MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
SECRETARIA DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
(Real Academia de Artilharia, Fortificação e Desenho – 1792)

Kleyna Moore Almeida

GARANTIA DA QUALIDADE DE SOFTWARE NO DESENVOLVIMENTO BASEADO EM REUSO: UMA ABORDAGEM NO CONTEXTO DO MINISTÉRIO DA DEFESA E DE SEUS COMANDOS SUBORDINADOS

Rio de Janeiro
2004

INSTITUTO MILITAR DE ENGENHARIA

KLEYNA MOORE ALMEIDA

GARANTIA DA QUALIDADE DE SOFTWARE NO DESENVOLVIMENTO BASEADO EM REUSO: UMA ABORDAGEM NO CONTEXTO DO MINISTÉRIO DA DEFESA E DE SEUS COMANDOS SUBORDINADOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia de Software do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências da Computação.

Orientador: Ten Cel Ulf Bergmann – D.C.
Co-orientador: Cel José Antônio Moreira Xexéo – D.C.

Rio de Janeiro

2004

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

??? Almeida, Kleyna Moore

Garantia da qualidade de *software* no desenvolvimento baseado em reuso: uma abordagem no contexto do Ministério da Defesa e de seus comandos subordinados / Kleyna Moore Almeida – Rio de Janeiro: Instituto Militar de Engenharia, 2004.
??? p.: il., tab. 29,7 cm.

Dissertação (mestrado) – Instituto Militar de Engenharia, 2004.

1. Processo de Qualidade. 2. Técnica de Leitura. 3. Revisão. 4. Indicadores. I. Instituto Militar de Engenharia. II. Título.

???CDD 625.1

INSTITUTO MILITAR DE ENGENHARIA

KLEYNA MOORE ALMEIDA

GARANTIA DA QUALIDADE DE SOFTWARE NO DESENVOLVIMENTO BASEADO EM REUSO: UMA ABORDAGEM DO MINISTÉRIO DA DEFESA E DE SEUS COMANDOS SUBORDINADOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia de *Software* do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências da Computação.

Orientador: Ten Cel Ulf Bergmann - D. C.

Co-orientador: José Antônio Moreira Xexéo – D. C.

Aprovada em 06 de agosto de 2004 pela seguinte Banca Examinadora:

_____

Ulf Bergmann – D. C. do IME - Presidente

_____

José Antônio Moreira Xexéo – D. C. do IME

_____

Soeli Teresinha Fiorini – D. C. da e-Dablio Consultoria e Projetos Ltda

Rio de Janeiro

2004

Ao vovô, meu marido e filhos pelo apóio e incentivo de meu aprimoramento. Aos meus pais que sempre me orientaram e contribuíram para minha formação educacional e profissional.

AGRADECIMENTO

Agradeço a todas as pessoas que me incentivaram, apoiaram e possibilitaram esta oportunidade de ampliar meus horizontes, especialmente:

- Aos meus orientadores Ulf Bergmann e José Antônio Moreira Xexéo, por suas disponibilidades e atenções;
- A todos os professores, pelos conhecimentos transmitidos e aos funcionários do Departamento de Engenharia de Sistemas do IME pelo apoio e atenção;
- Às amigas Claudia Hazan e Miriam Sayão, e à minha irmã Kenya pelo apoio nos momentos de dúvidas;
- Aos amigos de turma da PUC-RJ e do IME pelo companheirismo e amizade;
- À bibliotecária Rosane da PUC-RJ pela atenção em todas às vezes que precisei de livros e artigos para estudo e desenvolvimento do meu trabalho;
- Ao Centro de Apóio a Sistemas Operativos e à Diretoria de Administração da Marinha pelo apoio e incentivo;
- Ao Instituto Militar de Engenharia (IME) pelo apoio e atenção.

“Quando se pode medir o que está falando e expressar isso em números passa-se a conhecer algo mais sobre isso; mas quando não se pode expressar em números, seu conhecimento é superficial e insatisfatório”.

LORD KELVIN

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ABREVIATURAS

ADC – Acoplamento Direto da Classe

CMC – Coesão entre Métodos na Classe

MAD – Medida de Acesso aos Dados

MAF – Medida de Abstração Funcional

MAg – Medida de Agregação

NHq – Número de Hierarquia

NM – Número de Métodos

NMA – Número Médio de Antecessores

NMP – Número de Métodos Polimorfos

TIC – Tamanho da Interface da Classe

TPC – Tamanho do Projeto em Classes

SÍMBOLOS

Σ – Símbolo matemático de somatório

LISTA DE SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

ACM – *Association for Computing Machinery*

APF – Análise por Pontos de Função

CGQS – Consultor de GQS

CMM – *Capability Maturity Model*

CMMI – *Capability Maturity Model Integration*

COCOMO – Constructive Cost Modeling

COTS – *Commercial Off The Shelf*

GGQS – Grupo da Garantia da Qualidade de *Software*

GQS – Garantia da Qualidade de *Software*

GQT – Gerência pela Qualidade Total

GQM – *Goal/Question/Metric*

ICP – Indicador de Compreensibilidade do Projeto

IEC – *International Electro Technical Commission*

IEEE – *Institute of Electrical and Eletronics Engineers*

IEfP – Indicador de Efetividade do Projeto

IEsP – Indicador de Extensibilidade do Projeto

IFPUG – *International Funcion Point Users Group*

IFuP – Indicador de Funcionalidade do Projeto

IFxP – Indicador de Flexibilidade do Projeto

IME – Instituto Militar de Engenharia

IPPVR – Indicador de Produtividade do Processo de Verificação e Revisão

IPqM – Instituto de Pesquisa da Marinha

IQPOOR – Indicadores de Qualidade do Projeto OO Baseado em Reúso

IQPVR – Indicador de Qualidade do Processo de Verificação e Revisão

IRP – Reusabilidade do Projeto

ISO – *International Organization for Standardization*

KPA – *Key Process Areas*

LOC – *Lines Of Code* (linhas de código)

MD – Ministério da Defesa

NIST – *National Institute of Standards and Technology*

OO – Orientado a Objeto

PAA – Processo de Aprovação de *Assets*

PCQ – Processo de Controle da Qualidade

PDCA – *Plan-Do-Check-Act* (Ciclo de Deming)

PF – Pontos de Função

PSM – *Practical Software Measurement*

PGQS – Processo da Garantia de Qualidade de *Software*

PGQSP – Plano de Garantia de Qualidade de *Software* do Projeto

QSMGQ – Questionário de Sugestões para Melhoria da Garantia da Qualidade

SADT – *Structure Analysis and Design Technique*

SEI – *Software Engineering Institute*

SEPIN – Secretaria da Política de Informática

SPICE – *Software Process Improvement and Capability dEtermination*

SPM – Subprocesso de Medição e Avaliação

SPR – Subprocesso de Revisão

SPV – Subprocesso de Verificação

TLH – Técnica de Leitura Horizontal

TLOO – Técnica de Leitura Orientada a Objeto

TLV – Técnica de Leitura Vertical

UML – *Unified Modeling Language*

RESUMO

A garantia da qualidade de *software* no desenvolvimento baseado em reuso é um dos principais fatores para assegurar e controlar a qualidade do processo de desenvolvimento de *software* e dos artefatos gerados, visando posterior reutilização.

Um Processo de Garantia de Qualidade de *Software* é definido por meio de um conjunto de atividades sistemáticas para assegurar a adequação ao reuso de artefatos, visando obter maior produtividade e melhoria da qualidade de um processo de desenvolvimento de *software* baseado em reuso de *asset*s (artefatos criados com o propósito explícito de serem posteriormente reutilizados). Este processo promove o desenvolvimento de projetos e de *asset*s de forma eficaz com base em aspectos qualitativos e quantitativos. Os indicadores quantitativos asseguram um melhor planejamento e controle de projetos e de *asset*s, do reuso no desenvolvimento de aplicações, utilizando métricas e modelos de *software*, indicadores de qualidade, produtividade e reusabilidade.

Este trabalho está baseado nas definições de padrões de processos mais utilizados atualmente pelas industrias e organizações, tais como, CMM, CMMI, ISO 9000, ISO 12207 e ISO 15504, além da Norma IEEE1517. Os processos estabelecidos neste trabalho abrangem diferentes níveis de maturidade do CMM, tais como, pontos de revisão e *checklists* nas diversas fases do processo de desenvolvimento de *software* (Nível 3 - Definido), e também a definição de um processo de medição (Nível 4 - Gerenciado). As verificações e revisões empregam técnicas de leitura baseada em perspectiva e técnicas de leitura horizontal e vertical respectivamente.

A implantação do processo desenvolvido neste trabalho possui quatro fases: implantação de um processo de garantia de qualidade no processo de desenvolvimento de *assets*; definição dos modelos de qualidade dos artefatos, dos objetivos da avaliação e de indicadores; aplicação de técnicas de qualidade; e utilização dos indicadores para promover a melhoria do processo.

Além disso, é apresentado um estudo de caso aplicando o processo de garantia de *software* estabelecido em um projeto piloto, Controlador Tático, desenvolvido com base no processo de Desenvolvimento de *Software* Baseado em Reuso no Contexto do Ministério da Defesa e de seus Comandos Subordinados de GURGEL (2004). Este estudo de caso aponta que a estratégia utilizada pode proporcionar a melhora da qualidade do modelo conceitual final assim como no processo de produção de *software* para reuso mais produtivo.

ABSTRACT

The software quality warranty in development based on reuse is currently a topic of a great interest, mainly due to the need of assuring and controlling the quality of the software development process and of the generated products, under the perspective of subsequent reuse.

In order to increase productivity and improve the quality of a software development process based on assets reuse (software products created with the explicit purpose of later reuse), a Software Quality Warranty Process is defined by a group of systematic activities designed to assure the adaptation needed to reuse software products. So, this process promotes assets and projects development in an effective way based on qualitative and quantitative aspects. The quantitative indicators consist on assuring a better planning and control of assets and projects, as well as the reuse on applications development, by applying metric, software models, quality indicators, productivity and reusability.

This work is based on the process standard definitions more frequently used presently by industries and organizations, such as, CMM, CMMI, ISO 9000, ISO 12207 and ISO 15504, and also the IEEE1517 Standard. The processes established in this work include different CMM maturity levels, such as, revision points and checklists in the several phases of the software development process (Level 3 - Defined), and also the measurement process definition (Level 4 - Managed). The verifications and revisions use reading techniques based on perspective, as well as horizontal and vertical reading techniques respectively.

The process implantation developed in this work is divided into four phases: implantation of quality warranty process on the assets development process; definition of the software products quality models; definition of the evaluation objectives and indicators; usage of quality techniques; and usage of the indicators to promote process improvement.

Besides this, a case study is presented applying the software warranty process defined on a pilot project, a Tactical Controller, developed according to a reuse based Software Development Process, under the Defense Ministry Context and of their Subordinate Commands of GURGEL (2004). This case study points out that the used strategy can increase quality of the final conceptual model as well as of the software production process, focusing on a more productive reuse.

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A era da informática teve início na década de 60 com a difusão dos computadores nos diferentes setores da sociedade através da utilização de *software* para os serviços e funções. O modelo de ciclo de vida de *software* surgiu para atender a necessidade do mercado no desenvolvimento de *software* de forma planejada, controlada e dentro da estimativa de prazos estabelecidos. Nos anos 80, a redução progressiva do preço do *hardware* incentivou a busca pelo menor custo do desenvolvimento de *software*. Nesta época, os modelos para estimar custos e recursos começaram a surgir, assim como a preocupação pela qualidade e produtividade no desenvolvimento de *software* (BASILI et al., 1991). A necessidade de garantir a qualidade no processo de desenvolvimento do *software* resultou na utilização de processos criteriosos de avaliação de *software* para identificar seus pontos fortes e oportunidades de melhoria (SEPIN, 2002).

A qualidade dos serviços e dos produtos de *software* está fortemente relacionada com a qualidade do processo de desenvolvimento do *software* (CROSBY, 1996; ABNT12207, 1998; HAZAN, 1999; ROCHA et al., 2001). Assim, o número de defeitos apresentados no *software* é diretamente proporcional à qualidade do processo usado para a construção do *software* (JACOBSON et al., 1997; HENNINGER, 1999; ROCHA et al., 2001). Isto provocou uma grande preocupação com a definição e melhoria de um processo de *software*, resultando no surgimento de vários modelos e normas para a definição de processos e garantia de qualidade.

Os modelos mais utilizados para definir processos de *software* são: o Modelo de Maturidade da Capacitação para *Software* (CMM) (PAULK et al., 1993) e a Norma ISO/IEC (*International Organization for Standardization/International Electro Technical Commission*) 12207 (ABNT12207, 1998). O CMM classifica as organizações quanto ao nível de maturidade, e a Norma ISO/IEC 12207 está baseada em *framework* para processos de ciclo de vida - processos fundamentais, processos de apoio e processos organizacionais (veja APÊNDICE 3). A Norma ISO/IEC 12207 e sua versão brasileira são conhecidas por 67% das empresas pesquisadas pela Secretaria da Política de Informática do Ministério da Ciência e Tecnologia (SEPIN, 2002), mas somente 12% das empresas utilizam-na.

Os produtos de *software*, comumente denominados de artefatos, são um conjunto de informações em diferentes níveis de abstração e um conjunto de transformações e decisões. A aplicação de um programa de controle de qualidade na produção de *software* garante maior qualidade no processo de desenvolvimento do projeto de *software* (HAZAN, 1999; ROCHA et al., 2001).

Existem diferentes definições de qualidade e a seguir são apresentadas algumas destas definições pesquisadas na literatura, a saber:

- Qualidade é a satisfação total do consumidor (DEMING, 1990). Nesta definição, a qualidade está ligada à satisfação do usuário interno ou externo, abrangendo suas necessidades e expectativas mensuráveis, que devem ser medidas através das características da qualidade dos produtos[1] ou serviços finais ou intermediários da organização. Estas características compreendem ausência de defeitos e presença de atributos que agregam valor ao produto ou serviço de forma confiável, acessível, segura e no tempo certo às necessidades do cliente. Essas características são: previsibilidade e confiabilidade de todas as operações, treinamento contínuo, objetividade e clareza da informação, gerência por processo e não por resultados, de forma a ter uma gerência preventiva de problemas, não permitindo que o mesmo problema se repita pela mesma causa;
- Qualidade é a adequação ao uso (JURAN, 1990). Diferentes usuários podem usar os mesmos produtos de diversas formas, ou seja, os produtos devem ter vários atributos de qualidade, tais como, capacidade, usabilidade, desempenho, confiabilidade, instabilidade, manutenibilidade, documentação e disponibilidade;
- Qualidade é a conformidade com os requisitos dos clientes (CROSBY, 1996). Os requisitos[2] devem ser claramente especificados para que, durante o processo de desenvolvimento do produto, as medições possam ser realizadas para determinar a conformidade aos requisitos especificados. A não-conformidade detectada é a ausência de qualidade, ou seja, o defeito encontrado no produto. Portanto, esta definição está diretamente relacionada com a falta de defeito no produto;
- Qualidade é o conjunto de características incorporadas ao produto através do projeto e manufatura que determinam o grau de satisfação do cliente (FEIGENBAUM, 1991);

[1] Como o produto e o processo estão fortemente relacionados e não podem ser separados quando se analisa a qualidade (DEMING, 1996; ABNT12207, 1998; ROCHA, 2001), então a qualidade do processo é fundamental para se ter qualidade do produto – *software*.

[2] Requisitos entendidos como necessidades e expectativas do cliente.

- Qualidade é o grau em que um sistema – pode ser artefato ou processo de *software* – atende aos requisitos especificados, como também, às necessidades e expectativas mensuráveis de todos os usuários (ROCHA et al., 2001).

A não-conformidade, no processo de comunicação e transformação da informação no desenvolvimento de produtos de s*oftware*, são enganos, interpretações errôneas, problemas de comunicação, evolução de requisitos, defeitos ou erros que podem ocasionar o mau funcionamento do sistema (ROCHA et al., 2001).

A não-conformidade pode ocorrer nas especificações de requisitos, projeto de sistema, casos de testes ou documentação (PFLEEGER, 2001). Assim, as não-conformidades encontradas pelas verificações e revisões de *software* referem-se aos defeitos. O defeito varia de acordo com a situação a que se destina o *software*. Por exemplo, o que é defeito para uma aplicação pode não o ser para outra (ROCHA et al., 2001).

As classes de defeito não são ortogonais, ou seja, um defeito pode se enquadrar em mais de uma categoria. Segundo TRAVASSOS et al. (1999), a taxonomia de defeitos pode ser classificada em:

- **Omissão:** omissão de informações necessárias sobre o sistema no artefato;
- **Fato incorreto:** contradição entre as informações do artefato e a especificação de requisitos/conhecimento do domínio;
- **Inconsistência:** inconsistentes nas informações contidas em um artefato;
- **Ambigüidade:** ambigüidade nas informações do artefato, que pode levar a uma implementação incorreta;
- **Informação estranha:** as informações são verdadeiras, mas não se aplicam ao domínio.

Na última revisão da ISO/IEC 9000:2000 (ABNT9000, 2001) a principal modificação é o objetivo da garantia da qualidade. Na versão anterior, o objetivo significava atendimento aos requisitos especificados, que passou nesta nova versão para a satisfação do cliente. A satisfação do cliente envolve tanto os requisitos explícitos como os implícitos, ou seja, a satisfação está diretamente relacionada com a qualidade dos produtos e serviços fornecidos, como, por exemplo, suporte ao usuário final.

As empresas têm investido mais no aumento da eficiência do desenvolvimento de *software* (prazos menores e menos recursos gastos) e no reuso de *software*, visando garantir a satisfação do cliente em obter um produto de *software* cada vez mais rápido, de qualidade e de baixo custo de produção. O aumento da eficiência do desenvolvimento de *software* refere-se

ao desenvolvimento de linguagens mais eficientes e ferramentas de apoio, melhora do processo produtivo, gerencial e de treinamento do pessoal, como também, obtenção de um bom arquivo de dados/informação. O reuso, considerado como chave da estratégia de algumas organizações, se torna uma ferramenta facilitadora que permite o ganho em qualidade (redução de falhas pelo reuso de artefatos já testados e aprovados), redução de custos de produção (ao médio/longo prazo, com alteração do enfoque organizacional), diminuição de redundâncias e melhoria no desenvolvimento (aumento da agilidade). Desta forma, o reuso pode ser definido como a utilização de um artefato fora de seu contexto de criação, isto é, uma equipe é a responsável pelo desenvolvimento e outra pelo seu emprego apropriado (JACOBSON et al., 1999).

Um dos pontos críticos da reutilização do produto de *software* é o tempo despendido pelos desenvolvedores para compreender, adaptar e integrar o artefato a ser reusado. A compreensão do artefato reutilizável e o momento de integrá-lo no ciclo de vida de desenvolvimento de *software* faz com que a documentação do *asset*[3] desempenhe um papel vital na viabilização da reutilização. (FAVARO, 1996; FICHMAN, 2001; JACOBSON et al., 1999; POULIN, 1997; SAMETINGER, 1996)

A utilização de um processo definido e gerenciado constitui um elemento fundamental para o desenvolvimento efetivo de aplicações de *software*. Assim, o processo é o principal fator determinante para custo, qualidade e reusabilidade. Mesmo o melhor desenvolvedor não pode ter um bom desempenho se o processo não for bem compreendido, ou seja, não estiver bem definido (STAA, 2002). O processo deve assegurar a utilização de procedimentos definidos de acordo com os critérios de qualidade e de reusabilidade requeridos pela organização. Desta forma, o processo deve considerar a sua adequação às tecnologias envolvidas, ao tipo de *software* em questão, ao domínio de aplicação, ao grau de maturidade (ou capacitação) da equipe em engenharia de *software*, às características próprias da organização, às características do projeto e da equipe, aos papéis dos interessados (*stakeholders*), às principais atividades, aos tipos e pontos de controle da qualidade. Assim, em geral, os problemas relativos à qualidade devem residir na definição do processo e no controle dos artefatos produzidos durante o processo (PAULK et al., 1993; PRESSMAN, 2001; SOMMERVILLE, 2000; ABNT12207, 1998).

[3] *Asset* é um artefato ou conjunto de artefatos criados com o propósito explícito de ser reutilizado em outros esforços de desenvolvimento.

A Norma IEEE (*Institute of Electrical and Eletronics Engineers*) 1517 recomenda que os processos de *software*[4] devem prover *frameworks* para praticar reuso. Nestes *frameworks*, as atividades de reuso devem estar integradas ao ciclo de vida do *software*, de forma que o reuso se torne um modo natural e normal de trabalhar (veja APÊNDICE 3). Portanto, algumas considerações devem ser abordadas: como o reuso muda o ciclo de vida do *software*, exatamente onde ajustar reuso no processo de desenvolvimento de *software*; e, especificamente, quais são os processos, as atividades e as tarefas de reuso que devem ser incluídos no ciclo de vida do *software* (IEEE1517, 1999).

Para viabilizar a reutilização de artefatos é necessário construir uma arquitetura própria (uma arquitetura orientada a reuso), que permita a organização de artefatos. Através do uso em conjunto da organização e estruturação das informações, procura-se facilitar o acesso e o reaproveitamento efetivo dos artefatos (JACOBSON et al., 1997).

Para esta arquitetura, a definição dos requisitos de qualidade é uma parte significativa no processo de desenvolvimento de *software* (JACOBSON et al., 1997). Pois, baseados nestes requisitos, o controle de qualidade deve assegurar aos reutilizadores o nível de qualidade desejada nos *assets* para que eles possam ter confiança em reutilizar. Desta forma, o controle de qualidade deve especificar um processo de avaliação adequado ao método de desenvolvimento escolhido para o domínio da aplicação e a tecnologia adotada para construção dos artefatos reutilizáveis.

1.2 OBJETIVO

O objetivo deste trabalho é propor um processo de garantia de qualidade aplicado ao Processo de Desenvolvimento de *Software* baseado em Reuso no Ministério da Defesa (MD) e de seus Comandos Subordinados proposto por GURGEL (2004). Este trabalho deve fornecer subsídios para os Órgãos Descentralizados de Fornecimento de *Software*[5] promoverem o aumento da produtividade e a melhora da qualidade de seu processo de desenvolvimento de artefatos reusáveis (*assets*). Além disso, deve possibilitar um acompanhamento eficaz de projetos de desenvolvimento de *software* com base em reuso para assegurar melhor planejamento e controle de projetos.

---

[4] Processos de *software* são mapas ou *frameworks* para desenvolver aplicações de *software* (IEEE1517, 1999).
[5] Estrutura de organização proposta por GURGEL (2004) (veja ANEXO 1).

Para tal, foram estudados os processos de garantia de qualidade de *software* existentes na literatura e as particularidades existentes no desenvolvimento baseado em reuso. A definição do processo foi apoiada pela condução de um estudo de caso que permitiu avaliar a viabilidade de sua aplicação no contexto da proposta de GURGEL (2004) para MD.

1.3 VISÃO GERAL DO PROCESSO PROPOSTO

Para que o reuso de artefatos esteja presente em todo ciclo de vida do sistema, proporcionando economia de tempo e recurso, redução no retrabalho e aumento na produtividade, é fundamental que a qualidade do processo de desenvolvimento destes artefatos e os próprios artefatos tenham qualidade assegurada (JACOBSON, 1997). Desta forma, é necessário um controle de qualidade para incluir prevenção e remoção de defeitos dos artefatos (JONES, 1994).

Neste trabalho, foi adotado como definição de qualidade de *software* um conjunto de propriedades de *software* a serem satisfeitas em determinado grau, de modo a satisfazer as necessidades e expectativas mensuráveis de todos seus usuários internos e externos (ROCHA et al., 2001; ABNT9000, 2001). Assim, a partir deste conjunto de propriedades, a qualidade do produto de *software* pode ser descrita e avaliada de forma objetiva e padronizada.

Os princípios a serem adotados na implantação do processo proposto foram:

- Criação da área de qualidade para melhoria contínua da qualidade, medição e análise de processo e artefatos de forma a aumentar efetivamente a produtividade e a satisfação dos usuários externos e internos (KAN, 1995; SEI, 2002). Assim, os resultados de cada fase do processo de desenvolvimento de *software* devem ser avaliados para assegurar a qualidade dos artefatos e garantir um *software* de boa qualidade. Pois, em cada fase do processo, um artefato é produzido para um usuário da fase seguinte. Portanto, cada artefato possui atributos da qualidade que afetam a qualidade do produto final (HAZAN, 1999);
- O controle da qualidade deve ser exercido de forma sistemática (ABNT12207, 1998; ISO15504, 1998; PAULK et al., 1993; SEI, 2002);
- O *software* deve estar sem defeito (ausência de não-conformidades), bem documentado[6] e desenvolvido por métodos eficazes e técnicas adequadas (KAN, 1995).

[6] Um *software* bem documentado significa que a documentação existe, além disso, esta documentação está completa e atualizada de acordo com as alterações realizadas no *software* (ROCHA, 2001).

Além disso, um *software* de qualidade deve produzir resultados relevantes e confiáveis no tempo certo, ter as funcionalidades necessárias e as possíveis expectativas desejadas ao uso, ser mensurável, ser corrigível, adaptativo e evolutivo, operar com economia de recursos e ser desenvolvido economicamente dentro do prazo estipulado (FIORINI et al., 1998; STAA, 2002).

Para a qualidade ser efetiva dentro do processo de desenvolvimento de *software*, ela deve resultar na redução significativa do retrabalho (CROSBY, 1996). Desta forma, os defeitos introduzidos ao longo do processo de desenvolvimento de *software* devem ser identificados o quanto antes, se possível na própria fase em que são cometidos, pois o esforço gasto e os custos associados para corrigir um defeito aumentam proporcionalmente ao tempo entre a sua introdução e exclusão, ou seja, quanto mais cedo for descoberto o defeito, menor o esforço para corrigi-lo (JACOBSON et al., 1997; FICHMAN, 2001; ROCHA et al., 2001; JONES, 1997). Portanto, os benefícios da qualidade resultantes da ausência de não-conformidades possibilitam que as organizações minimizem o esforço do retrabalho, reduzindo o tempo de desenvolvimento de *software* e os custos. Segundo ROSENBERG et al. (1998), os problemas que não são encontrados e corrigidos até a fase de teste custam 14 vezes mais do que corrigi-los na fase de requisitos.

