, transformações de PIM para PSM podem ser realizadas através de uso de UML Profiles (extensões UML), uso de padrões (patterns), marcas (markings), metamodelos e transformações automáticas (via algoritmos) [MDA Guide Version 2003]. Os elementos centrais dessas transformações são os mapeamentos das

de regras e técnicas utilizadas para modificar, refinar ou transformar um modelo e se obter um outro modelo. Esse mapeamento, usando-se metamodelos (modelos que descrevem e especificam os modelos originais), facilita a automação. A Figura 10.2 descreve o Metamodelo MDA[MDA 2001]. Observa-se que PIM, PSM e técnicas de mapeamento são baseadas em metamodelos expressos preferencialmente com as tecnologias núcleo do OMG: UML, MOF ou CWM.

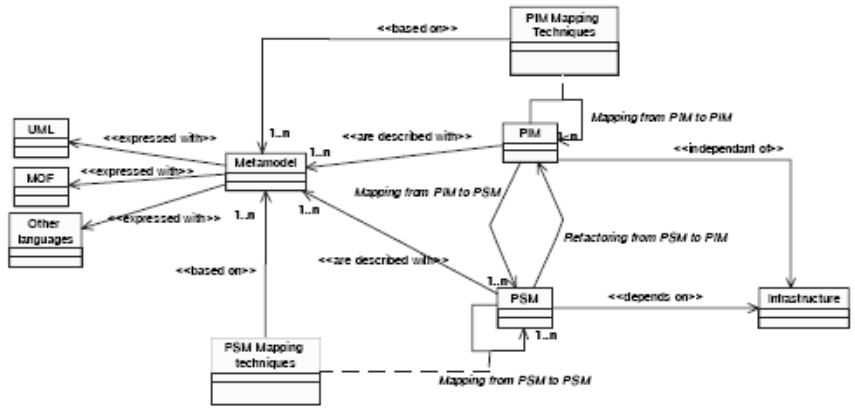


Figura 20.2 Metamodelo MDA

Nota-se ainda que o OMG não contempla nesse metamodelo MDA, seu próprio Modelo Independente de Computação (CIM) o que, sob a ótica de Engenharia de Requisitos e Engenharia de Software é como se deixasse uma grande lacuna "gap" a ser preenchida ou, como se o próprio PIM (UML) se unificasse com o CIM (UML), transmitindo a idéia de ser um único modelo capaz de representar de modo completo e consistente todo um sistema.

Para ilustrar o esquema geral de transformações através de metamodelos, considere a figura 10.3 onde um modelo 1 é transformado num modelo 2, usando como entrada do processo o metamodelo A do modelo 1 e produzindo o modelo 2 expresso no metamodelo B. É importante destacar que para realizar essa transformação é necessário ter regras de mapeamento precisas entre esses metamodelos.

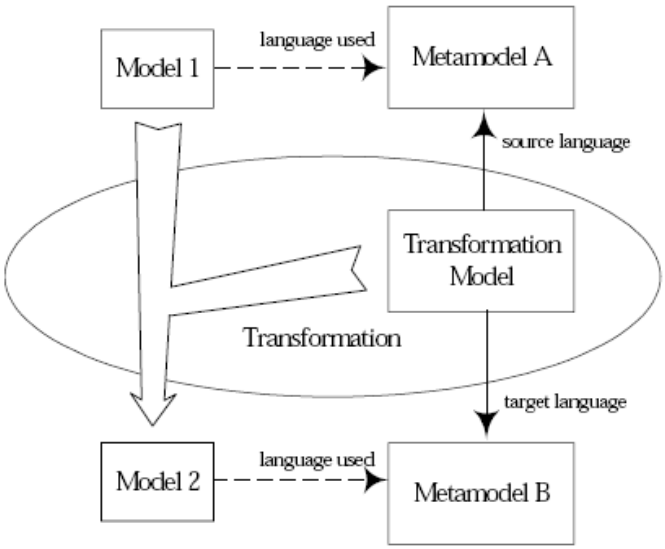


Figura 10.3. Transformações de mapeamentos por metamodelos

De fato, a Linguagem de Modelagem Unificada (UML) é um marco na história de modelagem visual de software, pois antes dela havia várias notações muitas delas incompatíveis entre si. Desde a sua primeira versão (UML 1.0) lançada em 1997, ela recebeu diversas críticas e propostas de extensão. Em 2001, o OMG publicou a UML 2.0.

Alguns dos novos aperfeiçoamentos da UML 2.0 foram:

- Melhor suporte de extensão para outros modelos(linguagens) através do uso de UML Profiles;
- Aperfeiçoamento da expressividade de modelar, incluindo modelagem de processos de negócios, suporte a modelagem de classificadores reusáveis e suporte para modelagem de arquiteturas distribuídas e sistemas heterogêneos;
- Integração com "Actions Semantics" que o desenvolvedor pode usar para definir a semântica de tempo de execução do modelo (aspecto funcional) e prover precisão semântica exigida para analisar modelos e transformá-los em implementações.

Robert B. France em “Model-Driven Development Using UML 2.0: Promises and Pitfalls” [France and Ghosh 2006] cita que padrão UML 2.0 contém um largo conjunto de conceitos de modelagem que são relacionados de um modo complexo. Para cobrir essa complexidade seus

- Infra-estrutura: elementos ou construtores básicos da linguagem.
- Super-estrutura: o próprio metamodelo UML.
- Linguagem de Restrição de Objeto (OCL): especificação de consultas, invariantes, restrições e operações em modelos UML.
- Intercâmbio de Diagramas: extensão do metamodelo (Super-estrutura) para dar suporte armazenamento e intercâmbio de informação de modelos UML.

Além de toda essa complexidade, UML carece de precisão semântica, pois muitos dos seus elementos (primitivas) têm diferentes interpretações e varia conforme entendimento do projetista. Isso causa ambigüidades [France and Ghosh 2006]. Também, Oscar Pastor [Pastor and Molina 2007] afirma que a maioria dos métodos baseados em UML tem conceitos como generalização, associação e agregação tão ambíguos e dependentes da interpretação do projetista que o resultado em termos do produto de software é imprevisível. Isso porque os relacionamentos de classes têm mais semântica do que o proposto por esses métodos. Assim, um modelo conceitual só será preciso se somente esses relacionamentos estão claramente definidos**.**

Essa imprecisão, aliada da ausência de formalismo de seu metamodelo, faz com que sua validação fique comprometida, e como consequência, erros e inconsistências sejam propagados, durante o refinamento desses elementos, para os níveis de menor abstração da UML [Pastor and Molina 2007].

Para dar um melhor suporte MDD, UML 2.0 lançou o conceito de UML Profile. Esse mecanismo de extensão auxilia a transformação de modelos PIM para PSM específicos, conforme esquema ilustrado na figura 10.4:

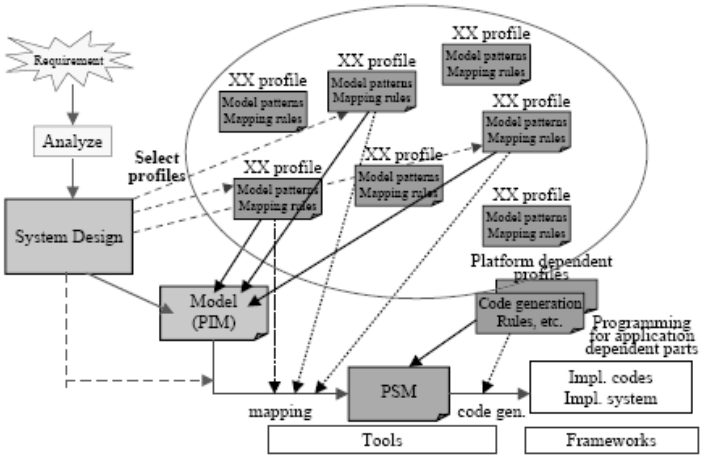


Figura 10.4. Transformações com UML Profile

Atualmente muitas extensões já estão padronizadas pela OMG, algumas estão em processo de padronização e outras ainda em discussão como mostrado na figura 10.5.

OMG(standardized)
- UML Profile for EAI (Enterprise Application Integration)
- UML Profile for EDOC (Enterprise Distributed Object Computing)
  - CCA (Component Collaboration Architecture)
  - Entities Profile
  - Events Profile
  - Business Process Profile
  - Relationship Profile
- UML Profile for Schedulability, Performance and Time
- UML Profile for CORBA

OMG(in process)
- UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms
- UML for Systems Engineering

JCP(standardized)
- UML Profile for EJB (JCP)

Others (discussing, topics, rumor)
- UML Profile for WSDL
- UML Profile for XML Schema
- UML Profile for Persistence Model
- UML Profile for Reverse Engineering
- UML Profile for Framework Architectures
- UML Profile for DCL
- UML Profile for Business Modeling
- UML profile for Business Analysis
- UML Profile for .NET
- UML profile for Interaction design
- UML Profile for Database Design
- UML profile for hypermedia
- UML for Ontology Development
- UML profile for DAML
- UML Profile for Web applications

Figura 10.5. UML Profiles da OMG

Enfim, devido sua imprecisão semântica e complexidade, UML 2.0

OMG na evolução do padrão [France and Ghosh 2006]. Isso não significa subestimar o valor inegável da UML no contexto da Engenharia de Software, entretanto, afirmar que UML vai ser mesmo o futuro do desenvolvimento de software dirigido por modelos (MDD) só o tempo dirá.

10.2.2. Padrões OMG e a Arquitetura MDA

O surgimento da arquitetura MDA em 2001 foi resultado da necessidade cada vez mais emergente de realizar manutenções em aplicações, integrá-las com outros sistemas, mudar suas infra-estruturas, alterar seus requisitos e lidar com a frequente evolução e criação de novas tecnologias. Além disso, MDA objetiva proporcionar os seguintes benefícios: produtividade, portabilidade, interoperabilidade

Para atingir esses objetivos e separar os níveis de abstrações, MDA [OMG 2003] foi definida pela OMG em três camadas conforme figura 10.6, tendo, na primeira camada de especificação (núcleo da arquitetura), padrões que ditam um conjunto de regras para estruturação da especificação expressa nos modelos e que não abordam características de plataformas específicas.

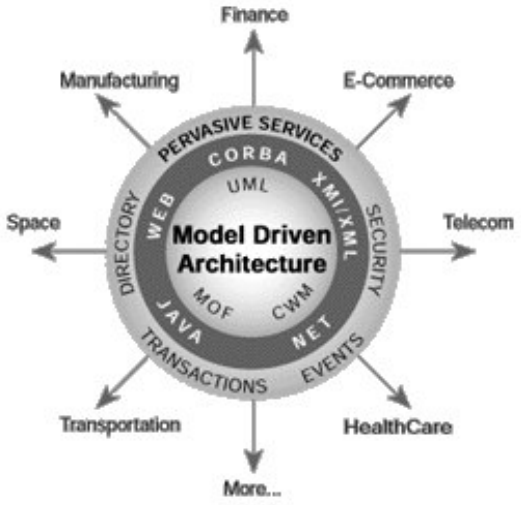


Figura 10.6. Padrões MDA

Esta camada representa o mundo (modelo) PIM. Os padrões também constituem definições propostas pela OMG. São eles:

- Unified Modeling Language (UML): padrão que define uma linguagem

- Common Warehouse Metamodel (CWM): padrão para armazenamento de dados que permite fácil manipulação dos mesmos entre ferramentas e plataformas de armazenamento em ambientes heterogêneos distribuídos;
- Meta Object Facility (MOF): padrão que define uma linguagem abstrata para definição de linguagens de modelagem (metamodelos). Ela é utilizada para descrever modelos da UML, CWM e do próprio MOF, além de definir o formato de intercâmbio para modelos, base do padrão XMI (XML Metadata Interchange);

Na segunda camada, encontram-se os modelos PSM que possuem características próprias a determinadas tecnologias e plataformas. Entre elas, algumas seguem padronização da OMG, elevando a resolução dos problemas de integração através da definição de especificações voltadas para interoperabilidade, que sejam abertas e independentes de fornecedores ou fabricantes específicos. São elas:

- XML Metadata Interchange (XMI): padrão para o intercâmbio de modelos através do mapeamento da linguagem definida pelo padrão MOF para o padrão XML do World Wide Web Consortium (W3C);
- Common Object Request Broker Architecture (CORBA): arquitetura que estabelece e simplifica a troca de dados entre sistemas distribuídos.

Na camada PSM, pode-se ter também outros padrões como JAVA EJB, Microsoft .NET, etc.

Na camada mais externa, são exibidos os serviços que a maioria dos domínios de aplicações necessita, para então, serem apresentados os múltiplos domínios que fazem uso desses serviços. Esses serviços podem ser de segurança, persistência, controle de transações, tratamentos de eventos, etc.

10.3 Abordagens MDD Modelos

Esta seção tem como principal objetivo descrever a abordagem MDD, chamada OO-Method, que apresenta características de um real ambiente MDD através de uma completa transformação de modelos. O OO-Method inova com o conceito de compilador de modelos "model compiler", que de fato, é uma máquina virtual de transformação de modelos. Além disso, o modelo de alto nível, chamado modelo conceitual, do OO-Method tem todos seus elementos (primitivas) descritos numa notação visual (gráfica) e que são especificados numa linguagem formal orientada a objeto (OASIS). Essas características fazem com que o OO-Method tenha precisão sintática e semântica suficientes para prover um ambiente capaz, inclusive, de fazer validação de modelos e consequentemente gerar um produto de software final de qualidade.

Nesta seção, será mencionada a ferramenta que implementa OO-Method,

Por fim, para não deixar de mencionar o poderoso mundo Open Source em expansão, esta seção também citará uma outra abordagem MDD não proprietária que está se tornando bastante popular: AndroMDA.

10.3.1 OO-Method

A primeira versão do OO-Method foi introduzida em 1992 através da tese de PhD de Oscar Pastor, juntamente com a da linguagem formal, de especificação de sistemas de informação – OASIS [Pastor and Molina 2007]. Deste então, o método incorporou um número de componentes até chegar a versão apresentada neste trabalho. Segundo autor [Pastor and Molina 2007], o método cobre todas as fases do processo de desenvolvimento de software, das fases iniciais de obtenção de requisitos e representação, passando pelo desenvolvimento correspondente do esquema conceitual OO, mais a geração do produto final de software numa plataforma específica. O centro do desenvolvimento do software dirigido por modelos do OO-Method é o Esquema (Modelo) Conceitual que tem como leitmotiv a seguinte afirmação do Prof. Antoni Olivé (Olivé 2005) [Pastor and Molina 2007]:

"Para desenvolver um sistema de informação é necessário e suficiente definir seu esquema conceitual"

Esta idéia aparece em trabalhos e propostas de alguns pesquisadores de prestígio. Toni Morgan, defende a idéia de usar "Extreme Non-Programing" [Morgan 2002] como argumento de que a principal atividade no desenvolvimento de software é modelagem, e não programação, pois modelagem está no espaço do problema enquanto programar está no espaço da solução. O objetivo final é tornar verdadeira a sentença "O Modelo é o Código", em vez de "O Código é o Modelo". Tudo isso é possível se obter, quando se tem um Modelo Conceitual Executável que abstrai de modo completo e consistente todos os aspectos estáticos, dinâmicos e de interação (interface usuário) de um sistema, tal como o do OO-Method, passível de transformação através de um compilador de Esquema Conceitual.

10.3.1.1. O processo básico de transformação

OO-Method estabelece uma distinção clara entre o espaço do problema, onde está definido o esquema conceitual, e o espaço da solução, onde é obtido o produto de software que representa o esquema conceitual. Na figura 10.7, o processo se inicia com uma a entrada que representa os requisitos do sistema, não importando por quais processos de engenharia de requisitos esses requisitos foram obtidos, nem o modelo de requisitos utilizado. De forma que esses requisitos são insumos para se criar (projetar) o esquema conceitual. Especificados os quatro modelos que compõe o esquema conceitual: modelo objeto, funcional, dinâmico e de apresentação, é gerado um repositório para conter todos os elementos (primitivas) especificados nos modelos desse esquema conceitual, utilizando-se a linguagem formal orientada objeto OASIS, conforme figura 10.7. Regras de mapeamentos das primitivas desse esquema conceitual para um modelo de aplicação específico de cada plataforma são definidas e, por fim, é

garante que há uma equivalência funcional entre toda primitiva definida no esquema conceitual com sua(s) respectiva(s) primitiva(s) no modelo de aplicação. No exemplo da figura 10.7, foi escolhido um modelo de aplicação (plataforma) constituído por uma arquitetura de três camadas: lógica da aplicação, persistência e interface com usuário.

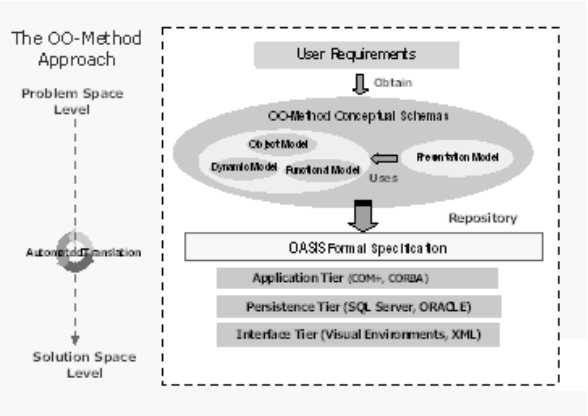


Figura 10.7. Abordagem OO-Method

10.3.1.2.Comparação com MDA

Os modelos do OO-Method, seus mapeamentos e transformações podem ser comparados com o padrão MDA, através da tabela

Tabela 10.1. Comparação do OO-Method com MDA

MDA	OO-Method
Platform-Independent Model (PIM)	Conceptual Model
Platform-Specific Model (PSM)	Application Model
Implementation Model (IM)	Application Code
PIM-to-PSM transformation	Mappings
PSM-to-IM transformation	Transformations

Entretanto, algumas propriedades do OO-Method estão ausentes no padrão MDA (OMG), tais como:

- Em termos de PIM, OO-Method provê uma solução semanticamente precisa, porque está especificado usando uma linguagem formal OASIS que é computacionalmente completa. Em outras palavras, os modelos do esquema conceitual do OO-Method são também computacionalmente completos e contém toda a informação que é necessária para gerar automaticamente código-fonte.

Essas propriedades não são vistas em abordagens que usam UML (Padrão MDA).

10.3.1.3. O Modelo Conceitual

O modelo ou esquema conceitual composto por quatro visões ou modelos que representam os requisitos funcionais de uma aplicação. Esses modelos são: modelo objeto, modelo dinâmico, modelo funcional e modelo de apresentação.

O Modelo Conceitual do OO-Method primou em ter uma notação visual (gráfica) parecida com UML, mas que só usa apenas parte dos seus conceitos (diagramas) que julga necessário e suficiente para representar um sistema de informação. Além disso, a grande diferença, quando comparado com UML, é que o modelo conceitual do OO-Method tem uma semântica e sintaxe bem precisas e, como base, a linguagem formal OASIS. Isso propicia a validação automática do modelo conceitual a fim de não deixar passar falhas (erros) para os modelos posteriores de mais baixos níveis.

Nas próximas seções, serão apresentadas, de modo geral, as principais características desses modelos. Para detalhes mais específicos, consultar referência bibliográfica [Pastor and Molina 2007].

- **Modelo Objeto**

  O modelo objeto especifica as propriedades estáticas do sistema, definido pelo diagrama de configuração de Classe que é composto por:

  - Classes
    - Atributos
    - Precondições e Serviços
    - Restrições de Integridade
  - Relacionamento entre as Classes
    - Associação, Agregação e Composição
    - Herança
  - Agentes

  OO-Method também representa precisamente os tipos relacionamentos como associação, agregação e composição. A associação considera aspectos de cardinalidade, papéis (roles) e também de temporalidade. A temporalidade de uma associação se define como estática ou dinâmica, conforme a associação seja constante (estática), desde o momento em que é criada até o fim do seu tempo de vida.

Além da precisão que o OO-Method trata os conceitos de associação, agregação e composição, ele também considera quais são os impactos que eventos de inserção, deleção e mudança de objetos têm sobre esses relacionamentos entre as classes.

O OO-Method suporta também os conceitos de Herança (generalização e especialização) e herança múltipla. Por fim, o modelo conceitual lida com gerenciamento de complexidade do modelo através da definição de subsistema que tem uma notação visual igual a de uma package em UML.

- **Modelo Dinâmico**

  Este modelo representa o comportamento do sistema, especificando suas propriedades dinâmicas através de dois diagramas:

  - Diagrama de Transição de Estado

    Especifica o ciclo de vida válido dos objetos de uma classe e seus serviços disponíveis em cada estado.
  - Diagrama de Interação de Objeto

Especifica as interações válidas entre os objetos através das transações, operações e gatilhos.

Um gatilho é uma condição sobre um estado do objeto que, tornando-se verdadeira, faz com que este objeto dispare eventos ou transações sobre si mesmo (self) ou sobre outros objetos (object) do sistema. A sintaxe de gatilhos na linguagem formal OASIS é: **<destination>::<condition>:<service>**

Onde, **<destination> := self | object | class | for all**

Sendo que "self" significa para a si mesmo, "object" para uma instância de outra classe, "class" para todas as instâncias da classe e "for all' para um subconjunto de objetos de uma classe.

- **Modelo Funcional**

  O modelo funcional especifica o relacionamento estático e dinâmico através de:

  - Definição semântica relacionada às transições de estado
  - Descrição de como a execução dos eventos muda o valor dos atributos das classes

O modelo funcional trata também questões de eventos de criação e destruição de objetos, além de transações e operações que afetam os estados (valores dos atributos) dos objetos. O modelo funcional do OO-Method, combinado com sua linguagem de fórmula, provê de modo completo e preciso uma solução para especificar os aspectos funcionais de um sistema via modelo conceitual.

O recurso de "Action Semantics" da UML 2.0, para suprir essa necessidade de modelagem de aspectos funcionais, não possui uma semântica definida claramente e precisamente, como também não os têm os elementos da UML. Assim, o modelo

- Acesso a dados de acordo com o Modelo Objeto
- Definição de lógica seqüencial
- Manipulação de classes e objetos
- Manipulação de relacionamentos
- Uso de operadores lógicos, aritméticos e relacionais

- **Modelo de Apresentação**

O modelo da apresentação especifica os requisitos de Interface de Usuário, modelando uma interface abstrata que é independente de plataforma ou dispositivo. Esse modelo de apresentação a nível de análise (conceitual) é considerado uma inovação do OO-Method, em relação outras abordagens que, na maioria, descrevem interface-usuário apenas em nível de implementação. No OO-Method, o modelo de apresentação é organizado em três níveis:

- **Nível 3**: Nível mais baixo, constituído pelos elementos básicos de entrada de dados, seleções, grupos de dados, filtros, critérios de classificação, conjunto de visualização, ações e navegação.

- **Nível 2**: **Unidades de Interação**

  Nível intermediário com conceito fundamental do modelo de apresentação que descreve um particular cenário de interação entre o usuário e o sistema. Geralmente, a interface de usuário de um sistema é definida como uma coleção relevantes unidades de interação e pelo modo como essas unidades estão estruturadas. OO-Method provê quatro tipos de unidades de interação, descritas a seguir:

- **Nível 1**: **Árvore de Hierarquia de Ação**

  Nível mais alto. Uma vez definidos os cenários de interação do nível 2 do modelo de apresentação, faz-se necessário determinar como essas unidades de interação serão estruturadas e apresentadas ao usuário. Essa estrutura caracteriza o nível mais alto da interface com o usuário, o que poderia ser descrito como o "Menu" principal da aplicação. A árvore de hierarquia de ação serve para esse propósito. Ela é estruturada hierarquicamente por uma árvore, tendo um nó raiz e respectivas ramificações até chegar às folhas. Por exemplo, um sistema de aluguel de carros (nó raiz) é organizado principalmente como tendo as seguintes ramificações: Veículos, Clientes, Aluguéis e Usuários.

A construção do modelo de apresentação pode ser realizada de modo "top-down", ou seja, partindo-se do nível 1 até chegar aos elementos do nível 3; ou "bottom-up" , partindo-se do nível 3 até a definição do nível 1.

Além dessas quatro visões do modelo conceitual, o OO-Method dá suporte a interoperabilidade e interface do sistema modelado com outros sistemas externos

Enfim, ao se definir os quadro modelos básicos (objeto, dinâmico, funcional e apresentação) do esquema conceitual do OO-Method, tem-se toda infra-estrutura necessária e suficiente para representar um sistema de informação no contexto do espaço do problema.

10.3.1.4. O Compilador de Modelos

Como OO-Method é segue o processo ideal MDD de transformação de modelos, ele precisa de alguma forma transformar o modelo conceitual, que contém todas as propriedades estáticas, dinâmicas e de interação com usuário (apresentação), em um modelo de implementação (código). Essa transformação é automatizada através do Compilador de Esquema Conceitual que pode ser visto como uma máquina virtual de programação. OO-Method inova com essa idéia de compilador de modelos, diferindo-o de outras abordagens que apenas focam a geração de código a partir de modelos através de técnicas diversas como integração, sincronização de código, etc.

Em comparação com outras abordagens existentes, o OO-Method se destaca por tratar de modo completo e preciso todos os aspectos da compilação de modelo. Um exemplo típico de problemas com outras abordagens são as insuficientes transformações para se obter o produto final de software. Mais grave ainda, a maioria das ferramentas que geram código a partir de modelos, consideram única e exclusivamente como seu modelo inicial de entrada o diagrama de classes em UML, negligenciando todos os demais diagramas que capturam os demais aspectos dinâmicos e de interação de uma aplicação.

10.3.1.5. OLIVANOVA

O OO-Method é implementado através do produto OlivaNova da Care Technologies [OlivaNova 2009] . O principal objetivo de OlivaNova é separar o que deve ser feito (espaço do problema), de como deve ser feito (espaço da solução). Ela é composta de duas principais ferramentas: o modelador e a máquina de transformação. O modelador permite:

- Modelar objetos e negócios;
- Modelar dados;
- Modelar integração;
- Modelar sistemas legados;
- Modelar regras e limitações;
- Definir conceitualmente interfaces do usuário;
- Suporte a UML;
- Suporte a XML.

A máquina de transformação é que implementa todo o processo de compilação de modelos do OO-Method, conforme descrito na seção anterior 10.3.1., gerando código fonte na plataforma de destino.

No sítio da OlivaNova [OlivaNova 2009] existe um quadro que a compara com várias outras ferramentas MDD comerciais, tais como Borland Together , Compuware OptimalJ, IBM Rational Software Architect, etc.

10.3.2. AndroMDA

AndroMDA [AndroMDA 2009] é uma poderosa ferramenta MDA Open Source. Possui arquiteturas como Spring, EJB, .Net, Hibernate, Struts e está desenvolvida sobre o Eclipse. Pode ser utilizada pelos servidores de aplicação Jboss e TomCat e suporta a UML2.0. Agora está em sua versão 4.0, disponível somente para "preview"e permite a criação e utilização de metamodelos no padrão EMF (Eclipse Model Framework) [Projetos Eclipse 2009] e, além disto, possibilita a definição de transformação de modelos PIM a modelos PSM para depois atingir a geração de código fonte, fazendo uso de transformações Model to Text.

Como framework, gerencia qualquer tipo de modelo (geralmente modelos UML guardados no formato XMI) produzido por outras ferramentas case, que combinados aos plugins que possui, permitem a geração de modelos e código fonte.

Em AndroMDA é possível gerar componentes para todas as linguagens: Java, .Net, HTML, PHP. Se desejarmos utilizar alguma tecnologia que ainda não esteja contemplada, somente temos que desenvolver plugins para isto (ou mudar algum que já exista). É mais comumente utilizado por programadores da tecnologia J2EE, inclusive

podendo gerar um projeto J2EE e seu código partindo de um modelo de classe. É possível definir gerar código para Hibernate, EJB, Spring, WebServices e Structs e o código gerado é automaticamente adicionado ao projeto e ao processo de compilação.

Sua realidade MDA permite fazer com que o trabalho de arquitetura e desenvolvimento seja mais curto e de mais qualidade, trabalhando com modelos independentes de plataforma que posteriormente serão refletidos em modelos UML (PSM). Isto permite, entre outras vantagens, ter foco no modelo e necessidades organizacionais (Modelo PIM) e a possibilidade de reutilizar o modelo PIM em outros projetos.

Como etapa seguinte se pode efetuar a transformação até o código fonte da aplicação, tendo como etapas intermediárias a geração de um ou mais modelos PSM. Neste ponto, é onde AndroMDA mais se destaca, por possuir muitos plugins já desenvolvidos e que realizam a transformação PIM > PSM em muitos tipos de linguagens e tecnologias diferentes. Estes plugins são chamados cartuchos "cartridges" e utilizá-los são bastante fáceis.

Além das vantagens citadas anteriormente, destacamos como pontos positivos de AndroMDA: não desenvolvimento de código redundante, o código reflete exatamente o que definem os modelos e a possibilidade de alterar de "cartridge" para que gere o mesmo sistema em outras linguagens. Para se desenvolver novos cartuchos para qualquer plataforma específica deve-se basicamente identificar as regras de transformação e criar um perfil (profile) UML

Entretanto, o nível mais alto de abstração de AndroMDA depende da ferramenta de modelagem que gera UML (PIM). Assim, o nível mais alto é o conceito de caso de uso. A ferramenta de melhor aceitação para modelar em UML e, fazer exportação do metamodelo UML em XMI é a Magicdraw.

AndroMDA não provê recursos de definição de interface usuário abstrata tal

(PIM E PSM) e trata questões de rastreabilidade e validação de modelos de forma limitada.

A versão AndroMDA 4.0 que está sendo desenvolvida visa aperfeiçoar o processo de transformação de modelos e, principalmente, receber como entrada "input", no modelo inicial de mais alto nível, metamodelos de qualquer linguagem de domínio específico. Isso, tornará o ambiente MDD de AndroMDA capaz de importar qualquer outro modelo de sistema, e não apenas, UML.

10.4 Problemas e Desafios dos Processos MDD

10.4.1. Visão Geral

Pode-se considerar que desenvolvimento de software dirigido a modelos ainda não está num nível de maturidade suficiente para realizar o sonho acalentado de todo desenvolvedor: ter um produto final de software com qualidade e 100% gerado automaticamente a partir dos seus requisitos. Nos melhores ambientes, o desenvolvedor ainda consegue modelar a nível de análise e projeto, mas não a nível de requisitos. Com isso, o desenvolvedor, dessa primeira década do terceiro milênio, ainda realiza esforço manual considerável a nível de projeto e implementação. E isso, tem impactos negativos nos custos, prazos e qualidade dos softwares produzidos. O paradigma dirigido a modelos traz uma promessa de tornar realidade esse sonho. Entretanto, muitos desafios haverão de ser superados.

Sabe-se que processo MDD ainda está na sua infância. Nem as linguagens (modelos) nem as ferramentas se desenvolveram o suficiente para concretizar suas promessas feitas. O processo MDA, padronizado pela OMG, é apenas uma referência e pode ser usado por qualquer outro processo específico de desenvolvimento de software existente (RUP, XP, OPEN, Agentes, Aspectos, Formais, etc.) desde que se adapte e seja dado um foco especial em modelos e suas transformações. Decerto que processos como RUP e OPEN, por suas próprias características, são mais fáceis de serem adaptáveis a um processo dirigido a modelos do que o XP cujo foco predominante é implementação.