A implantação de um programa de qualidade deve começar pela definição e implantação de um processo de desenvolvimento de *software*. Este processo deve ser documentado, compreendido e seguido por todos os profissionais nele envolvidos.

O método utilizado para exercer ação sobre o processo é o ciclo de Deming (DEMING, 1982), conhecido como PDCA (*Plan/Do/Check/Act*). Na **FIG. 1.3-1** é apresentada um diagrama explicativo do ciclo PDCA de Deming e da espiral de ascensão de qualidade. O ciclo de Deming é representado por um círculo em rotação no sentido horário, demonstrando um desenvolvimento iterativo e a necessidade da melhoria contínua. Na espiral de ascensão de qualidade, é apresentado o uso sistemático do PDCA que possibilita a melhoria da qualidade.

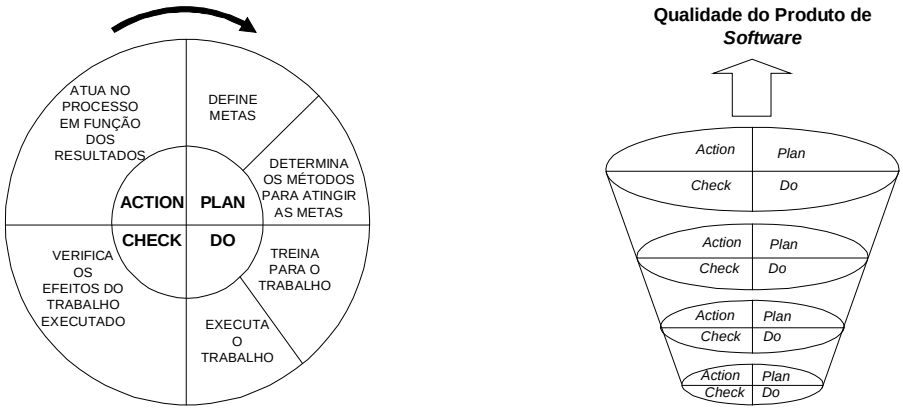


FIG. 1.3-1 (a) Ciclo PDCA de Deming, (b) Espiral de ascensão da qualidade. adaptado de (FALCONI, 1999).

O PDCA constitui em um processo que atua sobre os demais processos dividido em 4 etapas, a saber: (FALCONI, 1999)

1 - ***Pan*****:** etapa de planejamento que consiste em estabelecer metas e métodos para alcançar as metas propostas;

2 - ***Do*****:** etapa de execução que consiste na realização das tarefas de acordo com o planejado e na coleta de dados da verificação;

3 - ***Check*****:** etapa de verificação onde os dados coletados são comparados com os resultados esperados;

4 - ***Action:*** etapa ação onde é exercida uma atuação sobre o processo em si que pode ser de duas formas: se as metas foram atingidas, então o plano proposto deve ser adotado como padrão, ou se as metas não foram alcançadas, ações devem ser exercidas sobre as causas do não alcance das metas.

1.4 ORGANIZAÇÃO

Esta dissertação é composta por cinco capítulos. No Capítulo 1, **Introdução**, é apresentada uma breve história da informática desde sua criação até os dias atuais para relatar como a qualidade e a reusabilidade de *software* estão inseridas dentro dos processos de engenharia de *software*. Também são abordadas algumas definições de qualidade apontadas pela literatura e estabelecidos os princípios da qualidade adotados para esta dissertação.

O Capítulo 2, **Fundamentos Teóricos**, são abordados conceitos gerais da engenharia de qualidade, garantia da qualidade, técnicas aplicadas ao controle e melhoria da qualidade do

processo de desenvolvimento de *software*. Além disso, são apresentados conceitos e tipos de reuso, problemas relacionados a reuso identificados na literatura e os principais modelos e métricas voltados a reusabilidade.

A proposta da dissertação é detalhada no Capítulo 3, **Processo da Garantia da Qualidade de *Software* baseado em Reuso Sistemático**. Neste capítulo, é definida uma metodologia para assegurar a qualidade e validar os *assets* produzidos pelo Órgão Descentralizado de Fornecimento de *Software* proposto por GURGEL (2004). Para isso, é estabelecida uma estrutura organizacional do Grupo de Garantia da Qualidade de *Software*.

No Capítulo 4, **Estudo de Caso,** é apresentado um estudo de caso realizado para avaliar a viabilidade da proposta definida no capítulo anterior, e aplicar as técnicas especificadas, definir *templates* e analisar os resultados obtidos deste trabalho.

No Capítulo 5, **Conclusões**, são apresentados as conclusões e os principais benefícios desta dissertação, assim como, sugere possíveis trabalhos futuros.

2 FUNDAMENTOS TEÓRICOS

2.1 ENGENHARIA DE QUALIDADE

A Engenharia de Qualidade é responsável pela determinação e o planejamento do trabalho. Ela deve zelar pela qualidade em termos de determinar o que fazer para que o todo alcance os resultados propostos (CROSBY, 1996). Para isto, um Comitê de Qualidade deve ser formado por consultores da qualidade para apoiar as equipes de desenvolvedores/ montadores de projetos nas avaliações da qualidade dos artefatos produzidos durante todo processo de ciclo de vida do projeto, determinando de que modo o produto deve ser verificado, inspecionado, testado e controlado no decorrer de sua vida dentro e fora da organização (ROCHA et al., 2001). Este comitê deve, também, ajudar na identificação dos atributos de qualidade necessários para que cada um dos projetos alcance seus objetivos e a equipe de desenvolvedores/montadores enfoque no desenvolvimento de *software* baseado no reuso (IEEE1517, 1999).

A garantia da qualidade somente pode ser assegurada se existir um padrão de especificação confiável e seguido, e também controlado de forma sistemática. O grupo de garantia da qualidade de *software* deve utilizar técnicas, ferramentas, métodos, padrões,

normas, processos e bases de dados para apoiar os diversos projetos de *software*. Ele deve apoiar a equipe de desenvolvimento de *software* por meio de verificações e validações, que permitem avaliar os artefatos produzidos (ABNT12207, 1998; ISO15504, 1998; ROCHA et al., 2001; FIORINI et al., 1998).

A avaliação deve ser planejada e realizada, conforme o planejamento por revisores, verificadores ou testadores treinados e sem vínculo operativo no artefato examinado (PAULK et al., 1993; CROSBY, 1996; ABNT12207, 1998; ISO15504, 1998; ROCHA et al., 2001; SEI, 2002). As Normas CMM, CMMI, ABNT 12207 e ISO 15504 recomendam que os executores das atividades de verificação e validação não devem ter participação na elaboração do artefato a ser examinado (PAULK et al., 1993; SEI, 2002; ABNT12207, 1998; ISO15504, 1998). As tensões entre autores e revisores dos artefatos são apontadas na abordagem da Gerência pela Qualidade Total[7] (GQT) como grandes fontes geradoras de conflitos que comprometem a qualidade do produto (KAN, 1995). Esta fonte geradora de conflito deve ser minimizada através de esclarecimentos durante a etapa de treinamento da revisão e, também, com a introdução de um líder que exerce o papel de moderador nas reuniões, de forma a eliminar ou reduzir o número de conflitos (WIEGERS, 1995).

Cada atividade de verificação e validação deve apresentar dois resultados: produto de *software* é aceito ou rejeitado, e o registro de não-conformidades e de cálculos (ABNT12207, 1998). Acumulando os resultados dos cálculos e analisando-os, o Comitê de Qualidade pode determinar exatamente o estado do produto de *software*. Ele reúne os dados provenientes das atividades de verificação e validação, e os registra de maneira a ser de utilidade prática para todos os interessados. A gerência de reuso e de projeto precisa de dados acurados das tendências, para saber quando e onde deve agir (IEEE1517, 1999).

O controle da qualidade deve produzir um documento contendo todos os problemas identificados, e estes devem ser resolvidos. Após a cada alteração, é necessário repetir o controle da qualidade (ABNT12207, 1998; FIORINI et al., 1998).

Ishikawa propôs sete ferramentas básicas de estatística para controle da qualidade (PALADINI, 1994) que são úteis para o planejamento e controle de projetos de desenvolvimento de *software*. Dentre elas se destacam o Diagrama de Causa-Efeito, o *Checklist* e o Diagrama de Pareto.

---

[7] GQT é um sistema que combina técnicas de controle da qualidade e modelos organizacionais, onde a qualidade é reconhecida como uma vantagem estratégica com o apóio total da alta-administração.

O Diagrama de Causa-Efeito, conhecido como Gráfico Espinha-de-Peixe ou Gráfico de Ishikawa, identifica todos os fatores de causa de uma característica da qualidade em um único gráfico. Sua forma é similar à espinha de peixe, onde o eixo principal mostra a característica da qualidade, e as espinhas representam os fatores que afetam esta característica. (PALADINI, 1994)

O *Checklist* é um formulário que contém itens para serem verificados conforme as necessidades específicas de seus usuários, e por isso, apresentam extrema flexibilidade de elaboração, utilização e interpretação. Seu principal objetivo é facilitar a coleta de dados que vão ser analisados posteriormente. (PALADINI, 1994)

O Diagrama de Pareto, representado por barras de freqüência, é utilizado para classificar as causas de defeitos que atuam em um processo específico de acordo com seu grau de importância (PALADINI, 1994). A análise de Pareto retrata o princípio 80-20 (20% das causas são responsáveis por 80% dos defeitos). Este diagrama é bastante aplicado na Qualidade de *Software*, porque os defeitos do *software* ou densidade dos defeitos não seguem uma distribuição uniforme (HAZAN, 1999).

2.1.1 TESTES

Os testes são atividades de validação e verificação que consistem na análise dinâmica do *software*, ou seja, na execução do produto de *software* com o objetivo de verificar a presença de não-conformidades cometidas ao longo do processo de desenvolvimento do *software*. Para tal, deve-se construir e aplicar planos de teste capazes de identificar não-conformidades nos artefatos (ROCHA et al., 2001).

O Plano de Teste é um documento que descreve o planejamento das atividades e tarefas de teste, ou seja, define os objetivos do teste, escopo, estratégia, procedimentos do teste, ambiente de teste, casos de teste, itens a serem testados, tipos de testes a serem executados, escala de teste, recursos humanos, relato de procedimentos, critério de análise, riscos e plano de contingência. Ele deve ser iniciado logo na fase de elicitação das necessidades do cliente, pois ajuda a melhorar as interações das atividades de análise do domínio, projeto de domínio e codificação ou montagem de componentes. Nas fases posteriores o plano deve ser refinado com mais detalhes (LEWIS, 2000).

A falta de um planejamento das atividades de desenvolvimento é uma das causas da crise do *software* (PRESSMAN, 2001). O planejamento da atividade de teste deve estar inserido

dentro do planejamento do projeto, onde são estimados os recursos e definidos estratégias, métodos e técnicas de teste para formar um critério de aceitação do *software* em desenvolvimento (LEWIS, 2001).

Um caso de teste é um conjunto específico de dados de teste e *scripts* de teste (ROCHA et al., 2001). O *script* de teste é um guia para o testador na realização do teste e assegura consistência entre as partes de execuções do teste. Um teste também inclui resultados esperados para verificar se o teste encontrou o objetivo corretamente. Durante a fase de codificação, os *scripts* de teste e os dados de teste devem ser gerados. Na fase de aplicação de teste, os *scripts* de teste devem ser executados e os resultados devem ser analisados. Um ou mais planos de testes podem ser usados para cada tipo de teste (LEWIS, 2000).

Segundo HARROLD (2000), a atividade de teste possui algumas limitações:

- Dado uma seqüência de comando, identificar se é executada ou não;
- Dados duas seqüências de comandos de um programa ou de programas diferentes, identificar se as seqüências executam a mesma função;
- O programa pode apresentar um resultado correto para um dado de entrada específico, satisfazendo um requisito de teste e não revelando a presença de um erro;
- Inviabilidade de testar todas as possibilidades, pois o domínio dos dados de entrada normalmente é infinito ou muito grande (teste exaustivo);
- Limitação do tempo e recursos;
- Indisponibilidade de ferramentas adequadas para a realização dos testes.

Uma maneira de minimizar as limitações é estabelecer critérios de teste que devem ser estabelecidos no Plano de Teste. A princípio devem ser definidos os critérios de teste de aceitação, que vão servir de base para os critérios de testes de sistema, integração e de unidade. Na etapa de execução do teste, o processo deve ser inverso ao do plano de teste, começando pelo teste de unidade. Na **FIG. 2.1.1-1** é apresentado um diagrama que relaciona as fases de desenvolvimento de projeto e os tipos de testes correspondentes. Ao estruturar o processo de teste em cada fase do processo de desenvolvimento do *software*, diferentes tipos de defeitos e aspectos do *software* são abordados, resultando em maior cobertura e minimizando a complexidade da atividade (MYERS, 1979; MALDONADO, 1991).

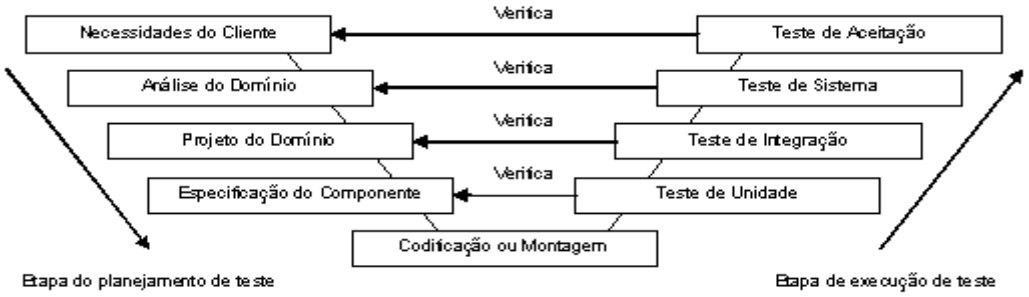


FIG. 2.1.1-1: Diagrama de relacionamento das etapas de desenvolvimento de projeto e os tipos de testes aplicados a cada etapa.

As técnicas de teste, quanto à origem da informação utilizada para estabelecer os requisitos de teste, podem ser: funcional ou caixa-preta (PRESSMAN, 2001), estrutural ou caixa-branca (PRESSMAN, 2001), com base em erros (DEMING, 1990) e com base em máquinas de estado finito (FUJIWARA, 1991).

Basicamente há quatro tipos de teste (LEWIS, 2000; ROCHA et al., 2001): teste de unidade, teste de integração, teste de sistema e teste de aceitação.

Os testes de unidade testam a lógica do código, independentemente do ambiente (testes de caixa branca), e os testes de integração focam na integração destas unidades testadas com ambiente do projeto (ROCHA et al., 2001). Tanto os testes de unidade como de integração devem ser escritos pelos programadores, por ambos requerem conhecimento específico das tecnologias envolvidas, tais como, linguagens de programa e parâmetros de integração entre as unidades (LEWIS, 2000). O teste de integração deve envolver todos os níveis de detalhes de implementação de base de dados e linguagens, componentes externos ou protocolos de comunicação utilizados para a construção do sistema (LEWIS, 2000).

O teste de sistema visa identificar erros de funções e atributos de qualidade que não estejam em conformidade com a especificação (PRESSMAN, 2001). O sistema deve ser testado no ambiente de operação computacional antes da realização do teste de aceitação. O teste de caixa-cinza, que combina as abordagens caixa-branca (teste estrutural) e caixa-preta (teste funcional), é freqüentemente aplicado neste tipo de teste (ROCHA et al., 2001).

O teste de aceitação certifica que o sistema de *software* satisfaz os requisitos originais. Este teste não deve ser realizado antes do completo sucesso do teste de sistema. Neste tipo de teste é empregada a técnica de caixa-preta para testar o sistema contra suas especificações. Em geral o teste de aceitação é um subconjunto de um ou mais testes de sistema (LEWIS, 2000).

Os casos de teste de aceitação devem incluir testes de desempenho, testes de carga, testes de reusabilidade e testes de compatibilidade de arquitetura de domínio. Os testes de reusabilidade, que incluem os testes de generalidade e de adaptabilidade, devem ser escritos pelos engenheiros de domínio, assim como o teste de compatibilidade de arquitetura de domínio. O teste de generalidade consiste em validar a modificação do domínio da aplicabilidade do *asset*, ao passo que o teste de adaptabilidade verifica a facilidade de reuso dos *assets*. Os testes de sistema devem ser escritos pela equipe de desenvolvimento/ montadores e os testes de aceitação pelos engenheiros de domínio e cliente (IEEE1517, 1999).

Em geral os testes devem ser executados por uma equipe diferente da que faz a programação, exceto os testes de unidade (CROSBY, 1996; LEWIS, 2000). A equipe de teste (testadores) é responsável pela execução dos testes e deve assegurar que o teste seja executado conforme o plano de teste (ROCHA et al., 2001).

Cada não-conformidade detectada durante os testes deve ser documentada. Assim, para cada teste deve ser gerado um Relatório de Teste e verificado o atendimento aos requisitos especificados nos casos de testes (LEWIS, 2000; ROCHA et al., 2001). O Relatório de Teste deve conter local e hora, recursos utilizados (*hardware*, *software* e pessoal), os resultados obtidos, problemas e não-conformidades detectadas, além do nível de adeqüabilidade (aprovado, desaprovado ou aprovado com pendências) (LEWIS, 2000).

Atualmente no mercado, existem muitas ferramentas para testes, conforme apresentado em (LEWIS, 2000). Elas provêem rapidez na execução, execução sem interferência humana, programável, repetível, reusável e fornece análise de cobertura de código depois da execução do teste. Mas, os principais fatores que limitam a ferramenta de teste são (LEWIS, 2000):

- Se o teste for executado apenas uma vez, uma ferramenta de teste pode não valer o tempo exigido e despesa;
- Se existe pressão para terminar os testes dentro de um prazo de tempo determinado, uma ferramenta de teste pode não ser apropriada, pois demanda tempo para aprender, configurar e integrar uma ferramenta de teste à metodologia de desenvolvimento;
- Se o sistema muda rapidamente a cada iteração do ciclo espiral de teste, mais tempo será despendido em manter a ferramenta de teste de regressão.

Portanto, a seleção da ferramenta de teste mais adequada para o ambiente de desenvolvimento é um fator crítico para o sucesso das atividades de teste. Ela deve ser empregada quando o processo manual é inadequado, como por exemplo, uma ferramenta de teste de carga pode simular vários usuários virtuais em condições controladas de estresse.

Para apoiar a seleção de uma ferramenta de teste, já que não é uma tarefa simples, Lewis aborda algumas questões por meio de um *checklist* em (LEWIS, 2000).

2.1.2 INSPEÇÕES DE *SOFTWARE*

Embora a aplicação de testes para o controle da qualidade do *software* seja importante, o uso exclusivo de testes é pouco produtivo. Mesmo sendo indispensável realizar testes bem feitos, a qualidade do *software* não melhora com o aumento do investimento em testes (LEWIS, 2000).

LAITENBERGER (2002) relata que a introdução de inspeção de código reduz 39% dos custos com defeito em relação à realização de testes e a inserção de inspeção de projeto reduz 44% dos custos com defeito comparados aos testes. Assim, os artefatos devem sofrer inspeções ao longo do processo de desenvolvimento de *software* para garantir a qualidade interna dos artefatos produzidos. Os objetivos das inspeções são:

- Verificar se o artefato examinado satisfaz as suas especificações e se está em conformidade com os padrões aplicáveis;
- Identificar desvios de padrões e especificações;
- Coletar dados de engenharia de *software*, como dados de esforço e defeitos.

A inspeção de *software* possibilita a detecção e a remoção de defeitos em artefatos assim que estes são criados (LAITENBERGER, 2002; FAGAN, 1986; DOOLEAN, 1992). Na **FIG. 2.1.2-1** é apresentada curvas de desenvolvimento de software em relação a escala de tempo com e sem inspeção (FAGAN, 1986).

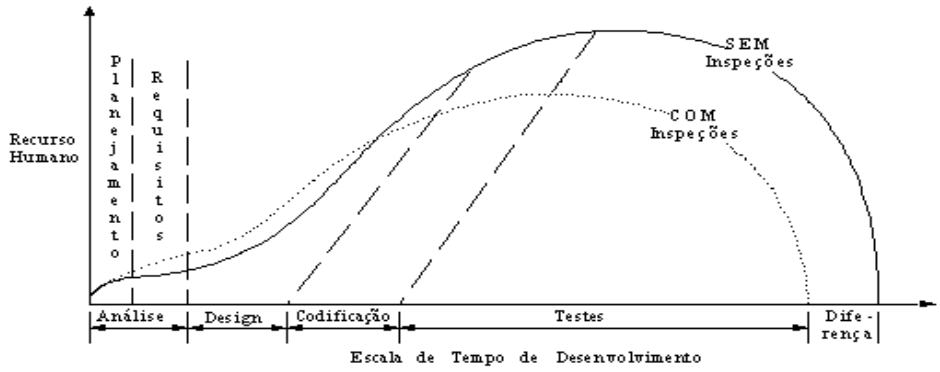


FIG. 2.1.2-1: Comparação de desenvolvimento de *software* com e sem inspeções.

As inspeções de *software* precisam estar integradas ao processo de desenvolvimento de *software*. Assim, a inspeção deve ser adequada para cada etapa de desenvolvimento do

projeto. Na **FIG. 2.1.2-2** é apresentado o Modelo V de Vorgehens (Modelo de Produto) que relaciona o produto desenvolvido e a inspeção de *software* (BRÖHL, 1995), aplicável a qualquer tipo de desenvolvimento de *software* (seqüencial, em paralelo ou de modo incremental).

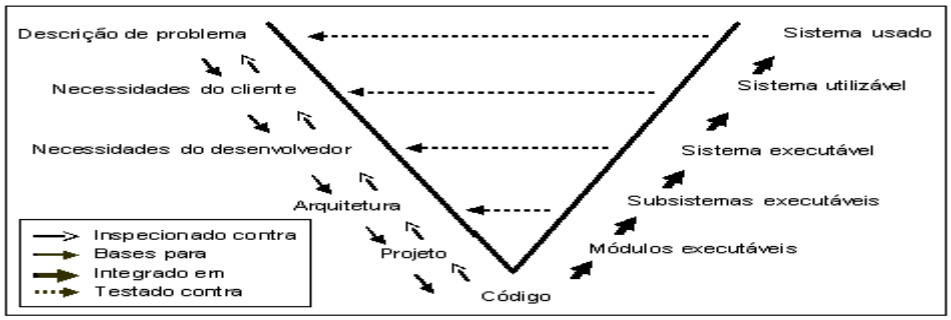


FIG. 2.1.2-2: Diagrama de relação dos artefatos produzidos e a inspeções de *software*.

2.1.2.1 REVISÕES

A revisão formal é uma técnica de revisão do *software* ou de alguns de seus artefatos, executada, sistematicamente, ao final de cada fase do processo de desenvolvimento de *software*, com o objetivo único de encontrar não-conformidades. Esta técnica é executada por uma equipe na qual cada membro tem papel preestabelecido. O desenvolvedor autor do artefato em inspeção participa, mas não coordena a reunião. (TAVARES, 2002)

As revisões podem ser implementadas via inspeções ou *walkthroughs* (reunião conjunta para listar problemas encontrados e ações decorrentes). A inspeção de s*oftware* pode utilizar técnicas de avaliação formal onde os artefatos são examinados de maneira criteriosa por uma pessoa, sem a participação do autor, para detectar faltas, violações de padrões de desenvolvimento e outras não-conformidades. As inspeções, também, podem ser usadas para obtenção de conhecimento ou como base para tomada de decisões (REGNELL et al., 1999).

Muitas variações têm sido propostas para o processo de revisão desde sua criação, mas todas são influenciadas pelas idéias iniciais de Fagan e Gilb (FAGAN, 1986; GILB, 1993): a realização de uma revisão criteriosa do artefato para identificar as possíveis não-conformidades (defeitos). Essas variações incluem a modificação na estrutura da reunião de revisão ou do número de revisores que devem participar da reunião, ou na eliminação da reunião de revisão (FAGAN, 1986; GILB, 1993; PORTER et al., 1995; VOTTA, 1993). Um exemplo dessas variações é a realização da revisão por uma equipe na qual cada membro tem

papel preestabelecido, onde o autor do artefato a ser revisado participa, mas não coordena a reunião (LAITENBERGER, 2002).

Para Fagan e Gilb (FAGAN, 1986; GILB, 1993), o método de revisão identifica as fases de planejamento, detecção, coleta e correção de não-conformidades. Entretanto, este método não fornece para o revisor uma diretriz de como as não-conformidades devem ser encontradas na fase de detecção. Ambos, também, admitem que a revisão individual de artefatos pode ser feita satisfatória e eficientemente apenas com o conhecimento do revisor (abordagem *ad-hoc*).

A revisão deve ser realizada através da inspeção individual pelo revisor que pode ser desenvolvedor/montador desde que não seja o próprio autor do artefato inspecionado, para identificar as possíveis não-conformidades (defeitos) através da técnica de leitura adequada ao artefato em questão. Mas, a parte principal do processo de inspeção é a detecção de defeitos por técnica de leitura de documentos e o registro de não-conformidades detectadas. Todo o material gerado do artefato deve ser lido, as não-conformidades anotadas e uma estatística das não-conformidades detectadas deve ser posteriormente realizada para fins de trabalho futuro da eficácia do procedimento. (ABNT12207, 1998; HAZAN, 1999; LAITENBERGER, 2002)

2.1.2.2 TÉCNICAS DE LEITURA DE *SOFTWARE*

Uma abordagem para identificar defeitos em artefatos, diferente da abordagem *ad-hoc* normalmente utilizada na maioria dos casos, é fornecida pelas técnicas de leitura de *software*. Uma técnica de leitura é uma série de procedimentos ou estratégias preparada para análise de um artefato que permite alcançar a compreensão necessária para uma determinada tarefa (BASILI et al.,1996). As técnicas de leitura aumentam a eficiência dos revisores por fornecerem diretrizes do que procurar, já que estas técnicas propiciam que o resultado da atividade de detecção de defeitos seja menos dependente dos fatores humanos como a experiência. Em vez dos revisores aplicarem apenas seus próprios conhecimentos e técnicas, as técnicas de leitura agregam o conhecimento sobre as melhores práticas para detecção de defeitos em um procedimento que pode ser seguido e repetido. Essas técnicas de leitura auxiliam a identificação de defeitos e melhoram o processo de revisão de *software* de outros artefatos não código-fonte (BASILI et al., 1996; FUSARO, 1997; PORTER et al., 1995; ZAHRAN, 1998). As principais técnicas de leituras e suas características são apresentadas em (LAITENBERGER, 2002). Na **TAB. 2.1.2.2-1** são apresentadas as duas técnicas de inspeção

mais utilizadas na indústria (*ad-hoc* e *checklist*) e duas variantes da técnica de leitura baseada em cenário em atual difusão (Leitura baseada em Defeito e em Perspectiva).