Este capítulo do livro procurou relatar alguns aspectos dessa tecnologia MDD. Como trabalhos futuros e desafios para o processo desenvolvimento de software dirigido por modelos, destacam-se os seguintes:

- Integração com a fase requisitos (Engenharia de Requisitos) para elevação do nível de abstração inicial a ser modelado (modelo CIM da MDA);

- Suporte a modelos orientados a metas “goals”, agentes e aspectos.
- Melhor precisão semântica dos modelos em relação às características estáticas, dinâmicas e de apresentação (interação-usuário);
- Melhores mapeamentos entre os modelos;
- Melhor transformação automática de modelos (automação);
- Melhor suporte à Validação de Modelos;
- Melhor integração com as plataformas específicas (PSM);

- Maior suporte à rastreabilidade;
- Melhor suporte à engenharia reversa;
- Suporte à computação autonômica;
- Suporte a ontologias.

10.4.2. Lições Aprendidas na adoção de soluções MDA

Muitas organizações que, nos últimos anos, vêm utilizando com sucesso soluções MDA, perceberam que um conjunto de práticas e passos consistentes devem ser considerados ao se adotar um processo MDD automatizado. A seguir, é mostrado um resumo com os passos necessários para se desenvolver uma solução MDA adequada:

- Examinar os modelos atualmente usados na empresa no seu processo de desenvolvimento e a conexão/correlação semântica entre os elementos desses modelos.
- Identificar as transformações candidatas para automação
- Especificar (documentar) os requisitos dessas transformações
- Criar os UML Profiles necessários
- Desenvolver o código da(s)s transformação(coes)
- Esboçar documentos de uso, empacotar e distribuir

10.4.3. O programa FastStart da OMG

Recentemente, o grupo OMG lançou o programa "FastStart" para ajudar as organizações aprenderem sobre MDA e aplicar MDA nas arquiteturas de seus sistemas, na integração dos sistemas e nos seus processos de desenvolvimento de software. Durante programa FastStart , a organização recebe consultores da OMG para realizarem as seguintes atividades:

1. Análise inicial MDA
2. Revisão da Arquitetura Empresarial MDA
3. Plano de Transição MDA
4. Seminários Executivos MDA
5. Prática MDA

Essas atividades duram em média 5 semanas e ajudam à equipe executiva e técnica da empresa a:

- Analisar claramente e planejar como MDA pode melhor ser introduzido e aplicado para melhor beneficiar a organização e seus processos de negócios chaves
- Capacitar a empresa em MDA para que ela própria, sem ajuda de provedores externos, desenvolva seu processo MDA/MDD.

10.5. Tópicos de Pesquisa

O Processo de desenvolvimento de software dirigido por modelos (MDD) ainda é um tema recente. Os benefícios do padrão MDA ainda não foram bem entendidos pelas organizações. Existem várias linhas e tópicos de pesquisa relacionados com MDD/MDA, entre estes se destacam:

1. Desenvolvimento de modelos precisos semanticamente e completos para facilitar transformações e validações.
2. Desenvolvimento ou aperfeiçoamento de ferramentas de apoio ao processo MDD
3. Adaptações do processo MDD aos processos específicos de desenvolvimento de software
4. Extensões MDA para novas plataformas
5. MDA em Linhas de Produtos de Software(LPS)
6. Rastreabilidade em processos MDD
7. Medição/Estimativas de projetos de software em ambientes MDD
8. MDD para sistemas embutidos (Embedded systems development)

10.6. Sugestões de Leitura

Este capítulo propôs dar uma visão sobre Desenvolvimento de Software Dirigido por Modelos. Entretanto, devido ao enorme escopo e dinamismo dessa área, realizou-se um esforço considerável para contemplar, bem como resumir de modo claro e objetivo, um maior número de tópicos possíveis sobre o tema, ou seja, tentou-se elaborar um trabalho que fosse o mais científico, atualizado e completo possível. Porém, sabe-se que qualquer estudo, por mais minucioso que seja, está longe de esgotar o assunto.

Para complementar o material apresentado neste capítulo, o autor sugere que o leitor realize as seguintes leituras:

- UML executável (OMG): Usando apenas os diagramas de classes, de estado e a extensão UML "Action Semantics" torna um modelo UML capaz de ser executável e o transforma, através de compiladores de modelos específicos, em plataformas como C++, Java, etc.

  Para mais detalhes sobre esse tópico, ver livro: **Executable UML, A Foundation for Model Driven Architecture**, Mellor-Balcer, Addison-Wesley, 2002 e o site http://en.wikipedia.org/wiki/Executable_UML.

- Grupo de pesquisa "Precise UML- PUML": Como visto nas seções deste capítulo 10.2.1[France and Ghosh 2006] e 10.3.1 sobre o OO-Method, UML carece de precisão semântica, por exemplo, dependendo da interpretação do projetista pode-se modelar uma um determinado objeto como agregação, composição, associação ou herança. Visando aperfeiçoar a precisão semântica e formalidade da linguagem de modelagem de propósito geral UML, há um grupo de trabalho desenvolvendo Precise UML-PUML em http://www.cs.york.ac.uk/puml/maindetails.html.

- Ambiente MDD "Moskitt" da Universidade Politécnica de Valencia: O Kit de Modelagem de Software (Modeling Software KIT-MOSKitt) é um ambiente case MDD de código aberto (livre) construído sobre o Eclipse IDE, desenvolvido pelo Ministério Regional de infra-estrutura e transportes de Valencia (Espanha). Para maiores informações ver http://www.moskitt.org/eng/moskitt0/.

- Ambiente MDD Open source "OPENMDX": Ferramenta concorrente do AndroMDA, roda no Eclipse IDE, sendo considerado também o estado da arte em desenvolvimento dirigido a modelos. Boa documentação adicional, inclusive ferramenta disponível para download em http://www.openmdx.org/.

- Livros específicos sobre o tema:

  - **MDA Explained: The Model Drive Architecture:Pracice e Promise** by Anneke Kleppe, Jos Warmer, Wim Bast (Editora Addison Wesley 2003): Obs. Muito bom para uma introdução sobre MDA.
  - **Model-Driven Software Development** by Sami Beydeda, Matthias Book , Volker Gruhn (Editora Springer 2005): Obs. Excelente pelo nível teórico básico , avançado e técnico.
  - **Real-Life MDA: Solving Business Problems with Model Driven Architecture (Interactive Technologies)** (Editora Morgan Kaufmann, 2006): Obs. Totalmente prático, focado em soluções de processos de negócios, utilizando-se MDA/MDD.

10.7. Exercícios

Para sedimentar melhor os conhecimentos teóricos abordados nesse capítulo, os seguintes exercícios foram elaborados:

1. MDA usa modelos distintos para separar os sistemas/aplicações em os níveis de abstrações/visões distintos. Quais são modelos definidos no padrão MDA e descreva seus propósitos.
2. Descreva a técnica de transformação de modelos através de metamodelos.
3. Quais os padrões que estão no núcleo da arquitetura MDA do grupo OMG.
4. Discuta: MDA determina o uso ou recomenda algum processo específico de desenvolvimento de software, tal como RUP, XP ou outro qualquer? Como se poderia adaptar o RUP ou XP para utilizar um processo dirigido a Modelos (MDD/MDA)?
5. Caracterize o modelo conceitual do OO-Method

7. Compare as ferramentas CASE de desenvolvimento de software dirigido a modelos: OLIVANOVA e AndroMDA.
8. Suponha uma ferramenta MDD que dá suporte completo aos modelos CIM, PIM e PSM. Durante as manutenções dos aplicativos desenvolvidos com essa ferramenta qual é o único desses modelos que, segundo o padrão MDA, deve ser alterado?
9. Quais os benefícios em se adotar uma processo de desenvolvimento de software dirigido a modelos?
10. Usando uma ferramenta de modelagem UML (MagicDraw, ArgoUML,etc) modele a seguinte aplicação simplificada para administração de alunos, utilizando o diagrama de classe da figura 10.8. As funcionalidades (operações) básicas da classe Aluno são inserir, excluir, listar e alterar, conforme diagrama de atividades da figura 10.9. Depois, instale uma das ferramentas free MDD (AndroMDA ou OPENMDX), configure-as adequadamente e importe/utilize o modelo UML. Por fim, tente usar ferramenta MDD para gerar seu aplicativo na plataforma JAVA JEE, inclusive com recursos de persistência num banco escolhido (Mysql, Postgresql, etc) e "deployment" num servidor JSP Tomcat Apache. Talvez, você tenha tido dificuldade em instalar e configurar esses ambientes de desenvolvimento, mas que achou do processo de transformar seu modelo PIM em PSM automaticamente? Que achou de obter o código do aplicativo automaticamente? Tente fazer uma alteração (evolução) do aplicativo para adicionar disciplinas, professores e seus respectivos relacionamentos com alunos, lembrando-se de que a alteração deverá ser realizada apenas no modelo PIM (mais alto nível de abstração).

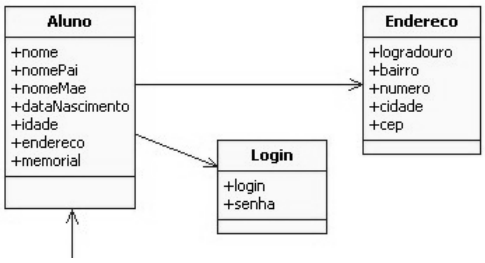


Figura 10.8 Diagrama de Classes

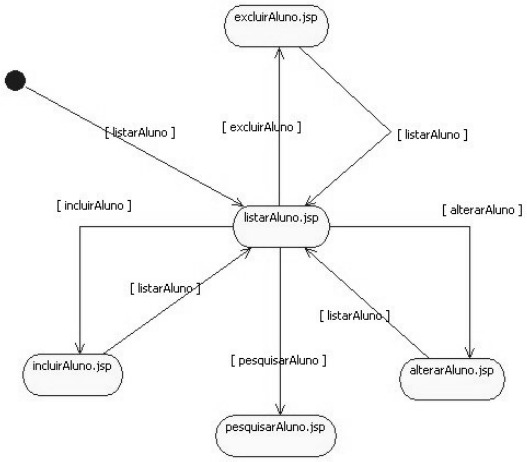


Figura 10.9 Diagrama de Atividades

Referências

MDA productivivity case study. The Middleware Company (2003). <http://www.omg.org/mda/presentations>. Acesso em julho 2009.

OMG: padrão MDA (2003).< http://www.omg.org/mda >. Acesso em julho 2009.

Pastor O.; Molina J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer Publisher 2007.

OlivaNova Care Technologies, Denia, Spain (<http://www.care-t.com> Acesso em julho 2009.

Kontonya, G. e Sommerville, I. (1998) Requirement Engineering: Processes and Techniques, John Wiley & Sons.

MDA Guide Version 1.0.1, Document Number: omg/2003-06-01 Date: 12th June 2003. http://www.omg.org/mda/

Hitachi M. O.; MDA and System Design. Presentation at MDA Information Day, OMG Technical Meeting, April 2002.

Model Driven Architecture (MDA). Document number ormsc/2001-07-01, Architecture Board ORMSC1, July 9, 2001. Disponível em <http://www.omg.org/mda/presentations>. Acesso em julho 2009.

France R. B.; Ghosh S.; Dinh-Trong T. : Model-Driven Development Using UML 2.0: Promises and Pitfalls. In IEEE *Computer*, vol. 39, no. 2, pp. 59-66, Feb. 2006.

Morgan T (2002) Business rules and information systems – aligning IT with business goals.Addison-Wesley, Reading,MS.

Pastor, O.: Model-Driven Development: The OO-Method Aproach. Presentation at UFPE, Recife, Brasil August 2008.

AndroMDA. <http://www.andromda.org>. Acessado em julho 2009.

Projetos Eclipse <http://www.eclipse.org/projects/> Acesso em julho 2009.

Capitulo 16

Modelagem de Processos

André Luis Rodovalho Bezerra

1. Introdução

Modelagem de Processos significa desenvolver diagramas (Diagramas de Processos) que mostram as atividades da empresa, ou de uma área de negócios, e a seqüência na qual são executadas. Muitos negócios são relativamente complexos, assim um modelo poderá consistir de diversos diagramas, e o alvo da modelagem é ilustrar um processo completo, permitindo aos gestores, consultores e colaboradores melhorarem o fluxo e aperfeiçoarem o processo.

É para a construção desse diagramas atualmente temos algumas linguagens para a construções desse diagramas de processos, onde neste capítulo iremos focar nas linguagens **BPMN e SPEM** alguns dos seus objetivos, especificações e notações, mostrando algumas ferramentas existentes para modelagem das mesmas e finalizando com uma comparação entre elas.

Comment [A31]: Formatar Tabulação para 1,27cm (ver os demais parágrafos)
- Também na formatação de todos os parágrafos, o **espaçamento antes** é de 6 pontos

1.1. O que é Modelagem de Processos

BRANÇO

Comment [A32]: Não devem existir espaços (linhas em branço) entre as seções e/ou parágrafos, basta ajustar a formatação do parágrafo para ter **espaçamento antes** de 6 pontos

Modelar processos ajuda a entender como funciona uma organização. Modelar um processo pode ser bastante difícil na prática, principalmente quando é a primeira vez, e lembrando que um processo pode permear diversas áreas funcionais, o que requer um trabalho conjunto de pessoas destas áreas funcionais. Durante este trabalho, os participantes apresentam um aumento do entendimento do negócio.

O **modelo** é um ponto central para que os participantes definam mudanças para melhoramento do processo ou mesmo um desenho completamente novo. Pode ser identificado se um processo é eficiente e eficaz, ou mesmo antecipar sua complexidade, redundâncias e não conformidades (problemas). Se o processo é alguma coisa nova que a empresa está planejando executar, o modelo pode ajudar a assegurar sua eficiência desde o início.

Comment [A33]: Evitar colocar / . O mais correto seria usar o sinal de pontuação parenteres: (eficaz)

A **comunicação do processo**, de forma eficiente, para outras pessoas é fundamental. Por melhor que seja um processo, se a comunicação para outros for deficiente, principalmente para aqueles que vão implementar o processo, o esforço desenvolvido pela equipe terá sido em vão. Bons modelos de processos ,são a chave para a comunicação.

Os resultados da modelagem de um processo são essencialmente: acréscimo de valor para o cliente e redução de custos para a empresa. O que, conseqüentemente, conduz a empresa ao aumento de lucros. Um diagrama de modelo de processo de negócio é uma

ferramenta, ou seja, um meio para se atingir um fim determinado e não um resultado de desempenho por si só.

A saída final deve ser a melhoria na maneira como o processo do negócio funciona, o foco das melhorias está nas ações que geram valor agregado ao negócio, ou seja, que melhoram o serviço e a experiência do cliente, além de reduzir tempo e esforço gastos.

Utilizamos BPI(Business Process Improvement), para a melhoria de processo de negócio, ele é uma metodologia (abordagem) que ajuda a otimizar e entender os processos de negócio com objetivo de alcançar as metas e melhorar os resultados dos processos.O primeiro passo da BPI é determinar o cenário atual dos processos, ou seja, AS-IS(que o cenário atual) e depois definir o cenário futuro, ou seja, TO-BE.

OS dois tipos principais de modelos de processo de negócio:

- Modelo "**as is**" or baseline (a situação atual);
- Modelo "**to be** "(a nova situação pretendida).

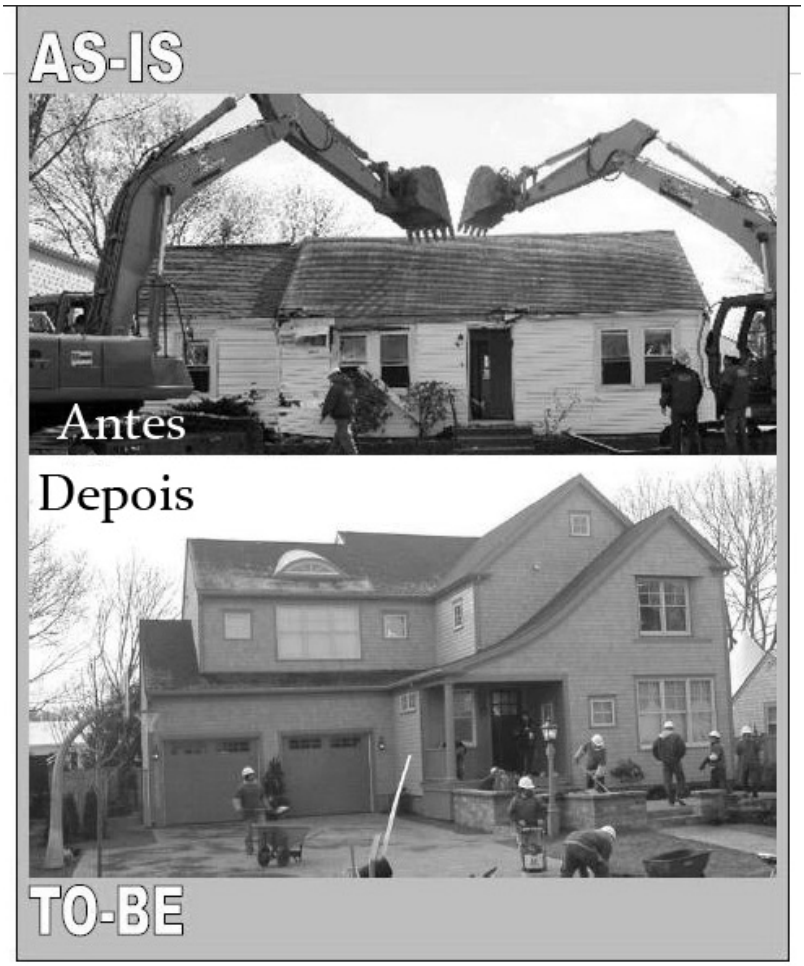


Figura x- exemplo de define os modelos AS-IS e TO-BE

Os quais são utilizados para analisar, testar, implementar e melhorar o processo. outras conseqüências secundárias que se alcançam com a modelagem de processo bem sucedida podem ser o aumento da vantagem competitiva, o crescimento no mercado, e a melhoria de moral e retenção de colaboradores.

Não há nenhuma regra absoluta para o escopo ou extensão de um modelo de processo Em termos de departamentos e atividades cobertas.

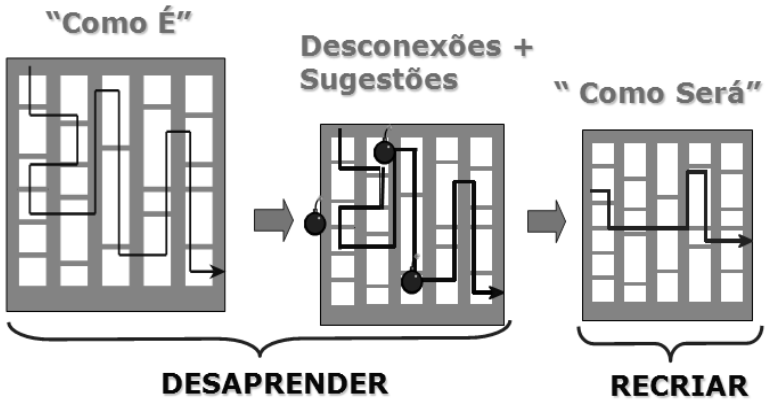


Figura 1(Modelagem de Processos)

Comment [A34]: Não tenho certeza, mas acho que as figures devem ter fontes de onde foram extraídas (referências bibliográficas)
- Fonte de figuras deve ser Helvética 10

2.1. Objetivo da Modelagem de Processos Software

Comment [A35]: Suponho que esta deva ser a segunda seção do capítulo, ou seja, 16.2 . Portanto, deveria ser com fonte tamanho 13

A modelagem de processos de software tem sido utilizada na engenharia de software ao longo dos anos para melhor entender, gerenciar e controlar o processo de desenvolvimento. Contudo, a descrição dos processos do cliente oferece uma nova perspectiva aos engenheiros de software: a necessidade de diferentes abordagens e o uso de diferentes notações e técnicas.

O principal objetivo da modelagem de processos é representar os processos de uma maneira clara e formal em diferentes níveis de abstração. A disponibilidade de modelos completos permite uma análise crítica das atividades existentes para definir melhorias e racionalizações dos processos

3.1 Vantagens e Desvantagens da Utilização de Modelagem de Processos

Comment [A36]: Fonte 13, seção 16.3

A. Vantagens

Comment [A37]: Não precisa colocar Vantagens e Desvantagens como subseções, da seção 16.3, basta colocá-as em destaque ou como tópicos num nível superior sem numeração.:
Ex:
- Vantagens
  - X, Y,Z
- Desvantagens
  - W,V, T ..

Temos algumas vantagens com a modelagem de Processos:

- Bons modelos de processos, são a chave para a boa comunicação.
- Se o processo, é alguma coisa nova que a empresa está planejando executar, o modelo pode ajudar a assegurar sua eficiência desde o início.
- Revelar anomalias, inconsistências, ineficiências e oportunidades de melhoria, permitindo à organização que se compreenda melhor e auxiliando na reengenharia desses processos.
- Fornecer visão clara e uniformizada das atividades, suas razões e formas de execução.
- Utilizar o modelo como um meio para distribuição de conhecimento dentro da organização e treinar as pessoas, ajudando-as a conhecer melhor seus papéis e as tarefas que executam.

B. Desvantagens

4.1. Linguagem de Modelos de Processos

Comment [A38]: Fonte 13, seção 16.4 . Fazer um parágrafo simples de apresentação da seção, antes de falar da linguagem.

4.1.1. BPM

Comment [a39]: Esta seria a seção 16.4.1 do capítulo.

O que é Gestão de Processos de Negócios (BPM) do acrônimo em inglês “Business Process Management “, segundo esses autores:

“Um conjunto de técnicas para garantir que os processos sejam continuamente monitorados e melhorados.”[RUMMLER, G.;BRACHE]

Comment [A40]: Referência incorreta, falta o **ANO** de publicação . Idem para as demais do Capítulo

“Desenvolvimento e manutenção de uma arquitetura de processos de negócios, que se utiliza de técnicas, metodologias e gerenciamento humano para garantir que os processos sejam continuamente melhorados e monitorados.”[Pinto Filho, J. B.]

BPM , envolve modelagem, execução, monitoramento e análise de processos de negócios e, nesse contexto , “Modelagem de processos de negócios, é o conjunto de conceitos e técnicas que visam a criação de um modelo com os processos de negócio existentes em uma organização. Esta "modelagem" é utilizada no contexto da gestão de processos de negócio. [1]

Comment [A41]: Referência não está conforme

Figura 2 (BPM)

Comment [A42]: - Fonte de figuras deve ser Helvética 10

O Business Process Management Initiative (BPMI) desenvolveu um padrão Business Process Modeling Notation (BPMN). O BPMN especificação 1.0 foi liberado ao público em maio de 2004.Esta especificação representa mais de dois anos de esforços por parte do BPMI Notation Working Group. Em 2005 BPMI se funde com o grupo OMG, e BPMN agora está na versão 1.2 e com uma proposta de especificação da 2.0.

Comment [A43]: Falar que em 2005 BPMI se funde com o grupo OMG, e BPMN agora está na versão 1.2 e com uma proposta de especificação da 2.0 .
Ver http://www.omg.org/bpmn/

Ela possui uma forma de apresentações mais amigável aos usuários de negócios que

negócios e TI, assim permitem aos analistas de negócios entederem um modelo de processos independente da ferramenta de modelagem utilizada, bem como construírem modelos de processos que sejão entendidos por outros analistas sem a nescessidade de treinamento especial.

Muitas vezes, o processo inclui atividades que são realizadas fora da empresa (realizado porterceiros, por exemplo) e não temos gerência sobre a execução destas atividades Utilizamos um modelo abstrato para representar uma “entidade”

independente, com processos próprios, mas que não podemos modelar (por não conhecer o processo) ou não nos interessa modelá-lo;

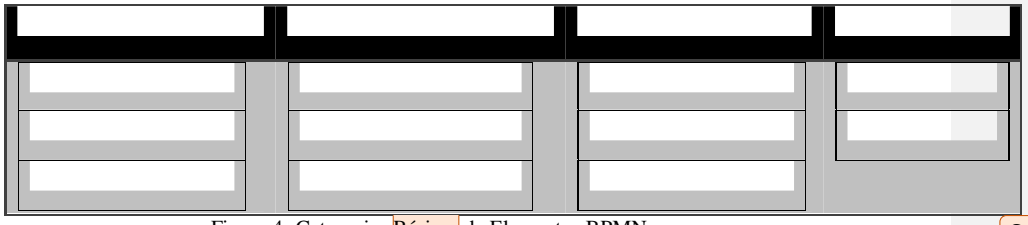


Figura 4- Categorias Básicas de Elementos BPMN

Comment [A44]: - Fonte de figuras deve ser **Helvética** 10
** Idem para todas Figuras do Texto.

A BPMN define um único tipo de diagrama, chamado de BPD (Business Process Diagram).Como forma de exemplificar isto, observe que na Figura 5.5 são dispostos

alguns dos elementos que formam esta notação e o diagrama representa parte da modelagem do processo de compra de um produto em uma loja

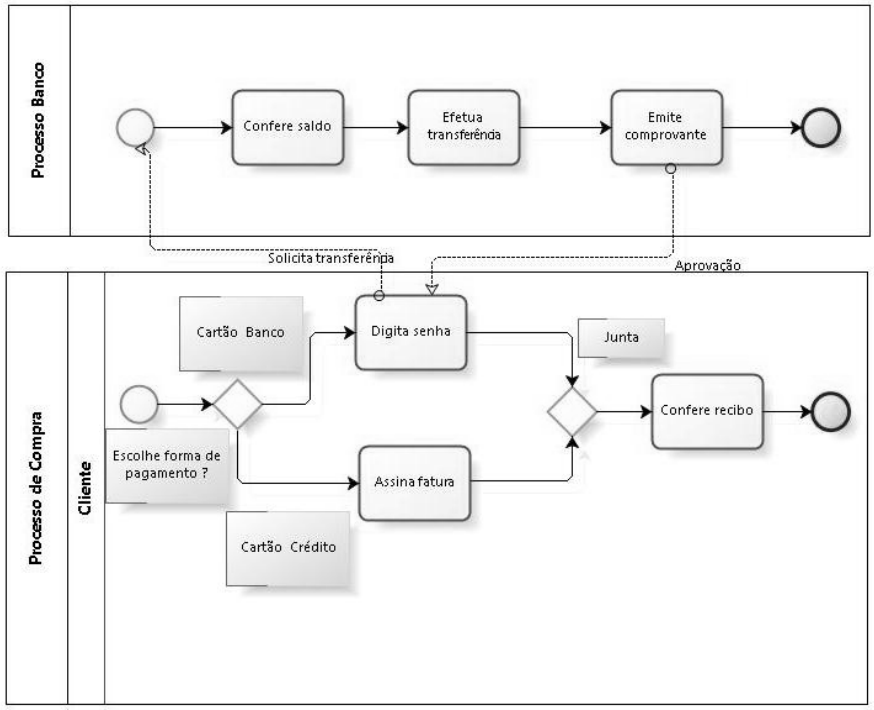


Figura x- Diagrama de Processos de uma loja.

Comment [a45]: Colocar um exemplo de processo modelado com BPMN

O SPEM foi desenvolvido e é mantido pelo OMG (Object Management Group) desde 2002 resultado do esforço coletivo de pesquisadores e consultores, tais como: Empresas: IBM, Rational, Computer Associates, Toshiba, Siemens, etc. e pesquisadores como: Philippe Kruntchen, Craig Lairman, e diversos outros. para tentar suprir a necessidade de um padrão para as técnicas de modelagem de processo

software, utilizando uma abordagem orientada a objetos e a UML (Unified Modeling Language, linguagem gráfica para modelagem de sistemas discretos, de maior aplicabilidade na área de projeto de software orientado a objetos; a UML 1.4, de Jan 2001, é a versão referenciada pelo SPEM) como notação. É Atualmente o SPEM está na versão 2.0 de Abril de 2008. O SPEM é baseado em uma arquitetura de quatro camadas, como mostra a figura 5.

O SPEM utiliza mecanismos de extensão da semântica padrão da UML, para adaptá-la ao propósito da modelagem de processos, que são: estereótipos, valores atribuídos e restrições. Um estereótipo, por exemplo, estende o vocabulário da UML, permitindo a criação de novos blocos de construção derivados dos já existentes. Um fator que favorece a escolha do SPEM para a definição de processos é que ele tanto define capacidades de modelagem dedicadas ao domínio do processo de software, quanto se beneficia da expressividade da UML. Assim, desenvolvedores de software que estejam

familiarizados com a UML podem reutilizar seus conhecimentos de modelagem de software no domínio da modelagem de processos de software.

AQUI

Comment [a46]: COLOCAR UM EXEMPLO SIMPLES DE PROCESSO MODELADO COM SPEM, pode ser o RUP, o mesmo de um material enviado pelo Prof.

uma pré- condição.

4.2.1 OMG(Object Management Group)

Comment [a47]: Ver numeração da seção que seria 16.4.2

Modelagem Padrões da OMG's, incluindo a Unified Modeling Language ™ (UML ®) e Model Driven Architecture ® (MDA ®), que permitem um desenho com um visual forte, e a execução e manutenção de software e de outros processos, incluindo sistemas de TI Modelagem de Processos e Gestão Empresarial. Padrões de middleware OMG's e perfis com base no objeto comum Request Broker Architecture (CORBA ®) que é suportado em uma ampla variedade de indústrias.

Comment [a48]: Incluir também neste parágrafo SPEM e BPMN , vistos nas seções anteriores, como padrões da OMG.

OMG é também o produtor do evento para a Internacionalização. OMG vem produzindo eventos durante quinze anos e fornece uma gama completa de serviços de conferência de gestão e desenvolvimento de competências e conferência, produção e

quatro Reuniões Técnicas da OMG e nove conferências e workshops a cada ano para os seus membros, em locais espalhados pelo mundo.

explícita do processo de negócio é usada como ponto de partida para a configuração destes sistemas.

Business Process Management “Concepts, languages, Architectures”,

Comment [a49]: Colocar texto descritivo , caso o leitor queira se aprofundar no assunto , para cada r sugestão de leitura

Referencias

[2] http://www.bpmn.org

Comment [a50]: Retirar os colchetes das referencias e para cada referencia web, complementar com "acessado em ..... "

- Também colocar as referencias em ordem alfabética.

Comment [a51]: Referencias [1] e [2], [15] colocar no padrão, p exemplo: BPMN , HTTP://www.bpmn.org, Acessado em outubro 2009.

Índice do Capítulo

6.1. INTRODUÇÃO 195

6.2. O QUE É QUALIDADE? 195

6.3. COMPETITIVIDADE X PRODUTIVIDADE 196

6.3.1. CONCEITO DE PRODUTIVIDADE 197
6.3.2. CONCEITO DE COMPETITIVIDADE 198

6.4. QUALIDADE TOTAL 200

6.4.1. DEMING 200

6.4.2. JURAN 201

6.4.3. CROSBY 201

6.4.4. FEIGENBAUN 202

6.4.5. ISHIKAWA 202

6.5. CONTROLE DA QUALIDADE TOTAL 204

6.5.1. APRESENTAÇÃO DO CONTROLE DA QUALIDADE TOTAL 205

6.5.3. PRINCÍPIOS DA QUALIDADE TOTAL 207

6.6. CONTROLE DE PROCESSO 210

6.6.1 CONCEITO DE PROCESSO 211
6.6.2 CONCEITO DE CONTROLE 211
6.6.3 CONCEITO DE CONTROLE DE PROCESSO 212
6.6.4 MÉTODO DE CONTROLE DE PROCESSO 212

6.7. FERRAMENTAS DA QUALIDADE 214

6.7.1. AS SETE FERRAMENTAS DA QUALIDADE 214

6.8. GESTÃO DA QUALIDADE 218

6.8.1. GERENCIAMENTO PELAS DIRETRIZES 218
6.8.2. GERENCIAMENTO DA ROTINA 219

6.9. GARANTIA DA QUALIDADE 220

6.10. QUALIDADE NA INTERFACE COMPRAS/VENDAS 221

6.10.1. QUALIDADE NAS VENDAS 221
6.10.2. QUALIDADE NAS COMPRAS 223

6.11. IMPLANTAÇÃO DO TQC 225

6.11.1 FUNDAMENTOS 225
6.11.2 ORGANIZAÇÃO PARA IMPLANTAÇÃO 225
6.11.3 SISTEMA DE GERENCIAMENTO DA IMPLANTAÇÃO DO TQC 227

6.12 TÓPICOS DE PESQUISA 228

6.13 SUGESTÕES DE LEITURA 228

6.14 EXERCÍCIOS 228

6.15 REFERÊNCIAS 229

O grande objetivo das organizacões humanas é atender às necessidades do ser humano

que atende perfeitamente, de forma confiável, de forma acessível, de forma segura e no tempo certo, às necessidades do cliente.

$$\text{Produtividade} = \frac{\text{Output}}{\text{Input}}$$

$$\text{Produtividade} = \frac{\text{Valor Produzido}}{\text{Valor Consumido}} = \text{Taxa de valor agregado}$$

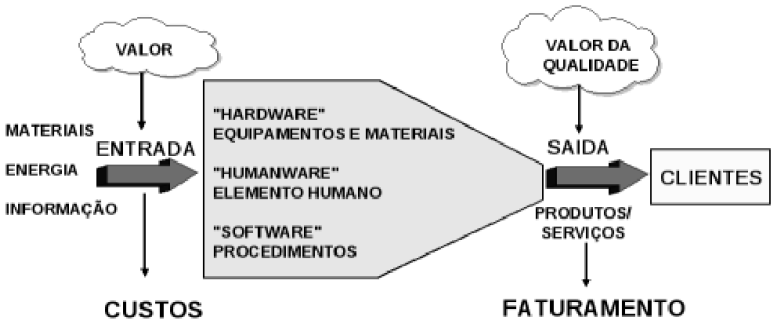
Quanto maior a produtividade de uma empresa, mais útil ela é para a sociedade, pois está atendendo às necessidades dos seus clientes a um baixo custo.

$$\text{Produtividade} = \frac{\text{Qualidade}}{\text{Custos}}$$

$$\text{Produtividade} = \frac{\text{Faturamento}}{\text{Custos}}$$

- É necessário fazer "aporte de conhecimento" de maneira a aumentar o ativo de conhecimento da empresa;

Comment [abv52]: Seria interessante colocar uma "nota de rodapé" informando o que quer dizer esse "aporte de conhecimento"


VALOR
VALOR DA QUALIDADE
MATERIAIS
ENERGIA
INFORMAÇÃO
ENTRADA
"HARDWARE"
EQUIPAMENTOS E MATERIAIS
"HUMANWARE"
ELEMENTO HUMANO
"SOFTWARE"
PROCEDIMENTOS
SAIDA
CLIENTES
PRODUTOS/
SERVIÇOS
CUSTOS
FATURAMENTO


Sobrevivência
Competitividade
Produtividade
Qualidade - Preferência do Cliente
Projeto Perfeito
Fabricação Perfeita
Segurança Do Cliente
Assistência Perfeita
Entrega No Prazo
Custo Baixo

A abordagem de lshikawa nasceu a partir da compilação de diversos aspectos do trabalho de vários especialistas como Deming, Juran e Shewart, acrescendo a eles uma grande preocupação com a participação do elemento humano e trazendo para o controle

da qualidade uma visão humanística sob a influência dos trabalhos de Maslow, Herzberg e McGregor.

Este trabalho tratará apenas do TQC no estilo japonês, procurando em alguns pontos demonstrar a coerência com as outras abordagens. É importante salientar que

Campos, que procurou adaptar alguns aspectos à cultura local, bem como, estruturar o sistema administrativo TQC em etapas bem claras para facilitar a sua implementação.

Segundo Ernest & Young (1993), o TQC consiste na criação de uma vantagem competitiva sustentável, que se dá por meio do constante aprimoramento do processo de identificação e atendimento das necessidades e expectativas dos clientes quanto aos produtos e serviços requeridos, e da utilização eficiente dos recursos existentes de modo

Cada empresa é diferente, mesmo atuando no mesmo ramo. Isto significa dizer que o método não é uma receita de bolo e que, em cada caso, deverão ser respeitadas as

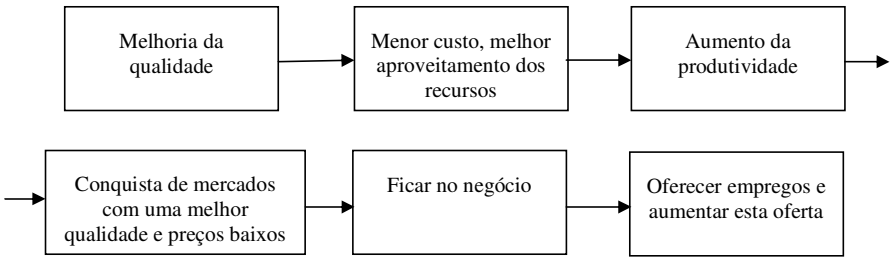
da região. Enfim, o método permite que, caso a caso, as particularidades sejam analisadas e, quando necessário, sejam feitas adaptações.

$$TQC = (Controle + Qualidade)Total$$

$$TQC = Controle\ Total + Qualidade\ Total$$

Controle total é o controle exercido por todas as pessoas da empresa, de forma harmônica (sistêmica) e metódica (baseado no ciclo PDCA). Qualidade total é o verdadeiro objetivo de qualquer organização humana: satisfação das necessidades de todas as pessoas. TQC é o controle exercido por todas as pessoas para a satisfação das necessidades de todas as pessoas.

Comment [W53]: Referenciar dizendo que isso vai ser dito lá na frente.

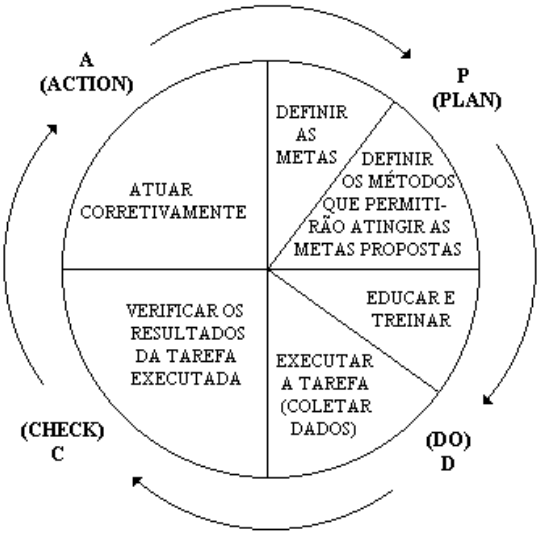


A solução de problemas é iniciada pela identificação dos mesmos. A prática da

questão ou a grande soma de dinheiro envolvida. Estes critérios de seleção, no entanto, geralmente não levam em consideração os clientes envolvidos.

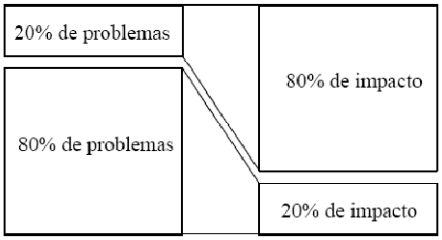
Para entendimento do conceito de “controle”, supõe-se o seguinte caso: Em um novo aeroporto, a primeira coisa a ser feita é o **planejamento do processo,** que inclui as várias **metas** e vários **procedimentos-padrão** de pouso**.** Vários aviões aterrisam e decolam sem dificuldades **cumprindo os procedimentos-padrão**. Porém, certo dia um avião se acidenta durante os procedimentos de pouso. Ocorreu um desastre. Foi localizado um **problema** (resultado ou efeito indesejado do processo). Assim, a **causa** do problema passa a ser procurada. Como as causas podem ser várias (um conjunto de causas é um processo, como já dito anteriormente), procurar a causa é conduzir uma **análise de processo**.

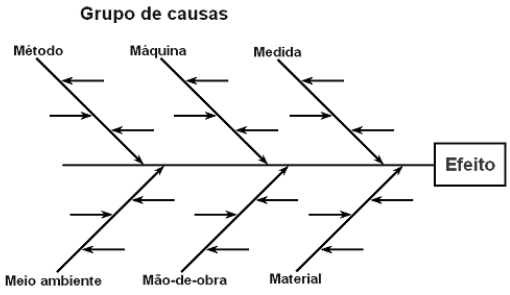
Comment [W54]: Trocar depois esse exemplo por um ligado a software.

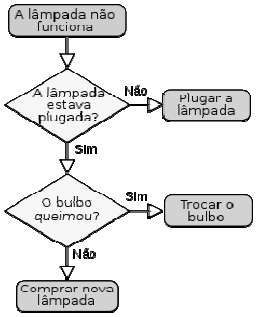

A
(ACTION)
P
(PLAN)
DEFINIR
AS
METAS
DEFINIR
OS MÉTODOS
QUE PERMITI-
RÃO ATINGIR AS
METAS PROPOSTAS
ATUAR
CORRETIVAMENTE
EDUCAR E
TREINAR
VERIFICAR OS
RESULTADOS
DA TAREFA
EXECUTADA
EXECUTAR
A TAREFA
(COLETAR
DADOS)
(CHECK)
C
(DO)
D

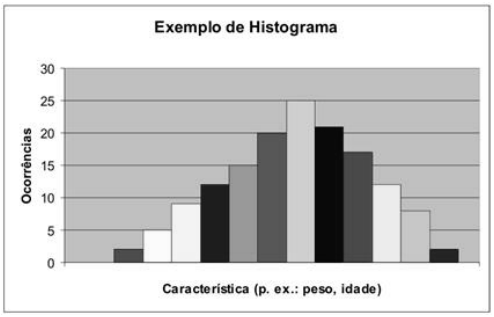
FOLHA DE CONTROLE DE PROCESSO	
x x x x x x	𝍸 𝍸 / /
x x x x x x x	𝍸 ///
x x x x x x x	𝍸 𝍸 //
x x x x x x x	𝍸 /
x x x x x x x	///

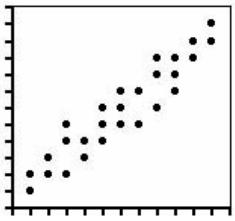
FORMULÁRIO DE PESQUISA		
Perguntas	S	N

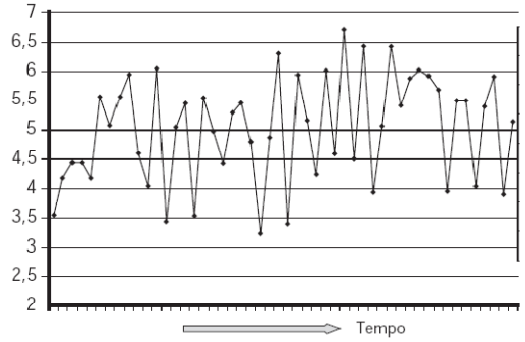



Grupo de causas
Método
Máquina
Medida
Efeito
Meio ambiente
Mão-de-obra
Material


A lâmpada não funciona
A lâmpada estava plugada?
Não
Plugar a lâmpada
Sim
O bulbo queimou?
Sim
Trocar o bulbo
Não
Comprar nova lâmpada


Exemplo de Histograma
30
25
20
15
10
5
0
Ocorrências
Característica (p. ex.: peso, idade)





O gerenciamento pelas Diretrizes, conduzido pela alta administração da

A “Garantia da Qualidade” é uma função da empresa que visa confirmar que todas as atividades da qualidade estão sendo conduzidas da forma requerida, atendendo às necessidades do cliente (antecipando seus anseios) de forma completa e melhor que o concorrente. Por esse motivo de estar voltada a verificar continuamente se as necessidades do cliente estão sendo atendidas, a garantia da qualidade é considerada, segundo Campos [Campos 1992], como a “embaixatriz” do cliente na empresa.

Comment [W55]: Dizer em algum lugar que “empresa” pode significar uma organização sem fins lucrativos também ou qualquer outra entidade que implanta qualidade.

O ciclo da garantia da qualidade começa no cliente. Por meio de uma pesquisa de mercado, dados são coletados e classificados em necessidades de novos produtos e necessidades de melhorias em produtos existentes. Essas necessidades são enviadas para o desenvolvimento e são alinhadas com o

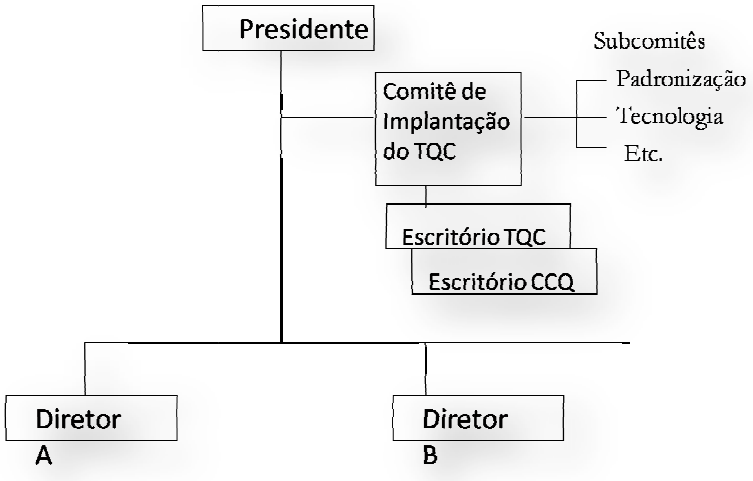
é iniciada, verificando sempre se a qualidade planejada está sendo seguida. Após a produção, o produto passa pela inspeção final e fecha-se o ciclo da garantia da qualidade, retornando ao cliente.

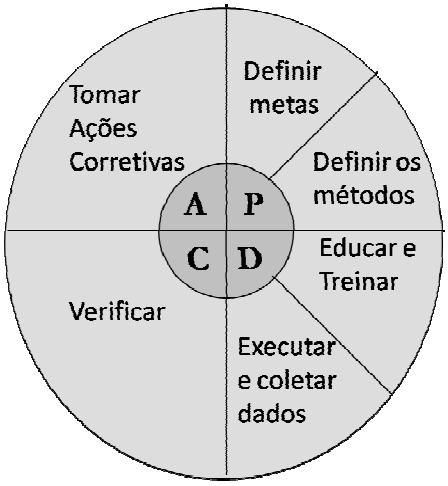
Porém, em algumas empresas brasileiras, a conscientização do pessoal ligado ao setor de vendas sobre qualidade tem sido muito baixa. Nesses casos, o pensamento predominante é que o culpado pela falta de qualidade é a "produção" e as "reclamações

de seu processo. Sendo assim, o marketing é diretamente responsável pela qualidade do produto perante o consumidor [Campos 1992].

Os métodos com que muitas empresas brasileiras atuam no setor de compras são

num relacionamento fornecedor/comprador que não prima, na maioria dos casos, pela confiança mútua.


Presidente
Subcomitês
Comitê de Implantação do TQC
Padronização
Tecnologia
Etc.
Escritório TQC
Escritório CCQ
Diretor A
Diretor B


Definir
metas
Definir os
métodos
Educar e
Treinar
Executar
e coletar
dados
Verificar
Tomar
Ações
Corretivas
A
P
C
D

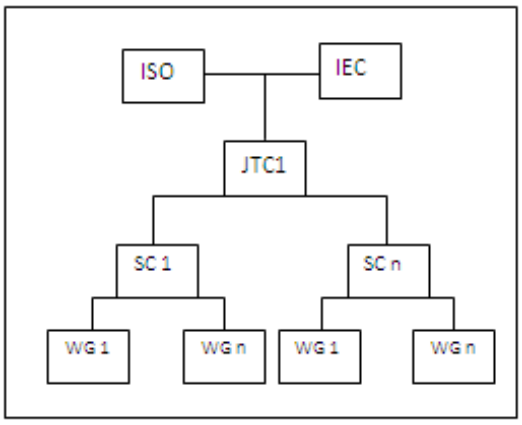
Walton, Mary. (1989) “O método Deming de administração”, Marques Saraiva, Rio de Janeiro.

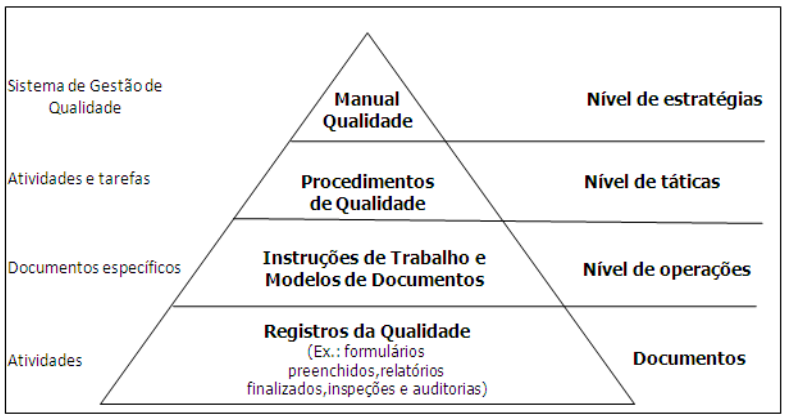
Adjacente a este conceito enquadra-se um conjunto de normas internacionais

e certificar características de processos e produtos, firmando assim garantia e segurança no desenvolvimento de sistemas de informação.Dentre estas normas destacam-se a série ISO 9000, com os requisitos mínimos para implantação e avaliação de um Sistema de Gestão para Qualidade (SGQ), a ISO/IEC 12207, responsável por ditar os processos mínimos essenciais para projetos em organizações, e a ISO/IEC 15504, responsável por nortear todos os processos utilizando-se de modelos de referência e medição para facilitar o desenvolvimento dos mesmos e suas etapas componentes.

Atualmente existe uma grande quantidade de organismos normativos espalhados pelo mundo. Grande parte deles aborda assuntos que condizem normas técnicas e normas de procedimentos relacionadas à avaliação de qualidade, como por exemplo, a ISO 9001, ou para características naturais destinadas ao meio ambiente, como por exemplo, a ISO 14000[1].

Com o intuito de conceder um controle para os documentos de normas, essa entidade ganhou relevante importância e respeito ao longo de sua história. Entre a data de sua fundação até os dias atuais, a publicação de aproximadamente 17500 padrões internacionais [ISO 2009b] para áreas como ciências exatas, saúde e humanas, transforma o pensamento de organizações, empresas e órgãos governamentais em 162 países[2] dos cinco continentes, que aos poucos utilizam as normas com uma visão mais coerente e realista sobre as necessidades de investimentos que precisam ser integradas nas organizações para que o diferencial qualitativo alcançado se torne um fator prioritário de negócios.


ISO
IEC
JTC1
SC 1
SC n
WG 1
WG n
WG 1
WG n


Sistema de Gestão de Qualidade
Atividades e tarefas
Documentos específicos
Atividades
Manual Qualidade
Procedimentos de Qualidade
Instruções de Trabalho e Modelos de Documentos
Registros da Qualidade
(Ex.: formulários preenchidos,relatórios finalizados,inspeções e auditorias)
Nível de estratégias
Nível de táticas
Nível de operações
Documentos

- **Liderança:** Os líderes devem ser conscientes que precisam despertar os

a norma instaura releva para os líderes saberem implantar valores como dedicação, determinação e empenho dos envolvidos.

A ISO 9001:1994 surgiu como a primeira versão em caráter avaliativo para a certificação de Sistemas de Gestão de Qualidade. Baseada em vinte elementos chaves[3] para facilitar a administração das organizações, esta certificação adotou políticas definidas principalmente para gerência de processos e produtos para fábricas em vários níveis de produção [MUTAFELIJA e STROMBERG, 2003]. Melloti et al. 2007 descreve que esta norma possuía uma visão desmembrada de negócios para organizações. A adoção de seus requisitos era instaurada nos processos para a formação de um sistema de qualidade, porém de forma paralela as relações existentes entre as organizações e os fornecedores, muitas vezes dificultando a exclusão de problemas que influenciavam em todo o sistema de gestão adotado.

A nova e recém formulada certificação para sistemas de gestão de qualidade é a ISO

implantação dos requisitos nos sistemas de gestão de qualidade adotados [MELLO et al. 2009].

A versão em uso da NBR ISO 9001:2008 no Brasil é a segunda, publicada em novembro de 2008 e validada em dezembro do mesmo ano [ABNT 2008]. Adequada do modelo original elaborado pelo comitê ISO/TC 176[4], esta certificação possui no escopo termos definidos como "generalidades" que capacitam os consultores a estipularem planos de análises para processos de acordo com os requisitos e seus fatores de implantação. A [ABNT 2008a] descreve o sumário da ISO 9001:2008 na seguinte abrangência de assuntos de gestão para qualidade:

- Introdução: Possui características correlatas as generalidades superficiais da norma destacando o conceito da abordagem de processo e cliente, com o PDCA, a relação da certificação com a ISO 9004 e a compatibilidade com outros sistemas de gestão[5]

- 5 – Responsabilidade da direção: Destina-se a conscientização para com os líderes das organizações. A alta direção deve definir estratégias para serem executadas nos níveis

[4] "Comitê técnico *Quality managements and quality assurance* (ISO/IEC 176), subcomitê *Quality systems* (SC 2),conforme a ISO/IEC Guide 21-1:2005" [ABNT 2008,p. v]

táticos e operacionais (Ver Figura x). A norma cita que um fator diferencial para que se obtenha isto é a especialização da comunicação entre as categorias que formam o sistema, além da análise crítica de realimentação de mudanças que suscitam progressos baseados em ações de acompanhamento com contenções e prevenções.

Outro ponto importante é a adoção de uma política e objetivos da qualidade. A alta direção impõe um plano de metas que devem ser analisadas pelas gerências e

suas aplicações. Cumprimento de prazos, redução de erros e contenções de gastos, dentre outros detalhes ínfimos que fazem a diferença, idealizam o atendimento das necessidades explicitas e implícitas dos clientes, fornecedores e da organização durante o processo de implantação deduzindo-se então que a qualidade pode ser aplicada sem nenhuma restrição.

O mapeamento e a descrição dos processos também se integram nos requisitos para a obtenção da certificação. Metodologias como o *Business Modeling Process*[6] e a utilização do PDCA facilitam a abordagem dos processos delineando a padronização e identificação de procedimentos, instruções e características que controlam as atividades e tarefas básicas para a elaboração de um plano de sistematização de qualidade aplicável, tornando-se este, padrão para a organização e como modelo de gestão para ser adotado.

A abrangência genérica para a sistematização da qualidade em organizações inserida pelas certificações ISO traz conceitos que muitas vezes não identificam as práticas específicas para a gestão de processos ou produção de software, para serem implantadas em projetos. A ISO, em parceria com a IEC, desenvolveu um guia de referência que buscasse complementar a aplicação da certificação ISO 9001 com o propósito de normatizar e qualificar a gestão da qualidade baseado nas chamadas *Fábricas de Software*[7].

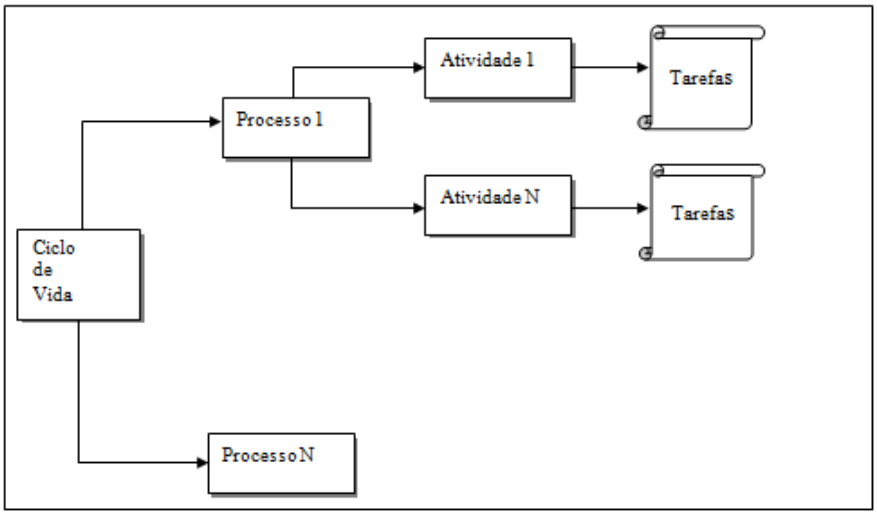
Para Cortês (2008), a descrição e a terminologia da ISO/IEC 90003 estão relacionadas de acordo com os requisitos descritos na ISO 9001. Para cada requisito, e o conjunto de componentes que os formam, são realizadas interpretações que adéquam as características genéricas de gestão de qualidade para sistemas, ao contexto do desenvolvimento de produtos de software, através de orientações operacionais e técnicas provenientes da certificação, tais como *shall* (deve fazer), *should* (poderiam ou convém que) e *may* (podem fazer), melhorando o fluxo de funcionamento do ciclo de

[6] O *Busineess Process Modeling* é um conjunto de alternativas para a construção de modelos de processos

vida dos processos com práticas seguras que geram confiabilidade nas ações e decisões evidenciadas pela alta direção sendo executadas pelos colaboradores dos níveis táticos e técnicos.

- Teste e validação: Externa a necessidade para a elaboração de planos de testes do software e a homologação dos resultados em vários níveis. Um plano de testes

documentação, nos casos de testes e principalmente na quantificação e análise de comparação efetivada com os dados.


Atividade 1
Tarefas
Processo 1
Atividade N
Tarefas
Ciclo
de
Vida
Processo N

- **Processo de aquisição:** A aquisição ocorre quando a organização busca uma solução customizada para a fabricação do software e o atendimento imediato das necessidades do cliente. A aquisição de ferramentas pode ocorrer através de um contato com outra fábrica de software ou simplesmente com a compra de "produtos de prateleira", como por exemplo, editores de textos e ferramentas RAD[8] de desenvolvimento. As atividades prioritárias que descrevem este processo são o lançamento de uma proposta, o pedido de formulação de um contrato, a monitoração da relação de um fornecedor e seu cliente, além da aceitação e conclusão de todos os fatores e aspectos que idealizam e justificam o início do projeto.

- **Processo de manutenção:** Contém as atividades de solução de problemas do produto em questão. O processo é executado quando são realizadas alterações de código, documentações técnicas ou contratuais, em virtude da correção de erros para impor melhorias contínuas que possibilitem a implementação da modificação, a revisão dos fatores avaliados, a aceitação das alterações, a migração de dados, ferramentas, tecnologias e operações, como também a descontinuação do software através de

modificações de sistemas legados e suas devidas reparações quanto as funções e responsabilidades.

- **Processo de gerência:** Neste processo é descrito a necessidade em implantar as atividades de gestão para processos e produtos nos projetos desempenhados pela

A norma descreve o processo de gerência como uma fonte para determinar se o projeto obterá fracasso ou sucesso em sua realização de acordo com sua administração.

Observando a realidade das organizações em definirem ou melhorarem seus ciclos de desenvolvimento baseados nos preceitos da ISO/IEC 12207, a ISO integrou um processo auxiliar com o intuito de prover uma adequação conveniente a estrutura da organização a medida que as alterações forem sendo realizadas de forma que os

O início da norma ISO/IEC 15504 remarca uma volta aos anos noventa, mais precisamente em 1991. O JTC1 seguia uma linha de pensamento que idealizava a normatização de conceitos que facilitassem a perspectiva de denominação e definição de características para processos. Em 1993, teve início o projeto intitulado *Software*

Process Improvement and Capability dEtermination (*SPICE*), que segundo Koscianski e Soares (2007) possuía três objetivos principais:

Este capítulo buscou de forma sucinta e objetiva apresentar os principais conceitos de algumas das principais normas para qualidade de processos de software utilizadas atualmente. Foram abordadas as entidades que regem essas normas, a ISO, o IEC e a ABNT, bem como histórico, objetivos, modelo, aplicação

sistemas de gestão para qualidade, além do ciclo de vida para o desenvolvimento de software e alguns modelos de acompanhamento e avaliação de processos para obtenção de qualidade em processos de software.

6. O processo de adaptação da norma ISO/IEC 12207 envolve algumas práticas administrativas essenciais que todas as organizações deveriam adotar no seu

esboço que associe essas práticas aos processos primários, organizacionais e de apoio.

2009. Disponível em: <http://eulerhm.googlepages.com/cea446-gestãodaqualidadedesoftware > Acesso em 15 Out. 2009

8.2.HISTÓRICO272

8.3.CMMI272

8.3.1. REPRESENTAÇÕES DO MODELO CMMI 275
8.3.1.1. REPRESENTAÇÃO POR ESTÁGIOS 275
8.3.1.2. REPRESENTAÇÃO CONTÍNUA 276
8.3.1.3. REPRESENTAÇÃO POR ESTÁGIOS X CONTÍNUA 278
8.3.2. MÉTODO DE AVALIAÇÃO DO CMMI (SCAMPI) 279
8.3.2.1. CONCEITO CENTRAL 280
8.3.2.2. PARÂMETROS OBSERVADOS NO SCAMPI 280
8.3.2.3. PRAZO E EXIGÊNCIA DE PESSOAL 280
8.3.2.4. CARACTERÍSTICAS ESSENCIAIS DO MÉTODO DE SCAMPI 280
8.3.2.5. MODOS DE USO 281
8.3.2.6. DESCRIÇÃO DO MÉTODO 281

8.4.MPS.BR285

8.4.1. REPRESENTAÇÃO DO MODELO MPS 286
8.4.1.1. NÍVEL G – PARCIALMENTE GERENCIADO 287
8.4.1.2. NÍVEL F – GERENCIADO 287
8.4.1.3. NÍVEL E – PARCIALMENTE DEFINIDO 288
8.4.1.4. NÍVEL D – LARGAMENTE DEFINIDO 288
8.4.1.5. NÍVEL C – DEFINIDO 289
8.4.1.6. NÍVEL B – GERENCIADO QUANTITATIVAMENTE 289
8.4.1.7. NÍVEL A – EM OTIMIZAÇÃO 289
8.4.2. MÉTODO DE AVALIAÇÃO DO MPS.BR (MA-MPS) 289
8.4.2.1. PRAZO E EXIGÊNCIA DE PESSOAL 291
8.4.2.2. DESCRIÇÃO DO MÉTODO 291

8.5.CMMI X MPS.BR294

8.6.EXERCÍCIOS295

8.7.SUGESTÕES DE LEITURA296

8.8.TÓPICOS DE PESQUISA296

8.9.REFERÊNCIAS297

Diante deste cenário, a área de desenvolvimento de software se tornou um nicho lucrativo para as empresas da área de Tecnologia da Informação. Buscando uma maior inserção no mercado de desenvolvimento de software, diversas corporações começaram a fazer grandes investimentos para desenvolver sistemas diferenciados com mais qualidade. Para isto, investiu-se também na melhoria do processo de desenvolvimento de software e passou-se a buscar a adoção de modelos de qualidade de software com reconhecimento internacional que pudessem certificar que os sistemas desenvolvidos pela organização são sinônimos de qualidade. Com isso, foram criados os modelos de

desenvolvimento do software. Os mais conhecidos são: ISO/IEC 15504, CMMI e MPS.BR.