TAB. 2.1.2.2-1: Técnicas de inspeção de *software* e suas características.

Técnicas de Leitura	**Características**						
	Contexto da aplicação	**Usabilidade**	**Repetibilidade**	**Adaptabilidade**	**Risco**	**Necessidade de treinamento**	**Validação**
Ad-Hoc	Todos os produtos	Não	Não	Não	Não	Não	Prática industrial
Checklist	Todos os produtos	Não	Não	Sim	Depende do caso	Não	Prática industrial
Leitura baseada em Defeito	Todos os produtos	Sim	Depende do caso	Sim	Alto	Sim	Validação experimental
Leitura baseada em Perspectiva	Todos os produtos	Sim	Sim	Sim	Alto	Sim	Validação experimental e iniciado uso industrial

O método *ad-hoc* é uma técnica não sistemática, onde todos os revisores desempenham as mesmas tarefas utilizando seus próprios conhecimentos. O método *checklist* é similar ao método *ad-hoc*, mas uma lista de defeitos é fornecida. Já no método cenário, os revisores têm responsabilidades fragmentadas e coordenadas (LAITENBERGER, 2002).

Os métodos *ad-hoc* e *checklist* são os dois métodos de detecção de defeitos mais utilizados (LAITENBERGER, 2002). Entretanto, Cheng identifica alguns dos principais problemas dos revisores ao utilizar esses métodos (CHENG et al., 1996), tais como:

- Os revisores são sobrecarregados com informação excessiva;
- Desconhecimento dos revisores sobre o assunto de negócio e detalhes do projeto;
- Responsabilidades ambíguas;
- Não tem ponto de partida bem definido devido aos itens acima;
- Interação limitada entre revisores e equipe de projeto;
- Participação de revisores inexperientes;
- Procedimento de revisão não sistemática e conjunto de perguntas despreparadas para questionar sobre o projeto.

No método de Leitura baseado em Cenário, o cenário é um procedimento que o revisor de um documento deve seguir durante a inspeção. O cenário pode ser benéfico ao dirigir a atenção dos revisores para os defeitos mais críticos da organização (LAITENBERGER, 2002). Neste método, os revisores lêem um documento com diferentes perspectivas (REGNELL et al., 1999), de forma que diferentes revisores tenham diferentes

responsabilidades e sejam direcionados em suas leituras por um cenário específico que se agrega ao construir o modelo invés de uma leitura passiva. Existem duas variantes da técnica de Leitura baseada em Cenário: Leitura baseada em Defeito (PORTER et al., 1995) e Leitura baseada em Perspectiva (BASILI et al., 1996). A Leitura baseada em Defeito concentra-se em especificar classes de defeitos, enquanto que a Leitura baseada em Perspectiva concentra-se na visão dos usuários do documento (BASILI et al., 1996) (projetistas na fase de projeto, testadores na fase de testes e usuários na fase de operação).

A principal idéia da técnica de Leitura baseada em Perspectiva é que um produto de *software* deve ser inspecionado sob a perspectiva de diferentes *stakeholders* (BASILI et al., 1996) (BRIAND et al., 1998; LAITENBERGER et al., 1997; LAITENBERGER, 2000; LAITENBERGER, 2002). Para cada perspectiva, tanto um como vários cenários são definidos, consistindo de atividades repetíveis que um revisor precisa executar e de perguntas que um revisor deve responder. A técnica de Leitura baseada em Perspectiva foi aplicada para inspecionar documentos de requisitos (BASILI et al., 1996), modelos de projeto orientado a objeto (LAITENBERGER et al., 1997) e documentos de código (LAITENBERGER, 2000).

A principal idéia da Leitura baseada em Defeito é que diferentes revisores focalizam diferentes classes de defeito enquanto examinam o mesmo documento (MILLER et al., 1998; PORTER et al., 1998; PORTER et al.,1995; SANDAHL et al.,1998; LAITENBERGER, 2002). Para cada classe de defeito, há um cenário que consiste em um conjunto de perguntas que um revisor deve responder enquanto lê. Ao responder as perguntas, o revisor deve estar detectando defeitos daquela classe em particular.

O método de Leitura baseado em Cenário tem apresentado uma taxa mais alta de detecção de defeitos do que *ad-hoc* e *checklist* (PORTER et al., 1995) (MILLER et al., 1998) (REGNELL et al., 1999): taxa superior a 35%. Isto ocorre, porque os cenários ajudam a focar classes específicas de defeitos e independem da habilidade de identificar outras classes de defeitos.

2.1.2.2.1 TÉCNICA DE LEITURA BASEADA EM PERSPECTIVA

Na Técnica de Leitura baseada em Perspectiva, o revisor, ao preencher a lista de conferência, deve assumir uma perspectiva específica (revisor/avaliador), visando o objetivo de inspecionar e validar o documento. A lista de conferência deve ser composta por um conjunto de aspectos e cada aspecto deve conter uma série de questões relacionadas. Os

aspectos devem considerar o tipo de produto a ser especificado e os atributos de qualidade necessários para atender a satisfação do cliente. Cada questão deve ter opções de resposta e deve ser elaborada de forma que a resposta positiva assegura adequação ao objetivo do aspecto proposto. Após a conclusão da inspeção, deve ser elaborado um documento, onde devem estar registrados os resultados obtidos, comentários e sugestões de melhorias. (PAGLIUSO et al., 2002)

Nesta técnica, os revisores são solicitados a desenvolver abstrações do sistema pelos requisitos, utilizando diferentes pontos de vista. Para um projeto OO, as abstrações das informações importantes já existem: a informação está descrita em diferentes diagramas e documentos, como diagramas de classes e interação, descrição de classes e outros. Mas, apesar disso, os defeitos ainda existem e precisam ser identificados. (ROCHA et al., 2001)

2.1.2.2.2 TÉCNICA DE LEITURA ORIENTADA A OBJETO

A Técnica de Leitura Orientada a Objeto (TLOO) é um conjunto de sete técnicas diferentes para leitura de projeto OO. Estas técnicas foram desenvolvidas para fornecer um procedimento para as inspeções individuais dos diferentes diagramas e documentos de projeto OO e identificar as classes de defeitos genéricos relativos à especificação de requisitos e ao projeto. (TRAVASSOS et al., 2003)

O processo de leitura utilizando TLOO baseia-se em técnicas horizontais e verticais para apoiar as revisões. Na **FIG. 2.1.2.2.2-1** é apresentado o conjunto de TLOO (TRAVASSOS et al., 2003), onde cada linha entre os diferentes artefatos de *software* representa uma técnica de leitura definida para ler um documento comparando-o com outro.

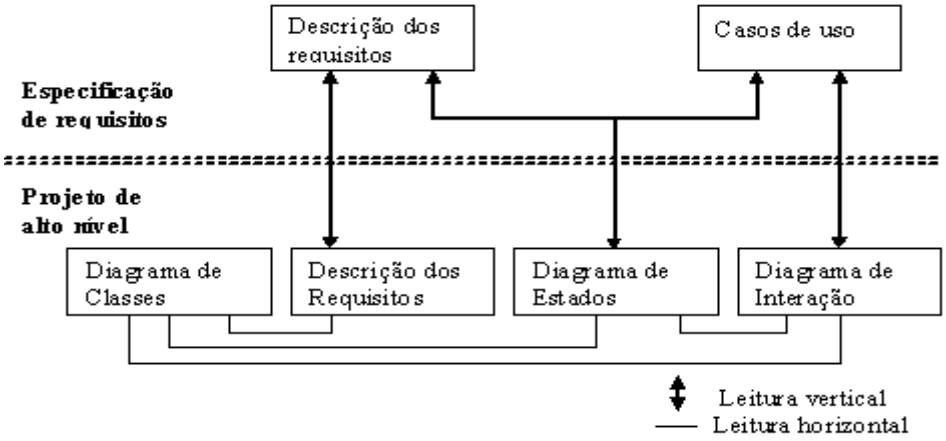


FIG. 2.1.2.2.2-1: Conjunto de TLOO.

Através das leituras horizontais assegura-se que todos os diagramas de projeto estejam consistentes. E, através das leituras verticais assegura-se à consistência entre os artefatos de projeto e os requisitos do sistema. (TRAVASSOS et al., 2003)

A vantagem em utilizar esta técnica está na seleção de apenas um subconjunto de técnicas correspondentes aos artefatos que devem ser inspecionados ou que são importantes em um determinado momento no processo de desenvolvimento de projeto OO. (ROCHA et al., 2001)

2.1.3 PROGRAMA DE MEDIÇÃO DE *SOFTWARE*

A implantação de um processo de medição de *software* visa fornecer ao Gerente de Reuso um conjunto de dados úteis para planejar e controlar os projetos de *software* com rigor e precisão (PRESSMAN, 2001). Para isto, é necessária a motivação de todos os profissionais que atuam no processo de desenvolvimento de *software*. Também é necessário que a gerência de reuso estabeleça um programa de medição corretamente planejado e implementado para avaliar o progresso do programa de qualidade e de reuso, como também assegurar que os objetivos de qualidade e de reusabilidade estão sendo cumpridos (IEEE1517, 1999).

O Gerente de Reuso deve determinar os atributos de qualidade apropriados para a avaliação do projeto de forma a este atingir seus objetivos. Além disso, os dados obtidos da avaliação devem ser armazenados de forma consistente para sua utilização em projetos futuros. (JACOBSON et al., 1997; IEEE 1517, 1999)

O processo de coleta de dados deve ser simples e ferramentas automáticas para extração destes dados devem ser utilizadas, já que a quantidade de dados coletados e o gerenciamento dos mesmos é um aspecto crítico. Caso não seja possível a coleta de dados automatizada, a organização deve definir padrões (*templates*) para obtenção dos dados, assegurando assim a melhoria da qualidade dos dados obtidos (MCGARRY et al., 2001).

Dentro do contexto de programa de medição os termos indicadores, medida e métrica são empregados conforme a definição a seguir (STAA, 2002):

- **Métrica**: medida quantitativa de um atributo. As métricas podem ser objetivas ou subjetivas (medidas relativas ao alto ou baixo grau de precisão respectivamente), diretas ou indiretas (medidas simples ou derivadas de outras medidas respectivamente) e de produto ou de processo (medidas da qualidade do produto ou da qualidade do processo respectivamente). Por exemplo, número de defeitos por linhas de código;

- **Medida**: avaliação de um resultado perante uma unidade padrão. Por exemplo, 1000 linhas de código;
- **Indicador**: forma de representação quantitativa de características de produtos e de processos utilizado para acompanhar, avaliar e melhorar os resultados ao longo do tempo. Por exemplo, aumento no número de defeitos detectados na nova versão do produto de *software*.

O modelo de qualidade elaborado por MCCALL et al. (1977) foi o primeiro dos modelos que propôs qualidade de produto de *software* como uma hierarquia de fatores, critérios e métricas. Esforços internacionais também conduziram ao desenvolvimento de um padrão por medida de qualidade de produto de *software*, a norma ISO/IEC 9126. No APÊNDICE 3 é apresentada uma tabela resumida, **TAB. 7.3.1-1**, dos atributos de qualidade do produto da Norma ISO/IEC 9126 (ISO9126, 2000).

Bansiya e Davis relatam que todos os modelos de qualidade baseado em produto variam na definição hierárquica de qualidade, mas compartilham as seguintes dificuldades (BANSIYA et al., 2002):

- Os modelos são vagos nas suas definições e nas métricas necessárias para atingir uma avaliação quantitativa de qualidade de produto;
- Falta de um estudo mais detalhado para considerar a dependência entre atributos de qualidade, pois os atributos nem sempre são independentes entre si, podendo um ou um grupo de atributo de qualidade influenciar de forma conflitante na qualidade global. Por exemplo, um objetivo de qualidade para uma alta flexibilidade dificulta a possibilidade de alcançar um objetivo de baixa manutenibilidade;
- Os modelos não incluem como considerar o grau de influência dos atributos individualmente, pois ao avaliar a qualidade do produto como um agregado de um conjunto de atributos de qualidade, o significado de todos os atributos para a qualidade do produto não pode ser igual e, então, a influência de um atributo pode precisar ser mudada por um fator de peso. Por exemplo, para organizações com redes sofisticadas e processamento em tempo real, os atributos de desempenho e de confiabilidade podem ser os atributos mais importantes (KAN, 1995), ao passo que para as organizações com várias plataformas, os principais atributos são a portabilidade e extensibilidade.

A identificação de um conjunto de atributos de qualidade que completamente representa avaliação de qualidade não é uma tarefa trivial. Em geral, estes atributos estão relacionados aos objetivos gerenciais, metas empresariais, competição, economias e tempo alocado para o desenvolvimento do produto. Por isso, geralmente as organizações realizam um processo de

medição com um alto risco de insucesso. As dificuldades mais comuns na implantação de um processo de medições são as seguintes (DERBY, 2001; SINGER et al., 2002):

- Muitos programas de medição são iniciados com a medição de atributos que são convenientes e fáceis de se medir, ao invés de se medir os dados que são realmente necessários para uma tomada de decisão. Como resultado, tem-se uma grande quantidade de dados sem utilidade. Considerando que há uma demanda de esforço para obtenção e consolidação desses dados inúteis, o programa de medição acaba tornando-se muito custoso em relação aos benefícios e termina sendo descontinuado. É muito comum nas organizações, que a métrica se transforme em um objetivo, ao invés de uma informação útil. O segredo para manter a medição útil é colocar a atenção em seu significado;
- Muitas organizações não têm uma clara noção de como analisar e interpretar os dados coletados, deixando de utilizar suas medições. É fundamental que a organização aloque um grupo de métricas com conhecimento estatístico, responsável pela análise dos dados coletados. Também é desejável a utilização de ferramentas para apoiar a análise. E ainda, os dados coletados e os resultados de suas análises devem ser armazenados em um Banco de Dados Histórico, o qual deverá ser utilizado para o controle da melhoria do processo;
- Na maioria das vezes, a coleta não pode ser completamente automatizada, o que pode viabilizar a coleta de dados irreais e sem utilidade. Estes dados podem levar a interpretações errôneas, más decisões e ações de melhoria inadequadas;
- As pessoas envolvidas precisam entender por que se está coletando dados e como os dados serão utilizados. É importante que as pessoas conheçam o propósito para o qual os dados estão sendo coletados e que os dados realmente sejam utilizados para o propósito informado. É importante que os profissionais entendam que: o sucesso do processo de medição depende das equipes para coletar dados e estes dados devem ser acurados; eles não serão avaliados com base nestes dados; e se eles reportarem dados “viciados” (dados que destorçam o real desempenho para este parecer melhor) será obtido um banco de dados baseado em dados distorcidos, e isto pode ser pior do que não ter nenhum dado.

O Grupo de Métricas especializado em medidas deve ser desvinculado dos projetos, subordinado à área de gerência de reuso e do projeto, podendo estar ligado ao grupo de definição de padrões (SEI, 2002). É necessário que todos tenham confiança que o uso dos dados não será feito contra as pessoas e para isso é preciso estabelecer um protocolo claro e público, com o compromisso da gerência, que tranqüilize a equipe e a faça colaborar com os coletores de dados (BRIAND et al., 1998). Os dados coletados do trabalho de uma pessoa só

podem ser usados em benefício desta pessoa. Os dados que forem negativos ao desempenho profissional de uma pessoa só poderão ser tratados em caráter sigiloso com esta pessoa. Os dados de erros que poderão ser divulgados são exclusivamente aqueles dados globais consolidados, índices de erros e valores de desperdício da equipe do desenvolvimento ou da organização como um todo. Uma forma de aumentar a segurança dos empregados é organizar a coleta de modo que a gerência só observe dados agregados, nunca os resultados associados a um só empregado (LEWIS, 2000).

Segundo Jones e a Norma ISO/IEC 14598-5, os procedimentos de um processo de medição de *software* são: (JONES, 1997; PAIVA et al., 2001)

1 - Obter apoio da alta administração;
2 - Definir objetivos da medição;
3 - Desenvolver um plano de implantação;
4 - Definir métodos de medição;
5 - Definir a coleta de dados de medição;
6 - Designar profissionais que vão atuar no processo de medição;
7 - Treinar os profissionais envolvidos no processo de medição;
8 - Conscientizar, motivar e treinar gerentes e equipes de projetos;
9 - Estabelecer uma referência para medição inicial;
10 - Estabelecer o processo de medição;
11 - Integrar com ciclo de vida e processos de gerência;
12 - Monitorar e modificar o processo de medição se necessário.

Jones classifica as informações de um processo de medição em três tipos (JONES, 1997): *hard* (quantificadas com pouco ou nenhuma subjetividade, possuindo alto grau de precisão), *soft* (avaliadas subjetivamente, possuindo baixo grau de precisão devido à variação de opinião e comportamento comum entre pessoas) e de padronização (relativas às medições padrões usadas por propósitos corporativos para determinar a posição dos projetos em relação aos padrões definidos como meta pela organização, em termos de produtividade ou qualidade).

De acordo com a finalidade de uso, as medições de *software* podem ser estratégicas ou táticas. As medições estratégicas são aquelas que englobam fatores que influenciam diretamente no sucesso da corporação, enquanto que as táticas, influenciam nos resultados dos projetos. Jones ainda relata que um programa completo de medições deve incluir os dois tipos de medições (JONES, 1997).

2.1.3.1 MEDIÇÕES TÁTICAS E *HARD* PARA PROJETOS OO

Os conceitos básicos de OO - encapsulamento, herança e polimorfismo - requerem métricas diferentes das tradicionais de produto - tamanho, complexidade, desempenho e qualidade (CHIDAMBER et al., 1994; HITZ, 1996; LI, 1993). Além disso, muitos dos modelos de qualidade e de métricas disponíveis para análise de *software* OO só podem ser aplicados após a fase de implementação do produto. Então, com o intuito de melhorar a redução do retrabalho durante e após a implementação pela detecção e remoção de não-conformidades, como também, melhorar a elaboração dos planos de teste e planejamento dos recursos, há uma necessidade por métricas e modelos que possam ser aplicados nas fases do desenvolvimento de *software*.

Bansiya e Davis apresentam um Modelo de Hierarquia para Avaliação da Qualidade de Projetos OO (BANSIYA et al., 2002). Neste modelo, as estruturas e os comportamentos das classes, objetos e seus relacionamentos são avaliados usando um conjunto de métricas de projeto OO. Este modelo relaciona as propriedades de projeto com os atributos de qualidade. Bansiya fornece uma ferramenta de *software*, QMOOD++, para avaliação de qualidade para vários projetos OO, que permite tratar a avaliação de projeto automaticamente (BANSIYA, 1997). Esta ferramenta facilita a coleta de dados de medições de componentes de projeto implementados em C++.

Baseado nos atributos da ISO 9126, Bansiya e Davis selecionam e identificam seis atributos de qualidade, que são apresentados na **TAB. 2.1.3.1-1** (BANSIYA et al., 2002).

TAB. 2.1.3.1-1: Seis atributos de qualidade de projetos OO.

Atributos de Qualidade	Descrição
1 - Compreensibilidade	Evidencia a presença de características que revelam a facilidade em aprender e compreender.
2 - Efetividade	Evidencia a presença de características para alcançar a funcionalidade e o comportamento desejados usando conceitos e técnicas de OO.
3 - Extensibilidade	Evidencia a presença e o uso de características que permitem a introdução de novos requisitos no projeto.
4 - Flexibilidade	Evidencia a presença de características que permitem a um projeto de ser adaptado para fornecer funcionalmente uma capacidade específica.
5 - Funcionalidade	Evidencia a presença de características de responsabilidades atribuídas às classes de um projeto que estão disponíveis através de suas interfaces públicas.
6 - Reusabilidade	Evidencia a presença de características de projetos OO que permitem ao projeto ser reaplicado para um novo problema sem esforço significativo.

As propriedades de projeto[8], tais como, abstração, encapsulamento, acoplamento, coesão, complexidade e tamanho de projeto, são freqüentemente usadas como características de qualidade de projeto tanto para desenvolvimento estrutural como OO. As mensagens, composição, herança, polimorfismo e hierarquias de classe são conceitos do paradigma OO fundamentais para a qualidade de um projeto OO. Este modelo inclui onze propriedades de projeto identificadas e definidas na **TAB. 2.1.3.1-2**, e a métrica correspondente usada por (BANSIYA et al., 2002).

TAB. 2.1.3.1-2: As métricas de projeto para avaliar as propriedades de projeto.

Propriedades de Projeto	Métricas de Projeto
1 - **Abstração:** uma medida dos aspectos de especialização / generalização do projeto.	**Número Médio de Antecessores:** esta métrica consiste em contabilizar o número médio das classes que herdam informação.
2 - **Acoplamento:** define a interdependência entre objetos em um projeto.	**Acoplamento Direto da Classe:** esta métrica consiste em contabilizar o número de classes que são diretamente relacionadas pelas declarações dos atributos e passagens de parâmetros dos métodos.
3 - **Coesão:** avalia o relacionamento dos métodos e atributos em uma classe.	**Coesão entre Métodos na Classe:** esta métrica calcula a relação entre os métodos de uma classe baseada por uma lista de parâmetros.
4 - **Complexidade:** uma medida do grau de dificuldade no entendimento e compreensão da estrutura interna e externa das classes e de seus relacionamentos.	**Número de Métodos:** esta métrica consiste em contabilizar todos os métodos definidos em uma classe.
5 - **Composição:** medidas de relacionamentos do tipo “parte de”, “tem”, “consiste de” ou “parte do todo”, que são os relacionamentos de agregações em um projeto OO.	**Medida de Agregação:** esta medida mede a extensão do relacionamento do tipo “parte do todo” realizado pelo uso dos atributos.
6 - **Encapsulamento:** refere-se às classes que previnem quanto o acesso às declarações de atributos protegendo as representações internas dos objetos.	**Métrica de Acesso aos Dados:** esta métrica é a taxa do número de atributos privado (protegido) pelo número total de atributos declarados na classe.
7 - **Herança:** uma medida de relacionamento “é uma” entre as classes. Este relacionamento é relativo ao nível das classes quanto a uma hierarquia de heranças.	**Medida de Abstração Funcional:** esta media consiste na taxa do número de métodos herdados por uma classe pelo número total dos métodos acessíveis por métodos de membro da classe.
8 - **Hierarquia:** hierarquia é usada para representar diferentes conceitos de especializações/ generalizações em um projeto.	**Número de Hierarquia:** esta métrica consiste em contabilizar o número de classes “mãe” no projeto.
9 - **Mensagem:** uma quantia do número de métodos públicos que estão disponíveis como serviços para outras classes.	**Tamanho da Interface da Classe:** esta métrica consiste em contabilizar o número de métodos públicos das classes.
10 - **Polimorfismo:** a capacidade para substituir objetos cujas interfaces conciliam com outras em tempo de execução.	**Número de Método Polimorfos:** esta métrica consiste em contabilizar os métodos que podem ter comportamento polimorfo.
11 - **Tamanho do Projeto:** uma medida do número de classes usadas em um projeto.	**Tamanho do Projeto em Classes:** esta métrica consiste em contabilizar o número total de classes no projeto.

[8] São conceitos quantificáveis que podem ser avaliados diretamente ao examinar a estrutura interna e externa, relacionamento e funcionalidade dos componentes de projeto, atributos, métodos e classes.

Baseado nas propriedades de projeto (**TAB. 2.1.3.1-2**) para os atributos de qualidade (**TAB. 2.1.3.1-1**), Bansiya e Davis elaboram as funções de medição para estes atributos (**TAB. 2.1.3.1-3**). Os índices das parcelas das funções de medição apresentadas na **TAB. 2.1.3.1-3** revelam a influencia positiva ou negativa relativa às propriedades de projeto nos atributos de qualidade de forma que os valores calculados de todos os atributos de qualidade tenham a mesma abrangência de 0 a $\pm$1 (BANSIYA et al., 2002).

TAB. 2.1.3.1-3: Parcelas das funções de medição de (BANSIYA et al., 2002).

Atributos de Qualidade	Índices das Parcelas das Funções de Medição dos Atributos de Qualidade										
	Abstração	Acoplamento	Coesão	Complexidade	Composição	Encapsulamento	Herança	Hierarquia	Mensagem	Polimorfismo	Tamanho do Projeto
Compreensibilidade	* (- 0.33)+	* (- 0.33)+	* 0.33 +	* (-0.33)+		* 0.33 +				* (- 0.33)+	* (- 0.33)+
Efetividade	* 0.2 +				* 0.2 +	* 0.2 +	* 0.2 +			* 0.2 +	
Extensibilidade	* 0.5 +	* (- 0.5)+					* 0.5 +			* 0.5 +	
Flexibilidade		* (- 0.25)+			* 0.5 +	* 0.25 +				* 0.5 +	
Funcionalidade			* 0.12 +					* 0.22 +	* 0.22 +	* 0.22 +	* 0.22 +
Reusabilidade		* (- 0.25)+	* 0.25 +						* 0.5 +		* 0.5 +

2.1.3.2 TÉCNICAS DE AVALIAÇÃO

2.1.3.2.1 GOAL/QUESTION/METRIC - GQM

A metodologia GQM - *Goal/Question/Metric* (Objetivo/Pergunta/Métrica) de Basili (BASILI et al., 1994) tem sido muito utilizada atualmente para definir um processo de medição que viabiliza o acompanhamento e a avaliação do processo de produção de *software*. Basili emprega esta metodologia para definir um processo de coleta e análise de dados (BASILI et al., 2000).

O primeiro passo de um programa de métricas é estabelecer uma referência, de forma que o progresso possa ser avaliado em termos dos objetivos e através de comparação com esta referência. Depois, executar a metodologia GQM da seguinte forma (ALMEIDA et al., 2003):

- Primeiro, selecionar alguns objetivos do processo que estejam em conformidade com os objetivos organizacionais e definir esses objetivos de modo que sejam quantificáveis e mensuráveis;

- Para cada objetivo, identificar perguntas para serem respondidas com a finalidade de coletar informação sobre a situação do objetivo, se está sendo alcançado ou não;
- Finalmente, identificar métricas que ajudem a responder cada pergunta. Algumas delas são diretas, simples itens de dados, por exemplo, o valor total gasto em reparo. Outras métricas são indiretas, calculadas a partir de um ou mais itens de dados.

Segundo MCGARRY (2001), o modelo GQM resolve o problema de definição de métricas, mas não de indicadores. Além disso, a aplicação do GQM pode gerar muitas métricas sem muita relevância para os responsáveis pelas decisões.

2.1.3.2.2 PRACTICAL SOFTWARE MEASUREMENT - PSM

O PSM (*Practical Software Measurement*) de MCGARRY et al. (2001) é um modelo para a estruturação da medição em um projeto de *software*. Este modelo é composto pelo Modelo de Informação de Medição e Modelo de Processo de Medição. O primeiro é um guia de como especificar formalmente os indicadores a serem usados, e o segundo é um guia de como conduzir o processo de medição.

O Modelo de Informação de Medição do PSM fornece um apoio para a definição da Necessidade de Informação (o *Goal* do GQM) e um guia para a partir desta chegar na especificação da métrica. Assim, este modelo serve como um guia para o planejamento e implementação das atividades de análise e coleta de dados.

O Modelo de Processo de Medição trabalha em conjunto com o Modelo de Informação de Medição para fornecer um *framework* para implementação de medição em um projeto. Este modelo foi automatizado pelo *software* PSM *Insight*[9]. O *software* fornece facilidades para a criação de indicadores, armazenamento de dados históricos e a geração de gráficos e relatórios para a análise.

2.2 LIDANDO COM REUSO

Existem várias abordagens para reuso (veja APÊNDICE 1), dentre elas as duas abordagens técnicas que mais tem se destacados nas pesquisas são: gerador de reuso e reuso baseado em partes. A técnica de gerador requer que um domínio de conhecimento seja

[9] Disponível gratuitamente no *site* do PSM: www.psmsc.com.

codificado em um gerador de aplicação ou uma linguagem de programação para que os componentes sejam selecionados e integrados automaticamente. O reuso baseado em partes, conhecido também como reuso de componentes, requer que um programador humano integre componentes de *software* em uma aplicação. Para atender o objetivo deste trabalho, o reuso de *software* abordado é o baseado em partes de acordo com um processo bem definido e repetível – reuso sistemático de *assets*. A abordagem de reuso baseado em partes é mais abrangente do que o gerador (IEEE1517, 1999), pois os artefatos ao serem construídos como *assets*, aumentam a reusabilidade[10].

O reuso de *software* pode ser aplicado a qualquer produto do ciclo de vida e não apenas a fragmentos de código-fonte. Desta forma, a equipe de desenvolvimento pode procurar reuso de documentos de requisitos, especificações de sistemas, estruturas de projetos e qualquer outro artefato (BARNES et al., 1991).

O fato de estar reutilizando artefatos tem todos os benefícios conhecidos da tecnologia de reutilização, sendo possível fazer uso do conhecimento e da experiência adquiridos em processos de projetos da empresa ou estabelecidos por outras organizações ou pesquisadores (FIORINI et al., 2000; HENNINGER, 1999; RISING, 1999).

Entretanto, a reutilização não é trivial, ela apresenta muitos problemas, os principais são:

- A localização e recuperação dos componentes de *software* a partir de uma grande coleção (GIORARDI, 1998);
- Ausência da prática de reuso nos processos de desenvolvimento de projetos das organizações (FICHMAN, 2001);
- O reuso requer mudança na forma de pensar (IEEE1517, 1999). Ao praticar reuso, espera-se que as equipes de desenvolvimento não se concentrarem em um único produto de *software*, mas em um grupo de produtos relacionados pelos requisitos, funções e aspectos comuns. Além disso, espera-se que os desenvolvedores construam *assets* para serem reusados sempre que ocorram os mesmos respectivos requisitos nas especificações[11] ou um subconjunto de requisitos que correspondam ao conjunto de requisitos do artefato a ser reutilizado. Portanto, isto requer freqüentemente uma generalização do *asset*, conformidade com padrões e uma estratégia de projeto para o produto de *software* (ALMEIDA et al., 2003);

Para o sucesso da prática do reuso de *software*, os seguintes critérios devem ser seguidos:

[10] Determina o grau em que um *asset* pode ser usado em mais de um sistema de software ou construir outros *assets*.

[11] Especificações podem ocorrer em vários níveis de abstração desde a especificação de requisitos de um sistema até a especificação de projeto detalhado de módulos ou classes.

- As organizações devem ter um programa de reuso que seja sistemático (FRAKES et al., 1994). O reuso deve estar explicitamente definido nos processos de ciclo de vida de *software* para assegurar o reuso repetidamente nos diversos produtos e projetos de *software* da organização (FICHMAN, 2001);
- As organizações devem estabelecer uma política de reuso: reuso a *priori* (artefato projetado para reuso) ou a *posteriori* (artefato transformado para reuso) (FICHMAN, 2001);
- As organizações devem ter um programa de reuso que utilize um esquema de classificação para catalogar componentes em uma base de *software*. Os esquemas devem especificar alguns atributos, chamados de facetas, para serem utilizados como descritores de um componente de *software*. Em geral, esses atributos devem ser relativos à ação que o componente realiza e aos objetos manipulados pelo componente. A classificação e a recuperação devem ser realizadas ao especificar termos para cada atributo no esquema. Os relacionamentos com termos ou frases devem ser através da combinação de atributos, o que possibilita melhor precisão nas buscas (GIORARDI, 1998);
- As organizações devem ter um programa de reuso onde os reutilizadores tenham conhecimento da existência dos *assets*, possam ser facilmente localizados e entendidos, e principalmente tenham suas qualidades asseguradas. Portanto, o reuso requer que o projeto de *software* seja examinado quanto à sua qualidade para maximizar oportunidades de reuso de implementação. (JACOBSON et al., 1997)

FICHMAN (2001) avaliou nos modelos de desenvolvimento de artefatos projetados para reuso como um dos mais promissores. A vantagem deste é que os desenvolvedores produzem artefatos reutilizáveis quando esses artefatos ainda estão recentes em suas mentes, e apenas uma única biblioteca de artefatos de alta qualidade precisa ser mantida e procurada. Entretanto, um esforço é despendido sem saber quando, ou se o artefato será efetivamente reusado. A vantagem do artefato transformado para reuso é ser um modelo "*just in time*" para a produção de componentes, que resulta na necessidade de pelo menos de duas bibliotecas: primária, contendo só artefato aprovado para ser reutilizável, e secundária para artefato não candidato ao reuso ou ao que ainda não foi transformado. A desvantagem deste último está no esforço extra para fazer artefatos reutilizáveis a partir de artefatos candidatos que o desenvolvedor não esteja familiarizado e na busca em mais de uma biblioteca por artefatos reusáveis ou artefatos candidatos a serem transformados para reuso. Também, não há nenhuma garantia que um desenvolvedor execute a busca do artefato ao invés de desenvolver novamente.

O reuso sistemático de *software* requer a inclusão de elementos fundamentais de controle de qualidade no processo de desenvolvimento de *asset*, tais como:

- Validação e verificação dos *assets*, ou seja, estes devem ser submetidos a um processo para assegurar que eles têm os atributos desejados (JACOBSON et al., 1997; FICHMAN, 2001);
- Medir progressos do reuso baseado em métricas relevantes que possam ajudar na otimização do programa de reuso sempre que necessário (FRAKESb et al., 1996);
- Incluir verificações de reuso nas revisões de projeto e inspeções (IEEE1517, 1999).

2.2.1 PROCESSO DE REUSO

O processo de reuso deve maximizar reuso e minimizar esforços básicos de desenvolvimento. Então, o primeiro passo no processo de reuso é encontrar e selecionar *assets* adequados. Para efetivo reuso deve ser mais fácil achar *assets* que os desenvolver para único uso (JACOBSON et al., 1997). Ao encontrar os *assets* adequados não significa ter achado exatamente o que precisa. Localizar *assets* semelhantes pode ser suficiente. Depois que os *assets* foram encontrados, eles devem ser entendidos para serem reusados. Achar e entender estão relacionados, porque para selecionar um *asset* para reuso, deve-se saber o que ele faz. Entender fica mais importante quando o *asset* tiver que ser modificado. Assim, a documentação adequada é fundamental para este passo. Naturalmente, restrições legais podem proibir a disponibilidade da implementação de um *asset* e podem permitir seu reuso somente como uma caixa preta. (IEEE1517, 1999)

Construir um sistema de *software* sem precisar modificar os *assets* é o ideal. Geralmente, pelo menos alguns dos *assets* devem ser adaptados às necessidades específicas do sistema de *software* a ser construído. Os *assets* podem ser modificados mudando características internas ou acrescentando novas características. Quando um *asset* oferece a funcionalidade exigida, ele deve ser incorporado no sistema de *software*. Porém, os *assets* existentes podem não ser suficiente para construir um novo sistema. Portanto, alguns *assets* devem ser construídos do zero para serem inseridos no repositório de *assets* reutilizáveis, facilitando o seu reuso em projetos futuros. (FICHMAN, 1999)

Desta forma, desenvolver para reuso implica em construir modelo de domínio e arquitetura, construir novos *assets*, ou reengenhar *assets* existentes para melhorar sua reusabilidade. Para apoiar desenvolvimento para reuso, devem ser acrescentadas ao ciclo de

vida do *software* as seguintes atividades (IEEE1517, 1999) (veja seção 7.3.3 do APÊNDICE 3):

- Executar análise de domínio;
- Executar projeto de domínio;
- Construir *assets* reutilizáveis.

Na **FIG. 2.2.1-1** é apresentado um resume das etapas do processo de reuso. O processo de reusar um *asset* existente requer as seguintes atividades: Busca, Entende e Adapta. A atividade Desenvolve é necessária quando nenhum *asset* reutilizável pode ser identificado. A atividade Integra deve ser feita mesmo que um *asset* existente seja reusado ou um novo tenha sido desenvolvido. Finalmente, o novo *asset* ou o *asset* modificado deve ser generalizado e classificado para ser armazenado no repositório de *assets* reutilizáveis para futuro reuso. (FRAKESb, 1996)

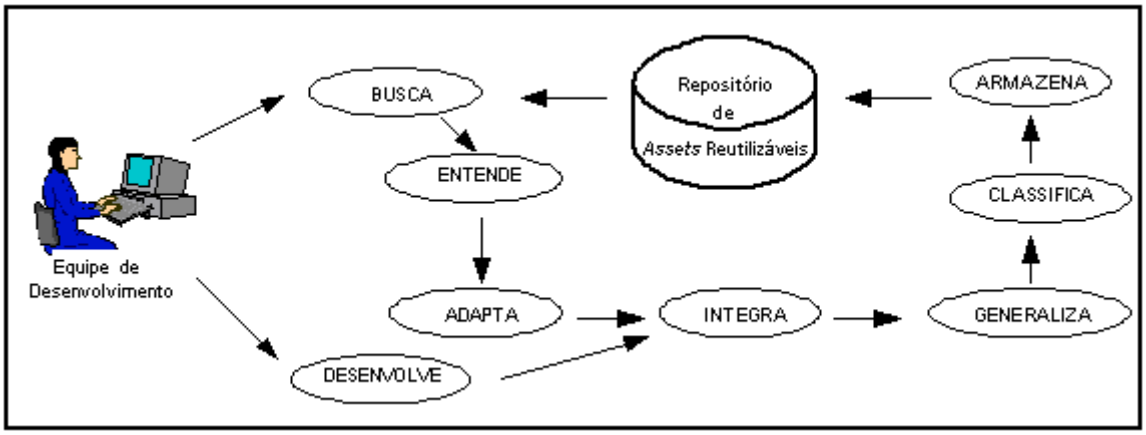


FIG. 2.2.1-1: Procedimentos para reuso de *software*.

É importante que a documentação seja considerada uma parte essencial do *asset*, pois sem sua própria documentação um *asset* é inútil. Nem pode ser recuperado quando for necessário e nem pode ser reusado e adaptado com esforço razoável. Desta forma, a documentação de cada *asset* deve ser clara, concisa e auto-suficiente para agilizar e viabilizar os seguintes aspectos: a avaliação de *assets* em um conjunto de possíveis candidatos, a compreensão da funcionalidade de um *asset*, o uso de um *asset* em um certo ambiente e a adaptação de um *asset* para necessidades específicas. (SAMETINGER, 1996)

2.2.2 MODELOS E MÉTRICAS

As métricas são necessárias para avaliar a robustez da arquitetura, em termos das mudanças de requisitos requererem mudança na arquitetura. Também são necessárias métricas para determinar a perda da reutilização em virtude de sucessivas manutenções. A perda de reutilização ocorre quando uma versão de artefato é particionada em várias versões coexistentes, em virtude da necessidade de cada uma dessas versões necessitar de atenção específica pela gerência de configuração e ao realizar novas alterações. (ALMEIDA et al., 2003)

A literatura destaca algumas métricas importantes (veja **TAB. 7.2-1** do APÊNDICE 2):

- **Métricas de processo:** produtividade do trabalhador, eficácia em termos de tempo, custo de produção, taxa de defeitos detectados, volume de mudanças de uma versão para outra;
- **Métricas de biblioteca ou repositório:** freqüência de acesso, tamanho do repositório, custo de certificação, qualidade e quantidade de componentes;
- **Métricas de arquitetura:** estabilidade;
- **Métricas do artefato:** tamanho do componente, complexidade da interface, coesão do sistema de componente e interface, nível de usabilidade do componente e interface, qualidade, custo dos sistemas padrões de componente, dificuldades em integrar componentes;
- **Métricas da aplicação:** volume de reuso, qualidade, volume de defeitos e suas causas, perfil da falta.

A quantidade de reuso está relacionada com o volume de reuso ou taxa de reuso (JACOBSON et al., 1997). A taxa de reuso é definida como a razão da soma dos tamanhos dos artefatos reutilizados dividido pelo tamanho total da aplicação. A taxa de reuso é usada para monitorar o reuso e, também, para estimar como o reuso vai incidir nas medidas de qualidade, tempo e custo.

Algumas organizações medem o tamanho do *software* utilizando os métodos recomendados pela SEI (PARK, 1992): LOC (linhas de código). Outras utilizam a técnica de Análise por Ponto de Função (APF) (IFPUG, 2000) ou Análise por Pontos de Caso de Uso[12] (KARNER, 1993). O tamanho dos artefatos não-códigos podem ser medidos em termos de linhas equivalentes de código ou pelo número de páginas de documentos.

As medidas de complexidade de *software*, determinada pela complexidade estrutural e tamanho, podem ser usadas para estimar densidade de defeito, probabilidade de erro e manutenibilidade. Já as medidas de coesão e acoplamento determinam o grau de aderência e independência de um sistema de componente ou que um componente tem dos outros. Estas medidas indicam o custo de integrar o componente e, também, de sua manutenibilidade (JACOBSON et al., 1997).

Frakes e Terry classificam as métricas e os modelos de reuso como (FRAKESb et al., 1996): Modelos de Custo / Benefício de Reuso, Modelo de Avaliação da Maturidade, Métricas de Quantidade de Reuso, Modelo de Fracasso, Métricas de Reusabilidade e Métrica de Bibliotecas de Reuso.

Os Modelos de Custo / Benefício de Reuso incluem a análise econômica de custo / benéfico assim como o custo da qualidade e produtividade. O Modelo de Avaliação da Maturidade categoriza programas de reuso pelo quanto à organização está avançada na implementação do reuso sistemático. As Métricas de Quantidade de Reuso são usadas para avaliar e monitorar um esforço de melhoria do reuso pelo acompanhamento do percentual de reuso de objetos do ciclo de vida. O Modelo de Fracasso é usado para identificar reuso e impedimentos de reuso em uma organização. As Métricas de Reusabilidade indicam a probabilidade que um artefato é reutilizável. As Métricas de Bibliotecas de Reuso são usadas para gerenciar e acompanhar o uso de um repositório de reuso. O critério de avaliação para indexar esquemas de bibliotecas de reuso é: custos, busca pela efetividade, suporte para entendimento e eficiência As organizações freqüentemente necessitam desses modelos e métrica nesta ordem apresentada por Frakes e Terry (FRAKESb et al., 1996).

A maioria dos modelos econômicos de engenharia de *software* precisa ser ajustada para incluir reuso, e customizada para cada reuso específico. Muitos autores têm modificado os modelos de custos, como COCOMO (BOEHM et al., 1988), que são usados para estimar tempo e esforço e para o desenvolvimento tanto de componentes quanto de aplicações usando componentes (POULIN, 1997).

Jacobson et al. (JACOBSON et al., 1997) apresentam como usar um modelo de esforço e custo de reuso simplificado, essencialmente reformulando a análise de Gaffney e Durek (GAFFNEY et al., 1989) e aplicando as recomendações de Poulin (POULIN, 1997). Ele também adiciona um fator de utilização de biblioteca, pois em uma biblioteca de sistemas de

---

[12] A Rational desenvolveu uma Análise por Pontos de Caso de Uso similar à Análise por Pontos de Função, que pode ser utilizada para estimar o tamanho de uma aplicação baseada no número e complexidade de atores e casos de uso.

componentes que contém vários sistemas de componentes, é esperado que ocorra mais reuso para os sistemas de aplicação. Jacobson ainda relata que alguns componentes podem ser mais reusados do que outros.

O APÊNDICE 2 apresenta uma tabela resumida das principais métricas e modelos apontados pela literatura, utilizando a classificação de Frakes e Terry (FRAKESb et al., 1996).

3 PROCESSO DE GARANTIA DA QUALIDADE DE *SOFTWARE* BASEADO EM REUSO SISTEMÁTICO DE *ASSETS*

Com o objetivo de garantir a qualidade dos artefatos gerados para compor os *assets* no ciclo de vida do desenvolvimento de projeto baseado em reuso apresentado por (GURGEL, 2004), este trabalho apresenta uma proposta de um Processo de Garantia de Qualidade de *Software* (GQS). Este processo estabelece um *framework* para auxiliar no desenvolvimento do ciclo de vida do *software* baseado em reuso sistemático de *assets* com o explícito propósito de contribuir para a qualidade e produtividade do projeto de *software* dentro do contexto do Ministério da Defesa e de seus Comandos Subordinados (**FIG. 3-1**).

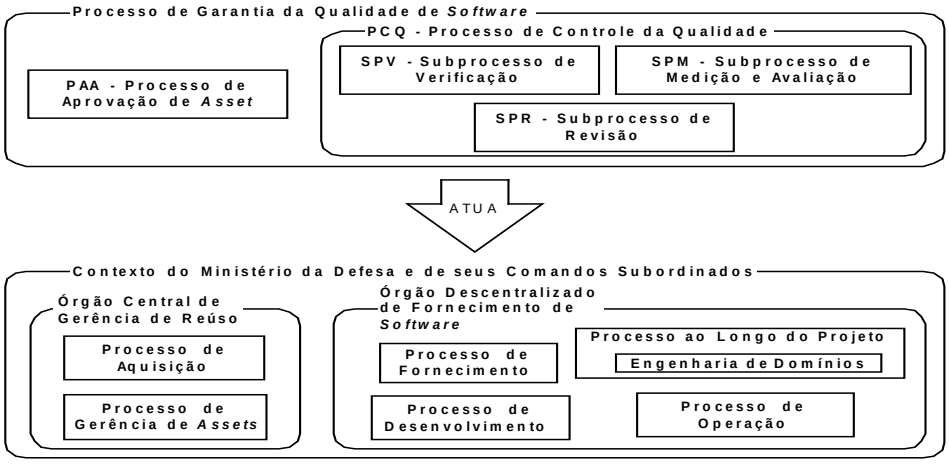


FIG. 3-1: Escopo do Processo de Garantia da Qualidade de *Software*.

Gurgel descreve sobre o contexto do Ministério da Defesa desde sua criação em 1999 à formação do Departamento de Ciência e Tecnologia nas Forças Armadas e, também, apresenta a atual estrutura dos Comandos da Marinha, Exército e Aeronáutica (GURGEL,

2004). O Ministério da Defesa é uma instituição de grande porte, hierarquizada e com núcleos de desenvolvimento descentralizados conforme apresentado em (GURGEL, 2004).

O modelo de produção de reuso utilizado por (GURGEL, 2004) para reuso sistemático de *software* para o contexto do MD consiste na combinação da análise de domínio *pre-project*[13] e da generalização em *in-project*[14] (FICHMAN, 2001), de forma a identificar artefatos reutilizáveis no domínio do projeto antes do desenvolvimento do projeto e sempre gerar produtos de *software* genéricos (o mais possível) e reutilizáveis. A proposta de (GURGEL, 2004) está apoiada na arquitetura da Norma ISO/IEC 12207 (ABNT12207, 1998) que é estendida pela Norma IEEE 1517 (1999) para atender um modelo de ciclo de vida de *software* com reusabilidade (veja ANEXO 1). Esta arquitetura foi empregada por utilizar uma terminologia bem definida composta de processos, atividades e tarefas para Aquisição, Fornecimento, Desenvolvimento, Operação, Engenharia de Domínio e Gerência de *Assets*.

O trabalho, aqui proposto, consiste em estabelecer um embrião de um processo estável de garantia da qualidade do *software* dentro desta nova abordagem – reusabilidade, aplicado à estrutura organizacional do Ministério da Defesa e ao Processo de Desenvolvimento de *Software* baseado em Reuso para MD de (GURGEL, 2004). Este trabalho está baseado nas definições de padrões de processos mais utilizados atualmente pelas industrias e organizações, tais como, CMM, CMMI, ISO 9000, ISO 12207 e ISO 15504, além da Norma IEE1517. Os processos estabelecidos neste trabalho abrangem diferentes níveis de maturidade do CMM e CMMI. Além disso, este trabalho se destina a propor um modelo de gestão de qualidade baseado na metodologia PDCA - *Plan/Do/Check/Action* - Ciclo de Deming (DEMING, 1982), que inclui técnicas e procedimentos que devem ser incorporados por todas as pessoas dos Órgãos de Fornecimento de *Software*.

De acordo com o Nível 3 (Definido) do CMM (PAULK et al., 1993) e o processo de apoio da Norma 12207 (ABNT12207, 1998), o Processo de Garantia da Qualidade de *Software* (GQS) proposto estabelece pontos de revisão e *checklists* nas diversas fases do Processo de Desenvolvimento de *Software* baseado em Reuso para MD de (GURGEL, 2004), através de subprocessos de Revisão (SPR) e Verificação (SPV) respectivamente (veja **FIG. 3-1**).

[13] Neste modelo, tenta-se identificar abstrações e desenvolver artefatos reutilizávéis com grande potencial de aplicação para uma família de projetos que residem no mesmo domínio. Em geral, estes artefatos são reutilizados em futuros projetos como "*as is*" ou podem ser especializados.

[14] Generalização durante o projeto.

Baseado no Processo de Apoio da Norma 12207 (ABNT12207, 1998) e no Nível 3 (Definido) do Modelo CMM, o processo proposto consiste em prover o gerenciamento, com a adequada visibilidade, do processo que está sendo utilizado pelo projeto de desenvolvimento de *software* e dos artefatos que estão sendo construídos. Este processo inspeciona os produtos de *software* e atividades para verificar se estão cumprindo os procedimentos e padrões aplicáveis. O objetivo é fornecer os resultados dessas revisões ao Gerente de Reuso e à equipe de desenvolvimento/montadores.

Além disso, com o objetivo de desenvolver e sustentar uma capacidade de medição usada para suportar gerencialmente as necessidades de informação, no processo proposto é definido um subprocesso de Medição e Avaliação (SPM) baseado na área de processo de Medição e Análise do Nível 2 (Gerenciado) do modelo CMMI (SEI, 2002). Esta área envolve os seguintes aspectos:

- Especificação dos objetivos de medição e análise de forma que estes sejam alinhados com os objetivos e as necessidades de informação identificadas;
- Especificação das medidas, mecanismos de coleta de dados e de armazenamento, técnicas de análise, e mecanismos de comunicação e de *feedback*;
- Implementação da coleta, armazenamento, análise e comunicação dos dados;
- Fornecimento de resultados objetivos que podem ser usados na tomada de decisão e implementação de ações corretivas adequadas.

Neste trabalho, um processo de Controle da Qualidade (PCQ) é especificado ao estabelecer os subprocessos de Verificação (SPV), Revisão (SPR), e Medição e Avaliação (SPM).

Para garantir que os *assets* definidos em (GURGEL, 2004) possam ser classificados e gerenciados para reuso, o processo proposto deve validar os artefatos que compõem os *assets*. Para isto, um processo de Aprovação de *Asset* (PAA) é definido, e este deve estar em sintonia com o processo de Controle da Qualidade (IEEE1517, 1999).

Com o objetivo de desenvolver com e para reuso de *assets* e de produzir *assets* de qualidade, a gerência de reuso deve implantar e utilizar os processos abordados neste capítulo. É fundamental que todas as pessoas dos Órgãos envolvidos com o desenvolvimento de *software* tenham incorporado as práticas deste processo em suas rotinas.

3.1 ESTRUTURA ORGANIZACIONAL DO GRUPO DE GARANTIA DA QUALIDADE DE *SOFTWARE*

Como o Ministério da Defesa é uma instituição centralizadora e hierárquica, que possui Comandos Subordinados, e estes comandos, por sua vez, contém vários órgãos de desenvolvimentos com diferentes áreas de atuação, este trabalho propõe, de acordo com a abordagem estabelecida em (GURGEL, 2004), uma estrutura organizacional para o Grupo da Garantia da Qualidade de *Software* (GQS).