Na representação por estágios, os níveis de maturidade não servem para analisar áreas do processo, mas sim para indicar melhorias na organização como um todo. Ao fazer a avaliação de uma organização, é possível mapear os valores de capacidade do processo para maturidade organizacional. Uma organização que apresente nível 2 para

capacidade mínimo igual a 3 para as áreas de processo correspondentes ao nível pretendido.

c) **Custo/Recurso Efetivos.** Eficiência em termos de custo de pessoas. O

avaliação, incluindo os recursos da organização, o impacto nos projetos avaliados e a equipe de avaliação.

Tabela 8.1. Fase 1 - Plano e Preparação para a Avaliação [Almeida 2007]

Tabela 8.2. Fase 2 - Administrando a Avaliação [Almeida 2007]

Tabela 8.3. Fase 3 - Relatório do Resultado [Almeida 2007]

MPS.BR

A Figura 8.5 ilustra a estrutura do Modelo MPS.BR que possui três componentes: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS).

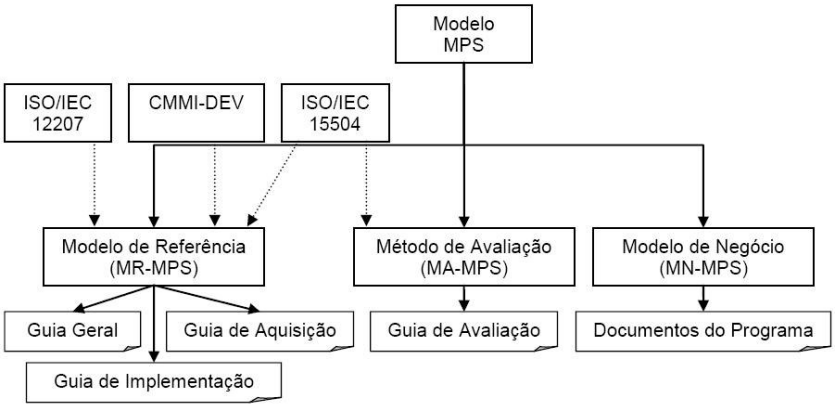


Figura 8.5. Estrutura do Modelo MPS.BR [SOFTEX 2009]

organizacional na execução dos seus processos de software. Este Método de Avaliação será detalhado na seção 3.4.2.

- O(s) avaliador(es) adjunto(s) da IA[9];

Tabela 8.4. Subprocesso Contratar a Avaliação [SOFTEX 2009]

Tabela 8.6. Subprocessos Realizar a Avaliação Final [SOFTEX 2009]

Tabela 8.7. Subprocesso Documentar os Resultados da Avaliação [SOFTEX 2009]

10

Comment [j56]: Formatar todas a tabelas do capítulo

Funcionalidade	Conjunto de atributos que evidencia a existência de um conjunto de funções e suas propriedades especificadas. As funções satisfazem às necessidades explícitas ou implícitas.

Comment [j57]: Deve ficar assim centralizado na célula e não como abaixo

Comment [j58]: Deve ficar assim Justificado na célula e não como abaixo

Funcionalidade (satisfação das	Adequação - execução do que é apropriado

	Interoperabilidade - interação com outros sistemas Conformidade - aderência às normas Segurança de acesso - bloqueio de uso não autorizado

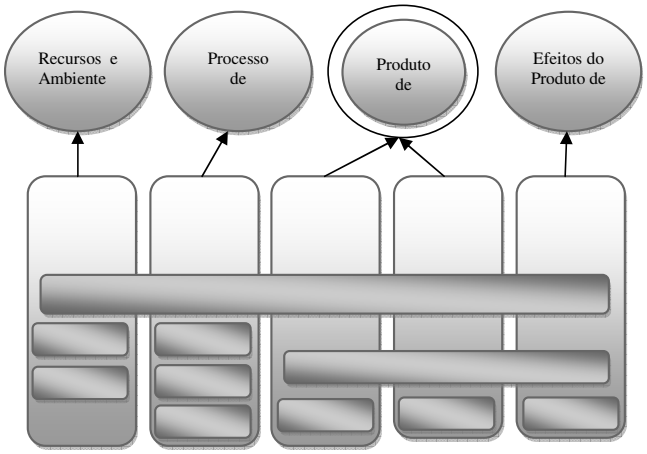
Comment [j60]: Deve ficar assim Justificado na célula e não como abaixo

	Relatório incluindo: identificação do produto, hardware e software utilizado, documentos utilizados, resultados dos testes, lista de não conformidade com os requisitos, lista de não conformidade com as recomendações, datas, etc.

Comment [j61]: Refazer essa tabela, pq alguns itens tem descrição e outros não, tabela muito confusa para leitor leigo

Certificação		

Norma		



Seção	

A norma SQuaRE surge de uma maneira muito sólida e por diversos motivos, pois abrange amplamente o assunto e estabelece uma base precisa, tanto para definir o modelo quanto para realizar a avaliação; Seus documentos contemplam um certo caráter didático: houve, por exemplo, a preocupação em fornecer uma extensa lista de exemplos de métricas; Os documentos são resultados de um esforço e consenso de centenas de pesquisadores; representam, assim, uma soma de experiências única no assunto; Outros modelos de qualidade elaborados por pesquisadores de maneira pontual podem ser mapeados para o modelo SQuaRE.

Comment [j62]: Falta referência

10.1.5. Norma SQuaRE

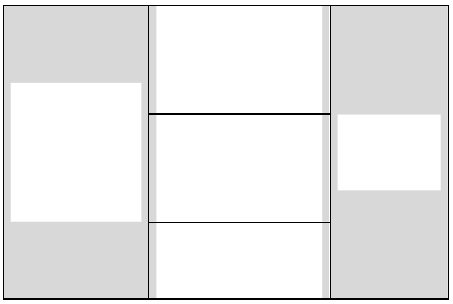


Figura 10.2. Partes componentes da norma SQuaRE

Comment [j63]: Referencia?

Cada divisão é composta por um conjunto de documentos e trata de um assunto específico, como apresentadas abaixo:

Comment [j64]: Foi tirado de onde?

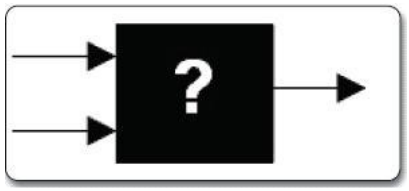
- **Gerenciamento**: os documentos desta divisão são voltados a todos os possíveis usuários dela, como gerentes, programadores, avaliadores ou compradores. São definidos os termos utilizados em todos os demais documentos e são feitas recomendações e sugestões de caráter geral sobre como utilizar o SQuaRE.
- **Modelo de Qualidade**: esta divisão corresponde principalmente à ISO/IEC 9126-1. São definidos os conceitos de qualidade externa, interna e em uso, que permitem orientar diferentes perspectivas de avaliação. Por exemplo,

Comment [j65]: reescrever

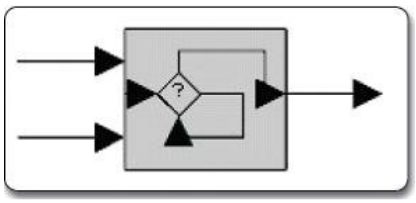
características de qualidade, permitindo que se faça uma descrição extensa e precisa do que cada ator espera de um produto.

Vale enfatizar que o projeto SQuaRE não surgiu para desmentir as normas 9126 e 14598, mas sim para organizá-las.

Comment [j66]: usar outra palavra



caracterizada por avaliar as funcionalidades internas dos componentes do software, baseando-se no código fonte e procurando exercitar estruturas de controle e de dados do programa. Sendo assim, faz-se necessário que o analista de testes tenha boa habilidade em programação de modo a entender todos os caminhos lógicos possíveis. A Figura 10.4 ilustra a abordagem estrutural.



- **Teste de aceitação:** o teste de aceitação corresponde ao teste realizado pelo usuário de fato do sistema, no momento em que todos ou quase todos os defeitos encontrados nas etapas anteriores já tenham sido corrigidos. O propósito deste teste é estabelecer a confiança do sistema; ele está mais relacionado com a

acordo com os requisitos especificados. Normalmente os testes de aceitação podem ser de duas categorias: testes *alfa* e testes *beta*. Os primeiros são realizados nas instalações do desenvolvedor, que fica observando os usuários utilizarem o sistema, e anotam os problemas identificados. Já os testes *beta* são realizados no ambiente real de trabalho do usuário, que instala o sistema e testa, sem a presença do desenvolvedor. Em seguida, um documento contendo os registros dos problemas encontrados é enviado à organização desenvolvedora.

Durante o planejamento de testes deve-se ter certeza de que os objetivos dos clientes e *stakeholders* foram entendidos de maneira correta [Graham et. al 2007]. Baseados neste entendimento, os propósitos da atividade de testes propriamente dita são estabelecidos, e assim, uma abordagem e plano para os testes é obtida incluindo especificação das atividades de teste. O planejamento de testes apresenta as seguintes atividades principais:

Comment [j67]: referência muito repetida, buscar outras fonts, para esta e para as demais seções

Esta é a atividade em que os objetivos gerais de testes são transformados em condições e projetos de teste tangíveis [Graham et. al 2007]. O propósito principal é identificar e descrever os casos de teste para cada versão de teste, e identificar e estruturar os procedimentos de teste, especificando como executar os casos de teste. As principais tarefas desta etapa podem ser destacadas em:

Comment [j68]: referência muito repetida, buscar outras fonts, para esta e para as demais seções

Uma vez que os casos e procedimentos de teste foram especificados em alto nível na etapa anterior, este é o momento em que o ambiente será preparado para que eles sejam executados e comparados com os resultados desejados [Graham et. al 2007]. Além disso, é a etapa em que os componentes necessários são implementados para que os testes sejam executados. As principais tarefas destas duas fases serão destacadas a seguir.

Comment [j69]: referência muito repetida, buscar outras fonts, para esta e para as demais seções

Esta é a fase em que se deseja observar se já foram executados testes suficientes para garantir a qualidade desejada do produto, sendo assim, critérios de saída são definidos com esta finalidade [Graham et. al 2007]. Estes critérios informam se uma dada atividade de testes pode ser considerada completa. As principais atividades são:

Comment [j70]: referência muito repetida, buscar outras fonts, para esta e para as demais seções

A atividade de encerramento de teste pode ser dada através de diversos fatores, como por exemplo, as informações necessárias do processo de testes já foram atingidas; o projeto é cancelado; quando um marco particular é alcançado; ou quando uma versão

Um modelo de desenvolvimento de software bastante conhecido é o modelo em cascata, que como o próprio nome já sugere, tem sua base voltada a um desenvolvimento seqüencial das atividades. As primeiras atividades começam no topo da cascata, e então vão seguindo seqüencialmente através das várias atividades de concepção do projeto, e finalmente terminando com a etapa de implementação. Após isso, é que as atividades de teste são introduzidas, e dessa forma os defeitos só podem ser detectados bem perto da fase de implementação [Graham et. al 2007]. A Figura 10.5 ilustra o modelo em cascata.

Comment [j72]: referência muito repetida, buscar outras fonts, para esta e para as demais seções

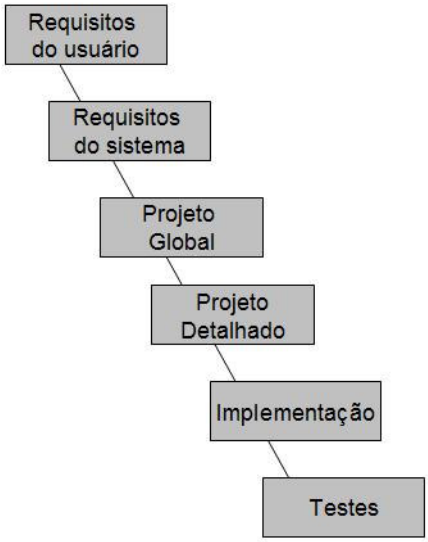


Figura 10.5 Modelo em cascata (adaptado de Graham 2007)

Comment [j73]: et al?

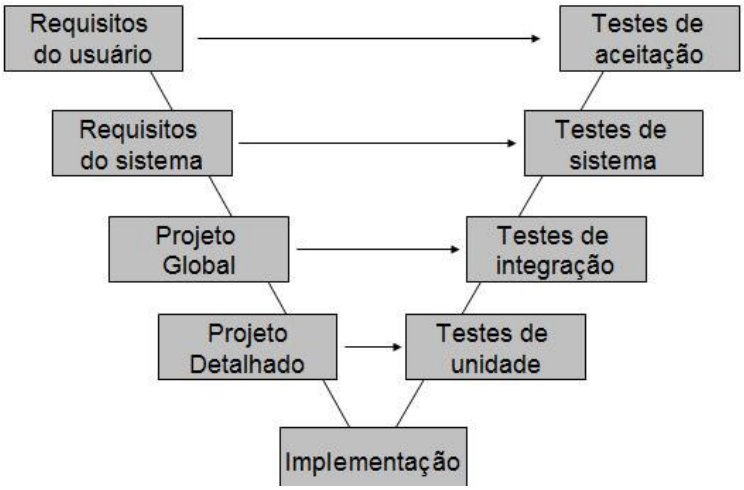
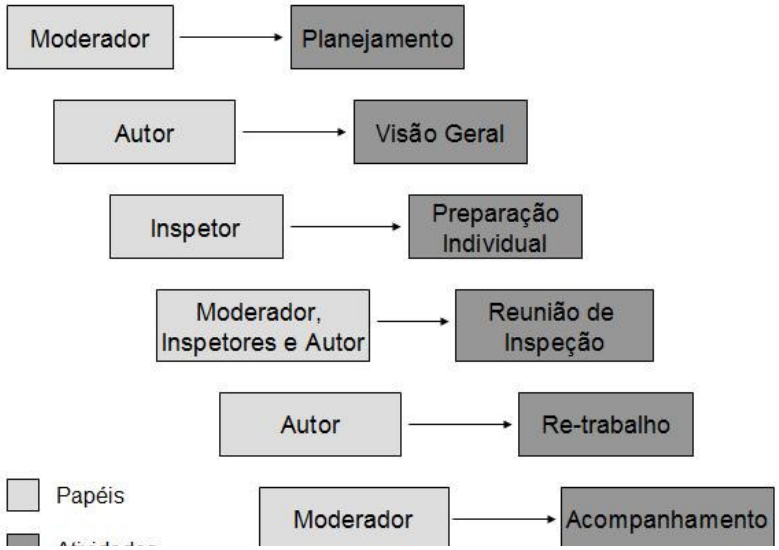


Figura 10.6 Modelo V (adaptado de Graham 2007)

Comment [j74]: Et al?

O modelo CMMI exige a realização de revisões como uma prática específica do processo de verificação, demonstrando assim sua importância na garantia da qualidade do produto. Segundo Fagan, a utilização de inspeções informais de software captura em torno de 60% dos erros em um programa [Fagan 1986]. Mills et al. sugere que uma aplicação mais formal de inspeção de software pode detectar até mais de 90% dos erros de um programa [Mills et al. 1987]. Selby e Basili comparam empiricamente a efetividade de inspeções e testes. Eles perceberam que a revisão de código estática se mostrava mais efetiva e menos cara do que a procura por erros utilizando testes [Selby et al. 1987].

Comment [j75]: Substituir por et. al



- **Reunião:** nesta etapa, o passo a passo principal consiste na leitura e interpretação do produto, pelo leitor; em seguida o autor tira quaisquer dúvidas

passar mais do que duas horas, e deve ser focada na detecção de defeitos, conformidade com o padrão e programação de má qualidade. O time de inspeção não deve discutir como estes defeitos poderiam ser corrigidos e nem sugerir mudanças em outros componentes.

- Assist – *Asynchronous/ Synchronous Software Inspection Support Tool* foi desenvolvida para prover inspeções individuais e em grupo. Como o nome sugere, permite inspeções síncronas e assíncronas, com reuniões tanto em locais diferentes como no mesmo ambiente. Utiliza uma linguagem de definição de processo de inspeção (IPDL) e um sistema flexível para o tipo de documento inspecionado, permitindo o suporte a qualquer tipo de processo de inspeção de software. Inspeção de código, coletas de dados para métricas e cálculos para apoio as inspeções também estão presentes nesta ferramenta. É baseada numa arquitetura cliente/servidor, em que o servidor é usado como um repositório central de documentos e outros tipos de dados. Um *browser C++* pode automaticamente apresentar itens relevantes de *checklist*[10] para a sessão de código inspecionado.
- IBIS – *Internet-Based Inspection System* é uma ferramenta baseada em WEB com notificações por *email* que auxilia no processo de inspeção desenvolvido por Fagan. Permite que as inspeções sejam realizadas entre pessoas geograficamente distribuídas e possui uma interface bastante leve e amigável, tendo toda sua estrutura e dados armazenados em arquivos XML. Ela não limita o tipo de artefato a ser inspecionado e provê suporte a decisões, apoio a

anotações e *checklists*. As principais vantagens desta ferramenta são: permite que os inspetores acessem a aplicação de seus próprios computadores; admite que a inspeção seja realizada com integrantes da equipe distribuídos em locais diferentes, até mesmo em países diferentes; permite que especialistas diferentes participem da reunião, podendo ser especialistas de outro departamento ou mesmo fora na organização.

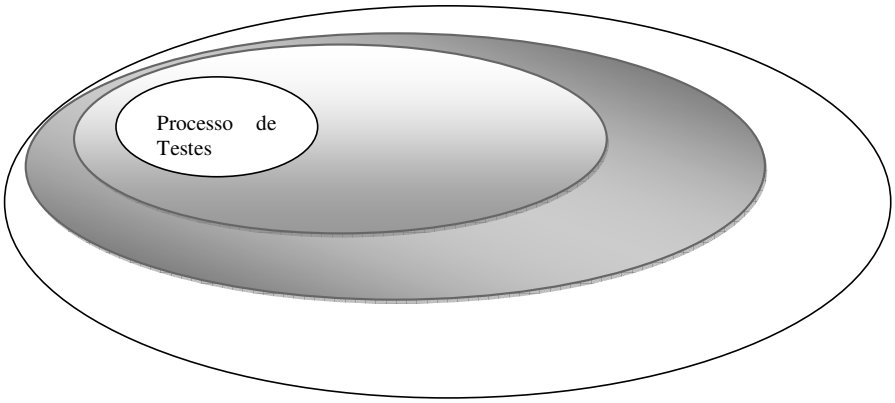
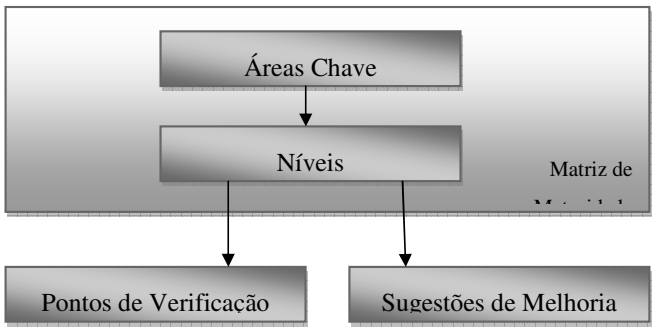
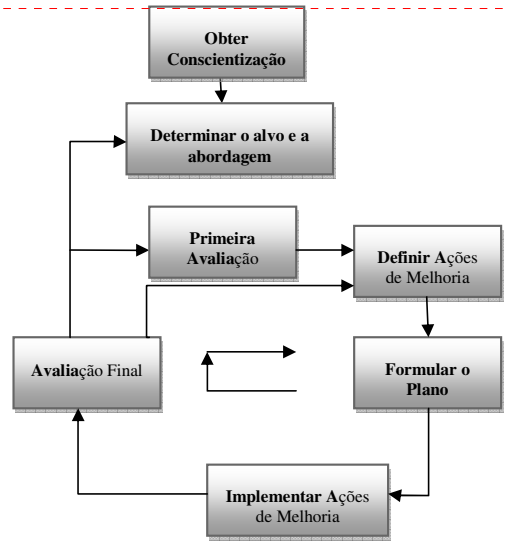


Figura 10.8. Processo Total de Desenvolvimento

Comment [j76]: referencia





Comment [j77]: corrigir setas soltas no meio da figura e colocar referencia

O *Testing Maturity Model* – TMM [Burnstein, et al 1998] foi desenvolvido pelo Illinois Instituty of Tecnology como um guia para melhoria de processos de testes. A estrutura do TMM foi baseada no CMM, e está aderente ao CMMI, consistindo de cinco níveis que avaliam o grau de maturidade de um processo de testes. Para cada nível de maturidade áreas de processo são definidas. Uma área de processo é um conjunto de atividades que, quando executadas de forma adequada, contribuem para a melhoria do processo de testes. Na Figura 10.11 pode-se observar a estrutura do TMM.

Comment [j78]: itálico

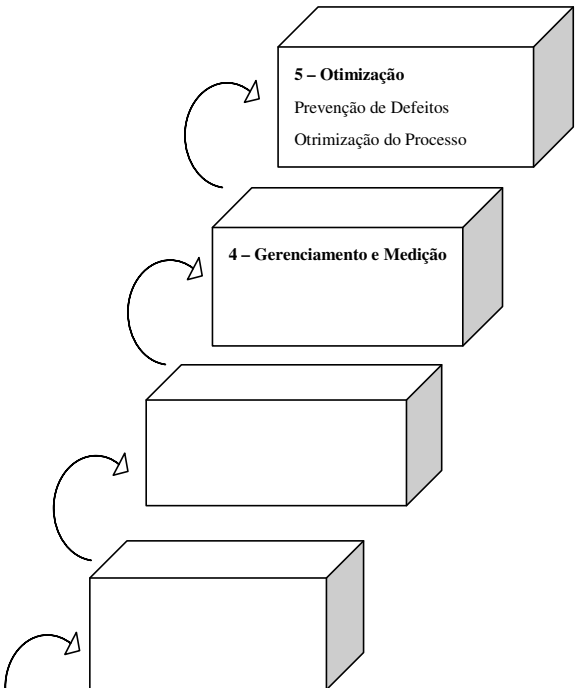


Figura 10.11. Estrutura do TMM

Comment [j79]: referemcia

Comment [j80]: referencia

10.4.2.1. Níveis de Maturidade do TMM

- Nível 1 – Initial

Comment [j81]: traduzir

- Nível 2 – Phase Definition

Comment [j82]: traduzir

- Nível 3 – Integration

Comment [j83]: traduzir

- Nível 4 – Management and Measurement

Comment [j84]: traduzir

- Nível 5 – Optimization

Comment [j85]: traduzir

10.4.3. TIM – Test Improvement Model

Comment [j86]: colocar entre parentese e colocar tradução

10.4.3.1. Modelo de Maturidade

Comment [j87]: referencia

- **Nível 1 – *Baselining***

Comment [j88]: traduzir

- **Nível 2 – *Cost-efectiveness***

- **Nivel 3 – *Risk-lowering***

- **Nivel 4 - *Optimizing***

10.4.3.2. Áreas Chave

Comment [j89]: traduzir

Comment [j90]: traduzir

Comment [j91]: traduzir

Comment [j92]: traduzir subtopicos, que puderem ser traduzidos

- No nível *Optimizing,* atividades de planejamento e a rastreabilidade é continuamente melhorada baseada na análise de métricas. Reuniões de *post-mortem*[11] são realizadas e os resultados armazenados e distribuídos.

[11] *Reunião de Post-mortem: é uma reunião com o objetivo de coletar de experiências eficazes e de baixo*

Considerações Finais

O desenvolvimento de software engloba um mercado de extrema competitividade. Tendo em vista que os sistemas que apresentam melhor qualidade garantem seu espaço no mercado, as empresas que os desenvolvem têm investido bastante para assegurar a qualidade de seus produtos e garantir a satisfação dos clientes. A qualidade de um produto pode ser definida como sua capacidade de cumprir os requisitos inicialmente estipulados pelos clientes, e sendo assim, está diretamente relacionada à qualidade do processo de desenvolvimento. Por este motivo, tem surgido uma grande demanda ao incentivo de pesquisas que levem em consideração à procura por formas de melhoria da qualidade dos produtos.

Comment [j93]: aumentam as possibilidades, mas nao garantem..

Exercícios

Sugestões de Leitura

Para aprofundar a leitura sobre TMM (*Test Maturity Model*), é sugerida a leitura do livro *A Model to Assess Testing Process Maturity*, Burnstein & Grom, 1998.

Tópicos de Pesquisa

Comment [j94]: acredito que esse não seja o formato de tópicos de pesquisa, dúvida!

Referências

Comment [j95]: padronizar

Uma introdução ao SWEBOK

Em uma conferência da OTAN em no ano de 1968, - *Software Engineering*: *Concepts and Techniques. Proceedings of the NATO Conferences* - Ronald Graham comentou "construímos sistemas como os irmãos Wright construíam aviões - constrói-se de uma só vez só, empurra-se para o despenhadeiro, deixa bater e começa tudo outra vez" (NAUR & RANDELL, 1968).

O aumento gradual e crescente da capacidade de processamento dos computadores revelou a necessidade de se criar processos que orientassem e organizassem a atividade de desenvolvimento de software, deixando de ser uma atividade que até então supria apenas as necessidades do hardware.

Comment [C96]: A versao do SWEBOK que voce está descrevendo é a mais nova?

Comment [C97]: Esta frase ficou incomplete. Precisa concluir o pensamento. É preciso falar aqui que nesta conferencia foi criado o termo Eng. SW motivado pela Crise de SW.

Comment [C98]: Coloque uma referencia aqui

Tais necessidades aumentaram a importância e responsabilidades dos especialistas ligados a uma das áreas da computação, conhecida como engenharia de softwareEngenharia de Software. Com isso, a *Institute of Electrical and Electronics Engineers* (IEEE) e a *Association for Computing Machinery* (ACM) conduziram estudos como uma forma dede modo a promoverção ativamente da Engenharia de SoftwareEngenharia de Software como uma profissão desde 1993, definindo as fronteiras que delimitam a Engenharia de softwareEngenharia de Software, e foi chamado deatravés do Corpo de Conhecimento emda Engenharia de SoftwareEngenharia de Software - *Software Engineering Body of Knowledge* (SWEBOK). (SWEBOK, 2004).

Neste capítulo iremos realizar apresentar uma descrição sobre a visão da Engenharia de softwareEngenharia de Software detalhada e apoiada por um processo de desenvolvimento realizado por profissionais, sociedade científica e órgãos públicos que culminou no guia tema do capítulo.

Comment [C99]: Que necessidades? É preciso encadear melhor as ideias.

Comment [C100]: Este parágrafo ficou confuse.

3.3.2.7.• Deixar claros os limites dae Eengenharia de Ssoftware com respeito a outras disciplinas como ciência da computação, gerência de projetos, engenharia da computação, matemática, entre outros;

Formatted: Indent: Left: 0,25", Hanging: 0,14", Bulleted + Level: 1 + Aligned at: 0,25" + Indent at: 0,5"

Tabela 11.1. Demonstrativo das cCategorias do Conhecimento conforme o SWEBOK

Comment [C101]: Por que esta distribuicao na tabela? Por que nao colocar um embaixo do outro?

Formatted Table

Especializado	

- **Certificação CSDA (Certificação de Associação no Desenvolvimento de Software);**

Comment [C102]: Esta é a traducao correta?

A certificação CSDA oferece os princípios fundamentais para o avanço do profissional de Eengenharia de Ssoftware, disponibilizando uma forte alavanca para experiência estudantil e as reais requisições do mercado de trabalho. CSDA é o primeiro passo para se tornar um *Certified Software Development Professional* (CSDP).

Comment [C103]: Nao gostei deste texto

O SWEBOK utiliza uma organização hierárquica, decompondo todaos aos KA´s em um conjunto de temas com rótulos reconhecíveis pela área de interesse do leitor sobre a Eengenharia de Ssoftware. A figura 11.1 apresenta o corpo de conhecimento do guia, como também seus níveis hierárquicos.

Comment [C104]: Nao gostei deste texto

11.2.2.1. Requisitos de Software

Comment [C105]: Nao usar 4 niveis de numeracao. Neste caso veja o padrao que fonte usar sem numeracao

A liberação é utilizada neste contexto para se referir a entrega de um item de

software estão disponíveis para entrega é freqüentemente necessário para recriar versões específicas de pacotes e os materiais corretos para entrega da versão.

à melhoria da qualidade de software através de mecanismos que proporcionam o gerenciamento automatizado do desenvolvimento de software. Diversas teorias, conceitos, formalismos, metodologias e ferramentas surgiram nesse contexto, enfatizando a descrição de um modelo de processo de software que é automatizado por um ambiente integrado de desenvolvimento de software.

crimes de computador. Respostas sociais, éticas e de legislação estão sendo desenvolvidas para procurar tratar adequadamente cada caso (Koscianski, 2006).

\\lecturer\processos\public_html\TAES3\Livro\Cap12-Gestao_de_Projetos\Gerenciando Projetos de Software.docx

14.1. PROCESSO DE COMUNICAÇÃOERRO! INDICADOR NÃO DEFINIDO.

14.1.1. A COMUNICAÇÃOERRO! INDICADOR NÃO DEFINIDO.

14.1.2. MODELO DE COMUNICAÇÃOERRO! INDICADOR NÃO DEFINIDO.

14.1.3. CANAIS DE COMUNICAÇÃOERRO! INDICADOR NÃO DEFINIDO.

14.1.4. A COMUNICAÇÃO EM ORGANIZAÇÕESERRO! INDICADOR NÃO DEFINIDO.

14.1.5. COMUNICAÇÃO EM PROJETOSERRO! INDICADOR NÃO DEFINIDO.

14.1.6. A COMUNICAÇÃO COMO DESAFIO PARA O GERENTE DE PROJETOERRO! INDICADOR NÃO DEFINIDO.

14.2. GERENCIAMENTO DE COMUNICAÇÃO EM PROJETOSERRO! INDICADOR NÃO DEFINIDO.

14.2.1. PLANEJAMENTO DAS COMUNICAÇÕESERRO! INDICADOR NÃO DEFINIDO.

14.2.2. DISTRIBUIÇÃO DAS INFORMAÇÕESERRO! INDICADOR NÃO DEFINIDO.

14.2.3. RELATÓRIO DE DESEMPENHOERRO! INDICADOR NÃO DEFINIDO.

14.2.4. GERENCIAR AS PARTES INTERESSADASERRO! INDICADOR NÃO DEFINIDO.

14.4. TÓPICOS DE PESQUISAERRO! INDICADOR NÃO DEFINIDO.

14.5. SUGESTÕES DE LEITURAERRO! INDICADOR NÃO DEFINIDO.

14.6. EXERCÍCIOSERRO! INDICADOR NÃO DEFINIDO.

REFERÊNCIASERRO! INDICADOR NÃO DEFINIDO.

O capítulo visa por meio de uma referência didática contribuir para a ampliação do conhecimento e auxiliar pessoas que necessitem aplicar, de forma eficaz, o processo de comunicação em projetos de software. Este capítulo aborda uma visão geral da comunicação, dos processos da Gerência de Comunicação de Projetos, bem como sugestões de leitura, tópicos de pesquisa e exercícios. Inicialmente serão abordadas questões referentes ao processo da comunicação em geral, em torno da sua definição, importância, seus elementos básicos e aspectos do uso da comunicação em organizações e projetos, como a comunicação representa um desafio para o gerente, concluindo com o gerenciamento da comunicação em projetos sendo detalhados seus respectivos processos.