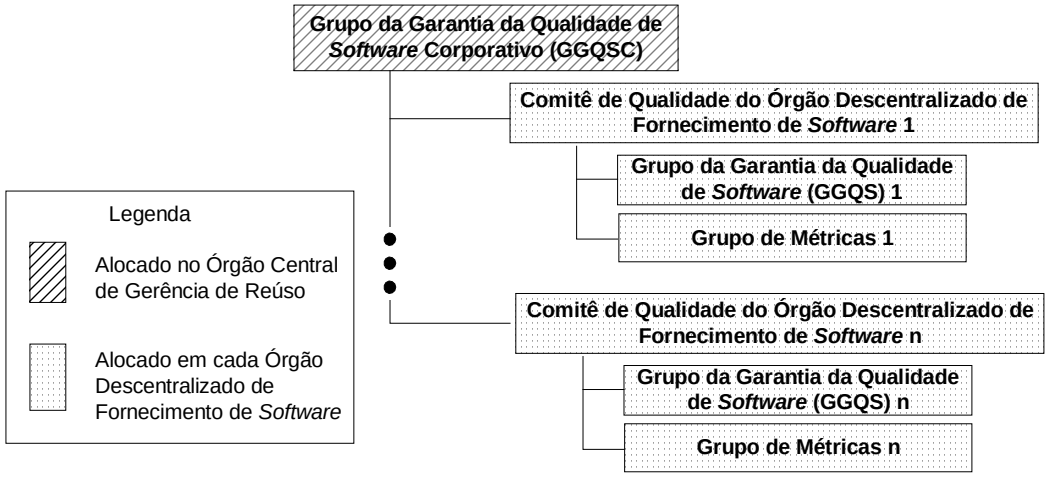


FIG. 3.1-1: Estrutura da área do Grupo da Garantia da Qualidade de *Software* (GGQS).

Na **FIG.3.1-1** está representada uma organização estrutural do Grupo da GQS (Garantia da Qualidade de *Software*), onde um Grupo de GQS Corporativo deve ser centralizador para ser "os olhos e ouvidos da Gerência de Reuso", e a este grupo vários Comitês de Qualidade devem estar subordinados. O objetivo destes comitês é garantir a reusabilidade e qualidade no desenvolvimento de *software* nas diversas organizações fornecedoras. Para cada comitê devem existir dois grupos subordinados: um Grupo de GQS e um Grupo de Métricas. O Grupo de GQS consiste na verificação e inspeção de artefatos/*assets* produzidos no processo de desenvolvimento de *software*, ao passo que o Grupo de Métricas deve ser responsável por medir e analisar os dados coletados e armazenados para avaliação do processo de desenvolvimento e do produto de *software*.

Esta organização estrutural aplicada ao contexto do Ministério da Defesa é apresentada na **FIG. 3.1-2** onde um Grupo de GQS Corporativo deve estar ligado ao Ministério da Defesa via Órgão Central de Gerência de Reuso, Departamento de Ciência e Tecnologia. Cada Órgão

Descentralizado de Fornecimento de *Software* deve ter seu Comitê de Qualidade formado por Consultores da Qualidade, e um Grupo de GQS e um Grupo de Métricas devem estar subordinados a este comitê.

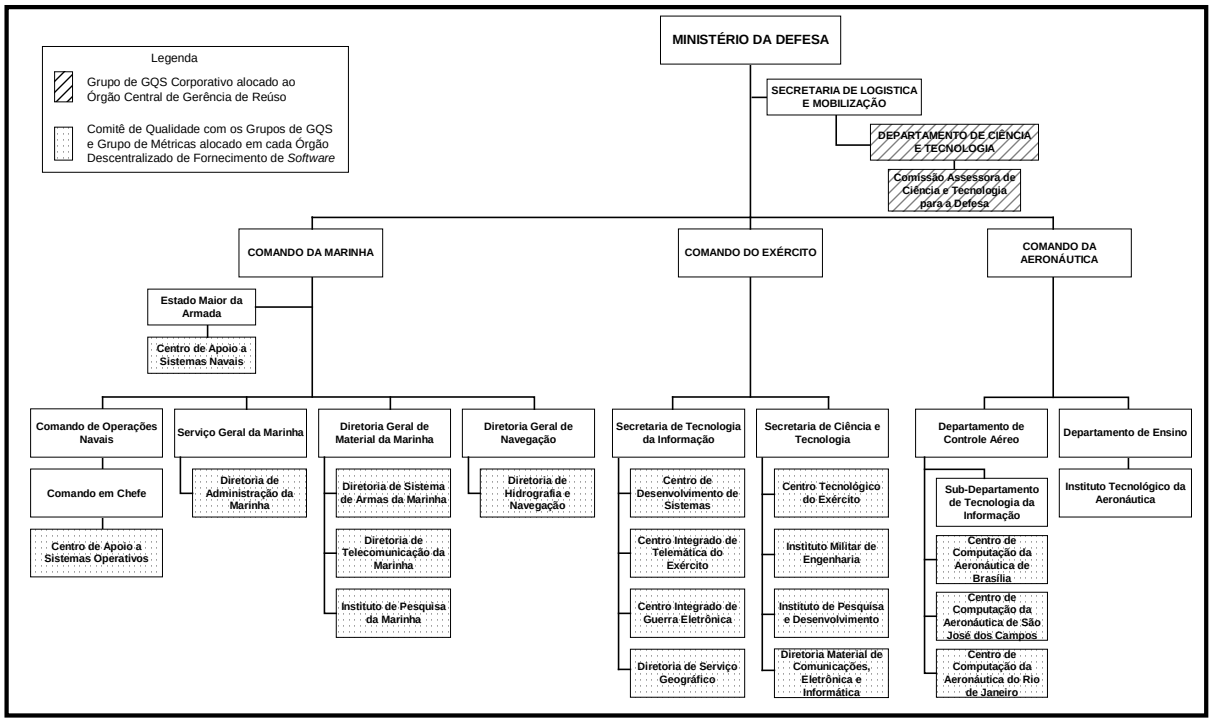


FIG. 3.1-2: Grupo da Garantia da Qualidade de *Software* (GGQS) no contexto do MD.

Os Consultores da Qualidade não devem fazer parte da equipe de desenvolvedores/ montadores. Já o Grupo de GQS deve ser formado por desenvolvedores/montadores selecionados pelo Comitê de Qualidade para exercerem o papel de verificadores/revisores de artefato, desde que não tenham vínculo operativo com o artefato em questão, cujo propósito é de garantir que os processos de Engenharia de Domínio e de Desenvolvimento estejam sendo seguidos (atividades, tarefas e artefatos).

Todos os Comitês de Qualidade dos Órgãos Descentralizados de Fornecimento de *Software* devem se reportar ao Grupo da GQS Corporativo de forma que este possa obter subsídios para verificar as necessidades e oportunidades para realizar a melhoria do processo de toda a corporação quanto ao processo de desenvolvimento de *software*.

De acordo com o Nível 4 (Gerenciado) do CMM (PAULK et al., 1993) e o Nível 2 (Gerenciado) do CMMI (SEI, 2002), um Grupo de Métricas deve ter especialistas em estatísticas para fornecer suporte aos projetos da corporação com finalidade de analisar os dados consolidados dos Órgãos Descentralizados de Fornecimento de *Software*. As

ferramentas estatísticas, como Gráficos de Controle, Histogramas e Pareto, devem ser usadas para análise e melhoria do processo.

É importante ressaltar que o Comitê da Qualidade não pode ser visto como fiscalizador. O papel do consultor ou facilitador da qualidade é apoiar as equipes de desenvolvimento para melhoria da qualidade de seus projetos. No entanto, não se pode fugir do papel de fiscalizador, caso a equipe seja resistente a trabalhar com base no processo. Todos os resultados devem ser reportados para a Gerência de Reuso, pois esta precisa ter visibilidade do que está acontecendo. Pode ser que o processo não esteja adequado para a organização. Para tirar o papel de fiscalização da qualidade, os dados são reportados sempre de forma consolidada, para não se enfatizar um culpado.

3.2 GARANTINDO A QUALIDADE

O processo de Garantia da Qualidade de *Software* (GQS) tem como propósito estabelecer um conjunto de diretrizes para o planejamento e a execução de um processo de avaliação de um produto de *software* ao definir as atividades que devem ser executadas.

O processo de Garantia da Qualidade de *Software* (GQS) se inicia juntamente com o Plano de Aquisição[15] pelo Grupo de Garantia da Qualidade de *Software* Corporativo através do artefato Necessidades do Cliente e Glossário[16] fornecido pelo Cliente no Processo de Aquisição (veja **FIG. 3.2-1**). A partir deste artefato, o Grupo de GQS Corporativo deve criar e documentar um Manual da Qualidade de Projeto, onde devem ser especificados os objetivos para assegurar a qualidade dos *assets* e processos utilizados para desenvolver os produtos de *software* (artefatos) que vão compor estes *assets*, além dos objetivos para garantir a satisfação do cliente. Neste manual também deve estar especificado o escopo da avaliação (das verificações, revisões e medições a serem realizadas no produto), os métodos utilizados e as responsabilidades de todos envolvidos no processo de avaliação.

Na **FIG. 3.2-1** são apresentadas as interações do Processo de Garantia da Qualidade de Software (PGQS) com os processos de (GURGEL, 2004), especificando os artefatos de entrada/saída com seus respectivos papéis e os responsáveis pelo PGQS.

[15] Artefato da atividade Preparação do Plano de Aquisição do Processo de Aquisição do Processo de Desenvolvimento de *Software* baseado em Reuso para MD de (GURGEL, 2004).

[16] O artefato Necessidades do Cliente e Glossário defini o domínio do problema, funcionalidades e características desejadas, e glossário de termos do domínio da aplicação.

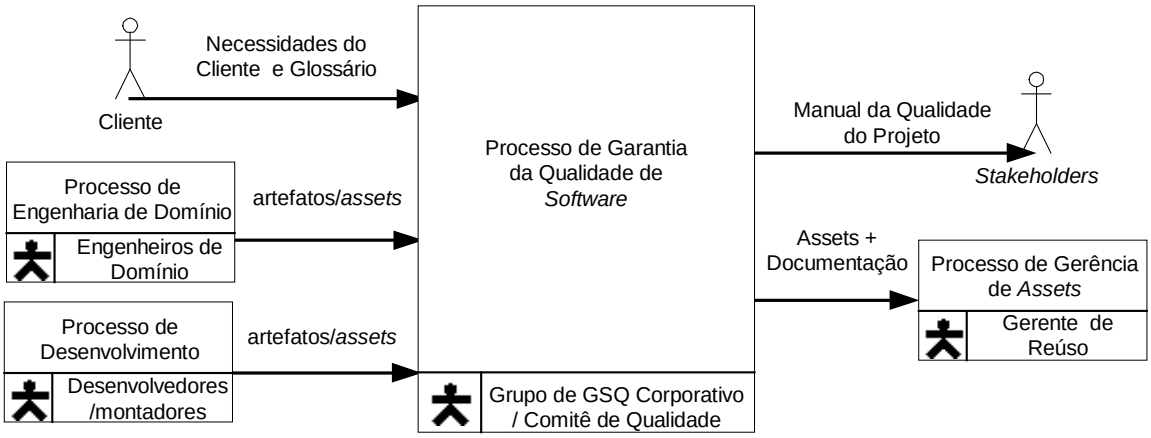


FIG. 3.2-1: Processo de Garantia da Qualidade de *Software*.

O Grupo de GQS Corporativo envia este manual junto com o Pedido da Proposta[13] ao Comitê da Qualidade do Órgão de Fornecimento de *Software* interno ou externo à organização para que este comitê possa estabelecer o Grupo de Garantia de Qualidade de *Software*, e definir um Plano e Cronograma de Garantia da Qualidade para o projeto em questão. Tanto o plano como o cronograma devem ser integrados pelo Plano de Gerência de Projeto[17].

Para cada referência estabelecida pelo Plano de Gerência de Projeto, este processo deve especificar os itens a serem verificados, revisados e armazenados de acordo com o Manual da Qualidade do Projeto e o Plano de Garantia da Qualidade do *Software* do Projeto (PGQSP). Este plano deve ser implementado e mantido durante a vigência do Contrato do Projeto[18] (ABNT12207, 1998; ABNT9000, 2001), e deve estabelecer os seguintes itens:

- **Responsável pelas atividades de GQS no projeto:** identifica o nome do consultor de GQS alocado ao projeto pelo Comitê de Qualidade;
- **Padrões e procedimentos para as atividades de GQS:** o consultor de GQS deve utilizar, na execução das atividades de GQS no projeto de *software* e na realimentação acerca destas atividades, os padrões e procedimentos estabelecidos no Plano de Gerência do Projeto versão <X.X>. As responsabilidades e autoridade do grupo de GQS devem estar definidas no Documento de Implementação de GQS do Órgão Militar;

[17] Este artefato é produto da atividade Preparação do Plano de Gerência de Projeto do Processo de Fornecimento proposto por (GURGEL, 2004). (Veja ANEXO 1).
[18] Este artefato é produto do Processo de Aquisição proposto por (GURGEL, 2004). (Veja ANEXO 1).

- **Padrões e procedimentos de projeto:** informa os padrões e procedimentos que devem ser seguidos pelo projeto de *software* a ser verificado / revisado pelo verificador / revisor no Plano de Gerência do Projeto;
- **Ferramentas:** informa a ferramenta, caso haja, que deve ser utilizada para registro e acompanhamento das atividades de GQS. Esta ferramenta deve fornecer o apoio necessário para a documentação e o rastreamento dos itens de não-conformidade até sua conclusão;
- **Treinamento:** necessidade de algum treinamento específico em GQS requerido por alguma circunstância no contexto do projeto. Esse item é opcional;
- **Histórico de versões:** informa dados de histórico das versões anteriores e da atual do artefato, tais como data da versão, número seqüencial da versão, breve descrição do motivo da versão, autor responsável pela versão, o nome do revisor e por quem foi aprovado;
- **Cronograma de verificações / revisões:** informa o cronograma para as verificações / revisões de GQS no projeto e o esforço em homens-hora empregado pelo consultor de GQS e pela equipe de desenvolvimento que participam dessas atividades.

As não-conformidades às necessidades do cliente detectadas no Processo de Controle da Qualidade devem ser registradas nos Relatórios de Verificação ou Revisão para análise e avaliação. Além disso, após a elaboração de cada relatório, um questionário de solicitação de sugestões de melhoria da qualidade deve ser respondido e comentado para capturar e aplicar as lições aprendidas no controle da qualidade de forma a contribuir para melhorar este processo. O preenchimento deste questionário pelo executor da verificação / revisão não deve ultrapassar a quinze minutos.

Na entrega do Projeto de *Software* ao Cliente, o Manual da Qualidade do Projeto deve conter todos os artefatos produzidos para avaliação e os resultados obtidos, os problemas e a resolução destes problemas que surgiram durante todo o ciclo de vida de desenvolvimento do Projeto, inclusive o Plano e Cronograma de Garantia da Qualidade.

É importante ressaltar que todos os artefatos do Processo de Garantia da Qualidade de *Software* devem estar disponíveis a toda equipe de desenvolvimento/montadores a qualquer momento para que, assim, estes artefatos possam contribuir como suporte na elaboração dos produtos de *software* das atividades dos processos de Engenharia de Domínio e Desenvolvimento de (GURGEL, 2004). Além disso, os registros das atividades e das tarefas de Garantia da Qualidade, dos problemas detectados e da avaliação dos problemas devem ser mantidos e disponibilizados a todos interessados (*stakeholders*).

Nas seções seguintes, através da técnica de modelagem SADT (*Structure Analysis and Design Technique*) desenvolvida por Douglas T. Ross (ROSS, 1984) e SOFTech, são definidos por meio dos actigramas o Processo de Controle de Qualidade (PCQ) ao Nível Técnico e seus respectivos subprocessos (veja **FIG. 3-1**), e o Processo de Aprovação de *Assets* (PAA).

3.2.1 PROCESSO DE CONTROLE DE QUALIDADE - PCQ

O Processo de Controle de Qualidade consiste na identificação de problemas com relação aos requisitos de qualidade estabelecidos ao longo de todo o desenvolvimento do *software*, ou seja, na especificação das necessidades e requisitos, modelagem conceitual, arquitetura de domínio, projeto lógico, implementação, testes e implantação. Desta forma, este processo deve avaliar os artefatos produzidos tanto nos níveis de gerenciamento do projeto como nos níveis técnicos durante a vigência do Contrato do Projeto (ABNT12207, 1998).

3.2.1.1 NÍVEL DE GERENCIAMENTO

O Processo de Controle de Qualidade (PCQ) ao Nível de Gerenciamento consiste em exercer avaliações gerenciais. No último dia útil do mês, o Gerente do Projeto deve realizar estas avaliações, verificando o andamento e situação do projeto conforme cronograma estabelecido no Plano de Gerência do Projeto e o gerenciamento dos recursos e treinamentos necessários para a equipe de desenvolvimento/ montadores do Projeto de *Software*. Os dados destas avaliações devem ser informados formalmente ao Subprocesso de Medição e Avaliação por meio de um relatório de avaliação do acompanhamento juntamente com o plano de trabalho atualizado – cronograma (veja **FIG. 3.2.1.1-1**). De posse deste relatório e do cronograma atualizado, o Grupo de Métricas deve alimentar sua base de dados gerencias sobre projetos para obtenção de indicadores de produtividade do processo de desenvolvimento do *software*.

Na **FIG. 3.2.1.1-1** são apresentas as interações do Processo de Controle da Qualidade (PCQ) ao Nível de Gerenciamento com os processos de (GURGEL, 2004), apresentando os artefatos de entrada/saída com seus respectivos papéis e os responsáveis pelo PCQ.

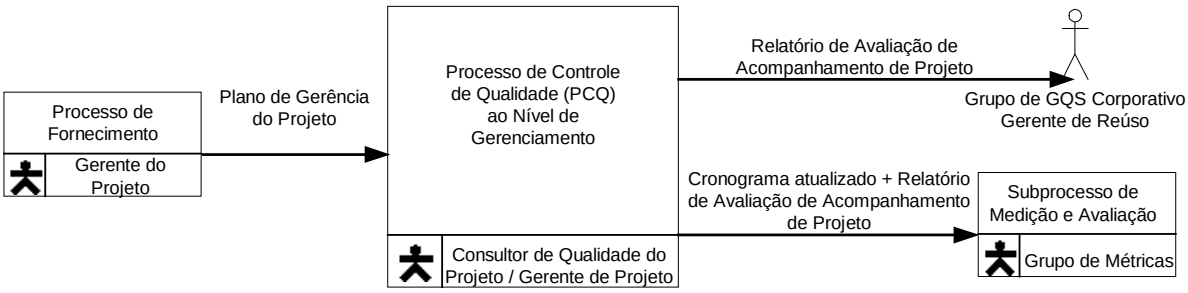


FIG. 3.2.1.1-1: Processo de Controle de Qualidade ao Nível de Gerenciamento.

A atividade do Processo de Controle de Qualidade ao Nível de Gerenciamento é a Avaliação de Gestão dos artefatos e recursos. Esta atividade deve ser realizada em reunião pelo Consultor de Qualidade do Projeto, exercendo o papel de revisor, e o Gerente de Projeto, como autor. A seqüência de tarefas desta atividade consiste em:

1 - Avaliar a situação do projeto em relação ao Plano de Gerência de Projeto e Cronograma;

2 - Registrar as situações de riscos e problemas detectados na tarefa anterior;

3 - Definir um esquema para categorizar e priorizar os problemas. Cada problema deve ser classificado por categoria e prioridade para facilitar a análise à resolução do problema;

4 - Através da análise de Pareto (ARTHUR, 1993), os problemas devem ser classificados por categoria em uma tabela ou utilizando o Diagrama de Pareto (PALMER, 1997) - quadro de barras de categorias de freqüência de ocorrência em cada categoria, com as categorias ordenadas da maior para a menor freqüência;

5 - Para cada problema, identificar as causas e analisá-las;

6 - Para cada problema, o registro do problema, a resolução do problema encontrada e sua aplicação devem ser documentadas no relatório de acompanhamento, e este enviado ao Grupo de GQS Corporativo, Gerente de Reuso e Grupo de Métricas. Ao Subprocesso de Medição e Avaliação, também é enviado o cronograma atualizado para que o Grupo de Métricas, juntamente com o relatório de acompanhamento, possa alimentar sua base de dados gerenciais para obtenção de indicadores de produtividade de processo de desenvolvimento de *software*.

O resultado desta atividade deve estar acordado pelos revisor e autor de forma a:

- Fazer com que as atividades de elaboração do artefato progridam de acordo com o Plano de Gerência do Projeto, baseadas em uma avaliação da situação do artefato;
- Manter o controle geral do projeto através da alocação adequada de recurso;

- Redirecionar o projeto ou determinar a necessidade de um planejamento alternativo;
- Avaliar e gerenciar situações de risco que possam comprometer o sucesso do projeto.

3.2.1.2 NÍVEL TÉCNICO

O Processo de Controle de Qualidade (PCQ) ao Nível Técnico consiste em exercer avaliações técnicas dos artefatos/*assets* produzidos nos processos de Engenharia de Domínio e de Desenvolvimento, assim como, do documento Necessidades do cliente e Glossário do processo de Aquisição do Processo de Desenvolvimento de *Software* baseado em Reuso para MD (GURGEL, 2004) (veja **FIG. 3.2.1.2-1**).

Na **FIG. 3.2.1.2-1** são apresentadas as interações do Processo de Controle da Qualidade (PCQ) ao Nível Técnico com os processos de (GURGEL, 2004), apresentando os artefatos de entrada/saída com seus respectivos papéis e os responsáveis pelo PCQ. Este processo abrange os subprocessos de Verificação (SPV), Revisão (SPR), e Medição e Avaliação (SPM), que serão detalhados nas próximas subseções.

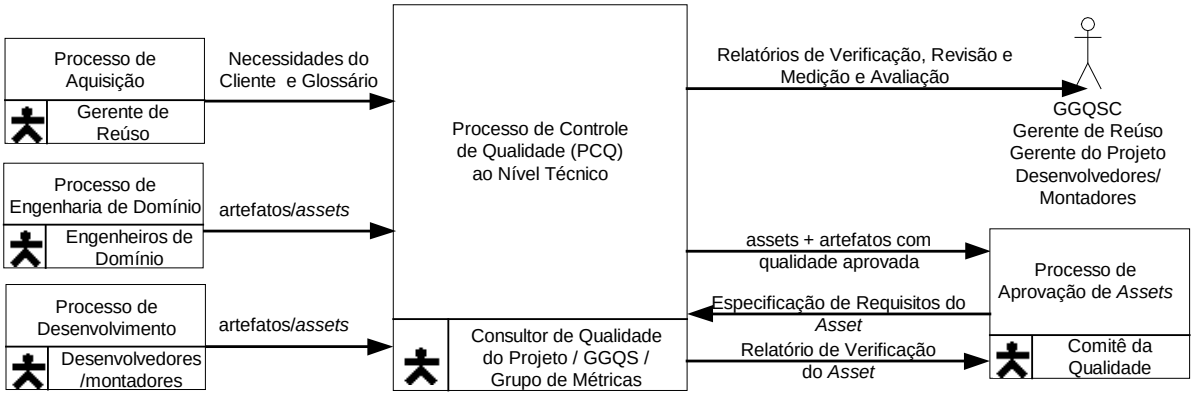


FIG. 3.2.1.2-1: Processo de Controle de Qualidade ao Nível Técnico.

Este processo inicia com o recebimento de um artefato prontificado pelos engenheiros de domínio ou pela equipe de desenvolvimento/montadores para ser inspecionado ou verificado, conforme o PGQSP e cronograma do projeto. No final de cada semana, o Gerente do Projeto deve receber do Comitê de Qualidade os relatórios das avaliações técnicas realizadas (Relatórios de Verificação e Revisão) na semana. Estes relatórios devem conter todas as não-conformidades detectadas e devem ser produzidos pelo Grupo de GQS e Consultor de Qualidade alocados ao projeto, conforme o PGQSP.

A cada nova versão do projeto, conforme estipulado no Plano de Gerência do Projeto, o Grupo de Métricas deve medir e analisar os dados coletados nos Subprocessos de Verificação e Revisão, e deve enviar um Relatório de Medição e Avaliação, juntamente com todos os Relatórios de Verificações e Revisões gerados no período de uma versão para a seguinte, aos Gerentes de Reuso e do Projeto uma semana após da prontificação da versão.

O Gerente de Projeto juntamente com o Consultor de Qualidade devem determinar quando e quais não-conformidades devem ser resolvidas pela equipe de desenvolvimento / engenheiros de domínio. Após a cada alteração ou quando uma nova referência for estabelecida, é necessário repetir o Controle de Qualidade.

Na **FIG. 3.1.2.2-2** é apresentada a expansão dos processos e subprocessos que compõem o Processo de Garantia da Qualidade do *Software*, mostrando todas as interações com os processos do Órgão Central de Gerência de Reuso e do Órgão Descentralizado de Fornecimento de *Software* de (GURGEL, 2004), além do *feedback* às equipes de desenvolvimento/montadores e gerência.