Comment [WU106]: Antes da descrição do capítulo faltou o nome da autora!

Comment [WU107]: Acho que esse parágrafo ficou muito grande, poderia ser melhor dividido pra facilitar o entedimento.

Visão Geral da Comunicação

Comment [WU108]: Acho que devia ser numerada como seção 14.1

Atualmente, dentre todas as formas nas quais a comunicação é utilizada vale destacar como esta é empregada nas empresas e/ou organizações. Segundo Maron [Maron 2008], uma organização nada mais é do que a reunião de pessoas integradas e constantemente se comunicando, a serviço de outras pessoas. Quando esta comunicação

problemas e dificuldades com simplicidade e criatividade, permitindo decisões com segurança e rapidez, atingindo os melhores resultados. Um ambiente aberto à comunicação permite que as pessoas se sintam respeitadas e satisfeitas por contribuírem e participarem ativamente. O resultado será sempre a conquista de maior produtividade, progresso para todos e resultados positivos em todos os níveis. A boa comunicação é primordial para o sucesso de quaisquer projetos e conseqüentemente da organização, pois além de todos os seus benefícios também "contamina" as pessoas com a alegria e o otimismo.

Os projetos das organizações são realizados por pessoas, as quais necessitam incondicionalmente da comunicação para cumprir os objetivos estabelecidos e conseqüentemente compreender como as tarefas devem ser realizadas nos projetos. Assim, a comunicação é um elemento essencial no gerenciamento de qualquer projeto, pois utiliza recursos de troca e partilha capazes de promover a compreensão mútua.

Comment [WU109]: O trema foi extinto de acordo com a nova ortografia da lingua portuguesa

Comment [WU110]: Idem ao comentário 4

- Estabelecer com a participação de membros de todos os níveis hierárquicos da organização, os objetivos organizacionais de forma que contemplem, não apenas os interesses da mesma, mas também de todos os seus membros.
- Definir, com a participação de membros de todos os níveis hierárquicos da organização, a estrutura organizacional, sendo ao nível do desenho organizacional, da distribuição de autoridade, responsabilidade e tarefas.
- Definir com a participação dos membros da organização, desde o nível do desenho organizacional até a distribuição de autoridade, responsabilidade e tarefas. Coordenar, fornecer apoio e controlar as atividades de todos os membros da organização.

Comment [WU111]: Acho que esse "com" deve ser excluido, ou então ser colocada uma vírgula antes do "com"

Comment [WU112]: Acho que a concordância correta é "responsabilidades"

Comment [WU113]: Idem ao comentário 6

Comment [WU114]: Texto praticamente idêntico ao do tópico anterior, acho que o texto deveria ser excluido ou alterado.

Tabela 14.1. Funções da comunicação na organização.

Comment [WU115]: Colocar a legenda na mesma página que a tabela.

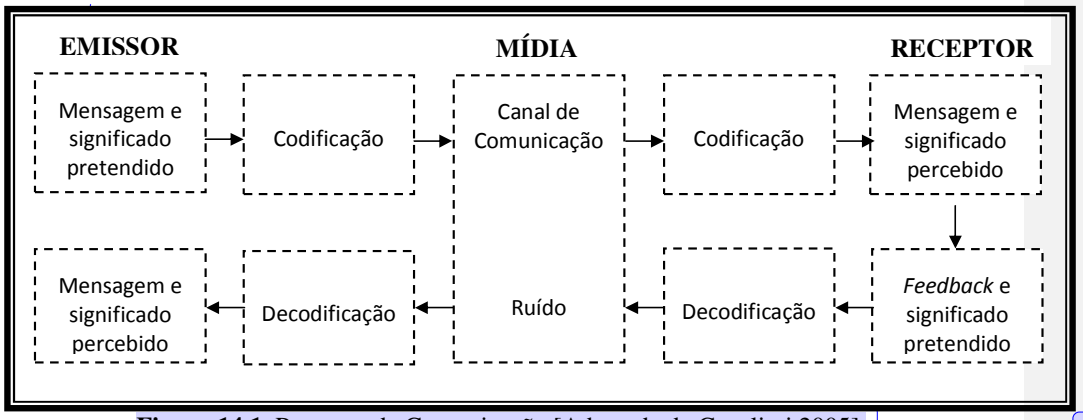


Figura 14.1. Processo de Comunicação [Adaptada de Cavalieri 2005].

Comment [WU116]: Figura está ultrapassando as margens determinadas do texto

O emissor (ou fonte da mensagem da comunicação) é o componente que

pensamento que se pretende transmitir em palavras, gestos ou símbolos que sejam compreensíveis por quem recebe a mensagem.

Receptor da mensagem representa quem recebe e decodifica a mensagem. Portanto, neste momento é necessário ter atenção no processo de decodificação da mensagem, a qual resulta efetivamente no que emissor pretendia enviar (por exemplo, em diferentes culturas, um mesmo gesto pode ter significados diferentes). Podem existir apenas um ou numerosos receptores para a mesma mensagem.

Comment [WU117]: Acho que faltou um "o" e o certo é "que o emisssor"

O *feedback* ou realimentação é a resposta do receptor ao emissor da mensagem e pode ser utilizada como uma medida do resultado da comunicação, para se certificar de que a interação está sendo mantida no momento em que a mesma está se processando, e ajuda no processo de conhecimento para saber se a mensagem foi enviada, como foi recebida e se foi ou não compreendida. Pode ou não ser transmitida pelo mesmo canal de transmissão.

Comment [WU118]: Parágrafo ficou um pouco extenso, tentar reduzir para facillitar o entendimento.

A comunicação oral possui como principal característica a presença do receptor (exclui-se, obviamente, a comunicação oral que utilize a televisão, a rádio, ou as gravações). Esta característica explica diversas das suas principais vantagens, nomeadamente o fato de permitir o *feedback* imediato, a passagem imediata do receptor ao emissor e vice-versa. A utilização de comunicação não verbal como os gestos, a mímica e a entonação, por exemplo, facilita as retificações e explicações adicionais, observar as reações do receptor, e ainda a grande rapidez de transmissão. Como principais desvantagens da comunicação oral destacam-se o fato de ser efêmera, não permitindo qualquer registro e, conseqüentemente, não se adequando a mensagens longas e que exijam análise cuidadosa por parte do receptor.

Comment [WU119]: Idem ao comentário 4.

A comunicação escrita teve o seu auge, e ainda hoje predomina, nas organizações burocráticas que seguem os princípios da Teoria da Burocracia enunciados

da comunicação escrita, podemos destacar o fato de ser duradoura, permitir um registro, além de exigir uma maior atenção à organização da mensagem, sendo assim adequada para transmitir políticas, procedimentos, normas e regras. Adequa-se também a mensagens longas e que requeiram uma maior atenção e tempo por parte do receptor tais como relatórios e análises diversas. Como principais desvantagens destacam-se a referida ausência do receptor, o que impossibilita o *feedback* imediato, não permite correções ou explicações adicionais e obriga ao uso exclusivo da linguagem verbal.

A comunicação em organizações é utilizada como estratégia competitiva de mercado, englobando todas as formas de comunicação utilizadas para alcançar e interagir com seus públicos de interesse, de maneira integrada com sua missão e valores. Deve principalmente ser trabalha de forma convergente com os propósitos que pretende alcançar em curto, médio e longo prazo.

Comment [WU120]: Acho que a palavra correta seria “trabalhada” e não “trabalha”

Inicialmente, é necessário saber os objetivos do projeto, quem será o líder ou gerente e os limites do projeto. Basicamente é uma maneira de dizer por que realizar um projeto, a quem se reportar inicialmente e de maneira mais rebuscada dizer 'isso não será feito' ou 'isso não é de responsabilidade deste grupo nesse período'. Outro ponto importante é deixar claro qual o tempo estimado para desenvolver o projeto, qual folga se pode ter em determinada tarefa, quanto se pode gastar nele, quanto do valor pode ser acrescido e em quais circunstâncias. Após a definição e o planejamento de todas as áreas que envolvem o planejamento de um projeto, é o momento de fazer o controle do mesmo, a fim de superar as barreiras da comunicação identificadas.

Comment [WU121]: Acho que é desencessário esse "o planejamento de"

Barreiras são elementos que interferem e distorcem o processo de comunicação, dificultando ou impedindo o correto entendimento entre o emissor e o receptor. Essas barreiras podem ser de: **conhecimento**, onde inclui o despreparo para lidar com o processo oral ou escrito de comunicação, o uso da linguagem não familiar aos envolvidos e a falta de conhecimento do assunto a ser comunicado; **comportamentais**, quando ocorrem desconfianças entre as partes, atitudes hostis ou preconceituosas, ansiedade, desinteresse, a omissão intencional de fatos ou informações, não saber ouvir e falta de atenção ao assunto; **organizacionais e técnica**, quando as estruturas organizacionais são inflexíveis ou excessivamente burocráticas, possuem excesso de regras, procedimentos, padrões e equipamentos inacessíveis ou inadequados.

Comment [WU122]: Acho que o certo é "comportamental"

Comment [WU123]: Acho que o certo é "organizacional"

A comunicação se tornou um insumo estratégico para as empresas, um valor agregado aos seus negócios e conseqüentemente uma vantagem competitiva. Em razão destes motivos, as empresas necessitam construir uma identidade corporativa sólida, real, que pode ser realizada por meio da sua comunicação, o que realmente faz e finalmente, como é percebida pelos seus públicos [Maron 2008]. Fazer comunicação exige conhecimento, planejamento, execução e finalmente, mensuração de resultados, principalmente tratando-se de projetos, pois caso contrário não se alcança o objetivo proposto.

Comment [WU124]: Idem 4

- A comunicação eficaz também atua como a “cola” que irá manter unida uma equipe propiciar o alto desempenho. Mensagens claras são enviadas, recebidas e interpretadas de forma acurada.

Comment [WU125]: Acho que o certo é “propiciando”

O fato é que, não importa o quanto à tecnologia avance projetos, sempre serão executados por pessoas e, dependerão muito delas para que sejam bem sucedidos. Saber lidar com os desafios da comunicação é um fator crítico de sucesso para o projeto, e uma questão de sobrevivência no mercado para o gerente de projeto. No entanto, ainda existem vários casos em que excelentes profissionais, detentores de sólida formação técnica, por vezes se vêem em dificuldades no exercício da gerência de projetos, porque descobrem que além do perfil técnico, precisam por em prática uma série de habilidades

Comment [WU126]: Acho que a vírgula está no lugar errado, o correto seria “avance, projetos”

Comment [WU127]: Acho que essa virgule deveria ser excluída. Na maioria dos casos não se usa vírgula após um “e”

- *Reunião de kick-off:* marca o início efetivo do projeto. É uma oportunidade dos participantes se conhecerem, se manifestarem quanto suas expectativas;

- *Reuniões para registro e acompanhamento de mudanças*: formalizam as possíveis mudanças no planejamento e execução do projeto, que certamente trará algum impacto no mesmo. A partir dessas reuniões, será gerado um documento formal, padronizado, que posteriormente poderá ser ou não avaliado e aprovado ou por um comitê executivo do projeto;

- Preparar de uma pauta;

- Realizar de intervalos em reuniões longas;
- Utilizar o máximo do tempo e o esforço de todos os envolvidos (os profissionais poderiam tratar do mesmo assunto (ou decisão) por outro meio de comunicação (por exemplo, e-mail, carta, teleconferência, etc);

Comment [WU128]: Acho que o certo é "quanto às suas expectativas"

Comment [WU129]: Acho que o certo é "trarão" já que se refere a mudanças

Comment [WU130]: Acho que esse "ou" deve ser excluído pois está sobrando na frase.

Comment [WU131]: Esse "de" está sobrando. Ou tira ele ou coloca "Preparação" ao invés de "preparar"

Comment [WU132]: Idem ao 26

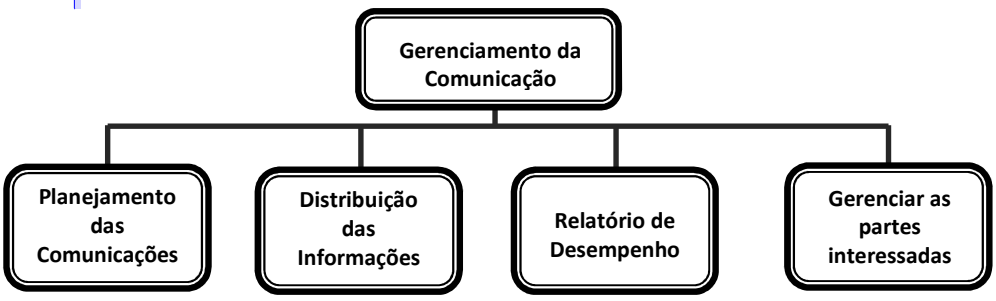
Comment [WU133]: Esse parêntese foi aberto e não foi fechado.

As comunicações do projeto sempre foram e continuarão sendo um ingrediente importante na fórmula para o seu sucesso. O PMBoK[12] considera a área de conhecimento "comunicação", como sendo vital para projetos e seu sucesso. Por isso, a gerência da comunicação é considerada uma das áreas mais importantes na gerência de projetos, apesar de ser muitas vezes negligenciada

O gerenciamento das comunicações do projeto é a área de conhecimento que emprega os processos necessários para garantir a geração, coleta, distribuição, armazenamento, recuperação e destinação final das informações sobre o projeto de forma oportuna. O PMBOK considera como uma boa prática da gestão de projetos utilizar os ativos de processos organizacionais, que será explicado na Seção 14.2.1. Apesar de um projeto ser único e temporário, as informações geradas e o histórico de um projeto podem e devem ser considerados como base de dados para outro projeto semelhante. Para se ter uma boa gestão da comunicação em um projeto, segundo o PMBOK [PMBOK 2004], além do planejamento é preciso cuidar da distribuição das informações, do relatório de desempenho e gerenciar as partes interessadas como mostra a Figura 14.2.

Comment [WU134]: O correto é "serão explicados"

Comment [WU135]: Corrigir a formatação dessa linha.



[12] PMBOK é um *Guide to the Project Management Body of Knowledge* desenvolvido pelo *Project Management*

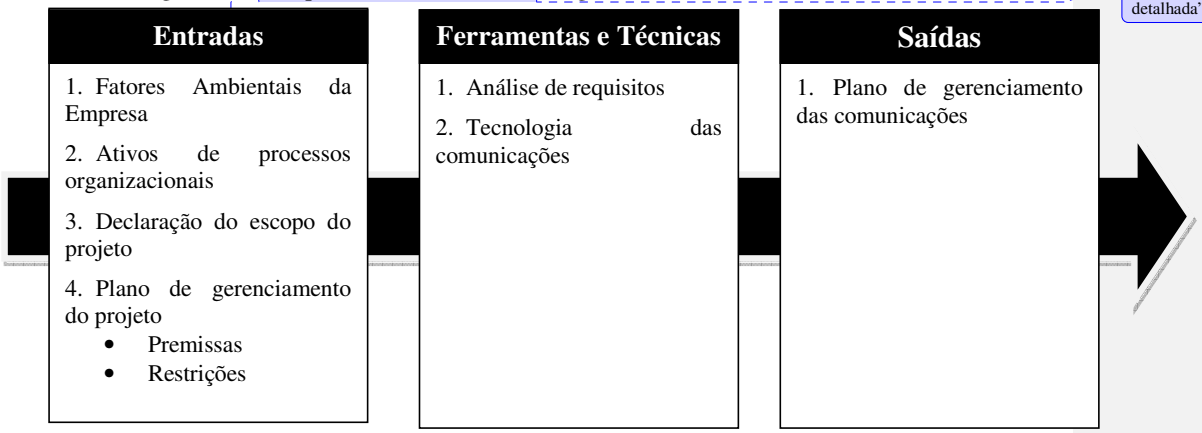
Figura 14.2. Visão geral do gerenciamento das comunicações do projeto [Adaptada de PMBOK 2004].

A seção anterior deste capítulo apresentou algumas questões relacionadas ao processo de comunicação em geral, sob o ponto de vista de sua utilização e importância no gerenciamento de projetos. O gerenciamento das comunicações em projetos estabelece, monitora e controla o fluxo de informações durante todo ciclo de vida dos projetos, sendo vital para o sucesso dos mesmos. Portanto, é de extrema importância que as comunicações em projetos sejam realizadas segundo processos organizados e disciplinados, capazes de gerar informações corretas e completas.

Os processos conforme mostrados na figura 2 se relacionam e interagem durante a condução do projeto, a descrição de cada um desses processos é feita por meio dos seguintes termos: *Entrada:* são representadas por documentos, planos, desenhos; *Ferramentas e Técnicas*: são aplicadas as entradas; S*aídas*: são representadas por documentos, resultados, produtos.

O processo de planejamento das comunicações determina as necessidades informações e comunicações das partes interessadas PMPOK [PMBOK 2004]. O planejamento envolve a identificação e definição das seguintes informações: quais são as informações e quem as precisa; quando precisarão e com qual freqüência; como será fornecida e por quem. Embora todos os projetos compartilhem a necessidade de comunicar informações, as necessidades das informações e os métodos de distribuição variam amplamente. Identificar as necessidades de informação dos interessados e determinar uma forma para atender a essas necessidades, é fator importante para o sucesso do projeto.

Em um número significativo de projetos a maior parte do planejamento da comunicação é feita como parte das fases iniciais do projeto. Entretanto, os resultados deste processo devem ser revistos regularmente durante o projeto e revisados se necessário para garantir aplicabilidade continua. Esse planejamento é freqüente e firmemente relacionado ao planejamento organizacional, visto que a estrutura organizacional do projeto terá um maior efeito nos requerimentos de comunicação. A composição do processo do planejamento das comunicações pode ser visualizada na Figura 14.3, as quais serão detalhadas a seguir.



Comment [WU136]: Dar um espaçamento vertical antes e depois da figura e corrigi-la para que ela fique dentro das margens estabelecidas.

Comment [WU137]: Corrigir a referência para a figura "14.2"

Comment [WU138]: O certo seria "necessidades de informações"

Comment [WU139]: O certo seria "quem precisa delas"

Comment [WU140]: Excluir o trema

Comment [WU141]: O certo é "a qual será detalhada" pois se refere a composição.

1. Entradas para o Planejamento das comunicações:

Ao longo do desenvolvimento do planejamento das comunicações do projeto, os Fatores Ambientais da Empresa devem ser considerados todos e quaisquer sistemas e fatores ambientais da empresa que influenciam o sucesso da comunicação no projeto. Isso inclui, porém não se limita aos seguintes itens:

- Cultura e estrutura organizacional ou da empresa;
- Normas governamentais ou do setor;

- Recursos humanos existentes (por exemplo, com facilidades, habilidades no ato de comunicar-se);

Comment [WU143]: Acho que o número dessa seção deveria ser 14.2.1.1.

Comment [WU144]: Rever esta frase que contém trechos repetidos

Comment [WU145]: Retirar esse "ou"

Comment [WU146]: Idem 39

Comment [WU147]: Não entendi esse exemplo.

Ao longo do desenvolvimento do planejamento da comunicação e da documentação subseqüente do projeto, todos ativos de processos organizacionais usados para influenciar o sucesso da comunicação do projeto podem ser obtidos a partir dos ativos de processos organizacionais. Todas e quaisquer organizações envolvidas no projeto podem ter políticas, procedimentos, planos e diretrizes formais e informais cujos efeitos devem ser considerados. Estes ativos também representam o aprendizado e o conhecimento das organizações obtidos de projetos anteriores; por exemplo, cronogramas terminados, dados de risco e dados de valor agregado. Tais ativos podem ser organizados de diversas formas, dependendo do tipo de setor, organização e área de aplicação [PMBOK 2004]. Por exemplo, os ativos de processos organizacionais poderiam ser agrupados em duas categorias: Processos e procedimentos da organização para realizar o trabalho base de conhecimento corporativo da empresa para armazenar e recuperar informações. É importante ressaltar que as lições aprendidas, bem como as informações históricas podem fornecer decisões e resultados com base em projetos anteriores semelhantes relacionados a problemas de comunicações.

A declaração do escopo do projeto descreve detalhadamente as entregas do projeto e o trabalho necessário para criar as entregas. Esta declaração também fornece um entendimento sobre o escopo do projeto para todas as partes interessadas no projeto e descreve os principais objetivos do projeto. Além disso, permite que a equipe do projeto realize um planejamento mais detalhado, orienta o trabalho da equipe durante a execução de tarefas e fornece a linha de base para avaliar solicitações de mudanças ou trabalho adicional e verificar se estão contidos dentro ou fora dos limites do projeto. O nível e o grau de detalhamento com que a declaração do escopo do projeto define o

Comment [WU148]: Idem ao 35

Comment [WU149]: Essa frase está sem sentido.

Comment [WU150]: O certo é "da organização obtido de projetos anteriores"

Comment [WU151]: Rever a pontuação a ser usada aqui.

Comment [WU152]: Não ficou claro quais são as duas categorias de processos organizacionais

Comment [WU153]: Verificar a repetição de palavras. Só a palavra "projeto" apareceu 5 vezes nesse trecho.

projeto. O gerenciamento do escopo do projeto, por sua vez, pode determinar e eficácia com que a equipe planeja, gerencia e controla a execução do projeto. A declaração do escopo do projeto inclui, diretamente ou referencia outros documentos como:

Comment [WU154]: O correto seria "a" ao invés de "e"

O plano de gerenciamento do projeto inclui as ações necessárias para definir, coordenar e integrar todos os planos auxiliares em um plano de gerenciamento do projeto. O conteúdo do plano de gerenciamento do projeto varia de acordo com a área de aplicação e a complexidade do projeto. Este plano define como o projeto é executado, monitorado, controlado e encerrado. Esse plano documenta o conjunto de saídas dos processos de planejamento do Grupo de processos de planejamento[13] e inclui:

2. Ferramentas e Técnicas para o Planejamento das comunicações:

Comment [WU155]: Acho que o número dessa seção deveria ser 14.2.1.2.

- Logística de quantas pessoas serão envolvidas no projeto e em que locais;

Comment [WU156]: Acho que esse "de" deveria ser trocado por ":"

- Necessidades externas de informações (por exemplo, a comunicação com as contratadas ou com os meios de comunicação);

Comment [WU157]: Acho que você quis dizer "as empresas contratadas" seria bom explicitar a palavra empresas pra facilitar o entendimento.

- *A urgência da necessidade de informações:* O sucesso do projeto depende da pronta disponibilidade de informações atualizadas freqüentemente ou relatórios por escrito emitidos regularmente seriam suficientes?

Comment [WU158]: Idem ao 35

3. Saídas do Planejamento das comunicações:

Comment [WU159]: Acho que o número dessa seção deveria ser 14.2.1.3.

A distribuição das informações envolve disponibilizar as informações às partes

solicitações de informações não previstas. Além disso, é imprescindível neste processo ter a clareza de quais e para quem as informações devem ser enviadas, conforme mostra as Figuras 14.4 e 14.5 respectivamente.

Comment [WU161]: Acho que ficaria melhor assim "...quais informações e para quem estas devem..."

Comment [WU162]: "mostram"

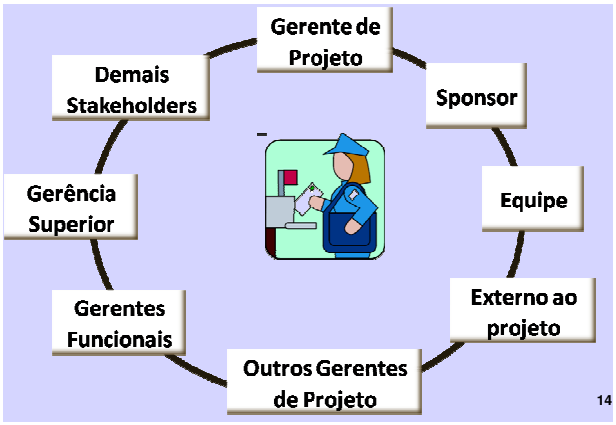


Figura 14.5. Para quem as informações que devem ser distribuídas [Adaptada de PERRELLI 2004].

Comment [WU163]: Dar um espaçamento vertical depois da figura, alterar a figura para escala de cinza e não colorida e retirar esse número "3" que está a direita dela

A composição do processo de distribuição das informações pode ser visualizada na Figura 14.6.

Comment [WU164]: Idem ao 31

Entradas	Ferramentas e Técnicas	Saídas
1. Plano de gerenciamento	1. Habilidades de	1. Ativos processos

1. Entradas para a Distribuição das informações:

Comment [WU165]: Acho que a numeração dessa seção deveria ser 14.2.2.1

2. Ferramentas e Técnicas para a Distribuição das informações

Comment [WU166]: Acho que a numeração dessa seção deveria ser 14.2.2.2

O sistema de coleta e recuperação das informações pode ocorrer por diversos meios, inclusive sistemas manuais de arquivamento, software de gerenciamento de projetos, bancos de dados eletrônicos, e sistemas que possibilitam o acesso à documentação técnica, como especificações de design, plano de testes e desenhos de engenharia.

Comment [WU167]: Acho que o certo é "possibilitem"

O processo de lições aprendidas se concentra na identificação dos sucessos e fracassos do projeto, incluindo sugestões para melhorar o desempenho futuro dos

projeto. As lições aprendidas são compiladas, formalizadas e armazenadas durante o projeto. Além disso, é importante destacar alguns resultados específicos das lições aprendidas, como:

3. Saídas da Distribuição das informações

A atualização dos ativos processos organizacionais é composta basicamente:

Comment [WU169]: Acho que a numeração dessa seção deveria ser 14.2.2.3

Comment [WU170]: Acho que o certo é "ativos de processos"

Comment [WU171]: Acho que o certo é "composta basicamente de:"

- Apresentações do projeto. A equipe fornece informações, formal ou informalmente, a algumas ou a todas as partes interessadas no projeto. Essas informações são relevantes para as necessidades da audiência.
- *Feedback* das partes interessadas. As informações recebidas das partes interessadas relativas às operações do projeto podem ser distribuídas e usadas para modificar ou melhorar o desempenho futuro do projeto.
- Notificações das partes interessadas. É possível fornecer informações às partes interessadas sobre problemas resolvidos, mudanças aprovadas e andamento geral do projeto.

Comment [WU172]: Acho que seria ":" ao invés de "."

Comment [WU173]: Idem ao 67

Comment [WU174]: Idem ao 67

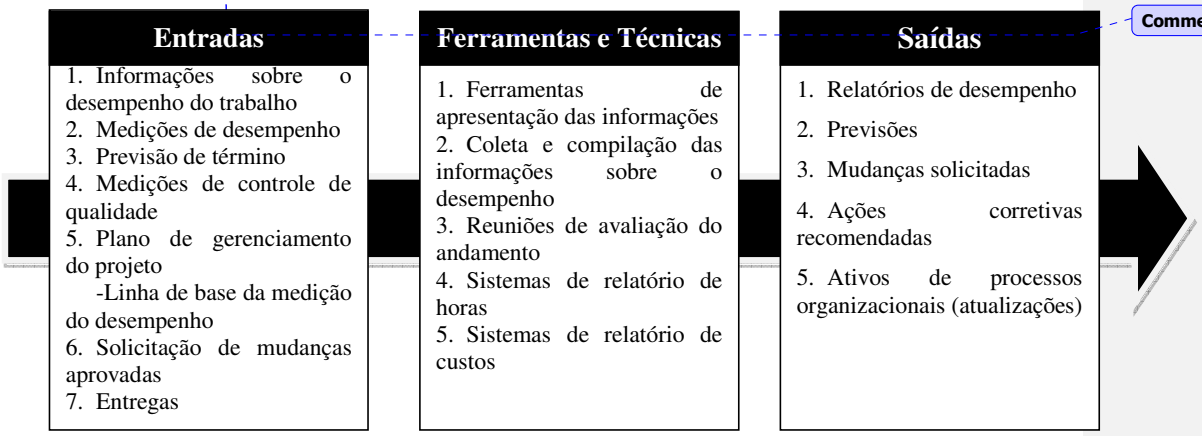
As mudanças surgidas no processo de Distribuição das informações devem causar mudanças no plano de gerenciamento do projeto e no plano de gerenciamento das comunicações. As mudanças solicitadas (adições, modificações, revisões) no plano de gerenciamento do projeto e nos seus planos auxiliares são revisadas e a destinação é gerenciada pelo processo Controle integrado de mudanças[15].

O processo de relatório de desempenho envolve a coleta de todos os dados de linha de base e a distribuição das informações sobre o desempenho às partes interessadas. Em geral, o relatório de desempenho contém informações, incluindo o modo como os recursos estão sendo utilizados para atingir os objetivos do projeto.

Comment [WU175]: Acho que o certo aqui seria "da" ao invés de "de"

O relatório de desempenho deve normalmente fornecer informações sobre escopo, cronograma, custo e qualidade. Muitos projetos também exigem informações

sobre risco e aquisições. Os relatórios podem ser preparados de forma abrangente ou com base em exceções. A composição do processo é mostrada na Figura 14.7.



Comment [WU176]: Idem ao 31

1. Entradas para o Relatório de desempenho

Comment [WU177]: Acho que a numeração dessa seção deveria ser 14.2.3.1

As informações sobre o desempenho do trabalho tratam a situação atual das entregas. Já referente ao que foi realizado no projeto, as informações são coletadas como parte da execução e alimentadas no processo Relatório de desempenho. As informações sobre o andamento das atividades que estão sendo executadas para realizar o trabalho do projeto são coletadas rotineiramente como parte da execução do plano de gerenciamento do projeto. Essas informações incluem, mas não estão limitadas a:

Comment [WU178]: Acho que o certo seria "alimentação do"

		Custo orçado do trabalho agendado a ser terminado em uma atividade ou o componente da EAP[16] até um determinado momento.
		A VC é igual ao valor agregado (VA) menos o valor planejado (VP). A variação de prazos será no final igual a zero quando o projeto for terminado, devido todos os valores planejados terem sido agregados.

Comment [WU180]: Acho que o certo aqui seria "VP"

As medições de controle da qualidade são os resultados das atividades de controle da qualidade fornecidos como *feedback* para o processo de Garantia da Qualidade (GQ)[17] para uso na reavaliação e na análise dos processos e padrões de qualidade da organização executora.

O plano de gerenciamento do projeto fornece informações sobre a linha de base. A linha de base da medição de desempenho trata-se de um plano aprovado para o trabalho do projeto em relação ao qual a execução do projeto é comparada, sendo medidos os desvios para o controle gerencial. A linha de base da medição de desempenho normalmente integra os parâmetros de escopo, cronograma e custo de um projeto, mas pode também incluir parâmetros técnicos e de qualidade.

Comment [WU181]: Excluir esse espaçamento vertical

2. Ferramentas e Técnicas para o relatório de desempenho

Comment [WU182]: Acho que a numeração dessa seção deveria ser 14.2.3.2

As ferramentas de apresentação das informações incluem os pacotes de software que por sua vez contêm análise de planilhas, relatórios de tabelas, apresentações ou capacidades gráficas podem ser usados para criar imagens de qualidade para a apresentação dos dados de desempenho do projeto.

Comment [WU183]: Acho que o certo é "gráficas que podem"

3. Saídas do Relatório de desempenho

Comment [WU184]: Acho que a numeração dessa seção deveria ser 14.2.3.3

Os relatórios de desempenho organizam e sintetizam as informações coletadas e apresentam os resultados de qualquer análise comparados com a linha de base da medição de desempenho. Os relatórios devem fornecer informações sobre o progresso e o andamento e o nível de detalhes exigido pelas diversas partes interessadas, conforme documentado no plano de gerenciamento das comunicações.

Comment [WU185]: Acho que deveria ser uma "," ao invés do "e"

- *PMO*: Se existir na organização executora, o PMO poderá ser uma parte interessada se tiver responsabilidade direta ou indireta pelo resultado do projeto.

Comment [WU186]: Acho que seria interessante explicitar o que é um PMO para melhorar o entendimento do capítulo.

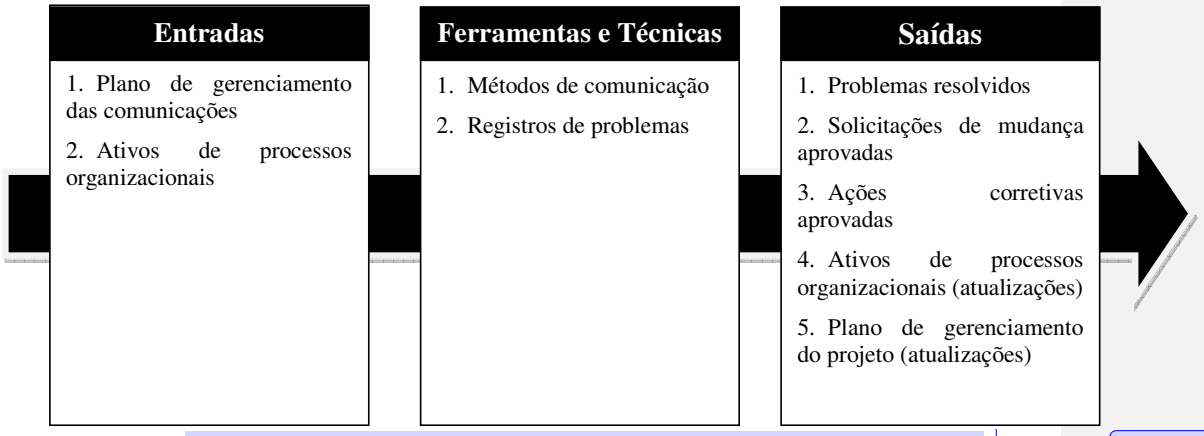


Figura 14.11. Gerenciar as partes interessadas [Adaptado de PMBOK 2004].

Comment [WU187]: Idem ao 31

1. Entradas para Gerenciar as partes interessadas

Comment [WU188]: Acho que a numeração dessa seção deveria ser 14.2.4.1

Os requisitos e expectativas das partes interessadas propiciam um entendimento das metas, objetivos e nível de comunicação das partes interessadas durante o projeto. As necessidades e expectativas são identificadas, analisadas e documentadas no plano de gerenciamento das comunicações (Seção 2.1.3).

Comment [WU189]: Verificar qual o número correto da seção a ser referenciada

2. Ferramentas e Técnicas para Gerenciar as partes interessadas

Comment [WU190]: Acho que a numeração dessa seção deveria ser 14.2.4.2

3. Saídas de Gerenciar as partes interessadas

Comment [WU191]: Acho que a numeração dessa seção deveria ser 14.2.4.3

Neste mercado cada vez mais competitivo, a comunicação não é apenas um diferencial estratégico, e sim uma necessidade humana, profissional e gerencial para conduzir mecanismos, processos, projetos e atividades de âmbito empresarial. A comunicação deve ser objetiva e clara, respeitando os limites estipulados por agentes definem o negócio, seja de um projeto ou organização. Tendo em vista que as organizações que apresentam uma comunicação mais eficaz garantem seu espaço no mercado, conseqüentemente apresentam resultados mais satisfatórios, assim, garantindo a satisfação dos clientes. Por este motivo, tem surgido uma grande demanda ao incentivo de pesquisas que levem em consideração à procura por formas de melhoria das práticas do processo de comunicação.

Este capítulo procurou introduzir ao leitor a importância e boas práticas que diz respeito à comunicação, apresentando o processo de comunicação em Organizações e projetos, sendo destacadas as principais barreiras, estratégias e quão é necessária a eficácia da comunicação. O gerenciamento das comunicações em projetos também foi destacado, pois estabelece, monitora e controla o fluxo de informações durante todo ciclo de vida dos projetos, sendo vital para o sucesso dos mesmos. Neste gerenciamento também foram apresentados um conjunto processos, os quais como mais uma tentativa de atingir os objetivos desejados, de maneira natural e transparente, minimizando os riscos, conflitos e frustrações que possam vir ocorrer ao longo do projeto.

Comment [WU192]: Acho que o certo seria "agents que definem"

Comment [WU193]: Excluir o trema

Comment [WU194]: Acho que ficaria melhor se esse texto fosse substituído por "referentes ao tema"

Comment [WU195]: Acho que o certo seria "conjunto de processos"

Comment [WU196]: Acho que esse "vir" deve ser retirado do texto

1. Comunicação Empresarial:

Comment [WU197]: Acho que a numeração deveria ser 14.4.1

A área de comunicação empresarial tem um papel importante na "administração de percepção" e na leitura do ambiente social da organização. Nessa perspectiva várias pesquisas, propostas, ferramentas vem contribuindo para a análise dos planos de negócios da organização, identificando problemas e oportunidades no campo da comunicação para garantir o sucesso da empresa no mercado.

Comment [WU198]: Acho que o certo seria "e" ao invés da ","

2. Comunicação em Desenvolvimento Distribuído de Software (DDS):

Comment [WU199]: Acho que a numeração deveria ser 14.4.2

Em Projetos Distribuídos, a comunicação é a base para definir como serão repassadas as informações para as partes interessadas envolvidas no projeto. Não existe

conclusão no prazo, dentro do custo e com qualidade. Na literatura, podem ser encontradas pesquisas e artigos com estudos focados neste assunto.

2. Para ampliar o conhecimento sobre o Gerenciamento da Comunicação é recomendada a leitura do livro Gerenciamento da comunicação em projetos, Chaves, L; Neto, F.; Pech, G.; Carneiro, M., 2006, o qual faz parte da Série Gerenciamento de Projetos das Publicações Fundação Getúlio Vargas. Este livro visa oferecer a estudantes, gestores e técnicos uma base didáticas para que estes possam aplicar o processo de comunicação em uma maneira eficaz em projetos de software.

3. Para um melhor conhecimento sobre a Comunicação Empresarial é sugerida a leitura das revistas: Comunicação Empresarial da Associação Brasileira de Comunicação Empresarial (Aberje) e a Comunicação Coorporativa do Valor Setorial, as quais retratam a comunicação empresarial sobre diversos aspectos relacionados a atualidade. Tais revistas podem ser visualizadas, respectivamente nos seguintes links: http://www2.ideavalley.com.br/aberje/flip/ e http://208.96.41.18/valoreconomico/home.aspx?pub=27&edicao=1.

4. Para ampliar o entendimento sobre Comunicação em Desenvolvimento Distribuído de Software é sugerida a leitura da dissertação Uma Proposta de Boas Práticas do Processo de Comunicação para Projetos de Desenvolvimento Distribuído de Software, Junior, L. Nesta dissertação é mostrada práticas da comunicação em DDS, a partir de uma experiência prática de empresas.

Comment [WU200]: Acho que o certo é "didática"

Comment [WU201]: Acho que o certo é "de"

Comment [WU202]: Seria interessante colocar qual foi a data do ultimo acesso, pois estas fonts podem sair do ar futuramente.

Comment [WU203]: Acho que o correto seria "são"

13. Cite quais informações e para quem devem ser distribuídas.

Comment [WU204]: Não ficou claro o objetivo dessa pergunta, melhor reformular ou excluir.

Vargas, R. (1999). Gerenciamento de Projetos - Estabelecendo diferenciais competitivos, 6ª edição, Editora Braspor.

Comment [WU205]: Acho que o nome da editor é "Brasport"

Verzuh, E. (2000). MBA compacto, Gestão de Projetos. 3. ed. Rio de Janeiro: Campus.398 p.

15.1. IMPORTÂNCIA DA MEDIÇÃOERROR! BOOKMARK NOT DEFINED.

15.2. O QUE SÃO MÉTRICASERROR! BOOKMARK NOT DEFINED.

15.3. MEDIÇÃO DE SOFTWAREERROR! BOOKMARK NOT DEFINED.

15.3.1. MÉTRICAS TÉCNICASERROR! BOOKMARK NOT DEFINED.

15.3.2. MÉTRICAS DE QUALIDADEERROR! BOOKMARK NOT DEFINED.

15.3.3. MÉTRICAS DE PRODUTIVIDADEERROR! BOOKMARK NOT DEFINED.

15.3.4. MÉTRICAS ORIENTADAS AO TAMANHOERROR! BOOKMARK NOT DEFINED.

15.3.5. MÉTRICAS ORIENTADAS À FUNÇÃOERROR! BOOKMARK NOT DEFINED.

15.3.6. MÉTRICAS ORIENTADAS À SERES HUMANOSERROR! BOOKMARK NOT DEFINED.

15.4. BOAS PRÁTICAS NA IMPLANTAÇÃO DE PROGRAMAS DE MEDIÇÃO NAS ORGANIZAÇÕES ERROR! BOOKMARK NOT DEFINED.

15.5. ESTIMATIVAS DE SOFTWAREERROR! BOOKMARK NOT DEFINED.

15.6. POR QUE É DIFÍCIL ESTIMARERROR! BOOKMARK NOT DEFINED.

15.7. TÉCNICAS DE ESTIMATIVAERROR! BOOKMARK NOT DEFINED.

15.7.1. FPAERROR! BOOKMARK NOT DEFINED.

15.7.2. PONTOS DE CASO DE USOERROR! BOOKMARK NOT DEFINED.

15.7.3. COCOMOERROR! BOOKMARK NOT DEFINED.

15.8. SUGESTÕES DE LEITURAERROR! BOOKMARK NOT DEFINED.

15.9. TÓPICOS DE PESQUISAERROR! BOOKMARK NOT DEFINED.

15.10. EXERCÍCIOSERROR! BOOKMARK NOT DEFINED.

REFERÊNCIASERROR! BOOKMARK NOT DEFINED.

Comment [A206]: Primeiro é Tópicos de Pesquisa e depois Sugestões de Leitura.

Neste capítulo são considerados conceitos fundamentais acerca da medição e estimativas de software. O capítulo começa trazendo uma visão geral sobre medição de software, com a motivação para se realizar este tipo de atividade, conceitos básicos e métricas de software. A discussão, em seguida, passa a ser a respeito de modelos de processo de medição de software.

Além do que já foi dito no parágrafo anterior, boas práticas na implantação de programas de medição nas organizações é uma abordagem que também é feita no capítulo, com o objetivo de trazer dicas ao leitor sobre como proceder bem quando do momento de levar à empresa a instituição de um programa de medição. Pontos de Caso de Uso, COCOMO, entre outras técnicas para estimativas em projetos de software, formam a última parte do capítulo.

É fato que a indústria de software continua, até hoje, lidando com projetos de software mal sucedidos. Uma pesquisa do [The Standish Group 2009] apontou que mais projetos estão falhando e menos estão tendo sucesso. E uma das causas que pode ser apontada para este problema é uma gestão não tão bem planejada e executada dos projetos de software.

Uma boa maneira de realizar uma gestão de projeto de software com um pouco mais de garantias de que o trabalho realizado não será um fracasso total, é executando, entre outras coisas, atividades de medição e estimativas. Afinal de contas, e em concordância com as palavras de [Tom DeMarco 1982], não se pode controlar o que não se pode medir.

Comment [A207]: Seria interessante neste espaço mostrar o objetivo do capítulo, antes de aborda o que o capítulo apresenta.

Comment [A208]: Acho que ficaria melhor deste jeito:
"*O capítulo aborda uma visão geral sobre medição de software, a sua importância, conceitos básicos, métricas de software, bem como os modelos de processo de medição de software.*

Comment [A209]: Acho que este parágrafo poderia ser continuidade do anterior. Acho que ficaria melhor assim:
"Além da visão geral, também é realizada uma abordagem sobre as boas práticas na implantação de programas de medição nas organizações e as possíveis técnicas para estimativas em projetos de software, como por exemplo: Pontos de Caso de Uso e COCOMO."

Comment [A210]: Acho que ficaria melhor assim:
Segundo a pesquisa de [The Standish Group 2009] foi detectado que existe um número mais significativo de falhas dos projetos do que o sucesso nos mesmos.

Comment [A211]: Uma das causas que reflete este problema é uma gestão mal planejada e organizada

Comment [A212]: Acho que esse paragrafo pode ser unido com o anterior e poderia ficar assim:
" Diante desse contexto, é importante destacar que uma boa maneira de realizar gestão de projeto de software é aplicando atividades de medição e estimativa, a fim de garantir que o trabalho

Quando o software é medido, é feito, entre outras funções, de dar ao gerente do projeto de software valores reais para que possa enxergar o projeto de uma maneira quantificada, apoiando a tomada de decisões e, subsídios para que ele possa realizar estimativas mais próximas da futura realidade final dos projetos que estão por vir. Além deste fator, o panorama encontrado em muitos projetos de software é o mesmo, e não é animador, apontando para a necessidade de medições. [Fernandes 1995] enumera 12 situações comuns a quem desenvolve software e que não contribuem em nada para uma gestão mais efetiva de projetos deste tipo de produto, as quais:

1. Estimativas de prazos, custos, recursos e esforço são realizados com base no julgamento pessoal do gerente de projeto.

7. Os fatores que impactam a produtividade e a qualidade não são determinados.

9. Os custos de não conformidade ou da má qualidade não são medidos.

Apesar da implantação de um programa de métricas na empresa ser uma atividade que represente mais trabalho e traga alguns custos imediatos, fora o fato de dar a impressão inicial de que tudo aquilo não está sendo útil, os ganhos futuros com a formação de uma base de dados composta por métricas de projetos de vários anos farão deste repositório uma "voz" muito precisa durante a estimativa de prazos, custos, esforço e recursos em outros projetos, dando a clientes e à própria equipe uma certeza mais absoluta do que será necessário, em todos os sentidos, para a realização do mesmo.

Diferentemente das outras áreas da engenharia, onde a medição é algo que rege o trabalho realizado dentro das atividades destas ciências, na engenharia de software a medição ainda se encaminha para uma tentativa de firmamento dentro dos processos que se valem desta área do conhecimento. O que acontece aqui, na verdade, é que o fato de medir software e os processos para o desenvolvimento de tais produtos parece, a muitos, algo tão abstrato e subjetivo. No entanto, medidas de software e métricas têm sido derivadas ao longo do tempo para que medições do produto sejam feitas.

No geral, a importância das métricas de software relaciona-se ao fato de darem

Comment [A213]: Acho que poderia ficar assim:
"Quando o software é construído, e avaliado, tem como finalidade fornecer ao gerente do projeto de software valores reais para que possa enxergar o projeto de uma maneira quantificada, apoiando a tomada de decisões e, subsídios para seja possível a realização de estimativas mais próximas da realidade final dos futuros projetos."

Comment [A214]: Poderia ficar assim:
"Além deste fator, o panorama encontrado em diversos projetos de software é o mesmo. Portanto, panorama não é animador, já que aponta para a necessidade de medições. De acordo com Fernandes [Fernandes 1995], existem doze situações comuns referente aos desenvolvedores de software, as quais não contribuem positivamente para uma gestão mais efetiva de projetos deste tipo de produto,tais como:"

Comment [A215]: Ao invés de números acho que fica melhor colocar marcadores.

Comment [A216]: Na produtividade e na qualidade.

Comment [A217]: péssima

Comment [A218]: acho que fica melhor **representar**

Comment [A219]: conceba

Comment [A220]: oferecer

Comment [A221]: seria melhor mudar essa expressão, pois ficou meio estranha no texto.

Comment [A222]: proporcionando

Comment [A223]: Acho que isso pode ser retirado.

Comment [A224]: Poderia retirar o dentro e ficar somente "…. nas atividades..."

Comment [A225]: Acho que poderia ficar assim*: "… na engenharia de software, por exemplo, …."*

Comment [A226]: Poderia retirar essa expressão e começar com o *"Na verdade..."*

Comment [A227]: Pode ser retirado

Comment [A228]: Feitas é no sentido de construídas ou de Realizadas?

Comment [A229]: Em geral fica melhor

regras claramente definidas, permitindo que os gestores do projeto tenham um entendimento imediato do que está sendo feito, e não posteriormente. Tudo isso faz com que eles possam descobrir problemas no decorrer do projeto antes que estes se transformem em algo muito mais difícil de ser resolvido mais tarde. Em suma, o processo de software é medido num esforço para melhorá-lo, ao passo que o produto é medido num esforço para aumentar sua qualidade [Pressman 1995].

15.2. O que são Métricas

Existem alguns conceitos comuns em discussões sobre medições de software e que valem a pena serem discutidos a fim de trazer uma maior clareza sobre o assunto. Primeiramente, a definição do termo métrica. Pode-se dizer, de certa forma, que não há nenhuma definição aceita como a mais correta para o termo. Alguns profissionais usam o termo métrica intercambiadamente com o termo medição. Já outros fazem a distinção entre medição e métrica, onde métrica indica uma medição e um modelo ou teoria baseados nela [Shepperd e Ince 1993].

A definição dada no parágrafo anterior para o termo métrica parece não esclarecer muito acerca desta. No entanto, [Shepperd e Ince 1993] dão um bom entendimento do conceito de métrica ao afirmarem que esta, em engenharia de software, é nada mais e nada menos, do que aquilo que transmite uma medição de um produto ou processo de software.

Já dá para se ter, a partir do que foi exposto, uma boa noção acerca da definição do termo métrica. Mas tomando como referência a própria definição citada por [Shepperd e Ince 1993], vem, de antemão, o questionamento sobre o que vem a ser medição e também, diretamente relacionado, o seu substantivo correlato: medida. Pode-se entender medição como o ato de obter valores de uma característica ou atributo de uma coisa ou entidade qualquer. Ou seja, tomando como exemplo uma pessoa. A respeito da entidade pessoa existe um conjunto de características ou atributos os quais são passíveis de receberem valores, tal qual a altura, o peso e etc. O peso de uma pessoa pode ser 85 e sua altura 180. Referente a esses valores obtidos, costuma-se atribuir unidades de medida que serão responsáveis por dar a noção de quantidade de um valor qualquer e que servem de base para comparações com outros valores com mesma unidade de medida. No caso, poderíasse ter peso igual a 85 kg e altura igual a 180 cm.

O mesmo acontece com o software. Enxergando o software e o processo usado para o seu desenvolvimento como entidades, existem atributos pertinentes a estas duas coisas que podem ser medidos. [Pressman 2006] exemplifica o abordado até agora dizendo que quando um único ponto de dados foi coletado (por exemplo, o número de erros descoberto em um único componente de software), uma medida foi estabelecida, e que uma métrica de software é aquela que relaciona medidas individuais de algum modo (por exemplo, o número médio de erros encontrados por teste de unidade).

Visto também como um conceito importante em discussões sobre medição de software é o termo indicador. O objetivo do engenheiro de software com a coleta de medidas do processo e do produto final é, a partir das medidas que ele tem em mãos, originar métricas de tal modo que chegue a indicadores os quais serão úteis para os gerentes de projeto ou à alta administração tomar as melhores decisões quanto ao

Comment [A231]: Acho q ficaria melhor assim: *"....do produto, o qual é desenvolvido e também o processo é utilizado com base em um conjunto de regras claramente definidas, permitindo que os gestores do projeto tenham um entendimento imediato do que está sendo realizado."*

Comment [A232]: Melhor mudar essa expressão.

Comment [A233]: Acho q fica melhor assim: *"...possam descobrir problemas no decorrer do projeto o mais rápido possível, antes que estes se transformem em problemas mais difíceis de serem resolvidos."*.

Comment [A234]: Definição de métricas

Comment [A235]:
Acho q ficaria melhor assim:
"Dentre esses conceitos, pode ser destacado a definição de métrica, sendo que não existe nenhum conceito mais correto para este termo. Portanto, alguns profissionais usam o termo métrica intercambiadamente com o termo medição. No entanto, outros fazem a distinção entre medição e métrica, onde métrica indica uma medição e um modelo ou teoria baseados nela [Shepperd e Ince 1993]."

Comment [A236]: Acho que pode juntar com o paragrafo anteriror e ficaria assim:
Apesa r das definições expostas, o conceito do termo métrica ainda não ficou claro. No entanto, [Shepperd e Ince 1993] oferecem um bom entendimento do conceito ao afirmarem que esta, em engenharia de software, é aquilo que transmite uma medição de um produto ou processo de software.

Comment [A237]: Acho que ficaria melhor assim:
De acordo com os conceitos expostos, ficou clara a definição do termo métrica.

Comment [A238]: Acho que ficaria melhor assim:
No entanto, ao tomar como referência a própria definição de Shepperd & Ince [Shepperd & Ince 1993], surge o questionamento sobre o que vem a ser medição e também, diretamente relacionado, o seu substantivo correlato: medida.

Comment [A239]: Acho que ficaria melhor assim:
A partir dessa indagação, pode-se entender medição como o ato de obter valores de uma característica ou atributo de uma entidade qualquer, como por exemplo, uma pessoa. A respeito da entidade ... [1]

Comment [A240]: O caso abordado anteriormente, também acontece com o software, onde o software e o processo usado para o desenvolvimento podem ser vistos como enti ... [2]

Comment [A241]: Não tem mt haver com o que foi abordado anteriormente.

Comment [A242]: Colocar virgule depois de indicadores

Comment [A243]: úteis para os gerentes de

fornecem profundidade na visão do processo de software, do projeto ou do produto, permitindo assim que os engenheiros de software/gerente do projeto possam tornar o software ou o processo para seu desenvolvimento melhores, tudo isto a partir das conclusões tiradas dos indicadores [Pressman 2006].

Comment [A244]: Colocar virgule depois do permitindo assim.

Comment [A245]: Retirar a barra e colocar o "e".

Comment [A246]: Retirar o tudo isto.

Comment [A247]: Acho que fica melhor "retiradas".

No mundo físico, as medições podem ser divididas em duas categorias: medidas diretas e medidas indiretas. Tomando como exemplo um pneu, pode-se considerar como uma medida direta deste, a sua circunferência. E como uma medida indireta do pneu pode-se considerar a sua qualidade, medida, por exemplo, através de testes de resistência [Caramoni et al. 2008]. Com o software, as coisas funcionam de forma idêntica.

Comment [A248]: Acho que ficaria melhor assim:
Tais medições também funcionam de forma

Entre as medidas diretas do processo de engenharia de software estão o custo e o esforço aplicados para tal. No software, linhas de código escritas, velocidade de execução, tamanho da memória e defeitos registrados ao longo de um determinado espaço de tempo são medidas diretas do produto. Funcionalidade, qualidade, complexidade, eficiência, confiabilidade, etc., são atributos de medidas indiretas do software [Pressman 2006].

Comment [A249]: Excluir esse pedaço.

Comment [A250]: É melhor substituir por entre outros.

Segundo [Pressman 1995], as métricas de software podem ser divididas em mais categorias próprias. O autor divide as métricas de software nas seguintes categorias:

Comment [A251]: Substituir por: Segundo Pressman [Pressman 1995]

Nas próximas seções cada uma das categorias de métricas de software é estudada.

Comment [A252]: Substituit por : sera apresentada.

15.3.1 Métricas Técnicas

Comment [A253]: Vais acrescentar mais assunto neste tópico?

A qualidade do software pode ser medida tanto durante o seu desenvolvimento quanto depois que o produto tiver sido implementado. As métricas de qualidade derivadas durante o desenvolvimento do produto formam uma base quantitativa para a tomada de decisões referentes ao projeto e aos testes que serão feitos com o software

vez, as métricas formadas depois da implementação do software dão uma indicação ao gerente e à equipe de projeto sobre a efetividade do processo de engenharia de software aplicada ao projeto. Especial atenção é dada ao número de defeitos descobertos e à manutenibilidade do sistema [Boaventura 2001]. ...

Comment [A254]: Substituir por oferecem.

Comment [A255]: Excluir essas palavras.

Comment [A256]: Acho que é melhor assim. Uma atenção em especial

Guarizzo, Karina, (2008) "Métricas de Software", http://bibdig.poliseducacional.com.br/document/?view=184.

Comment [A257]: Organizar as referências, de acordo com o padrão estabelecido para o livro.

17 GESTÃO DE PROGRAMAS 400

17.1 PROGRAMAS 400

17.2 GERENCIAMENTO DE PROGRAMAS 402
17.2.1 RELAÇÃO ENTRE GERENCIAMENTO DO PROGRAMA E GERENCIAMENTO DO PROJETO 403
17.2.2 TEMAS DO GERENCIAMENTO DE PROGRAMA 404
17.2.2.1 GERENCIAMENTO DE BENEFÍCIOS 404
17.2.2.2 GERENCIAMENTO DE STAKEHOLDERS 405
17.2.2.3 GOVERNANÇA 406
17.2.3 CICLO DE VIDA DO PROGRAMA 407
17.2.3.1 FASE 1: SET UP PRÉ-PROGRAMA 408

17.2.3.4 Fase 4: Benefícios Incrementais 410
17.2.3.5 Fase 5: Encerramento 411
17.3 Processos do Gerenciamento de Programa 411
17.3.1 Grupo Processos de Iniciação 412
17.3.2 Grupo Processos de Planejamento 414
17.3.3 Grupo Processos de Execução 416
17.3.4 Grupo Processos de Monitoramento e Controle 417
17.3.5 Grupo Processos de Encerramento 417

17.4 TÓPICOS DE PESQUISA 418

17.5 SUGESTÕES DE LEITURA 419

17.6 EXERCÍCIOS 419

17.7 REFERÊNCIAS 419

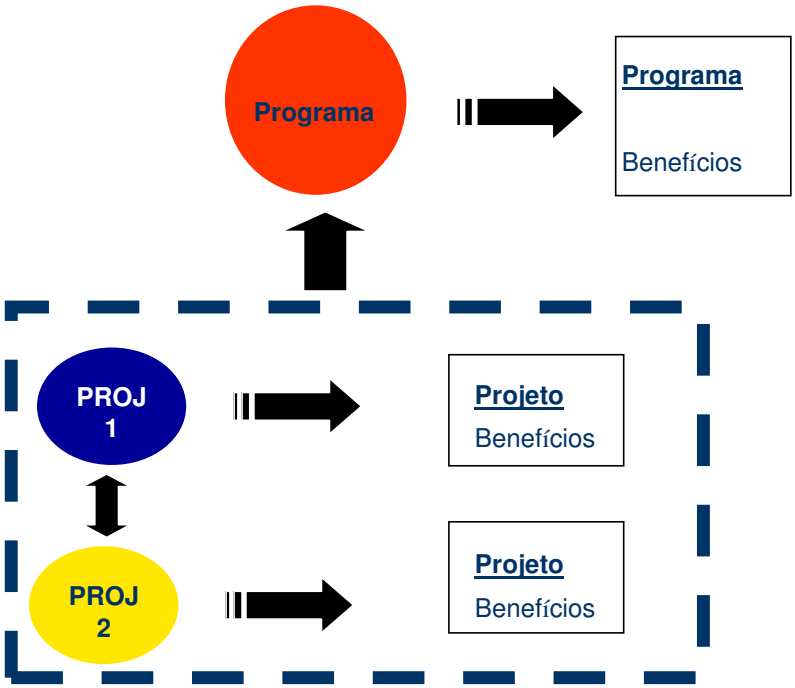
Comment [A258]: Criar um índice para figuras e tabelas.

Capítulo

.

- **Programas**

Comment [A259]: Corrigir a sequencia do tópicos com o capitulo.