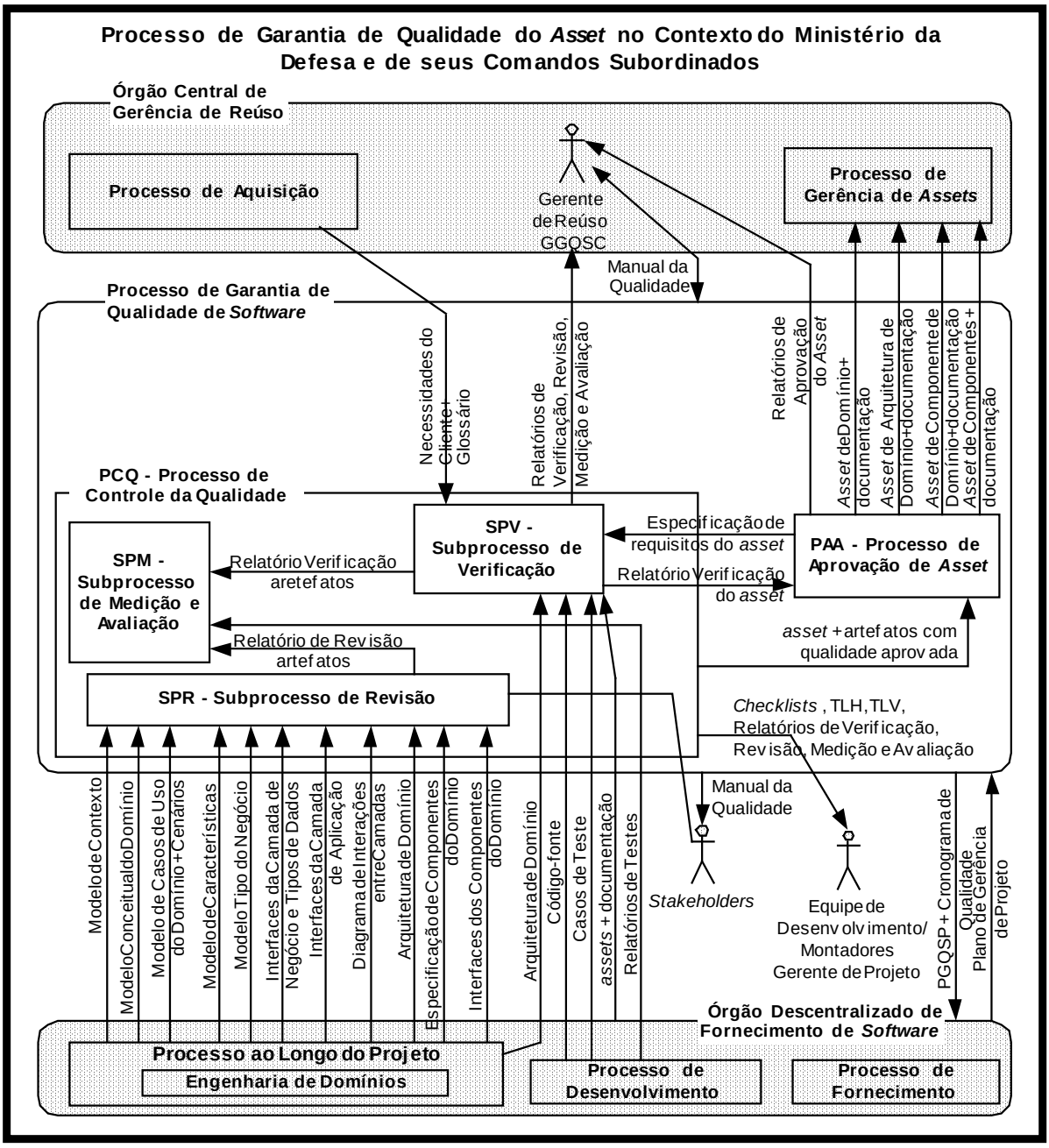


FIG. 3.2.1.2-2: Ampliação do Processo de Garantia da Qualidade do *Software*.

O Processo de Controle de Qualidade (PCQ), através das verificações do Subprocesso de Verificação, deve garantir que os artefatos Necessidades do Cliente e Glossário, a Arquitetura do Domínio, os Códigos-Fonte e os Testes de Qualificação do Desenvolvimento sejam aceitáveis pelo cliente e adequados quanto aos padrões, rastreabilidade, corretude, completude, recursos e escalabilidade. Além disso, através do Subprocesso de Revisão, o PCQ deve assegura que os artefatos e suas respectivas documentações produzidos pela Engenharia de Domínio estejam de acordo com o as necessidades e características desejadas

especificadas no documento Necessidades do Cliente e Glossário. Pelo Subprocesso de Medição e Avaliação, o PCQ deve garantir que as medições do processo e produto de *software* estejam de acordo com os procedimentos estabelecidos de reusabilidade e satisfação do cliente.

No caso de subcontratação, este processo deve garantir que os artefatos/*assets* da subcontratada satisfaçam as necessidades do Contrato original, repassando as necessidades e características desejáveis aplicáveis e executando verificações e revisões durante as fases de desenvolvimento de cada referência fornecida (ABNT12207, 1998).

3.2.1.2.1 SUBPROCESSO DE VERIFICAÇÃO – SPV

O Subprocesso de Verificação tem o propósito de definir as atividades para verificar artefatos/*assets*, via a Técnica de *Checklist* baseado em Aspecto, a fim de determinar sua conformidade com os requisitos e padrões estabelecidos para o projeto (veja **FIG. 3.2.1.2-2**). Os artefatos e os *assets* com suas respectivas documentações verificados são os seguintes produtos do Processo de Desenvolvimento baseado em Reuso para MD de (GURGEL,2004): Necessidades do Cliente e Glossário, Arquitetura de Domínio, Código-fonte, Casos de Testes de Qualificação, *Asset* de Domínio, *Asset* de Arquitetura de Domínio, *Asset* de Componente de Domínio e *Asset* de Componentes.

Este subprocesso abrange duas atividades, que são apresentadas no diagrama da **FIG. 3.2.1.2.1-1**: Planejamento (SPV-A1) e Verificação (SPV-A2).

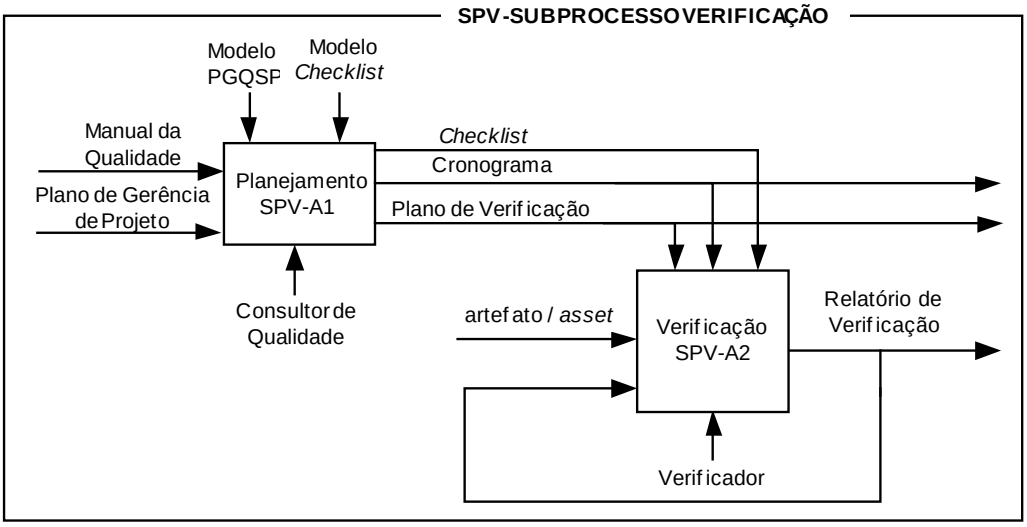


FIG. 3.2.1.2.1-1: Diagrama de Construção do Subprocesso de Verificação (SPV).

Este subprocesso inicia o Planejamento da Verificação (SPV-A1) quando do recebimento da Manual da Qualidade pelo Comitê de Qualidade para obtenção dos requisitos e objetivos do projeto, com o propósito de produzir um plano de verificação (PGQSP da Verificação), cronograma das atividades de verificação e elaboração das listas de conferência (*checklists*) adequadas aos artefatos/*assets* a serem verificados. O Plano de Gerência do Projeto (artefato do Processo de Fornecimento de (GURGEL, 2004)) deve ser aderente a este plano e cronograma de verificação.

É importante ressaltar que todos os produtos de qualidade gerados neste subprocesso devem ser incorporados no Manual da Qualidade. Portanto, o Gerente de Projeto, deve obter os dados do plano de verificação e cronograma da verificação para o Plano de Gerência do Projeto via Manual da Qualidade.

A atividade de Verificação (SPV-A2) inicia ao receber um artefato/*asset* supracitado que foi produzido para uma determinada versão conforme cronograma do projeto definido no Plano de Gerência do Projeto. Após dois dias úteis do recebimento, o verificador deve enviar o Relatório de Verificação ao Consultor de Qualidade alocado ao projeto.

Para apoiar os verificadores quanto aos objetos de verificação, o Consultor de Qualidade deve estabelecer lista de conferência (*checklist*) adequada ao artefato e requisitos do projeto, reusando um modelo de *checklist* aplicável ao artefato e requisitos do projeto a ser verificado, se existir. Os verificadores devem empregar o método *checklist* para detecção de defeitos, por ser sistemático ao ajudar a definir responsabilidades aos verificadores, por sugerir um modo aos verificadores para identificar não-conformidades, e principalmente por fornecer um modelo prático sem necessitar de treinamento prévio.

Laitenberger (LAITENBERGER, 2002) relata que 25 artigos apontam o método *checklist* como um dos métodos de detecção de defeitos mais utilizados na indústria de *software*. O *checklist* é sistemático, pois ajuda definir as responsabilidades dos verificadores e sugere um modo aos verificadores para identificar defeitos, ou seja, apresenta listas de perguntas (itens do *checklist*) que devem ser respondidas durante a leitura do documento. Os itens do *checklist* capturam lições importantes adquiridas de verificações anteriores dentro de um ambiente ou domínio da aplicação. Os itens de *checklist* particularmente podem enumerar características de defeitos, priorizar diferentes defeitos ou apresentar perguntas que auxiliam os verificadores descobrirem defeitos.

Entretanto, este método apresenta algumas dificuldades como (LAITENBERGER, 2002):

- Freqüentemente esta abordagem se tornar não sistemática devido às perguntas serem gerais e não adequadas suficientemente a um determinado ambiente de desenvolvimento, provendo pouco apoio para o verificador entender o artefato inspecionado;
- Falta de uma estratégia de como usar o *checklist* para responder a uma determinada pergunta. Não fica claro que abordagem os verificadores devem seguir ao usar um *checklist* e como eles chegam aos seus resultados - defeitos detectados. O verificador pode conduzir uma única pergunta por todo o artefato, responde a pergunta e segue para a próxima pergunta, ou o verificador lê o documento e depois responde as perguntas do *checklist*;
- As perguntas do *checklist* são freqüentemente limitadas à detecção de defeitos que pertencem a tipos específicos de defeitos. Como os tipos de defeito estão baseados em informação de defeito ocorrido em experiências passadas (CHERNAK,1996), os verificadores podem não focalizar nos tipos de defeito que não tenham sido anteriormente detectados, ocasionando perda de classes inteiras de defeitos.

Para tratar das dificuldades supracitadas, neste subprocesso, a lista de conferência (*checklist*) deve seguir os seguintes princípios:

- O comprimento da lista de conferência não deve exceder a uma página ou a trinta e cinco questões, mas se for necessário ultrapassar esta medida, então designar verificadores responsáveis por diferentes partes da lista;
- As perguntas da lista de conferência devem ser frases tão precisas quanto possíveis;
- A lista de conferência deve ser estruturada de forma que o requisito de qualidade esteja claro para o verificador e a pergunta forneça sugestões de como assegurar o requisito de qualidade;
- A lista de conferência deve ser utilizada em artefatos que o Comitê de Qualidade tenha conhecimento adequado e experiência suficiente para realização das listas de perguntas de forma a abranger maior número de classes de defeitos.

Durante o processo de preenchimento da lista de conferência (*checklist*), o verificador, embasado pela técnica de Leitura baseada em Perspectiva, deve assumir uma perspectiva específica com o objetivo de inspecionar e validar o artefato. O *checklist* deve ser formado por um conjunto de aspectos e cada um destes aspectos deve ser composto por um conjunto de questões que devem obter o maior número possível de respostas adequadas para assegurar o objetivo proposto de cada aspecto. Para obter esse conjunto de aspectos, deve ser considerado o tipo de produto a ser verificado e a necessidade de se obter um artefato adequado ao objeto de trabalho.

As duas atividades do Subprocesso de Verificação são detalhadas a seguir.

SPV-A1 PLANEJAMENTO

Na atividade Planejamento, a partir do Plano de Gerência de Projeto, o Consultor de Qualidade alocado ao projeto deve definir um Plano de Verificação conforme o PGQSP, usando um modelo e cronograma adequado ao projeto de desenvolvimento do *software*, cuja verificação deve ser realizada pelo Grupo de GQS do Órgão Descentralizado de Fornecimento de *Software* contratado.

A seqüência de tarefas da atividade Planejamento consiste em:

SPV-A1-1 - Analisar as necessidades e características desejadas do projeto em função dos recursos e riscos associados que podem causar a não atingir os objetivos;

SPV-A1-2 - Estabelecer *checklists* para verificar os artefatos baseados em aspecto e selecionar responsáveis qualificados para realização das verificações. Para cada aspecto, um ou mais objetivos são definidos, e um conjunto de perguntas que o verificador deve responder é estabelecido dentro do contexto dos objetivos. Para estes *checklists* devem ser reutilizados modelos aplicáveis, se existirem;

SPV-A1-3 - Determinar os recursos, responsáveis e cronogramas para as verificações dos seguintes artefatos: Necessidades do Cliente e Glossário (Processo de Aquisição) para assegurar padronização da documentação, qualidade, completude, rastreabilidade e corretude das necessidades e características desejadas do projeto pelo cliente; Arquitetura (Engenharia de Domínio) para garantir corretude, rastreabilidade, clareza e padrões de estrutura; código-fonte (Processo de Desenvolvimento) para garantir codificação adequada aos padrões e requisitos estabelecidos no Contrato; e Casos de Testes de Componentes e de Sistema para assegurar a padronização da documentação, completude, corretude, rastreabilidade e qualidade do produto final.

SPV-A2 VERIFICAÇÃO

Na atividade Verificação, a partir da prontificação do artefato/*asset* pela equipe de desenvolvimento/ engenheiros de domínio, o verificador alocado à verificação do artefato/*asset*, conforme o cronograma e PGQS, deve realizar uma análise prévia do artefato/*asset*. Após a análise, o verificador, embasado pela técnica de Leitura baseada em

Perspectiva, deve aplicar a lista de conferência (*checklist*) adequada ao artefato/*asset* definida na atividade anterior para detecção de não-conformidades. Esta atividade termina com a produção do Relatório de Verificação e o envio deste ao Consultor de Qualidade alocado ao projeto.

Todas as não-conformidades detectadas no Relatório de Verificação devem ser corrigidas pelo responsável pela elaboração do artefato verificado e suas correções verificadas pelo verificador de qualidade responsável. As não-conformidades resolvidas e aprovadas devem constar no Relatório de Verificação. E, durante a realização da verificação, os problemas detectados que fogem ao escopo do responsável pela elaboração do artefato devem ser listados no Relatório de Verificação, assim como as ações decorrentes. Os resultados das atividades de verificação devem ser disponibilizados para a equipe de desenvolvimento/montadores, engenheiros de domínio, cliente e gerente de projeto.

3.2.1.2.2 SUBPROCESSO DE REVISÃO – SPR

O Subprocesso de Revisão tem o propósito de estabelecer as atividades voltadas para o Controle da Qualidade que devem ser aplicadas à medida que um conjunto de artefatos de especificação de requisitos e de projeto de arquitetura é produzido na Engenharia de Domínio para avaliar tecnicamente os artefatos das atividades de Análise e Projeto de Domínio proposto em (GURGEL, 2004). As atividades deste subprocesso devem implementar reuniões conjuntas com inspeções individuais baseadas na Técnica de Leitura Horizontal e Vertical. Os artefatos inspecionados são: Modelo de Contexto do Domínio, Modelo Conceitual do Domínio, Modelo de Casos de Uso do Domínio e Descrição dos Cenários, Modelo de Características, Modelo de Negócio, Interfaces da Camada de Negócio, Interfaces da Camada de Aplicação, Diagrama de Interações, Arquitetura de Domínio e Especificação de Componentes do Domínio.

Este subprocesso abrange quatro atividades, que são apresentadas no diagrama da **FIG. 3.2.1.2.2-1**: Planejamento (SPR-A1), Treinamento para Revisão Técnica (SPR-A2), Preparação para Revisão Técnica (SPR-A3) e Revisão Técnica (SPR-A4).

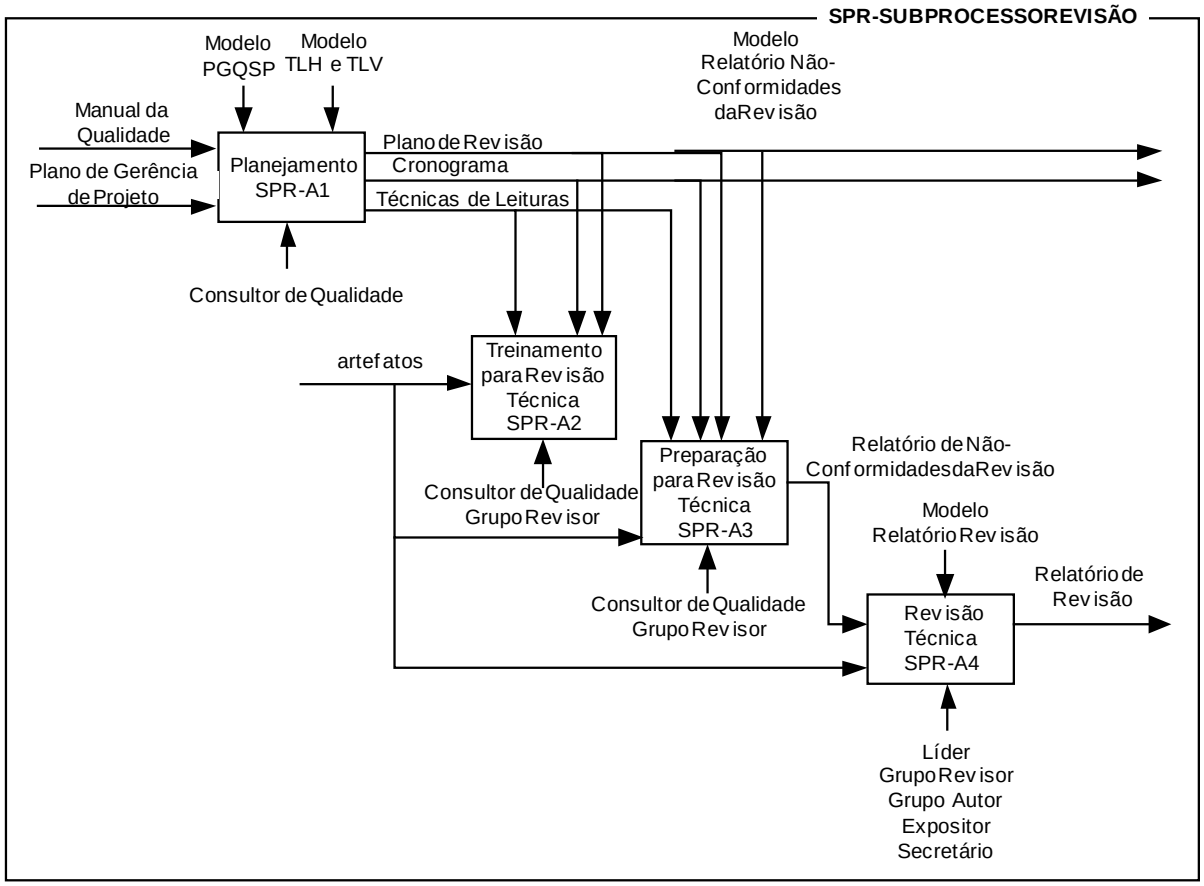


FIG. 3.2.1.2.2-1: Diagrama de Construção do Subprocesso de Revisão (SPR).

Este subprocesso inicia o Planejamento da Revisão (SPR-A1) quando do recebimento da Manual da Qualidade pelo Comitê de Qualidade para obtenção dos requisitos e objetivos do projeto, com o propósito de produzir um plano de revisão (PGQSP da Revisão), cronograma das atividades de revisão e elaboração dos procedimentos das Técnicas de Leitura Vertical (TLV) e Horizontal (TLH) adequados aos artefatos a serem inspecionados. O Plano de Gerência do Projeto, também, deve ser aderente a este plano e cronograma de revisão.

É importante ressaltar que todos os produtos de qualidade gerados neste subprocesso devem ser incorporados no Manual da Qualidade. Portanto, o Gerente de Projeto, deve obter os dados do plano de revisão e cronograma da revisão para o Plano de Gerência do Projeto via Manual da Qualidade.

O Consultor de Qualidade alocado ao projeto deve definir os integrantes que vão participar das atividades seguintes ao Planejamento (SPR-A1), ou seja, deve estabelecer os seguintes papéis (WIEGERS, 1995) para cada conjunto de artefatos a ser revisado:

- **Grupo autor:** grupo formado por autores dos artefatos a serem examinados;

- **Grupo revisor:** grupo formado por pessoas do Órgão Descentralizado de Fornecimento de *Software* contratado, e opcionalmente pelo(s) cliente(s) ou representante(s) legal(is), que tenham conhecimento e qualificação inerentes ao artefato, mas que não atuaram na elaboração do artefato;
- **Um secretário:** um membro do grupo autor do artefato que deve realizar a ata da reunião;
- **Um líder:** um consultor de qualidade alocado ao projeto;
- **Um expositor:** um membro do grupo revisor que expõe o problema.

A atividade de Treinamento (SPR-A2) inicia ao receber um conjunto de artefatos que foi produzido para uma determinada versão conforme cronograma do projeto definido no Plano de Gerência do Projeto. Na **TAB. 3.2.1.2.2-1** é especificado cada conjunto de artefatos que deve ser enviado ao expositor e quando do envio pelo Gerente do Projeto.

TAB. 3.2.1.2.2-1: Conjunto de artefatos para revisão.

Conjunto de Artefatos		Quando
1	▪ Modelo de Contexto ▪ Modelo Conceitual ▪ Modelo de Casos de Uso e Cenários ▪ Modelo de Características	Término da atividade Análise do Domínio do Processo ao Longo do Projeto.
2	▪ Modelo de Tipo de Negócio ▪ Interfaces da Camada de Negócio e Tipos de Dados ▪ Interfaces da Camada de Aplicação Arquitetura de Domínio ▪ Diagrama de Interações	Durante a atividade Projeto de Domínio do Processo ao Longo do Projeto.
3	▪ Especificação de Componente de Domínio ▪ Interfaces dos Componentes de Domínio	Término da atividade Projeto de Domínio do Processo ao Longo do Projeto.

Após um dia útil do recebimento, o expositor alocado ao conjunto de artefatos a ser revisado deve treinar o grupo revisor. Em seguida, este grupo juntamente com o grupo autor devem-se preparar para revisão técnica, conforme as respectivas atividades SPR-A2 e SPR-A3 especificadas mais adiante. Durante esta preparação, é aplicada a Técnica de Leitura Vertical (TLV) e Horizontal (TLH) para inspecionar os artefatos conforme são apresentados na **FIG. 3.2.1.2.2-2**. A inspeção de cada artefato não deve ultrapassar mais de uma hora.

A reunião da revisão deve ser realizada no dia seguinte da execução das inspeções do conjunto de artefatos recebidos para ser revisado. Esta reunião não deve demorar mais do que duas horas. A equipe de revisão deve ser composta pelo grupo revisor, grupo autor, um líder, um secretário e um expositor, conforme estabelecido na atividade de Planejamento (SPR-A1).

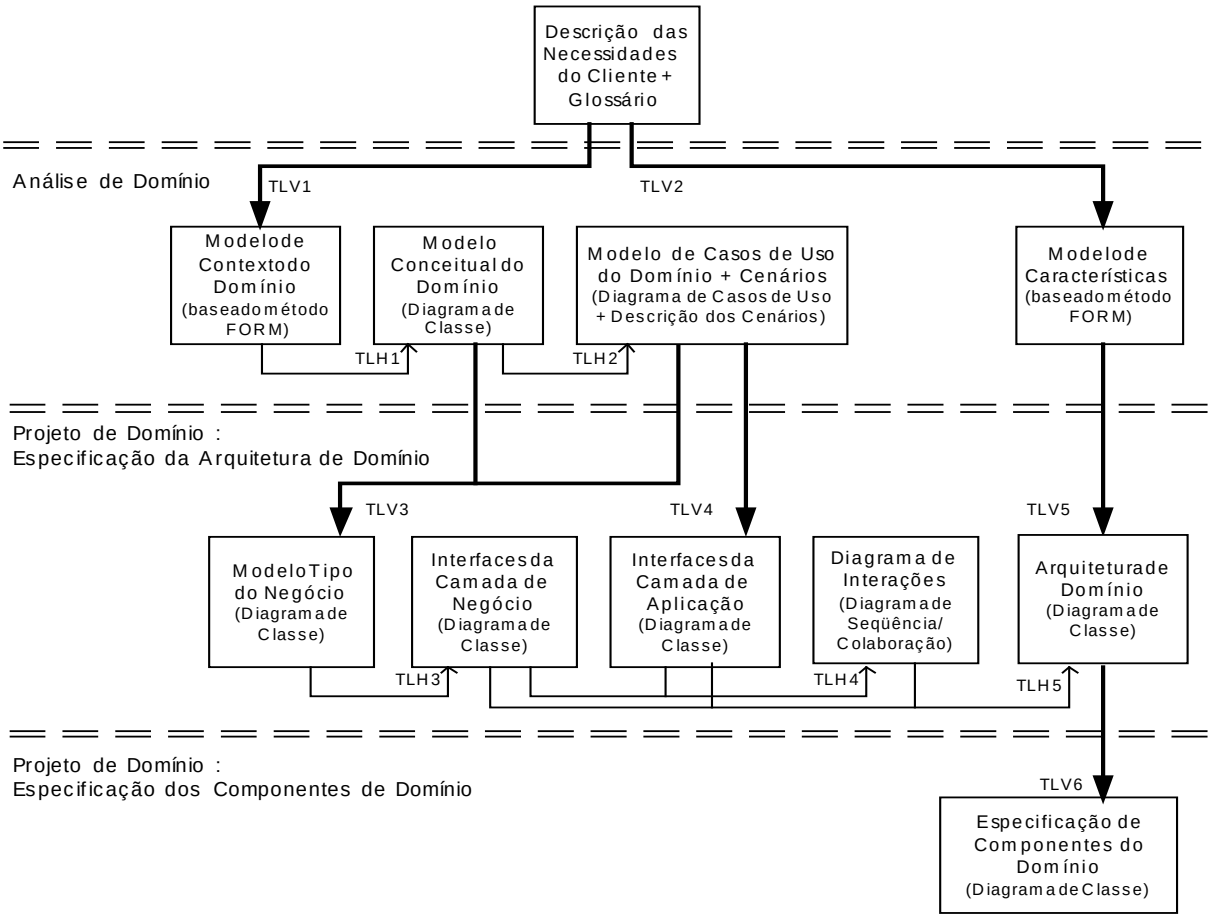


FIG. 3.2.1.2.2-2: Conjunto das Técnicas de Leitura Horizontal e Vertical.

Cada inspeção deve empregar a Técnica de Leitura Vertical (TLV) e Horizontal (TLH) baseada em (TRAVASSOS et al., 2003). E, cada revisor deve direcionar sua leitura por um cenário específico ao invés de uma leitura passiva. O cenário é um procedimento que o revisor deve seguir durante a inspeção, consistindo de atividades repetíveis que um revisor precisa executar e de verificações que um revisor deve certificar. Desta forma, esta técnica de leitura procura abranger as mais diversas classes de defeitos.