Programa
Programa
Benefícios
PROJ 1
Projeto
Benefícios
PROJ 2
Projeto
Benefícios

Ações relativas: Avaliar o valor e o impacto do programa na organização; analisar os impactos das mudanças; identificar as interdependências dos benefícios intermediários que serão entregues pelos projetos que compõem o programa. Garantir que os benefícios sejam:

Comment [PRCA260]: Acaharia mais interessante que fizesses uma pequena introdução às ações relativas, do que já jogá-las no texto de bate-pronto

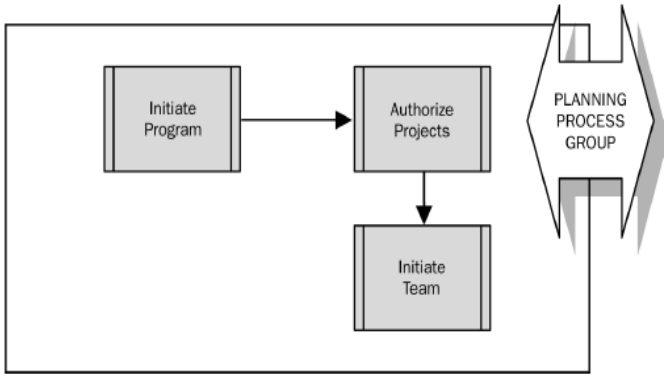
Programs						
Pre-Program Set Up	Program Set Up	Establish Program Management & Technical Infrastructure	Deliver Incremental Benefits	Close the Program	Transition	Ongoing Operations

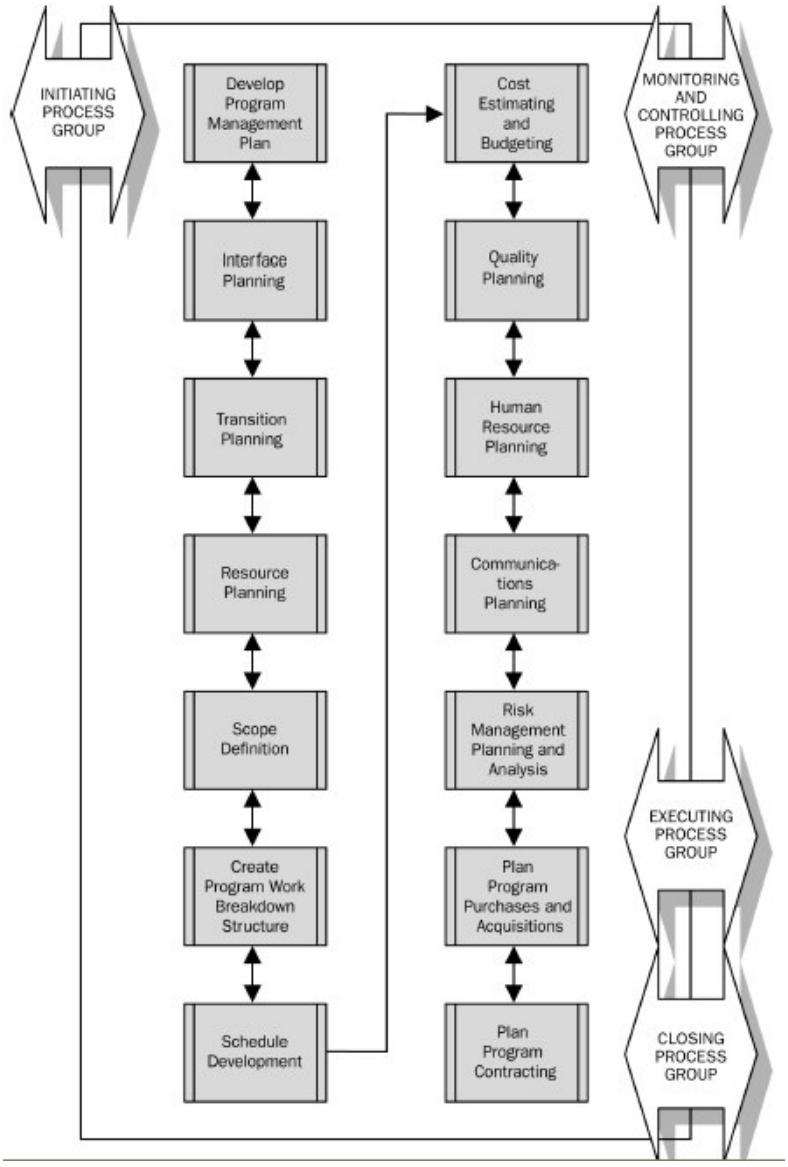
ENTRADA
• Declaração de escopo do programa • Critério de seleção dos projetos • Plano estratégico

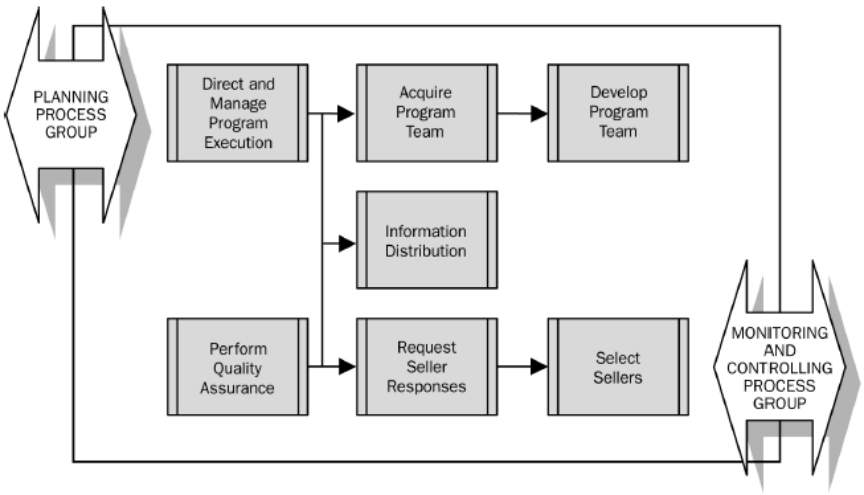
SAÍDA
• Requisitos de relatórios do Programa • *Project Charter* • Designação do gerente do projeto • Identificação do Patrocinador do projeto • Aprovação das reservas financeiras do projeto

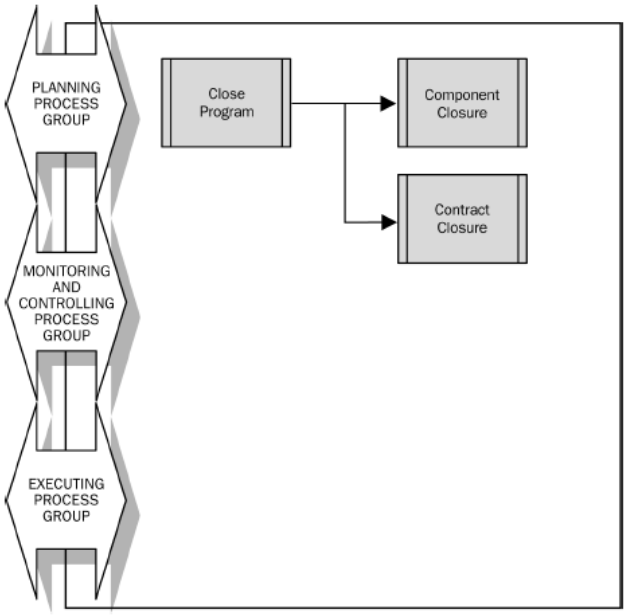
ENTRADA
• Prática de recrutamento • Descrição de recursos

SAÍDA
• Núcleo do time do programa designado • Gerente do programa designado • Time do Programa


Initiate
Program
Authorize
Projects
Initiate
Team
PLANNING
PROCESS
GROUP


INITIATING PROCESS GROUP
Develop Program Management Plan
Interface Planning
Transition Planning
Resource Planning
Scope Definition
Create Program Work Breakdown Structure
Schedule Development
Cost Estimating and Budgeting
Quality Planning
Human Resource Planning
Communica-tions Planning
Risk Management Planning and Analysis
Plan Program Purchases and Acquisitions
Plan Program Contracting
MONITORING AND CONTROLLING PROCESS GROUP
EXECUTING PROCESS GROUP
CLOSING PROCESS GROUP


PLANNING PROCESS GROUP
Direct and Manage Program Execution
Acquire Program Team
Develop Program Team
Information Distribution
Perform Quality Assurance
Request Seller Responses
Select Sellers
MONITORING AND CONTROLLING PROCESS GROUP


PLANNING PROCESS GROUP
MONITORING AND CONTROLLING PROCESS GROUP
EXECUTING PROCESS GROUP
Close Program
Component Closure
Contract Closure

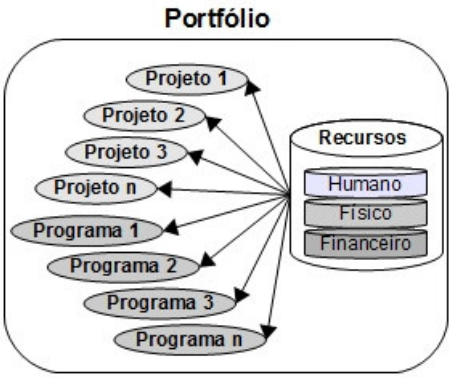
- **Referências**

Comment [PRCA261]: As Referências não possuem número de seção (ver template)

Comment [PRCA262]: Dispor as referências como no template

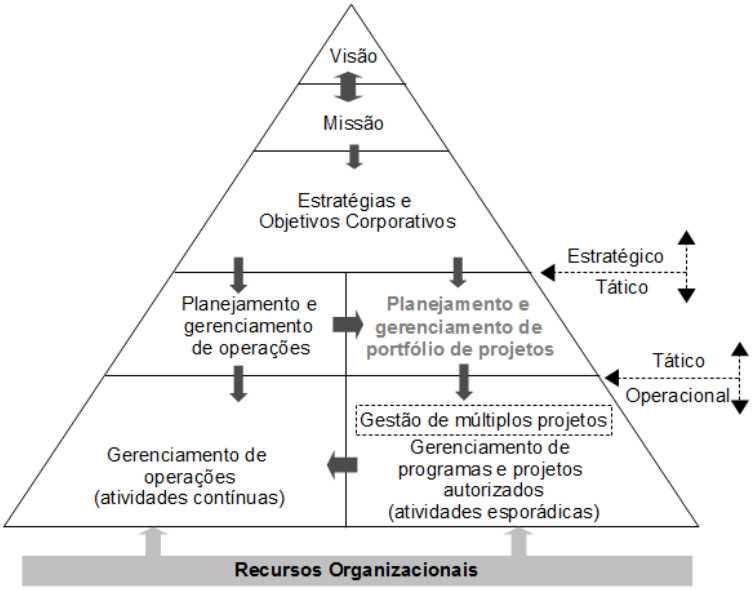
Sumário

15.13 REFERÊNCIAS BIBLIOGRÁFICAS.....120


Portfólio
Projeto 1
Projeto 2
Projeto 3
Projeto n
Programa 1
Programa 2
Programa 3
Programa n
Recursos
Humano
Físico
Financeiro

Na Figura 15-2, os itens Visão, Missão e Estratégias e Objetivos Corporativos

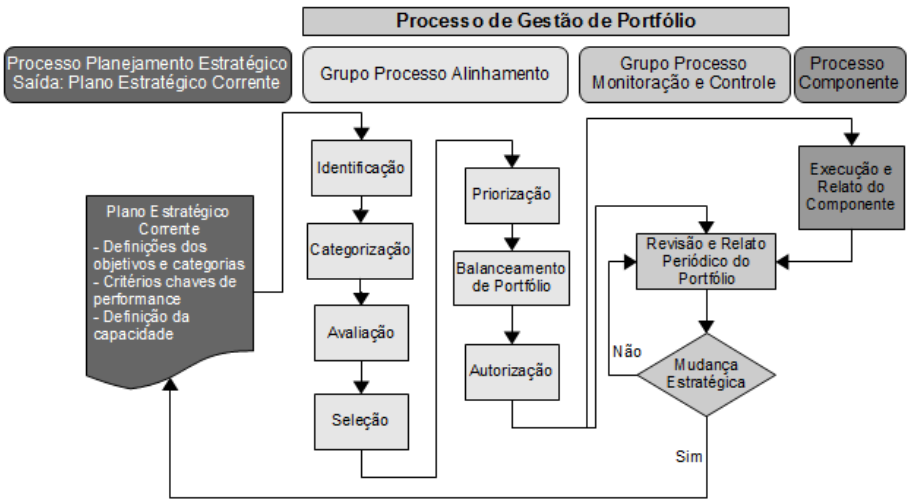
componentes do nível tático: Planejamento e gerenciamento de operações e Planejamento e gerenciamento de portfólio de projetos representam os processos que estabelecem as ações necessárias para realizar os objetivos da organização. Tais componentes interagem com os do nível operacional, que são: Gerenciamento de operações, Gestão de múltiplos projetos, abordada na próxima seção, e o Gerenciamento de programas e projetos autorizados, que são os componentes que tentam garantir à organização que suas operações e projetos sejam executados de forma eficiente [PMI 2006].



Sendo assim, a estratégia da organização impulsiona as definições das metas e objetivos estratégicos. Estes objetivos são passados para a gestão de portfólio que visa

objetivos da organização. Com base neste princípio, a gestão de portfólio irá selecionar, priorizar e aprovar projetos e programas para fazer parte do portfólio da organização. A gestão de portfólio também deve estar preocupada com o equilíbrio do portfólio em termos de retorno financeiro *versus* investimento e risco *versus* benefício, assim como, em negociar acordo entre as partes interessadas, por exemplo, entre a gerência executiva e os gestores de projetos [PMI 2006].

compreender, não só informações de alto nível sobre o andamento dos projetos, mas também, os detalhes da gestão de projetos, de modo a determinar se esta gestão está sendo eficiente ou não.

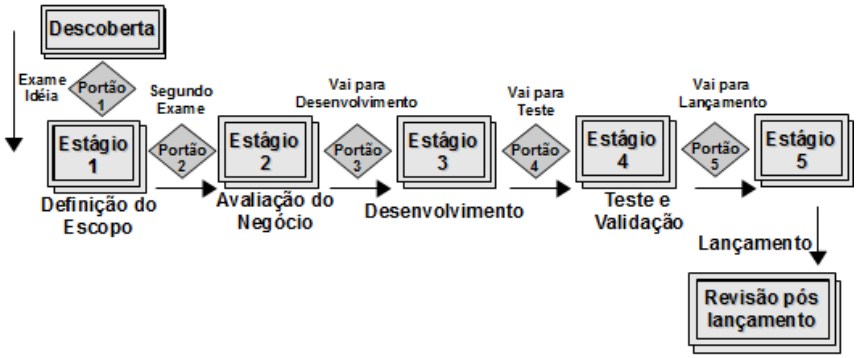

Processo de Gestão de Portfólio
Processo Planejamento Estratégico
Saída: Plano Estratégico Corrente
Grupo Processo Alinhamento
Grupo Processo Monitoração e Controle
Processo Componente
Plano Estratégico Corrente
- Definições dos objetivos e categorias
- Critérios chaves de performance
- Definição da capacidade
Identificação
Categorização
Avaliação
Seleção
Priorização
Balanceamento de Portfólio
Autorização
Execução e Relato do Componente
Revisão e Relato Periódico do Portfólio
Não
Mudança Estratégica
Sim

O objetivo deste processo é proporcionar a combinação de componentes com maior

benefícios da gestão do portfólio, que são a capacidade de planejar e alocar recursos **[ESPAÇO]** (financeiro, físicos e humanos) alinhado com as orientações estratégicas e a capacidade de maximizar o retorno d**O** portfólio dentro do que foi predefinido em termos de risco aceitável pela organização.

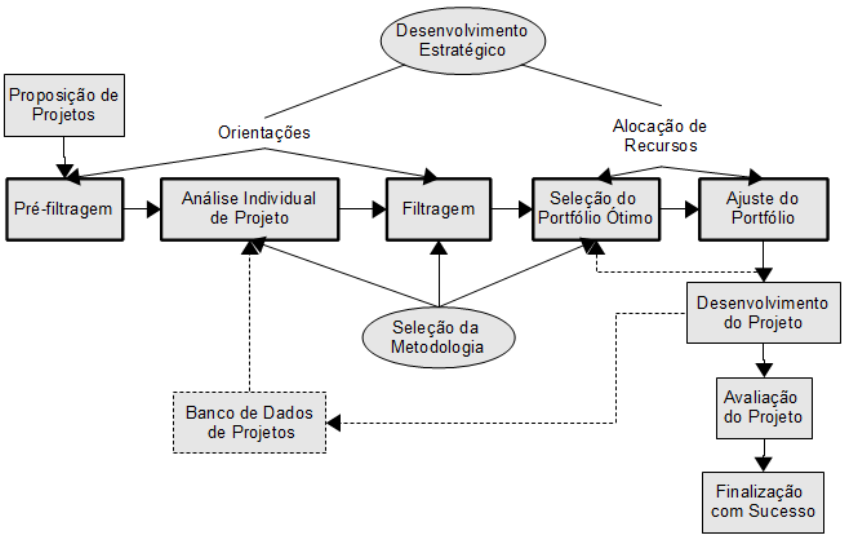
O objetivo deste processo é coletar indicadores de desempenho, apresentar relatórios

estratégias de negócios, além de garantir a utilização eficaz dos recursos. O ciclo de revisão examina todos os componentes durante um período determinado pela organização. Cada ciclo pode conter várias análises com um foco e profundidade diferente. Os indicadores chave de desempenho também podem variar a cada ciclo executado.



Este estágio tem como objetivo estabelecer os méritos técnicos e de mercado do projeto.

interessadas, e testes com usuários potenciais. O foco é detalhar informações quanto ao tamanho e ao potencial do mercado, além de sua possível aceitação. Paralelamente, uma avaliação técnica preliminar é realizada como o objetivo de verificar a viabilidade de desenvolvimento e de produção, além dos possíveis custos e tempo de execução do projeto.


Desenvolvimento Estratégico
Proposição de Projetos
Orientações
Alocação de Recursos
Pré-filtragem
Análise Individual de Projeto
Filtragem
Seleção do Portfólio Ótimo
Ajuste do Portfólio
Desenvolvimento do Projeto
Seleção da Metodologia
Banco de Dados de Projetos
Avaliação do Projeto
Finalização com Sucesso

A Empresa investe no desenvolvimento de soluções tecnológicas em Software

esferas federal, estadual, municipal, iniciativas do segmento acadêmico e sociedade [SERPRO 2009].

Assim como realizado na avaliação de complexidade, foram definidos intervalos de pontos para cada nível de importância estratégica, podendo ser baixo, médio, alto ou altíssimo, de acordo com o intervalo de pontos onde a nota estratégica do projeto esteja.

importância estratégica, será possível a definição de critérios para seleção e priorização dos projetos avaliados.

[SERPRO 2009] http://www.serpro.gov.br

Este capítulo descreve uma estrutura que surgiu a partir do reconhecimento da importância de uma gestão de projetos mais eficiente nas organizações. Tal importância é atribuída principalmente ao efeito positivo que as atividades que envolvem gestão de projetos têm dado sobre a lucratividade nos negócios. Diante disto, estas atividades acabaram demandando uma maior ênfase no profissionalismo o que culminou na necessidade de criação de departamentos especializados. Tais departamentos foram denominados Escritório de Projetos (PO – *Project Office*) ou Escritório de Gestão de Projetos (PMO – *Project Management Office*), e serão apresentados neste capítulo. Aqui serão descritos seus principais papéis, funções, objetivos, classificações, boas práticas de sua implantação e um estudo de caso demonstrando sua relevância, no que diz respeito a gestão de projetos, nas organizações contemporâneas.

Comment [Carlos Al263]: organizacional

Comment [Carlos Al264]: corporações

Comment [Carlos Al265]: causado

Comment [Carlos Al266]: disso

O Escritório de Projetos ou *Project Management Office* (PMO) tem sido uma das estruturas de Gerenciamento de Projetos mais exploradas atualmente pelas organizações. Isto tem se dado principalmente pela necessidade destas organizações possuírem uma entidade de apoio interna que preste suporte as atividades de implementação de princípios, práticas, metodologias, ferramentas e técnicas em Gerenciamento de Projetos [Marrom 2009].

Comment [Carlos Al267]: isso

Brian Hobbs e Monique Aubry [Hobbs e Aubry 2007] realizaram uma pesquisa pela internet em três etapas entre 2007 e 2008, envolvendo 500 organizações do Canadá, EUA e Europa. Tal pesquisa tinha como principal objetivo melhor entender o PMO e o valor agregado em sua implantação nas estruturas organizacionais. Dentre os principais destaques apresentados durante pesquisa foram listados:

Comment [Carlos Al268]: Colocar a referência no final do parágrafo

Comment [Carlos Al269]: Vírgula

O mercado contemporâneo tem forçado as organizações a definir estratégias gerenciais cada vez mais arrojadas na busca por uma maior produtividade dos projetos, maior eficiência dos processos e uma consolidação efetiva de informações estratégicas a serem repassadas a alta administração. Entre outros papéis e funções atribuídas atualmente ao PMO, podemos dizer que prover suporte a estes problemas é apenas um deles.

O papel dos PMOs tem se mostrado diverso e variado em nossa atual realidade. Ajuste de padrões e de metodologias para as gerências de projeto tem sido um dos grandes focos desta estrutura. Atividades ligadas a gerência de recursos humanos e também a responsabilidade na execução dos projetos também tem sido fortemente atribuídas ao PMO [Ribero 2007]. Além disto, as funções de unificar as estratégias e disseminá-las na organização como um Gerenciamento de Projetos responsável por preencher a lacuna entre a visão da empresa e de seus projetos, tem também sido atribuído a este tipo de estrutura [Kerzner 2003].

Comment [Carlos Al270]: pode-se

Comment [Carlos Al271]: disso

Comment [Carlos Al272]: Ficou estranho

Diante do que vem sendo apresentado neste capítulo nos resta um questionamento: Enfim, qual seria o principal objetivo dos PMOs? Os objetivos básicos, segundo [Dinsmore 2003] são: “orientar e dar suporte aos gerentes de projetos permitindo à empresa desenvolver seus projetos da forma mais eficiente e eficaz possível”.

Comment [Carlos Al273]: ...fica o seguinte quest...

- Tornar possível a execução dos projetos de forma que os mesmos fiquem alinhados com os objetivos da alta direção. Devido a disposição do PMO, torna-se possível a concentração das informações e condução dos projetos, facilitando assim o alinhamento entre os objetivos dos projetos e os objetivos organizacionais.

Comment [Carlos Al274]: vírgula

Podemos observar então que os objetivos de um PMO ainda são os mais variados e ainda não estão padronizados nas organizações. Todos os objetivos apresentados nesta seção buscam na realidade trazer uma melhoria na eficiência do planejamento e na condução dos projetos. Neste sentido, uma seleção e análise sobre os projetos existentes de forma ágil, torna-se de vital importância as organizações. Entregar os projetos no prazo, dentro do orçamento e atendendo aos objetivos de negócio da organização são critérios de suma importância segundo [Dai 2003].

Comment [Carlos Al275]: observa-se

- Escritório de Projetos de Grupo de Clientes;
- E Escritório de Projetos Corporativo.

Comment [Carlos Al276]: Tirar o "e"

O primeiro é caracterizado por gerenciar um conjunto crucial de recursos, dentro de um determinado setor, para a utilização em projetos peculiares do setor na organização. Os Escritórios de Projetos de Grupo de Clientes por sua vez, tem como objetivo principal progredir no diálogo com os clientes. Projetos de propósitos iguais e clientes são reunidos e acabam funcionando como uma organização provisória dentro da própria organização. Desta forma, segundo esta estrutura, pode existir vários escritórios de projeto de grupos de clientes sendo executados ao mesmo tempo na organização. Por fim, os Escritórios de Projetos Corporativos apresentam como principal característica acolher toda a organização concentrando-se nas questões estratégicas e corporativas da mesma. Neste sentido, este tipo de escritório tenta fazer com que todas as questões proeminentes para o sucesso dos projetos sejam discutidas e analisadas em seu âmbito, para que seja possível prestar uma maior assistência aos gerentes sobre suas decisões.

Comment [Carlos Al277]: Nesse

Na Equipe de Projeto Autônoma (APT – *Autonomus Project Team*), o

Comment [Carlos Al278]: Sigla já explicada

resultados positivos ou negativos. Este desenho ocorre caracteristicamente em circunstâncias em que o projeto não tem um relacionamento estreito com as outras partes da organização e a empresa não tem muita experiência em Gerenciamento de Projetos.

Vargas [Vargas 2003] afirma que este tipo de escritório de projetos destina-se ao Gerenciamento de um projeto ou programa muito específico. Neste cenário, as práticas de gerenciamento de projetos são provenientes da experiência do líder do projeto, não havendo um apoio nem mesmo cultura pela organização. A estrutura da APT tem a responsabilidade de gerenciar todas as áreas dos projetos, de forma centralizada.

Comment [Carlos Al279]: Letra minúscula

Segundo [Dinsmore 2003], existem, pontos cruciais para o sucesso de uma Equipe de Projeto Autônoma:

Comment [Carlos Al280]: Tirar a vírgula

Figura 1 - APT: Equipe Autônoma de Projeto – adaptado por Gerhard (Gehard 2004) de Dinsmore (Dinsmore 2003) e Vargas (Vargas 2003).

Comment [Carlos Al281]: A resolução da figura não está legível.

Comment [Carlos Al282]: Verificar a numeração da figura

No formato Escritório de Suporte de Projetos ou *Project Support Office*, é apresentado um apoio técnico ao projeto. Segundo este tipo de estrutura, [Vargas 2003] enfatiza que um PSO, destina-se ao suporte de diversos projetos concomitantemente, dando suporte especializado através do uso de ferramentas e recursos. De acordo com Dinsmore [Dinsmore 2003] e Vargas [Vargas 2003], um PSO em sua essência pode apresentar as seguintes opções de serviços:

Comment [Carlos Al283]: Substituir pela sigla PSO

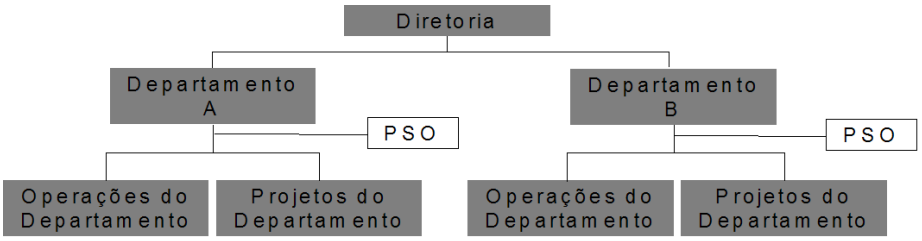


Figura 2 – PSO por departamento segundo Vargas [Vargas 2003].

Comment [Carlos Al284]: Figura ilegível

No modelo seguinte é apresentado o escritório de *Enterprise Project Support Office*. Segundo o autor neste cenário o escritório de projetos assume uma posição de planejador estratégico dos projetos da organização estando diretamente ligado ao alto escalão da organização.

Comment [Carlos Al285]: Colocar em português seguido da sigla em inglês

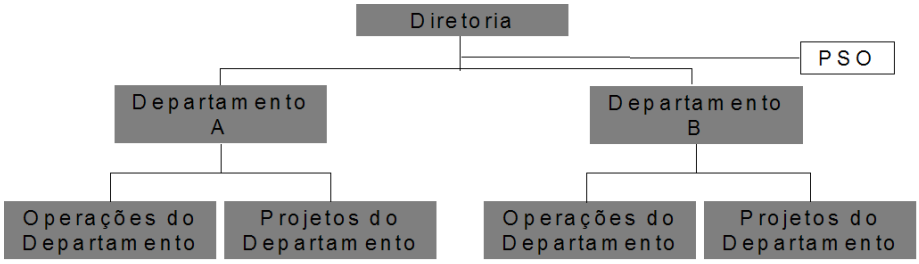


Figura 3 – PSO Corporativo segundo Vargas [Vargas 2003].

Comment [Carlos Al286]: Figura ilegível

O Centro de Excelência em Gestão de Projetos ou *Project Management Center Of Excellence*), apresenta como principal característica a fundamental referência em termos de conhecimento em gestão de projetos para a organização. Nada mais justo e necessário a um centro de excelência em Gestão de Projetos. Todavia não são de sua responsabilidade os resultados apresentados pelos projetos. O PMCOE é a estrutura responsável por disseminar a cultura de Gerenciamento de Projetos e manter as metodologias aplicáveis ao âmbito organizacional.

Comment [Carlos Al287]: Substituir pela sigla

Comment [Carlos Al288]: Tirar o ")"

O Escritório de Gerência de Programas ou *Program Management Office* por sua vez apresenta como característica a gerencia dos Gerentes de Projetos. Sendo assim, a responsabilidade sobre os resultados dos projetos recai sobre sua competência. Outro nome comumente atribuído a este tipo de estrutura é o de gerência de portfólio de projetos. O PrgMO tenta unir as funções do PMCOE e do PSO. Neste caso, projetos geridos isoladamente por uma determinada divisão, têm apoio do PrgMO sempre que preciso. Para que ele funcione com perfeição é preciso que tenha: poder, precedência corporativa e autoridade em âmbito empresarial.

Comment [Carlos Al289]: Substituir pela sigla

Figura 3 – PrgMO Corporativo segundo Vargas (2003).

Comment [Carlos Al290]: Figura ilegível

Para Block é primordial se identificar o patrocinador antes de tudo. Este é peça chave

funções e montagem do plano de comunicação a ser utilizado pelo PMO. Em seguida pode-se preparar o orçamento e tentar levantar os fundos necessários, inclusive a parte que cabe ao patrocinador, iniciando então logo em seguida o projeto piloto para avaliá-lo durante operação.

De acordo com Vargas [Vargas 2003], o primeiro passo é escolher o tipo de PMO que deve ser implantado na organização. Só ai então se pode levantar um patrocinador que lhe suporte. O próximo passo é criar uma estrutura com todas as possíveis dependências para que o PMO funcione. Motivar e promover o repasse de conhecimento entre os envolvidos no projeto é o passo posterior acompanhado de uma melhoria contínua em toda infra-estrutura. Após todos estes procedimentos pode-se de fato fundar um projeto piloto colocando-o em operação sempre o monitorando e observando quais os possíveis pontos de melhoria contínua.

Comment [Carlos Al291]: Frase estranha, reformular.

- **Repositório:** o PMO deve estruturar e gerenciar os documentos do repositório, para que seja possível reutilizar processos, procedimentos, *templates*, etc. [20].

Comment [Carlos Al292]: Tirar espaço

– Garantir que todo o trabalho realizado seja autorizado e tenha respaldo por documentos contratuais.

Comment [Carlos Al293]: Tirar espaço

O mercado de finanças públicas é ramo de atuação do SERPRO, o qual é composto atualmente pelo Ministério da Fazenda com suas secretarias e demais órgãos. Outro segmento igualmente importante é o de ações estruturadoras e integradoras da Administração Pública Federal cuja gestão e articulação compete ao Ministério do Planejamento, Orçamento e Gestão [Porto e Melo 2008]. A empresa atualmente destaca-se pelos seus pouco mais de 40 anos de atuação no mercado e por ser uma organização de destaque do governo federal para o segmento de tecnologia da

informação possuindo algo em torno de 10 mil funcionários e receita anual da ordem de R$1,5 bilhões-ano [Magalhães e Gusmão 2009].

Entre os principais clientes do SERPRO podemos citar: Presidência da República, Advocacia-Geral da União, Casa Civil da Presidência da República, Controladoria Geral da União, Ministério da Fazenda, Ministério da Educação, Ministério do Planejamento, Orçamento e Gestão, Ministério do Desenvolvimento, Indústria e Comércio Exterior, Ministério do Trabalho e Emprego, Ministério da Integração Nacional, Ministério das Minas e Energia, Ministério da Defesa, Ministério dos Transportes, Ministério da Saúde, Ministério das Cidades, Ministério da Justiça, Ministério do Desenvolvimento Agrário, Ministério dos Esportes.

Comment [Carlos Al294]: estão:

[Magalhães e Gusmão 2009] acrescentam ainda que a empresa em 2003, através de decisões de sua própria diretoria, saiu do Orçamento Geral da União (OGU), visando maior independência frente à questão comercial e financeira. O papel do OGU até então

dependeria exclusivamente de sua eficiência com a possibilidade de geração de receita própria.

suporte dos Gestores de Projetos e controle do desempenho dos projetos. Também foi criado o papel de Gestor de Projeto com a função de gerenciar e conduzir um projeto, o que inclui, liderança, planejamento e gestão do desenvolvimento de todos os resultados entregáveis do projeto. Com o objetivo de identificar as ações necessárias, foi considerado o contexto do gerenciamento de projetos adotados em nível corporativo, onde várias questões foram observadas de forma integrada e o encadeamento das ações foi uma premissa para garantir o sucesso da iniciativa.

Diante disto, decidiu-se decompor o processo em fases administráveis, baseadas

Comment [Carlos Al295]: Desativar a

mais adequadas ao SERPRO, com entrega de produtos bem definidos e com o envolvimento de todas as unidades desde o início, em trabalhos conduzidos por equipes

1.6.3.2 Fases

As fases foram organizadas considerando, também, os três pilares fundamentais em qualquer estratégia: processos, pessoas e tecnologia. Neste sentido, a primeira fase estava focada no amadurecimento do conhecimento e criação dos fundamentos básicos de Gerenciamento de Projetos na Empresa, o que se traduziu na elaboração de uma Política Corporativa, de um Processo de Gerenciamento de Projetos e de um Programa Corporativo de Implantação do gerenciamento de projetos no Serpro. Realização de benchmarking, disseminação (palestras, aquisição de livros especializados no assunto pelo CDI e criação da comunidade de Gestão de Projetos no SERPRO.) e conhecimento do uso atual de práticas de gerenciamento de projetos nas unidades do SERPRO ocorreram nesta fase inicial, em paralelo à capacitação em Gerência de Projetos, baseada nas práticas do PMI – *Project Management Institute*, com o objetivo de nivelar o grau de conhecimento e conceitos dos representantes das Unidades.

Comment [Carlos AI296]: Desativar a hifenização

A segunda fase constituiu-se da instituição da política e do processo corporativo de gerenciamento de projetos e focava em ações de conscientização e realização de projetos pilotos, inclusive para criação do Portfólio de Projetos. Em paralelo, ações de planejamento, de contratação de treinamentos e de especificação de ferramenta para apoio ao gerenciamento de projetos foram planejadas. Foi uma fase de curta duração, cujos resultados foram úteis para capacitação e iniciativas de implantação de gerenciamento de projetos nas Unidades na fase posterior. Foram previstas ações para divulgação maciça, instituição do Comitê Gestor, com representantes das Unidades e do

Comment [Carlos AI297]: Desativar a hifenização

Escritórios de Suporte e Controle de Projetos, aplicação do processo corporativo em pequenas iniciativas das Unidades envolvendo os profissionais que foram treinados e criação de um guia para orientar o planejamento de implantação pelas Unidades na próxima fase.