O resultado desta reunião deve ser uma lista de não-conformidades, as quais devem ser numeradas seqüencialmente e anotadas no Relatório de Revisão pelo secretário. Neste relatório também deve conter os nomes dos revisores, data da reunião, material avaliado e ações decorrentes. Essas revisões devem ser realizadas nos marcos do Plano de Gerência do Projeto de acordo com os términos das atividades de elaboração dos documentos acima descritos do Processo ao Longo do Projeto para avaliação dos artefatos produzidos, de forma a garantir aos usuários (internos e externos) deste artefato uma utilização de produtos pelo

menos com a qualidade mínima especificada. Esta revisão é de responsabilidade do Gerente de Projeto, Engenheiros de Domínio, Consultor da Qualidade e Cliente.

Em geral, as reuniões podem gerar conflitos humanos, portanto é fundamental que algumas abordagens para melhoria destes confrontos sejam tratadas (TAVARES, 2002) antes da reunião, principalmente na atividade Treinamento para Revisão Técnica (descrita mais adiante), tais como:

- O grupo revisor e grupo autor do artefato devem entender que o objeto da revisão é um artefato, estático naquele ponto, e não seus autores;
- As pessoas envolvidas em uma revisão devem exercer outros papéis nas revisões de outros artefatos, havendo um rodízio de papéis, agindo ora como revisores ora como autores ora como moderadores;
- Os grupos autor e revisor devem entender que críticas reincidentes a alguns aspectos do artefato examinado não significa críticas reincidentes ao grupo autor do artefato em inspeção.

As quatro atividades do Subprocesso de Revisão são detalhadas a abaixo.

SPR-A1 PLANEJAMENTO

O Consultor de Qualidade deve criar e documentar um Plano de Revisão para as fases de Análise e Projeto do Domínio da Engenharia de Domínio do Processo ao Longo do Projeto conforme o PGQSP, usando um modelo e cronograma adequado ao projeto de desenvolvimento do *software*, para definir para cada atividade os recursos, cronograma e procedimentos para gerenciar as revisões dos artefatos. A seqüência de tarefas da atividade Planejamento consiste em:

SPR-A1-1 - Agendar as reuniões das revisões periódicas nos marcos estabelecidos no Plano de Gerência do Projeto conforme o término das atividades Análise de Domínio, Especificação da Arquitetura de Domínio durante a atividade Projeto de Domínio e Especificação de Componentes do Domínio para avaliação dos artefatos produzidos;

SPR-A1-2 - Estabelecer as técnicas de leitura para revisões individuais dos artefatos que vão compor os *assets* da Engenharia de Domínio. Para cada técnica de leitura horizontal e vertical, um procedimento é estabelecido para verificação e certificação de um conjunto de informação dentro do contexto dos objetivos do tipo da técnica. Para estas técnicas devem ser reutilizados modelos aplicáveis, se existirem;

SPR-A1-3 - Selecionar os membros que vão compor a reunião da revisão relativa ao artefato a ser examinado. Além disso, deve definir local, instalações, *hardware*, *software* e ferramentas necessários para realização das revisões acordados pelos grupos (revisor e autor);

SPR-A1-4 - Para cada revisão, acordado pelos grupos revisor e autor, agendar as atividades pertinentes (treinamento, preparação e revisão técnica e de gestão).

SPR-A2 TREINAMENTO PARA REVISÃO TÉCNICA

A atividade Treinamento para Revisão Técnica deve ser realizada pelo Consultor de Qualidade. Esta atividade consiste em capacitar os revisores a inspecionar o conjunto de artefatos de acordo com a TLV e TLH. A duração desta atividade não deve ultrapassar à uma hora. Esta atividade é opcional, pois caso o grupo de revisor já tenha experiência com este tipo de revisão técnica e não necessite de instruções.

A seqüência de tarefas desta atividade consiste em:

SPR-A2-1 - Treinar o grupo revisor em que revisão de artefato eles vão examinar;

SPR-A2-2 - Associar técnicas de leitura horizontal e vertical aos revisores;

SPR-A2-3 - Fornecer impressão do documento relativo ao artefato a ser examinado para cada revisor, que deverá dispor de uma hora para leitura do documento.

No caso da não realização desta atividade, as tarefas SPR-A2-2 e SPR-A2-3 devem ser executadas pela atividade seguinte, ou seja, SPR-A3.

SPR-A3 PREPARAÇÃO PARA REVISÃO TÉCNICA

A atividade Preparação para Revisão Técnica deve ser realizada pelo Consultor de Qualidade. Esta atividade consiste em preparar os revisores sobre o conjunto de artefatos a ser inspecionado, e a duração desta atividade não deve ultrapassar a duas horas.

A seqüência de tarefas da atividade Preparação para Revisão Técnica consiste em:

SPR-A3-1 - Os revisores devem aprender sobre os artefatos a serem examinados e prepararem-se para desempenhar as leituras individualmente, verificando para o tipo de leitura os recursos necessários para execução do procedimento da leitura;

SPR-A3-2 - Uma vez que todos os revisores estão preparados, o grupo revisor procura não-conformidades, utilizando a Técnica de Leitura Vertical (TLV) e Horizontal (TLH) adequada ao artefato em revisão. Em cada leitura, o revisor deve procurar evidências de que o

artefato esteja completo, aderente aos padrões e especificações relacionados, como também o artefato esteja implementado adequadamente. A inspeção de cada artefato não deve ultrapassar mais de uma hora;

SPR-A3-3 - Cada revisor documenta os resultados da revisão no Relatório de Não-Conformidades da Revisão.

SPR-A4 REVISÃO TÉCNICA

A atividade Revisão Técnica consiste na realização da reunião da revisão do conjunto de artefatos inspecionados individualmente na atividade Preparação para Revisão Técnica (SPR-A3). Esta atividade deve ser realizada pelos integrantes estabelecidos na atividade Planejamento (SPR-A1), ou seja, um líder, um grupo de revisores, um grupo de autores do conjunto de artefatos inspecionados, um secretário e um expositor. O líder deve ser o Consultor de Qualidade alocado ao projeto. A duração da reunião não deve ultrapassar a duas horas.

A seqüência de tarefas da atividade Revisão Técnica consiste em:

SPR-A4-1 - A reunião se inicia pelo expositor apresentando o documento a ser examinado, e em seguida pela exposição do nível de adeqüabilidade e, se existir, das não-conformidades encontradas por cada revisor. Durante o debate dos grupos revisor e autor, pode haver intervenções do líder como mediador;

SPR-A4-2 - Durante toda a reunião o secretário preenche o Relatório de Revisão com os nomes dos membros da reunião, data-hora, local, recursos utilizados, como também, registra os problemas detectados e as ações decorrentes, inclui os resultados da revisão (as não-conformidades detectadas pelo revisor através da leitura individual) e registra o nível de adeqüabilidade (aprovado, desaprovado ou aprovado com pendências);

SPR-A4-3 - Das não-conformidades detectadas, o grupo autor pode apresentar justificativas dos motivos que levam a alguns deles não representarem problemas reais. Esta justificativa deve constar no Relatório de Revisão;

SPR-A4-4 - Os problemas detectados que fogem ao escopo do responsável pela elaboração do artefato devem ser listados no Relatório de Revisão, assim como as ações decorrentes.

O grupo autor deve corrigir as não-conformidades descritas no Relatório da Revisão, e o Gerente do Projeto deve atualizar o Plano de Gerência do Projeto conforme o resultado da revisão.

O Consultor de Qualidade deve acompanhar as correções apontadas no Relatório de Revisão se estão sendo realizadas eficazmente, e conforme a data estabelecida no Relatório de Revisão realizar nova revisão, se necessário, para verificar as correções e se defeitos secundários não foram introduzidos.

As não-conformidades resolvidas e aprovadas devem constar no Relatório de Revisão. E, os resultados das atividades de revisão devem ser disponibilizados para a equipe de desenvolvimento/montadores, engenheiros de domínio, cliente e gerente de projeto.

3.2.1.2.3 SUBPROCESSO DE MEDIÇÃO E AVALIAÇÃO – SPM

O Processo de Medição e Avaliação tem o propósito de definir procedimentos para medir e julgar, além do nível de reusabilidade, a qualidade do produto ou artefato quanto aos requisitos e satisfação do cliente.

Este subprocesso abrange cinco atividades, que são apresentados no diagrama da **FIG. 3.1.2.3-1** (ALMEIDA et al., 2003): Análise de Requisitos, Especificação da Avaliação, Planejamento da Avaliação, Execução da Medição e Avaliação.

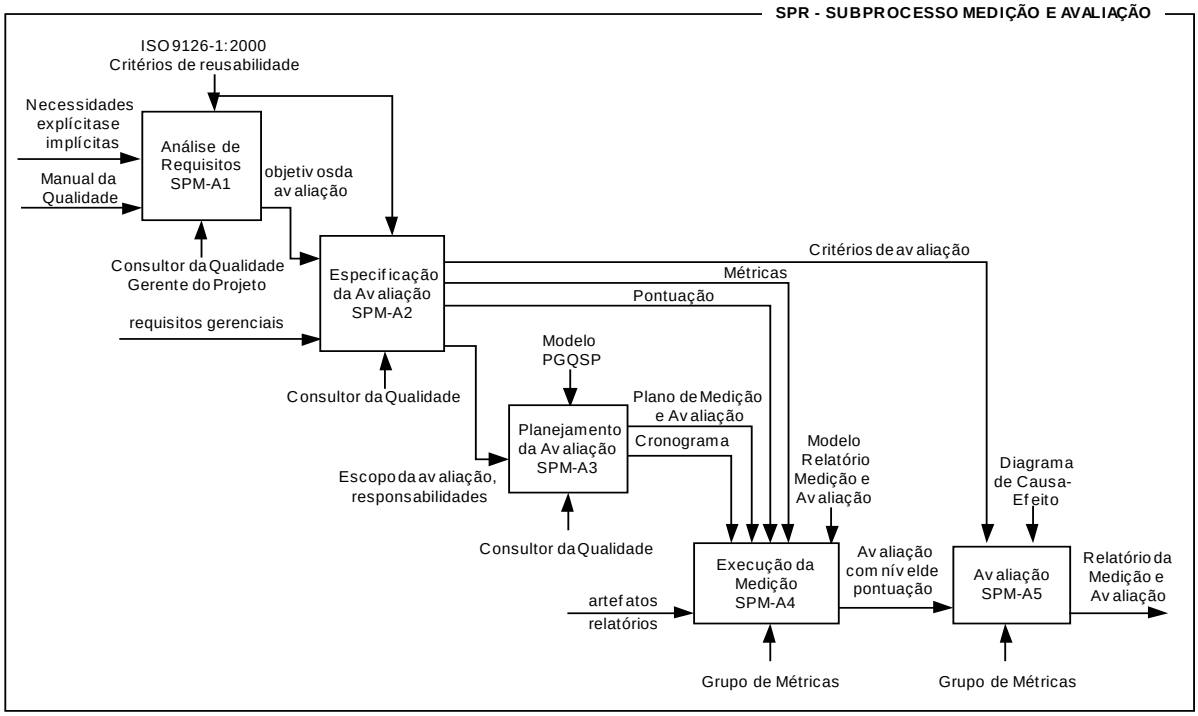


FIG. 3.2.1.2.3-1: Diagrama de Construção do Subprocesso de Medição e Avaliação (SPM).

Este subprocesso inicia a Análise de Requisitos (SPM-A1) quando do recebimento do Manual da Qualidade pelo Comitê de Qualidade para obtenção dos requisitos e objetivos do projeto, com o propósito de definir os objetivos da avaliação da medição. Baseados neste manual, o Consultor da Qualidade alocado ao projeto e o Gerente do Projeto devem definir os requisitos de qualidade e reusabilidade de acordo com as necessidades explícitas ou implícitas do projeto. Os objetivos da avaliação devem atender a Norma ISO 9126-1:2000 e os critérios de qualidade requeridos pelo Contrato.

A partir dos objetivos estabelecidos na atividade SPM-A1, o Consultor da Qualidade alocado ao projeto deve selecionar métricas, definir nível de pontuação e critérios de avaliação.

A cada nova versão do projeto, conforme o Plano de Gerência do Projeto, o Consultor de Qualidade alocado ao projeto deve enviar todos os relatórios de Verificação e Revisão realizados no período da versão anterior correspondente para o Grupo de Métricas. Este grupo deve medir e analisar os dados coletados destes relatórios e dos artefatos produzidos, e deve enviar um Relatório de Medição e Avaliação, juntamente com todos os Relatórios de Verificações e Revisões, aos Gerentes de Reuso e do Projeto uma semana após da prontificação da versão.

É importante ressaltar que todos os produtos de qualidade gerados neste subprocesso devem ser incorporados no Manual da Qualidade. Portanto, o Gerente de Projeto, deve obter os dados do plano de medição e avaliação e cronograma da revisão para o Plano de Gerência do Projeto via Manual da Qualidade.

Neste processo, deve ser adotada a abordagem GQM (*Goal/Question/Metric*) de (BASILI et al., 1994) para definir um processo de coleta e análise de dados. Primeiro, deve-se estabelecer uma referência para avaliar o progresso em termos dos objetivos e através de comparação com esta referência. Assim, após a cada nova referência no repositório da biblioteca, deve ser avaliada a qualidade e a reusabilidade do produto pelo Grupo de Métricas. O Grupo de Métricas deve ser formado por especialistas em estatísticas que pertençam ao Órgão Descentralizado de Fornecimento de *Software* para fornecer suporte na análise dos dados consolidados.

Os registros de informações (data, hora, usuário requerente, revisores) de entradas e saídas dos artefatos nos repositórios da biblioteca e a base de dados onde são armazenadas informações históricas e versões dos artefatos vão servir de informação de entrada para o Processo de Medição e Avaliação, além dos Relatórios de Testes, Verificação e Revisão.

Estas informações de entrada podem ser utilizadas para obtenção do tamanho do artefato, quantidade de reuso de *assets* por domínio, o número de alterações em um projeto, quantidade de defeitos detectados, economia de tempo e benefícios derivados do reuso de *asset*.

O Gerente do Projeto deve incluir requisitos de reusabilidade para *assets* nas especificações de atributos de qualidade para assegurar a qualidade dos *assets* selecionados para reuso no desenvolvimento do produto. Os **critérios para avaliar a qualidade de reusabilidade** são a confiança obtida com a experiência de *assets* reusados em projetos semelhantes, os defeitos detectados como resultados de inspeção dos *assets* e os resultados dos testes dos componentes diante os requisitos do componente e do sistema (IEEE1517, 1999).

Os dados coletados e armazenados durante as verificações, testes e revisões devem promover o acompanhamento da qualidade e da reusabilidade na produção de *assets*, como também a troca de experiência entre as equipes de desenvolvimento e engenheiros de domínio. O Grupo de Métricas deve fornecer os resultados da análise ao Comitê de Qualidade para divulgação dos resultados às gerências do Órgão Descentralizado de Fornecimento de *Software* e ao Grupo de GQS Corporativo. Este último deve analisar os pontos fortes e fracos dos processos empregados. Estas avaliações devem ser utilizadas para recomendar alterações nas diretrizes dos futuros projetos e para determinar avanços tecnológicos.

As cinco atividades do Subprocesso de Medição e Avaliação são detalhadas a abaixo.

SPM-A1 ANÁLISE DE REQUISITOS

A atividade Análise de Requisitos deve descrever as especificações dos requisitos de qualidade e reusabilidade, definindo os objetivos da avaliação de acordo do ponto de vista dos diferentes usuários do produto. Os requisitos a serem avaliados devem estar em conformidade com a norma ISO 9126-1:2000 e devem atender as necessidades explícitas e implícitas do usuário.

SPM-A2 ESPECIFICAÇÃO DA AVALIAÇÃO

A atividade Especificação da Avaliação estabelece o escopo da avaliação, as medições a serem realizadas no produto / artefato, a definição do nível de pontuação e do critério de avaliação, os métodos utilizados e as responsabilidades de todos envolvidos no processo de

avaliação. Esta atividade deve procurar selecionar os objetivos que facilmente podem prover medidas mais do tipo *hard* (menos subjetiva e mais objetiva) e que melhor atendem às necessidades de verificação. Baseado no modelo de (BASILI et al., 2000) devem ser identificados e definidos objetivos, questões e métricas (*Goal/Question/Metric*) para processos (por exemplo, tempo de análise, tempo de codificação, tempo de teste) e para produtos / artefatos (por exemplo, linhas de código, número de *assets* reusáveis) de maneira objetiva.

SPM-A3 PLANEJAMENTO DA AVALIAÇÃO

O Consultor de Qualidade deve criar e documentar um Plano para a Medição e Avaliação da qualidade e reusabilidade adequado ao projeto de desenvolvimento com todas as atividades de implantação e administração do processo, reusando um modelo aplicável de plano de administração de medição e avaliação, se existir, para definir os recursos, cronograma, os procedimentos e a metodologia para gerenciar e implementar a medição e avaliação de produto / artefato.

O Plano de Medição e Avaliação deve incluir os seguintes aspectos:

- Recursos, cronograma, responsabilidades e procedimentos para realização da medição e análise do processo e produtos;
- Prover infra-estrutura (métodos, ferramentas, práticas, recursos humanos, treinamento) para assegurar a execução do processo.

SPM-A4 EXECUÇÃO DA MEDIÇÃO

O Grupo de Métricas deve medir o produto / artefato mediante as métricas selecionadas na atividade Especificação da Avaliação (SPM-A2) e os recursos e a metodologia estabelecidos conforme o Plano de Medição e Avaliação. Os dados obtidos da medição devem ser coletados e armazenados. O Grupo de Métricas, diante dos resultados coletados, deve registrar no Relatório da Avaliação o nível de pontuação referente a cada métrica de acordo com a pontuação estabelecida na atividade Especificação da Avaliação (SPM-A2). Além disso, o grupo deve verificar o produto / artefato se está em conformidade com os requisitos, a especificação e o Plano de Medição e Avaliação. Finalmente, deve gerar o Relatório da

Avaliação, que deverá ser divulgado dentro da organização e, também, para o Grupo de GQS Corporativo.

A seqüência de tarefas da atividade Execução da Medição consiste em:

- Coletar dados que medem as métricas identificadas na atividade Especificação da Avaliação;
- Armazenar os dados;
- Analisar os dados para fornecer o nível de pontuação;
- Verificar o produto / artefato está de acordo com os requisitos, especificações e Planejamento da Avaliação;
- Pontuar a avaliação.

SPM-A5 AVALIAÇÃO

O Grupo de Métricas deve analisar os resultados para avaliar o desempenho, a estabilidade e capacidade do processo, prever custos e desempenhos futuros, identificar tendências, identificar oportunidades de melhoria da qualidade e de reusabilidade do produto / artefato. Este grupo deve produzir o Relatório de Medição e Avaliação, e disponibilizar os dados resultantes da mesma, inclusive as anomalias do processo encontradas. Para identificar as causas das anomalias do processo, o Grupo de Métricas pode utilizar o Diagrama de Causa-Efeito (PALMER, 1997). Uma vez diagnosticada as causas efetivas do problema detectado, elas devem ser documentadas no Relatório de Medição e Avaliação para que o Gerente de Reuso possa verificar e quantificar a viabilidade da mudança do processo.

3.2.2 PROCESSO DE APROVAÇÃO DE *ASSETS*

O Processo de Aprovação de *Assets* consiste na aprovação dos tipos e da documentação dos *assets* para que estes possam ser posteriormente classificados e armazenados nos repositórios da biblioteca pelo Processo de Gerência de *Assets* do Processo de Desenvolvimento de *Software* baseado em Reuso na MD de (GURGEL, 2004). A documentação do *asset* deve propiciar o completo entendimento aos reutilizadores do *asset* de forma concisa e clara.

Este processo inicia quando do recebimento, após a cada versão definida do projeto no Plano de Gerência do Projeto, de todos os artefatos que compõe o *asset* com qualidade

aprovada pelo Processo de Controle de Qualidade e a documentação do *asset* (veja **FIG. 3.2.2-1**). De posse do *asset* e de sua documentação, o Comitê de Qualidade deve identificar o tipo de *asset* de acordo com a **TAB. 3.2.2-1** e especificar os requisitos do *asset* para o Subprocesso de Verificação (SPV) do Processo de Controle de Qualidade (PCQ) validar via *checklist* características de reusabilidade, compatibilidade, padronização da documentação, rastreabilidade, corretude, facilidade no entendimento das funcionalidades e objetivos do *asset*. A partir da qualidade aprovada do *asset* pelo Relatório de Verificação, o Comitê de Qualidade deve enviar o *asset* aprovado ao Processo de Gerência de *Assets* para catalogação, notificação e gestão do *asset* no repositório de *assets* reutilizáveis e um Relatório de Aprovação do *asset* ao Grupo de GQS Corporativo e ao Gerente de Reuso.

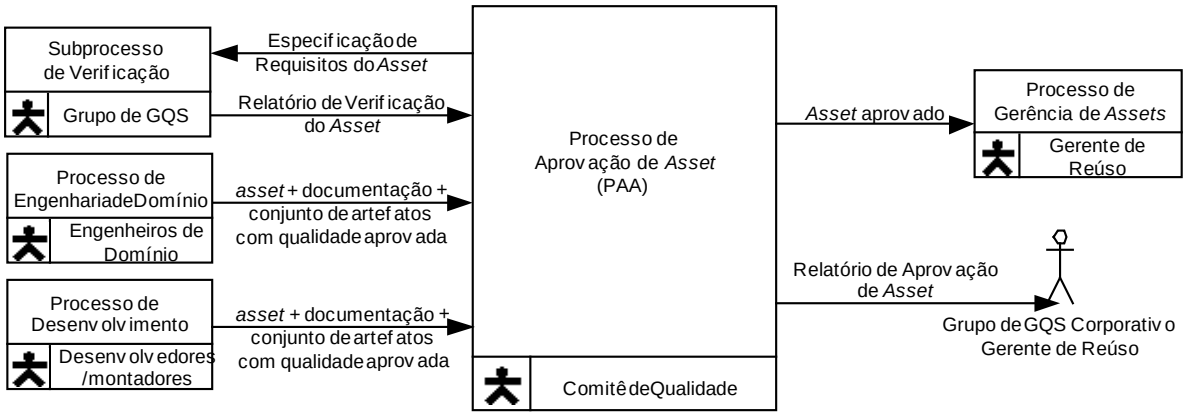


FIG. 3.2.2-1: Processo de Aprovação de *Assets* (PAA).

Na **FIG. 3.2.2-1** são apresentadas as interações do Processo de Aprovação de *Asset* (PAA) com os processos de (GURGEL, 2004) e de Controle da Qualidade (PCQ), apresentando os artefatos de entrada/saída com seus respectivos papéis e os responsáveis pelo PAA. E, na **TAB. 3.2.2-1** são apresentados os *assets* que devem ser gerados pelo órgão contratado para fornecimento do *software* com os artefatos que o compõe frutos de uma atividade específica de um processo definido de (GURGEL, 2004).

Para o *asset* ser aprovado, ele deve atender a um critério de aprovação inerente aos artefatos gerados que o compõe. Além disso, a documentação deve ser legível, identificável e completa, e também protegida, de acordo com o grau de segurança da informação contida no *asset* conforme especificado no Contrato (IEEE1517, 1999).

TAB. 3.2.2-1: Tipos de *assets* produzidos pelo Processo de Desenvolvimento de *Software* baseado em Reuso na MD.

Assets	**Artefatos**	**Atividade**	**Processo**
Domínio	▪ Modelo de Contexto ▪ Modelo Conceitual ▪ Modelo de Casos de Uso do Domínio e Cenários ▪ Modelo de Características ▪ Ontologias ▪ Taxonomias ▪ Linguagens de Domínio	Análise de Domínio	Engenharia de Domínio
Arquitetura de Domínio	▪ Modelo Tipo de Negócio ▪ Interfaces da Camada de Negócio e Tipos de Dados ▪ Interfaces da Camada de Aplicação ▪ Arquitetura de Domínio	Projeto de Domínio	Engenharia de Domínio
Componente de Domínio	▪ Especificação de Componentes de Domínio ▪ Interfaces de Componentes de Domínio	Projeto de Domínio	Engenharia de Domínio
Componentes	▪ Código de Componente ▪ Teste de Componentes	Construção do *Software*	Processo de Desenvolvimento

Os verificadores devem aplicar a técnica de leitura baseada em perspectiva que possibilita a verificação do *asset* sob a perspectiva de diferentes reutilizadores. Para cada perspectiva, tanto um como vários cenários devem ser definidos, consistindo de atividades repetíveis que um verificador deve executar, e de perguntas que um verificador deve responder. Para essas verificações devem ser reutilizados modelos aplicáveis, se existirem.

A documentação de cada *asset* deve conter as seguintes características apontadas por (BRAUN, 1994; KARLSSON, 1995; KRUEGER, 1992; MEYER, 1994):

- **Geral:** contém informação geral sobre o *asset* e a partir desta informação deve ser possível decidir se o *asset* é candidato para um certo cenário. Entretanto, esta documentação não pode ser muito detalhada para evitar que o processo de seleção de candidato seja demorado. Porém, este processo de seleção pode requerer busca de mais informação que deve estar documentada em algumas das características apresentadas a seguir;
- **De Reutilização:** contém todas as informações necessárias de instalação, uso e adaptação do *asset*;
- **Administrativa:** contém informações administrativas como restrições legais e suporte disponível;
- **De Avaliação:** contém informações mais detalhes para a avaliação de um *asset* como problemas conhecidos, restrições e declarações de qualidade. É importante ressaltar que para

aumentar o ganho em reuso, o *asset* deve ter o menor número de restrições técnicas possíveis e a maior abrangência quanto seu escopo;

- **Complementar:** deve conter qualquer detalhe a mais que não foi suprido pelas primeiras quatro características.

Segundo Sametinger, a documentação de um *asset* deve conter os seguintes itens de acordo com as cinco características acima descritas (SAMETINGER, 1996):

1 - **Características Gerais:**

- **Nome do *asset*:** nome, dando uma possível sugestão sobre a funcionalidade do *asset*;
- **Identificação:** declaração inicial concisa e clara sobre a funcionalidade do *asset* para seleção inicial, e inclusive com descrição geral de todas as operações externamente visíveis, se for um componente;
- **Especificação:** funcionalidade do *asset* que deve ser detalhada;
- **Classificação:** classificação do *asset* como uma lista de palavras chaves para habilitar recuperação futura, por exemplo, se *asset* for um componente, indicar o tipo (classes de C++, funções de C, aplicação de OpenDoc).