Na terceira fase, o foco estava na implantação efetiva do processo nas unidades, sob coordenação do Escritório de Suporte e Controle de Projetos da Unidade e acompanhamento pelo Escritório Estratégico de Projetos. Um Plano de ações foi elaborado por cada Superintendência, com base no guia elaborado na segunda fase, prevendo treinamentos seguidos de aplicação em projetos reais não críticos e de curto prazo, servindo como marco efetivo da implantação da cultura na Unidade. Foi uma fase com maior tempo de duração, pois previa também o desenvolvimento ou a aquisição de ferramenta para apoio ao gerenciamento de projetos, que dependia de processo de licitação, promoção de oficinas de trabalho para aplicação de técnicas adotadas no processo corporativo e incentivo a certificação PMP – *Project Management Professional.*

Comment [Carlos AI298]: Desativar a hifenização

Na quarta fase, o Escritório de Suporte e Controle de Projetos das Unidades, assessorado pelo Escritório Estratégico de projetos, deveria dar ênfase à extensão da implantação do processo corporativo a projetos de toda natureza realizados pela unidade, ao uso das lições aprendidas e à certificação PMP de seus profissionais. Todos

Comment [Carlos AI299]: Desativar a hifenização

os resultados deveriam ser objeto de avaliações do processo corporativo em busca de melhoria e da viabilidade de integração com as demais ferramentas existentes no Serpro.

Na quinta fase, o foco era na transição para a melhoria contínua, onde as ações deveriam ser direcionadas para a melhoria dos processos, atualização de software e meios de comunicação, reciclagem da equipe e avaliação da maturidade da Empresa em

Comment [Carlos Al300]: Desativar a hifenização

• Melhoria na qualidade da gestão dos projetos conduzidos pela organização, com impacto positivo percebido, inclusive, pelo cliente, que começa a requisitar próativamente os artefatos e atividades previstas no Processo de Gerenciamento de P

Comment [Carlos Al301]: Desativar a hifenização

Comment [Carlos Al302]: Tirar o enter

1.6.5 Melhoria Contínua

Durante o primeiro semestre de 2008, enquanto encerrava-se a 4° fase, realizou-se a transição do projeto de implantação para o momento de melhoria contínua. A definição da estratégia de evolução deu-se essencialmente através de discussões entre Escritório Estratégico de Projetos e a alta liderança da empresa. Como produto das discussões foi criado o Sistema de Gerenciamento de Projetos do Serpro (SGPS) compreendendo todo o conjunto de componentes responsáveis pela execução e evolução da solução de gerenciamento de projetos da empresa, como, por exemplo, estruturas funcionais, cursos, processos, ferramentas, etc. Também foram criados o Processo de Estruturação do SGPS, que agrega todas as ações de evolução do SGPS, e o Grupo de Trabalho de Estruturação do SGPS (GTSGPS), grupo inter-área de funcionários que assumiu a responsabilidade por conduzir as atividades do processo. Tanto o processo quanto o GT-SGPS são coordenados pelo Escritório Estratégico de Projetos.

Comment [Carlos Al303]: Desativar a hifenização

O Processo define um canal para controlar o ciclo de vida das demandas de melhorias através do sistema corporativo para gestão de demandas da empresa (Sistema

Comment [Carlos Al304]: Desativar a hifenização

Continuamente a coordenação do grupo monitora mudanças no Balde, na situação do projeto e subprojetos e na estratégia da instituição, de forma a manter o alinhamento com o cenário empresarial. A título de ilustração, descrevemos as principais demandas selecionadas para implantação em 2008:

Comment [Carlos Al305]: descreve-se

Durante a escrita deste capítulo foram identificadas algumas linhas de pesquisa desenvolvidas pela comunidade de Gerência de Projetos no que diz respeito a escritório de projetos. Entre os principais tópicos apresentados podemos citar:

Comment [Carlos Al306]: estão:

– **Analisando a Figura abaixo, a qual representa um organograma organizacional de uma empresa fictícia, que tipo Escritório de Projetos melhor se enquadraria segundo a caixa destacada em cinza?**

Comment [Carlos Al307]: Figura ilegível

– **Analisando a Figura abaixo, a qual representa um outro organograma organizacional de uma segunda empresa fictícia, que tipo Escritório de Projetos melhor se enquadraria segundo a caixa destacada em cinza?**

Comment [Carlos Al308]: Figura ilegível

Departamento de Administração da Faculdade deEconomia, Administração e Contabilidade da Universidade de São Paulo, São Paulo, São Paulo, Brasil, 2002.

20.1. INTRODUÇÃO A MATURIDADE EM GESTÃO DE PROJETOS 102

20.2. MODELOS DE MATURIDADE EM GESTÃO DE PROJETOS 103

20.2.1. ORGANIZATIONAL PROJECT MANAGEMENT MATURITY MODEL - PMI 103
20.2.2. PROJECT MANAGEMENT MATURITY MODEL – PM SOLUTIONS 104
20.2.3. MODELO DE MATURIDADE EM GERENCIAMENTO DE PROJETOS – DARCI PRADO 106
20.2.4. PORTFOLIO, PROGRAMME AND PROJECT MANAGEMENT MATURITY MODEL – OGC 107
20.2.5. KERZNER PROJECT MANAGEMENT MATURITY MODEL – HAROLD KERZNER 109

20.3. OPM3 109

20.3.1. ESTRUTURA DO MODELO 109
20.3.2. AVALIAÇÃO DA MATURIDADE 111
20.3.3. IMPLANTAÇÃO DO MODELO 112

20.4 MMGP 114

20.4.1. ESTRUTURA DO MODELO 114
20.4.2. AVALIAÇÃO DA MATURIDADE 115
20.4.3. IMPLANTAÇÃO DO MODELO 116

20.5. KPMMM 117

20.5.1. ESTRUTURA DO MODELO 117
20.5.2. AVALIAÇÃO DA MATURIDADE 118
20.5.3. IMPLANTAÇÃO DO MODELO 124

20.6. UM ESTUDO DE CASO 125

20.6.1. METODOLOGIA 125
20.6.2. RESULTADOS COLETADOS 126
20.6.3. PERFIL DOS PARTICIPANTES 126
20.6.4. SEGMENTAÇÃO POR NÍVEL DE MATURIDADE 128
20.6.5. SEGMENTAÇÃO POR PERCENTUAL DE ADERÊNCIA AOS NÍVEIS DE MATURIDADE 130
20.6.6. CONCLUSÃO 133

20.7. ANÁLISE COMPARATIVA 134

20.9. TÓPICOS DE PESQUISA 102

20.10. EXERCÍCIOS 102

REFERÊNCIAS 103

Além disso, no final do capítulo, será demonstrado um estudo de caso da avaliação da maturidade de empresas juniores no Brasil utilizando o MMGP, uma análise comparativa entre os principais modelos de maturidade apresentados, algumas sugestões de leitura para quem desejar se aprofundar em algum dos temas abordados, tópicos de pesquisa referentes ao tema e por fim exercícios para fixação do conteúdo trabalhado.

Antes de começar a abordar o tema da maturidade em gerenciamento de projetos é necessário definir alguns termos que serão frequentemente abordados durante todo este trabalho. Para isso é importante que o significado dos mesmos seja conhecido objetivando garantir o pleno entendimento de todo o contexto.

A primeira e talvez a mais importante definição a ser citada seja a da palavra projeto que de acordo com o PMBOK [PMI 2005] é um empreendimento único, com início e fim definidos, que utiliza recursos limitados e é conduzido por pessoas, visando

Comment [C309]: Disto,

Comment [C310]: ao final do capítulo

Comment [C311]: torna-se

Comment [C312]: capítulo

Comment [C313]: achei desnecessário este trecho

Comment [C314]:

Comment [C315R314]:

Comment [C316R315]: .

atingir metas e objetivos pré-definidos estabelecidos dentro de parâmetros de prazo, custo e qualidade.

Além da definição de projeto também é necessário entender o que é gerenciamento de projetos. Segundo o PMBOK [PMI 2005] gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas nas atividades do projeto a fim de atender os requisitos do projeto.

Maturidade pode ser definida como o desenvolvimento de sistemas e processos que são, por natureza, repetitivos e garantem uma alta probabilidade de que cada um deles seja um sucesso[Kerzner 2002]. Contudo, processos e sistemas repetitivos não são por si só, garantia de sucesso, apenas aumentam sua probabilidade. Assim, podemos observar que a maturidade em gestão de projetos está ligada a quão hábil uma organização está no exercício de gerenciar seus projetos [Prado 2004].

Atualmente existe uma necessidade de padronização dos processos de uma organização com o objetivo de elevar o nível de eficácia em gerenciamento de projetos possibilitando competir no acirrado mercado globalizado. Para isso, estão disponíveis cerca de 27 modelos de maturidade [Oliveira 2006] que indicam caminhos pelos quais a implementação de padrões pode tornar uma organização mais produtiva e competitiva.

O estudo da maturidade em gerenciamento, assim como o estudo do próprio gerenciamento de projetos, é assunto que vem sendo discutido recentemente, mas tem ocupado lugar de destaque [Leal 2008].

Muitos dos modelos de maturidade para gerenciamento de projetos apresentam a estrutura de cinco níveis proposta pelo CMM/CMMI, contudo algumas diferenças são observadas no conteúdo de cada nível. Dentre os modelos existentes podem ser destacados PMMM [Crawford 2002], MMGP [Prado 2004], P3M3 [OGC 2006], OPM3 [PMI 2003] e o KPMMM [Kerzner 2003], que serão apresentados a seguir.

Comment [C319]: Outra definição relevante…

Comment [0320]: Estes conceitos não são apresentados em capítulos anteriores?
Seria interessante fazer um link caso isto seja verdade?

Comment [C321]: Maturidade, por sua vez…

Comment [C322]: ,

Comment [C323]: remover

Comment [C324]: mas aumentam a probabilidade de sucesso.

Comment [C325]:

Comment [C326R325]: Além destes conceitos, essências para entendimento do restante deste capítulo, atualmente torna-se nítida a necessidade...

Comment [C327]: Colocar por extensor entre parenteses

Comment [C328]: nas organizações

Comment [C329]: em particular…

Esse modelo foi concebido em 2003 e lançado no início de 2004 pelo PMI – *Project Management Institute* contando com a participação de aproximadamente 800 voluntários de mais de 35 países, inclusive do Brasil, na sua elaboração.

O modelo inclui a recomendação de 603 boas práticas, ou seja, as experiências vividas ao longo dos anos por diversas empresas e profissionais da área de gerência de projetos. Desse modo o modelo retrata uma trilha aparentemente segura e referenciada, capaz de orientar os gestores organizacionais nos seus investimentos e iniciativas de aprimoramento da operação de gestão de projetos.

Uma das questões bastante discutidas na literatura é a relação muitas vezes inexistente entre o planejamento estratégico e o planejamento de Sistemas de Informação, principalmente no que diz respeito à estratégia organizacional e os projetos da organização.

O OPM3 como outros modelos de maturidade organizacional procura reconhecer e sinalizar as fases de amadurecimento progressivo da organização a cada degrau alcançado. Ao aplicar esse modelo a empresa pode obter benefícios como:

O modelo de maturidade PMMM (*Project Management Maturity Model*) foi concebido, em 2002, pelo Center for Business Practices, área que cuida das pesquisas da PM Solutions, organização voltada para consultoria e treinamentos em gerenciamento de projetos, leva em consideração as 9 áreas de conhecimento da gestão de projetos descritas em [PMI 2005], que são: integração, escopo, tempo, custo, qualidade, recursos humanos, comunicações, riscos e aquisições, além disso, a avaliação do grau de maturidade é baseada em cinco níveis cujas características [Prado 2004] serão detalhadas melhor abaixo:

- Nível 1 – Processos iniciais

Comment [C330]: remover itálico

Comment [C331]: colocar por extensor e entre parenteses

Comment [C332]: , muitas vezes inexistente,

Comment [C333]: , como outros modelos de maturidade organizacional,

Comment [C334]: Tb extenso e entre parenteses

Comment [C335]: melhor detalhadas

Comment [C336]:

Comment [C337R336]: colocar estas partes em negrito para destacar

- Análise informal da performance dos projetos;

Comment [C338]: itálico????

- São ativados processos para melhorar a eficiência da performance dos projetos;

Comment [C339]: itálico????

A Figura 20.1 demonstra a proposta da PM Solutions que foi criar um modelo de maturidade incorporando os 5 níveis de maturidade evolucionais existentes no *Capability Maturity Model* (CMM) desenvolvido pelo *Software Engineering Institute* (SEI) e as nove áreas de conhecimento do PMBOK.

20.2.3. Modelo de Maturidade em Gerenciamento de Projetos – Darci Prado

Comment [C340]: extenso

Comment [C341]: colocar numérico tb

Comment [C342]: Talvez fosse necessário escrever conforme os demais modelos... tem pouco conteúdo em relação aos demais não?

Ambos os modelos são baseados em cinco níveis de maturidade e possuem um sistema de avaliação similar onde se deve responder a um questionário contendo 40 perguntas de múltipla escolha e a partir das respostas obtidas é calculado o nível de maturidade seja ele setorial ou corporativo.

O modelo de maturidade P3M3 (*Portfólio, Programme and Project Management Maturity Model*) foi desenvolvido pela OGC, *Office of Government Commerce*, a Secretaria de Comércio Governamental do Reino Unido, mesma instituição que desenvolveu o PRINCE2. A versão inicial do P3M3 foi lançada em fevereiro de 2006, porém desde junho de 2008 está disponível a versão 2.0 do modelo que atualmente está em fase de consulta pública, sujeita a comentários e sugestões de usuários e especialistas.

O modelo P3M3 pode ser usado como base para a melhoria dos processos de gerenciamento de projetos, programas e portfólio. Ele está estruturado em 5 níveis de maturidade que são:

Comment [C343]: extenso

Comment [C344]:

Comment [C345R344]: .

Comment [C346]: remover

Comment [C347]: Diante disto, a partir …

Comment [C348]: Entre parenteses não?

Comment [C349]: extenso

Nível 5 – Processo Otimizado	

Comment [C350]: otimizado

20.2.5. Kerzner Project Management Maturity Model – Harold Kerzner

Comment [C351]: Talvez fosse necessário escrever conforme os demais modelos... tem pouco conteúdo em relação aos demais não?

Comment [C352]: Diminuir tamanho da figura para caber ainda na pagina anterior

- Domínios de abrangência, sobre os quais se desenvolvem as indicações e recomendações de amadurecimento organizacional. Os domínios mencionados no Modelo OPM3 referem-se a projetos, programas e portfólio;
- Estágios de amadurecimento dos processos organizacionais, associados às fases do modelo de melhoria contínua de processos. Os estágios de amadurecimento são descritos no Modelo OPM3 como Estágio de Padronização, de Mensuração, de Controle e de Melhoria Contínua.

Comment [C353]: :

Comment [C354]: :

Este questionário é composto por 151 perguntas abrangendo os domínios de Projetos, Programas e Portfólios. Inicialmente estas perguntas foram divididas de acordo com o domínio que avaliavam (70 perguntas para Projeto, 38 perguntas para Programas e 43 perguntas para Portfólio).

Comment [C355]: extenso

Comment [C356]:

Comment [C357R356]: extenso

Comment [C358]: extemso

Comment [C359]: extemso

- Ele se estrutura efetuando uma ligação entre o planejamento estratégico da organização e seus projetos. Assim os resultados dos projetos podem ser melhor avaliados em decorrência de estarem diretamente ligados ao sucesso da organização;

Comment [C360]:

Comment [C361R360]: estar?

Os cinco níveis de maturidade existentes no MMGP setorial são os seguintes: inicial, conhecido ou linguagem comum, padronizado, gerenciado e otimizado. O relacionamento existente entre as dimensões e os níveis de maturidade será mostrado na tabela a seguir [Prado 2004]

Comment [C362]: são

Comment [C363]: mostrados

Depois de associar o valor correspondente a cada uma das respostas dever-se-ia aplicar a fórmula a seguir para a partir da mesma extrair o valor referente ao nível de maturidade alcançado.

Comment [C364]: , a partir da mesma, extrair

- Interesses pessoais vem à frente de interesses corporativos;

Comment [C365]: vêm

A avaliação é realizada através de um questionário que contém 183 questões sendo dividas em 80, 20, 42, 25 e 16 questões, respectivamente para os níveis de 1 a 5.

Comment [C366]:

Comment [C367R366]: numeros extenso

O questionário referente ao nível de maturidade 1 contém 80 perguntas de múltipla escolha com cinco alternativas de resposta para cada questão, sendo que apenas uma delas é a resposta correta. O conteúdo destas questões são os princípios básicos do gerenciamento de projetos contidos no PMBOK [PMI 2005]. Apesar do PMBOK ser dividido em 9 áreas do conhecimento, para simplificar o questionário as áreas de conhecimento, gerenciamento de escopo e de integração foram integradas como uma única área. Sendo assim as o questionário foi dividido em 10 perguntas para cada uma das seguintes áreas de acordo com a seguinte tabela:

Comment [C368]: numero extenso

- Se sua organização obter menos de 60 pontos em todas as categorias, existe uma deficiência. Para pontuações inferiores a 30 em qualquer uma das categorias é necessário um treinamento rigoroso sobre os princípios básicos de gerenciamento de projetos. A organização aparenta estar altamente imatura em gerenciamento de projetos;

Comment [C369]: obtiver

Para fazer o levantamento da Avaliação da Maturidade nas Empresas Juniores brasileiras foi realizada uma pesquisa referente aos modelos de maturidade mais utilizados hoje em dia para identificarmos qual se adaptaria melhor ao estudo proposto e foi identificado que o modelo mais adequado para esta pesquisa seria o Modelo de Maturidade em Gerenciamento de Projetos – MMGP proposto por Darci Prado [Prado 2004].

Comment [C370]: ,

O questionário continha uma área inicial para cadastro dos respondentes e para futura segmentação e análise dos dados de acordo com as necessidades composto por oito questões referentes à identificação da empresa júnior (nome, área de atuação, cidade, estado e instituição de ensino superior a qual está vinculada) e também do respondente (cargo ocupado, nome e e-mail para contato).

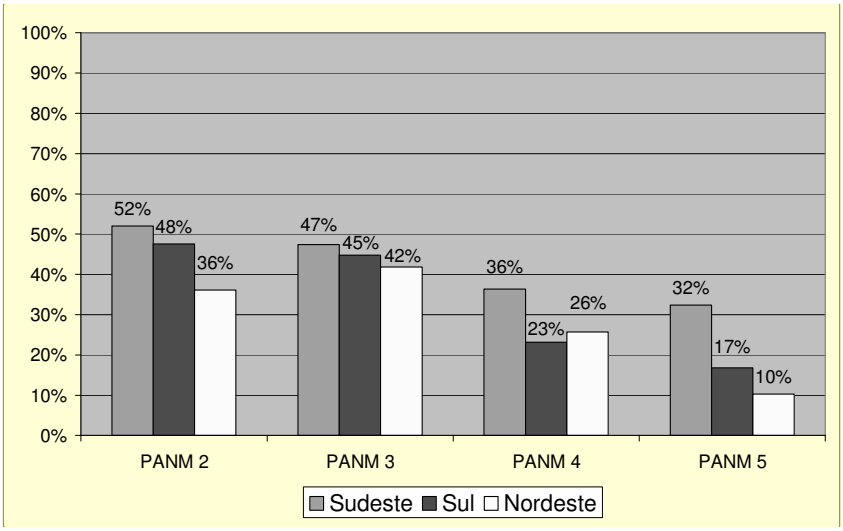
Comment [C371]: Remover espaço adicional

A maturidade média das EJ's brasileiras que responderam a pesquisa é de 2,41.

o Modelo Prado-MMGP setorial e o Modelo de Categorias de Archibald, quando foi obtido o índice médio de 2,42. Desta pesquisa participaram 258 empresas de 4 tipos de organizações, iniciativa privada, governo (administração direta), governo (administração indireta) e terceiro setor. Essa comparação entre as duas pesquisas confia credibilidade a ambas visto que a maturidade de um país tem alguma estabilidade no tempo. O valor médio 2,41 possui a distribuição descrita na figura abaixo:

Detalhando-se um pouco mais os dados percebemos que o estado que possui o maior nível de Maturidade é Minas Gerais, seguido pelo Rio de Janeiro, ambos representantes da região sudeste, o Paraná vem em seguida representando a região Sul e por fim temos Bahia, Pernambuco e Paraíba representando a região Nordeste. Os estados de Santa Catarina e Sergipe não estão presentes no gráfico acima, pois foram consideradas apenas unidades federativas com pelo menos três EJ's respondentes enquanto que estes possuíram apenas uma EJ participante.

Comment [C372]: , com pelo menos três EJ's respondentes,



Comment [C373]:

Comment [C374R373]: Reduzir tamanho do gráfico para colocá-lo junto ao seu texto

Podemos obter dados importantes através da comparação com a pesquisa de Archibald e Prado [Prado, Archibald 2006] a qual indica que o nível de maturidade das Empresas Juniores brasileiras se assemelha bastante com os níveis das organizações privadas, do terceiro setor e do governo que possuem administração indireta. Isso gera uma confiabilidade maior as empresas juniores perante a sociedade, visto que isto equipara este setor aos outros.

Comment [C375]: Aos níveis

Tabela 20.9:Dados Gerais dos Modelos de Gestão de Projetos [Adaptado de Costa 2009].

Comment [C376]:

Comment [C377R376]: Tabela incompleta

20.9. Tópicos de Pesquisa

Comment [C378]:

Comment [C379R378]: Colocar outros tópicos

21. GOVERNANÇA EM TIC 107

21.1 GESTÃO EM TIC 107

21.1.1 RELEVÂNCIA E EVOLUÇÃO DO PAPEL DA TIC NAS ORGANIZAÇÕES 109
21.1.2 DA GESTÃO À GOVERNANÇA EM TIC 112

21.2 MODELOS DE GESTÃO EM TIC 115

21.2.1 COBIT 115
21.2.2 ITIL 115
21.2.3 BSC 116
21.2.4 IT FLEX 116
21.2.5 COSO 117
21.2.6 ISO/IEC 20000 118
21.2.7 VAL IT 118
21.2.8 CMMI SOB A PERSPECTIVA DE GOVERNANÇA DE TI 119

21.3 ITIL 119

21.3.2 HISTÓRICO 119
21.3.3 REGULAMENTAÇÃO DO ITIL 120
21.3.3.1 DIREITOS AUTORAIS 121
21.3.3.2 CERTIFICAÇÕES / TREINAMENTOS 121
21.3.3.3 PUBLICAÇÃO DE CONTEÚDOS OFICIAIS 121
21.3.3.4 FÓRUM DE FOMENTO (ITSMF) 122
21.3.4 ESTRUTURA DO ITIL 122
- *SERVICE STRATEGY* (ESTRATÉGIA DE SERVIÇOS) 122
- *SERVICE DESIGN* (PLANEJAMENTO DE SERVIÇOS) 122
- *SERVICE TRANSITION* (TRANSIÇÃO DE SERVIÇOS) 122
- *SERVICE OPERATION* (OPERAÇÃO DE SERVIÇOS) 122
- *CONTINUAL SERVICE IMPROVEMENT* (APRIMORAMENTO CONTÍNUO DE SERVIÇOS) . 123
21.3.5 O QUE NÃO É ITIL 124
21.3.6 FRONTEIRAS COM OUTROS MODELOS E LIMITAÇÕES 125
21.3.7 PONTO DE PARTIDA 126
21.3.8 COMENTÁRIOS SOBRE PRÁTICAS DE SUCESSO 127
21.3.9 PÚBLICO ALVO 128
21.3.10 UTILIZAÇÃO DO ITIL 129

21.4 COBIT 130

21.4.1 DEFINIÇÃO 130
21.4.2 HISTÓRICO 131
21.4.3 REGULAMENTAÇÃO DO COBIT 132
21.4.3.1 CERTIFICAÇÕES / TREINAMENTOS 132
21.4.3.2 DIREITOS AUTORAIS 132
21.4.3.3 PUBLICAÇÃO DE CONTEÚDOS OFICIAIS 133
21.4.3.4 FÓRUM DE FOMENTO (ISACA) 133
21.4.4 ESTRUTURA DO COBIT 133
21.4.4.1 PRIMEIRA DIMENSÃO DO CUBO – PROCESSOS DE TI 134
21.4.4.2 SEGUNDA DIMENSÃO DO CUBO – CRITÉRIOS DE INFORMAÇÃO 136
21.4.4.3 TERCEIRA DIMENSÃO DO CUBO – RECURSOS DE TI 137
21.4.5 NÃO É COBIT 138
21.4.6 FRONTEIRAS COM OUTROS MODELOS 139
21.4.7 PONTO DE PARTIDA 140
21.4.8 COMENTÁRIOS SOBRE PRÁTICAS DE SUCESSO 141
21.4.9 PÚBLICO ALVO 141
21.4.10 UTILIZAÇÃO DO COBIT 142

21.5 INICIATIVAS DE INTEGRAÇÃO DOS PRINCIPAIS MODELOS 143

21.6 IMPLANTAÇÃO DE MODELOS DE GESTÃO 144

21.7 TÓPICOS DE PESQUISA 146

21.8 SUGESTÕES DE LEITURA 147

21.9 EXERCÍCIOS 149

21.10 REFERÊNCIAS 151

Capítulo

Gestão em TIC

Na **era da Integração e da Reestruturação do Negócio**, iniciada em meados de

de TI. A integração tecnológica flexibilizou e simplificou o intercâmbio e o acesso às informações otimizando o funcionamento das organizações. A TI passou a ser reconhecida como o fator crítico de potencialização do negócio das organizações, principalmente através das telecomunicações, o que possibilitou a eliminação de barreiras físicas e temporais, nas atividades de serviços e colaboração. Segundo Ken [KEN 1996], de modo súbito, estas mudanças se aceleraram em quase todas as áreas de negócio e da tecnologia. A convergência das tecnologias, as transformações e utilização das ferramentas de TI se tornaram globais e as distinções entre computador e comunicação desapareceram. Neste contexto o termo TI também se transformou, assumindo sua denominação mais recente "Tecnologias da Informação e Comunicação - TIC".

- A liderança do CIO[18]: tradicionalmente a figura do CIO tem sido a de se apresentar como o "paladino das causas do departamento de TIC", procurando defender os investimentos em infraestrutura de TIC, e

atuando no máximo em nível tático. É necessário, contudo, que esta figura se reposicione estrategicamente na organização, respondendo diretamente ao CEO[19] e apoiando-o no processo de decisão estratégica da organização. Para que isso aconteça, no entanto, é necessário que a TIC deixe de ser um centro de altos custos da organização e passe a atuar na camada estratégica do negócio, como setor de inovação e diferencial competitivo.

Para a devida abordagem do papel estratégico da TIC é necessário a existência de um processo estruturado para gerenciar e controlar as iniciativas de TIC nas organizações, garantindo o retorno de investimentos e adição de melhorias nos processos organizacionais. Neste contexto o termo Governança de TIC é utilizado como forma de obter controle e conhecimento em TIC, assegurando mais transparência na gestão estratégica. Neste ambiente surgiram e prosperam as propostas de metodologias

guias de referência, conjuntos de “boas práticas” e frameworks de que permitem a implantação da governança de TIC nas organizações, a racionalização dos investimentos em TIC e fornecem métricas para avaliação dos resultados destes. Dos quais podemos destacar: COBIT [ISACA 2009] e ITIL [ITGI 2009].

O termo Governança em TI é definido como uma estrutura de relações e processos que dirige e controla uma organização a fim de atingir seu objetivo de adicionar valor ao negócio através do gerenciamento balanceado do risco com o retorno do investimento de TI. Criar estruturas de governança significa definir uma dinâmica de papéis e interações entre membros da organização, de tal maneira a desenvolver a

controle e conhecimento em TI, é o modelo que assegura mais transparência na gestão estratégica [KOSHINO 2004].

The Stationary Office [TSO 2000] é o maior "publicador" de volumes no Reino Unido

documentos e serviços. O TSO é o "publicador" oficial da documentação do ITIL. Pode-se obter versões em PDF, para download, dos livros antigos do ITIL através do site deste grupo. A Figura 23.2 ilustra os cinco livros do ITIL.

Esse volume trata das boas práticas relacionadas à entrega e controle de serviços, focando a estabilidade do serviço. O livro basicamente mostra como administrar serviços já na etapa de produção, lidando com os problemas diários do serviço. Cobre

processo, administração de aplicação, infraestrutura e operações, fatores de sucesso e controle e funções de processos.

A relação entre tecnologia e processos é bastante complexa, e o ITIL é cuidadoso ao distinguir pontos de relação sem ficar intrinsecamente envolvido em problemas de tecnologia ou de arquitetura. Sua arquitetura já foi planejada para evitar confusões como a já relatada, onde os fornecedores de software que afirmam ser *ITIL-*

quatro aspectos básicos para o sucesso, que são: os Serviços, Processos, Organização e Tecnologia (SPOT) [DROGSETH 2004].

material. Nenhum outro direito ou permissão é concedida em relação ao material do COBIT.

Financeiros/Seguradoras, Governo, Transporte, Saúde. Dentre as empresas que prestam serviços de consultoria de TI que implantaram o COBIT, para melhoria dos seus processos, estão: a Datasec IT Security and Control, a Dongbu Hitek, a Jefferson Wells, The Manta Group, a Sun microsystems e a Unisys Corporation [ISACA 2009a].

- Para obter maiores informações sobre o Modelo Ágil no Apoio à Governança de TIC – MAnGva, deve-se ler a Dissertação de Mestrado: [LUNA 2009a], LUNA

Governança de Tecnologia da Informação e Comunicação inspirado no Paradigma das Metodologias Ágeis. Programa de Pós-graduação *stricto sensu* em Ciência da Computação. Centro de Informática, Universidade Federal de Pernambuco. Dissertação de Mestrado. Disponível em: <www.cin.ufpe.br/~ajhol/publicacoes>. Acesso em: 17/11/2009.

- Pode-se obter a versão completa do COBIT 4.1, em diversos idiomas através da

<http://www.isaca.org/Content/NavigationMenu/Members_and_Leaders/COBIT6/Obtain_COBIT/Obtain_COBIT.htm>. Acesso em: 13/01/2009.

ISACA. COBIT *Case Studies by Industry*. 2009a. Disponível em:<

Page 396: [1] Comment [A239] **Alinne** **10/24/2009 1:56:00 PM**

Acho que ficaria melhor assim:

A partir dessa indagação, pode-se entender medição como o ato de obter valores de uma característica ou atributo de uma entidade qualquer, como por exemplo, uma pessoa. A respeito da entidade pessoa existe um conjunto de características ou atributos, os quais são passíveis de receberem valores, tal como: altura, peso, entre outros. Para estes atributos pode-se atribuir os valores 85 para o peso e 180 para altura. No entanto, para tais valores costumam-se atribuir unidades de medida, as quais são responsáveis por oferecer uma noção de quantidade de um valor qualquer e que servem de base para comparações com outros valores com mesma unidade de medida. Portanto, para este exemplo o peso seria igual a 85 kg e altura igual a 180 cm.

Page 396: [2] Comment [A240] **Alinne** **10/24/2009 2:01:00 PM**

O caso abordado anteriormente, também acontece com o software, onde o software e o processo usado para o desenvolvimento podem ser vistos como entidades, existindo atributos pertinentes para ambos que podem ser medidos. Segundo Pressman [Pressman 2006], o que já foi abordado pode ser exemplificado quando um único ponto de dados foi coletado (por exemplo, o número de erros descoberto em um único componente de software), uma medida foi estabelecida, e que uma métrica de software é aquela que relaciona medidas individuais de algum modo (por exemplo, o número médio de erros encontrados por teste de unidade).