2 - **Características de Reutilização**

- **Usabilidade:** descrição de como o *asset* pode ser utilizado corretamente;
- **Instalação:** descrição de como adaptar o *asset* em uma nova aplicação;
- **Adaptação:** descrição de como e para que necessidades específicas pode ser adaptado o *asset*;
- **Implementação:** descrição de como o *asset* está implementado;
- **Suporte**: onde adquirir ajuda, se necessário for, por exemplo, ao adaptar o componente ou o que fazer se um problema ocorrer.

3 - **Características Administrativas**

- **Histórico:** histórico de versões, datas de liberações, desenvolvedores responsáveis, principais diferenças entre as versões;
- **Restrições comerciais e legais:** restrições comerciais ou legais no uso do componente, por exemplo, aquisição, licença especial ou permissão requerida.

4 - **Características de Avaliação**

- **Restrições técnicas:** restrições técnicas no uso do *asset*, por exemplo, capacidades, linguagem de programação, dependências de sistema operacional;

- **Qualidade:** indicar a situação do *asset*, quanto à qualidade, revisões, verificações e casos de testes aplicados, inclusive os resultados obtidos;
- **Performance:** recursos do sistema necessários para usar o *asset*, por exemplo, memória, processador, canais de comunicação;
- **Alternativas:** relação de *assets* semelhantes que poderiam ser usados no lugar deste;
- **Problemas:** relato de problemas conhecidos e custo requerido;
- **Interdependências:** o *asset* pode ser usado *stand-alone* ou deve ser usado junto a outros *assets*;
- **Custo recomendado:** custo conhecido para melhorar performance, tornar o *asset* mais robusto e / ou estender o escopo de reuso.

5 - **Características Complementares**

- **Indexação:** provê um índice para *assets* complexos que requerem documentação extensa;
- **Referências:** referências para literatura ou outra documentação, por exemplo, documentação de sistemas.

Este processo abrange duas atividades, que são apresentadas no diagrama da **FIG. 3.2.2-1**: Identificação e Avaliação.

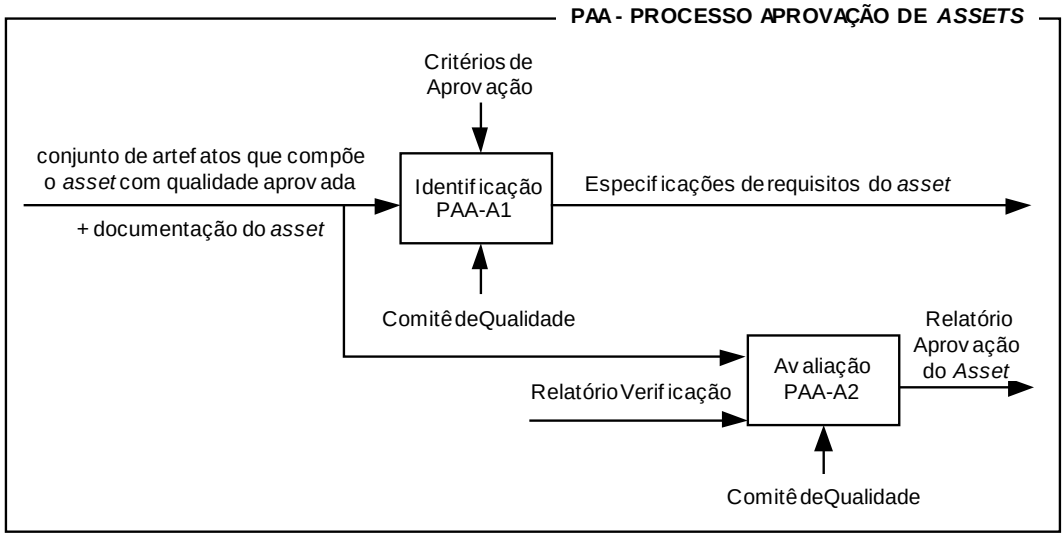


FIG. 3.2.2-2: Diagrama de Construção do Processo de Aprovação de *Assets* (PAA).

As duas atividades do Processo de Aprovação de *Assets* são detalhadas a abaixo.

PAA-A1 IDENTIFICAÇÃO

A partir do recebimento do *asset* juntamente com seus respectivos artefatos com qualidade aprovada que o compõe, o Comitê de Qualidade deve identificar e especificar os requisitos do *asset* submetido para aprovação, reusando um modelo aplicável de especificação de requisitos para aprovação de *asset*, se existir, segundo **Critérios de Aprovação** de *assets*. Essas especificações de requisitos para aprovação de *asset* devem ser fornecidas ao Subprocesso de Verificação do Processo de Controle de Qualidade para que os verificadores possam validar via *checklist* aspectos de reusabilidade e de compatibilidade, dentre outros aspectos como padronização da documentação, rastreabilidade, corretude e facilidade no entendimento das funcionalidades e objetivos do *asset*.

Para aprovação de *asset*, os requisitos de aprovação devem ser definidos, obedecendo, conforme o tipo do *asset*, aos seguintes **Critérios de Aprovação**:

1 - Utilidade e reusabilidade dos requisitos de domínio do componente;

2 - Utilidade e reusabilidade da arquitetura de domínio do *asset*;

3 - Grande potencial de reuso dos requisitos para serem usados em vários domínios;

4 - Grande potencial de reuso da arquitetura de domínio do *asset* ou parte dela para ser usada em vários domínios;

5 - Compatibilidade da arquitetura de domínio do *asset* com o modelo de domínio e componentes de domínio;

6 - Conformidade com os padrões de reuso da organização estabelecidos no Programa de Reuso[19] (IEEE1517, 1999).

A seqüência de tarefas da atividade Planejamento consiste em:

PAA-A1-1 - Identificar o tipo de *asset* em função dos artefatos com qualidade aprovada pelo Processo de Controle de Qualidade conforme especificado na **TAB. 3.2.2-1** e registrar pedido de avaliação do *asset*;

PAA-A1-2 - Identificar e especificar os requisitos para avaliação do *asset*, segundo **Critérios de Aprovação** de *asset* supracitados;

PAA-A1-3 - Enviar as especificações de requisitos para avaliação do *asset* ao Subprocesso de Verificação do Processo de Controle de Qualidade para estabelecer uma lista de conferência para avaliar o *asset* baseado na perspectiva dos reutilizadores e selecionar responsáveis qualificados para realização das verificações. Na perspectiva dos reutilizadores,

[19] Formalização de como o reuso deve ser implantado na organização.

os aspectos de reusabilidade e compatibilidade devem ser estabelecidos com seus objetivos definidos e conjunto de perguntas. Estas perguntas, que o verificador deve responder, devem atender aos objetivos do aspecto referente. Para estas listas de conferência devem ser reutilizados modelos aplicáveis, se existirem;

PAA-A1-4 - Determinar os recursos, responsáveis e cronogramas para as verificações dos seguintes *assets*: Domínio, Arquitetura de Domínio, Componente de Domínio e Componentes. Todos os *asset*, exceto Componentes, são produzidos no Processo de Engenharia de Domínio e o *asset* de Componentes é gerado no Processo de Desenvolvimento conforme apresentado na **TAB. 3.2.2-1**. As verificações dos *assets* são para assegurar padronização da documentação, qualidade, reusabilidade, compatibilidade, completude, rastreabilidade, corretude e facilidade no entendimento dos mesmos.

Todas as não-conformidades detectadas devem ser corrigidas pelo responsável pela elaboração do artefato verificado que compõe o *asset* e suas correções verificadas pelo verificador responsável. Durante a atividade Verificação os problemas detectados que fogem ao escopo do responsável pela elaboração do artefato devem ser listados no Relatório de Verificação, assim como as ações decorrentes. Os resultados da verificação devem ser disponibilizados para o Gerente de Reuso e aos órgãos envolvidos.

PAA-A2 AVALIAÇÃO

A partir da qualidade aprovada do *asset* pelo Relatório de Verificação correspondente, o Comitê de Qualidade deve estabelecer credencial de proteção de acesso ao *asset* conforme Contrato e liberar o *asset* e o Relatório de Aprovação do *asset* para o Processo de Gerência de *Assets* catalogar, armazenar, divulgar e administrar o controle de acesso no repositório de *assets* reutilizáveis.

4 ESTUDO DE CASO

Nesta dissertação, é utilizada a estratégia de um estudo de caso por ser um método que consiste na observação de um projeto em andamento, que pode prover subsídios para verificar os prós e contra da implantação de um processo específico em um determinado contexto (WOHLIN et al., 1999).

Jacobson et al. descrevem casos reais de organizações como AT&T, HP e outras que sentiram a necessidade de uma estratégia para implementação de um processo de reuso (JACOBSON et al., 1997). Estas organizações adotaram uma metodologia incremental: desenvolver a partir de um processo gradativo, ou seja, a partir de um projeto piloto. Pois, esta estratégia possibilita aumento em experiências e uma migração cultural passo a passo, minimizando riscos e reduzindo conflitos sociais entre as equipes de desenvolvimento.

Desta forma, uma experiência inicial de implantação da proposta desta dissertação foi realizada com o intuito de viabilizar o processo de Garantia da Qualidade de *Software* para o processo proposto em (GURGEL, 2004). Para tal, foi desenvolvido um projeto piloto - Controlador Tático - para possibilitar, na medida do possível, a avaliação do processo de desenvolvimento baseado em reuso de (GURGEL, 2004) e os possíveis ganhos na qualidade e reusabilidade advindos da utilização da estratégia aqui proposta.

Baseado no guia para estudo de caso de Kitchenham et al., o estudo de caso apresentado neste capítulo segue as etapas de Planejamento, Monitoração e Avaliação (KITCHENHAM et al., 1995). O Planejamento consiste nas seguintes definições para o estudo de caso: contexto, hipótese, condução, duração, participante e objetivos a serem atingidos. Já a Monitoração é o acompanhamento da realização das atividades de cada processo que compõe o processo de Garantia da Qualidade de Software proposto. Além de uma breve informação sobre o acompanhamento, é apresentado um quadro de monitoração contendo a técnica utilizada, o tempo despendido, os fatores de influência (condições, facilidades, dificuldades e/ou limitações encontradas), e análise dos resultados obtidos e do tempo. A última etapa, Avaliação, consiste na análise e interpretação dos resultados obtidos na etapa anterior.

4.1 PLANEJAMENTO DO ESTUDO DE CASO

4.1.1 DEFINIÇÃO DO CONTEXTO DO ESTUDO DE CASO

O projeto piloto – Controlador Tático - foi realizado baseado no projeto real TTI (Terminal Tático Inteligente) do IPqM (Instituto de Pesquisa da Marinha) para aplicação do processo de desenvolvimento de *software* baseado em reuso de (GURGEL, 2004) e do processo de Garantia da Qualidade de *Software* proposto neta dissertação, com o objetivo de criar artefatos reutilizáveis no domínio de controle tático de plataformas e também de

assegurar a qualidade destes artefatos de forma que estes possam ser efetivamente reutilizáveis pelos montadores

Este projeto piloto é adequado por ser típico ao contexto das Três Forças Armadas (Marinha, Exército e Aeronáutica) e também por ser adequado ao reuso. A reusabilidade deste projeto é relatada em (GURGEL, 2004) devido à estabilidade do domínio e grande aplicabilidade nas mais diversas plataformas das Três Forças.

Nós nos apoiamos nos produtos de *software* de (BRAGA et al., 1999; CHEESMAN et al., 2000) para gerar os artefatos dos processos de Engenharia de Domínio e Desenvolvimento do Controlador Tático, que são os artefatos de entrada para o processo proposto por esta dissertação.

Ficou estabelecido que todas as fases dos processos de Engenharia de Domínio e de Desenvolvimento da proposta de (GURGEL, 2004) seriam realizadas de forma a definir primeiramente todos os modelos e diagramas do projeto piloto necessários por cada atividade e, posteriormente, realizaria os artefatos de qualidade adequados a estes modelos e diagramas. Esta decisão foi tomada por dois motivos: aproximação do término do tempo para o estudo de caso da dissertação de (GURGEL, 2004) e mudanças constantes dos artefatos a serem produzidos. Entretanto, paralelamente, foi estabelecendo a técnica de qualidade apropriada ao projeto piloto, Controlador Tático, desenvolvido sob o paradigma OO – grande facilitador ao reuso.

4.1.2 HIPÓTESE DO ESTUDO DE CASO

A hipótese do estudo de caso para esta dissertação foi avaliar até que ponto o processo de Garantia da Qualidade de *Software* se mostrou adequado ao processo de desenvolvimento baseado em reuso de *assets* de (GURGEL, 2004) no domínio de controle tático de plataformas de forma a garantir a qualidade dos artefatos produzidos e prover uma referência para a reusabilidade.

A avaliação deste processo deve ser verificada através de sua implantação ao projeto piloto para obter dados de referência para futuros projetos desenvolvidos sob o processo de (GURGEL, 2004). Estes dados devem servir de base para comparar a qualidade e reusabilidade dos próximos projetos.

Desta forma, os artefatos de qualidade definidos para o projeto piloto devem ser analisados para se verificar a validação da hipótese.

4.1.3 CONDUÇÃO DO ESTUDO DE CASO

O projeto piloto seguiu o processo de Garantia de Qualidade de *Software* proposto no capítulo anterior, tendo como subsídio os artefatos reusáveis definidos para o Estudo de Caso de (GURGEL, 2004). Cada subprocesso foi conduzido conforme a etapa Monitoração de (KITCHENHAM et al., 1995) apresentada na seção 4.2.

4.1.4 DURAÇÃO DO ESTUDO CASO

Ficou estabelecido o prazo de 4 (quatro) meses para realização do estudo de caso, compreendendo de fevereiro a maio de 2004. A distribuição do tempo se deu conforme apresentado na **FIG. 4.1.4-1**.

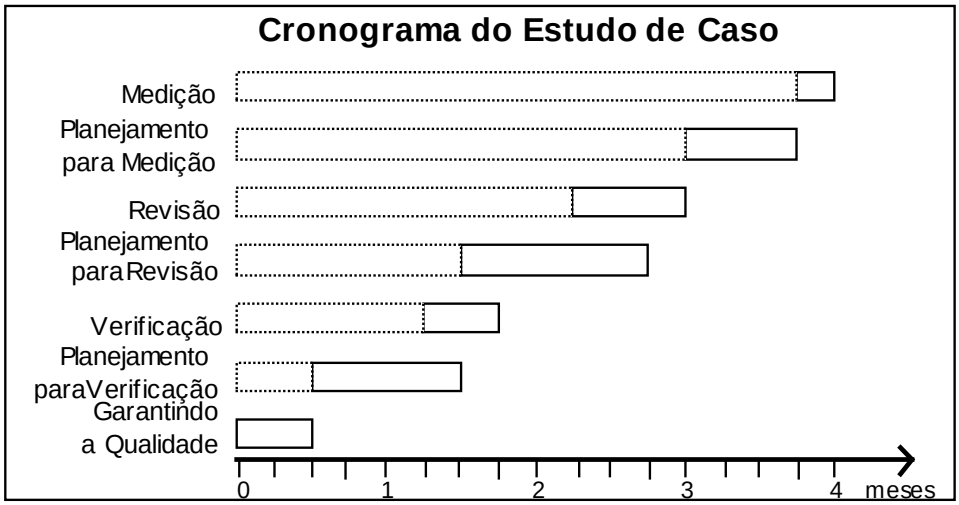


FIG. 4.1.4-1: Cronograma do estudo de casos do protótipo Controlador Tático.

4.1.5 PARTICIPANTES DO ESTUDO CASO

O projeto piloto teve apenas um consultor de qualidade e uma equipe acadêmica com 8 (oito) participantes para realização das técnicas de qualidade.

4.1.6 OBJETIVO DO ESTUDO DE CASO

O objetivo deste caso de uso é a redução de defeitos nos artefatos produzidos no processo de desenvolvimento de *software* baseado em reuso sob a ótica dos desenvolvedores e clientes, e o ganho dos atributos de qualidade como a reusabilidade, flexibilidade e compreensibilidade.

4.2 MONITORAÇÃO DO ESTUDO DE CASO

4.2.1 ETAPA: GARANTINDO A QUALIDADE

Para realização do processo de Garantia da Qualidade de *Software*, alguns modelos foram estabelecidos antes da execução das atividades do processo de Controle de Qualidade, a saber: Plano de Garantia da Qualidade de *Software* do Projeto (PGQSP), Cronograma para verificações / revisões de GQS do projeto, Histórico de Versões e Questionário de Sugestões para Melhoria da Garantia da Qualidade (QSMGQ). Os modelos foram realizados da seguinte forma:

- **Técnica Utilizada –** Consulta a consultores de qualidade de organizações que já têm introdução da prática de qualidade nos processos de desenvolvimento de projetos e pesquisa na literatura sobre modelos de qualidade.
- **Tempo Despendido –** Duas semanas.
- **Fatores de Influência –** Houve um ponto favorável: facilidade de acesso a consultores de qualidade de duas grandes organizações como SERPRO-RJ e SIEMENS-RJ.
- **Avaliação dos Resultados e do Tempo –** Foram realizadas no total de três entrevistas com os consultores de qualidade e algumas consultas por telefone. As entrevistas duraram em média 2 horas cada. Nas entrevistas, foram levantados vários tipos de artefatos de qualidade apontados pela Norma ISO9000:2000 (ABNT9000, 2001), mas nenhum dos consultores relatou sobre algum procedimento de avaliação dos produtos de qualidade.

Nesta etapa da Monitoração só foram definidos os modelos que nas etapas seguintes foram devidamente preenchidos.

Para o Plano de Garantia de Qualidade de *Software* do Projeto (PGQSP) foi utilizado um modelo que está apresentado na **FIG. 4.2.1-1**.

PGQSP - Plano de Garantia da Qualidade de *Software* do Projeto	
Objetivo:	Definir a estrutura de GQS adotada para o projeto, a forma de atuação das verificações / revisões, o cronograma das verificações / revisões para as atividades selecionadas e estimativas de esforço empregadas pelo consultor de GQS e equipe de desenvolvimento no processo de Verificação / Revisão.
Responsável:	
***Templates*:**	
Ferramentas:	
Exemplos:	
Listas de Verificação / Revisão:	
Técnica utilizada:	
Orientações Técnicas:	
Opcionalidade:	
Entrada para Atividade:	**Saída da Atividade:**

FIG. 4.2.1-1: Plano de GQS do Projeto (PGQSP).

Na **FIG. 4.2.1-2** é apresentado um modelo de Histórico de Versões para ser introduzido em todos artefatos produzidos com a data da versão, número seqüencial da versão, descrição sucinta da versão, autor responsável pela versão, o nome do revisor e por quem foi aprovado.

Data	Versão	Descrição	Autor	Revisor	Aprovado por
<dd/mm/aaaa>	<x.x>	<detalhes>	<nome>	<nome>	<nome>

FIG. 4.2.1-2: Histórico de versões.

O cronograma para as verificações / revisões de GQS dos artefatos do projeto piloto deve ser inserido no cronograma do projeto piloto, obedecendo aos itens apresentados na **FIG. 4.2.1-3.**

Nº Verificação /Revisão	Atividade	Tarefa	Data de Início	Data de Término	Esforço (homens/hora)	
					GQS	Equipe de Projeto
<Número seqüencial de verificações /revisões do projeto>	<Atividade em que se dá a verificação /revisão de GQS>	<Tarefa da Atividade a ser verificada /revisada>	<Data de início da verificação /revisão>	<Data de término da verificação /revisão>	<quantidade em homens-hora gastos pelo consultor de GQS>	<quantidade em homens-hora gastos pelos desen-volvedores>

FIG. 4.2.1-3: Cronograma para as verificações e revisões de GQS do projeto.

De acordo com a Norma ISO 9000:2000 (ABNT9000, 2001), foi estabelecido o QSMGQ para prover a melhoria do processo de qualidade de forma que o Comitê de Qualidade possa obter um *feedback* do grupo de qualidade. O Questionário de Sugestões para Melhoria da Garantia da Qualidade é apresentado na **FIG. 4.2.1-4.**

Questionário de Sugestões para Melhoria da Garantia da Qualidade		
Questões	**Respostas e Sugestões**	
1 - A verificação / revisão atingiu os objetivos do grupo da qualidade? Se não, por quê?		
2 – O grupo da qualidade sente que contribuiu para melhorar a qualidade do artefato ou *asset* pela verificação / revisão?		
3 - Todos tiveram tempo suficiente para fazer preparação? Se não, quanto tempo precisa?		
4 - Há necessidade de conhecimento específico para utilizar o *checklist* / técnica de leitura para este tipo de artefato / *asset* durante a preparação?		
5 - Foi útil para detectar não-conformidades?		
6 - O *checklist* / técnica de leitura pode ser melhorado? Caso afirmativo, como?		
7 - No subprocesso de Revisão, todos os participantes estavam presentes? Se não, quem estava ausente? Indicar e justificar se não havia necessidade.		
8 - Os procedimentos para as verificações / revisões foram seguidos corretamente?		
9 - No subprocesso de Revisão, como as reuniões poderiam melhorar?		
10 - Há necessidade de ajuda para realizar efetivamente *checklist* / técnica de leitura?		
11 - Há alguma outra sugestão para melhorar o Processo de Garantia da Qualidade?		
Data: / /	Nome:	Rubrica:

FIG. 4.2.1-4: Questionário de Sugestões para Melhoria da Garantia da Qualidade.

4.2.2 ETAPA: SUBPROCESSO DE VERIFICAÇÃO

4.2.2.1 PLANEJAMENTO DA VERIFICAÇÃO

O Planejamento do processo de Verificação foi realizado da seguinte forma:

- **Técnica Utilizada -** Os modelos definidos na seção 4.2.1 e os procedimentos propostos em SPV-A1 da seção 3.2.1.2.1, que inclui a definição dos *checklists* baseado em aspectos para os artefatos Necessidades do Cliente e Glossário, Arquitetura de Domínio, Código-fonte e Casos de Testes de Qualificação e os relatórios correspondentes.
- **Tempo Despendido** - Quatro semanas.
- **Fatores de Influência -** Foi utilizado uma planilha de texto (Microsoft Excel) que atendeu inteiramente ao propósito das listas de conferência (*checklists*).
- **Avaliação dos Resultados e do Tempo -** Apesar da facilidade da utilização de uma planilha de texto, a elaboração das listas de conferência demandou um tempo acima do estimado (uma semana a mais).

Como as listas de conferências seguiram a proposta apresentada no Capítulo 3, então foram preenchidos para cada artefato a ser verificado os modelos PGQSP, Cronograma e Histórico de Versões. Além disso, foram elaborados os *checklists* (veja na seção 7.4.1 do APÊNDICE 4) e os modelos de relatórios correspondentes de acordo com o artefato a ser verificado. A seguir na **FIG. 4.2.2.1-1** são apresentados os modelos supracitados em **(a) (b) (c)**, em **(d)** a lista de conferência e em **(e)** o Relatório de Verificação para Necessidades do Cliente e Glossário.

PGQSP - Plano de Garantia da Qualidade de *Software* do Projeto	
Objetivo:	Definir a estrutura de GQS adotada para o projeto piloto, Controlador Tático.
Responsável:	Consultor de GQS (CGQS).
***Templates*:**	VENC001/03..\ESTUDO-Caso\CT_Qualidade\ARTEFATOS-KLEYNA\Modelos_Checklist_v1.5..xls
Ferramentas:	Microsoft Excel
Exemplos:	..\ESTUDO-Caso\CT_Qualidade\ARTEFATOS-KLEYNA\EXEMPLOS\5_Checklist_Necessidades_Cliente_v1.5..xls
Listas de Verificação:	VENC001/03, VEAR001/03, VEPT001/03, VETU001/03, VETI001/03, VETG001/03, VETV001/03, VECJ001/03
Técnica utilizada:	*Checklists* baseados em aspectos.
Orientações Técnicas:	Na própria lista de conferência.
Opcionalidade:	Obrigatória.
Entrada para Atividade: Verificação	**Saída da Atividade:** Planejamento

FIG. 4.2.2.1-1: (a) PGQSP do projeto Controlador Tático.

Cronograma						
Código da Verificação	**Atividade**	**Tarefa**	**Data de Início**	**Data de Término**	**Esforço (homens/hora)**	
					GQS	**Verificador**
VENC001/03	Preparação do Pedido da Proposta	Elaboração das Necessidades do Cliente	09/12/2003	09/12/2003	0 h	1 h

FIG. 4.2.2.1-1: (b) Cronograma de verificação do artefato Necessidades do Cliente e Glossário.

Histórico de Revisões do *Checklist*
para Necessidades do Cliente e Glossário

Data	Versão	Descrição	Autor	Revisor	Aprovado por
21/11/2003	1.0	Elaboração do *checklist* com os aspectos e questões.	KLEYNA		
30/11/2003	1.1	Re-estruturação das respostas dos itens do *checklist*.	KLEYNA		
6/12/2003	1.2	Planilha para suporte.	KLEYNA		
9/12/2003	1.3	Inserção de espaço para comentário e identificação de cada defeito detectado.	KLEYNA		
23/12/2003	1.4	Alteração das colunas de respostas, verso do *checklist* e análise da verificacão.	KLEYNA		

Comentários sobre as versões:

FIG. 4.2.2.1-1: (c) Histórico de Versões do *checklist* para Necessidades do Cliente e Glossário.

***Checklist*: Necessidades do Cliente e Glossário**

Artefato Verificado: CTNC001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

1. Aspecto: Apresentação do Documento

Este aspecto deve analisar as questões gerais de apresentação do documento.	P	C	I
1.1 O documento está de acordo com o *template* padrão?	s	s	
1.2 O documento teve ortografia e gramática checada?	s	s	
1.3 O documento está livre de erros de *layout*?	s	s	
1.4 O glossário dispõe de todas as definições e explicações que o revisor necessita para o completo entendimento das necessidades e características desejadas?	s	s	
1.5 Os números das linhas do texto do documento estão impressos para facilitar a referência de localização específica durante a verificação?	s	s	

Total de itens do aspecto verificados:	5
Total de não-conformidades:	

FIG. 4.2.2.1-1: (d) Parte do *checklist* do artefato Necessidades do Cliente e Glossário.

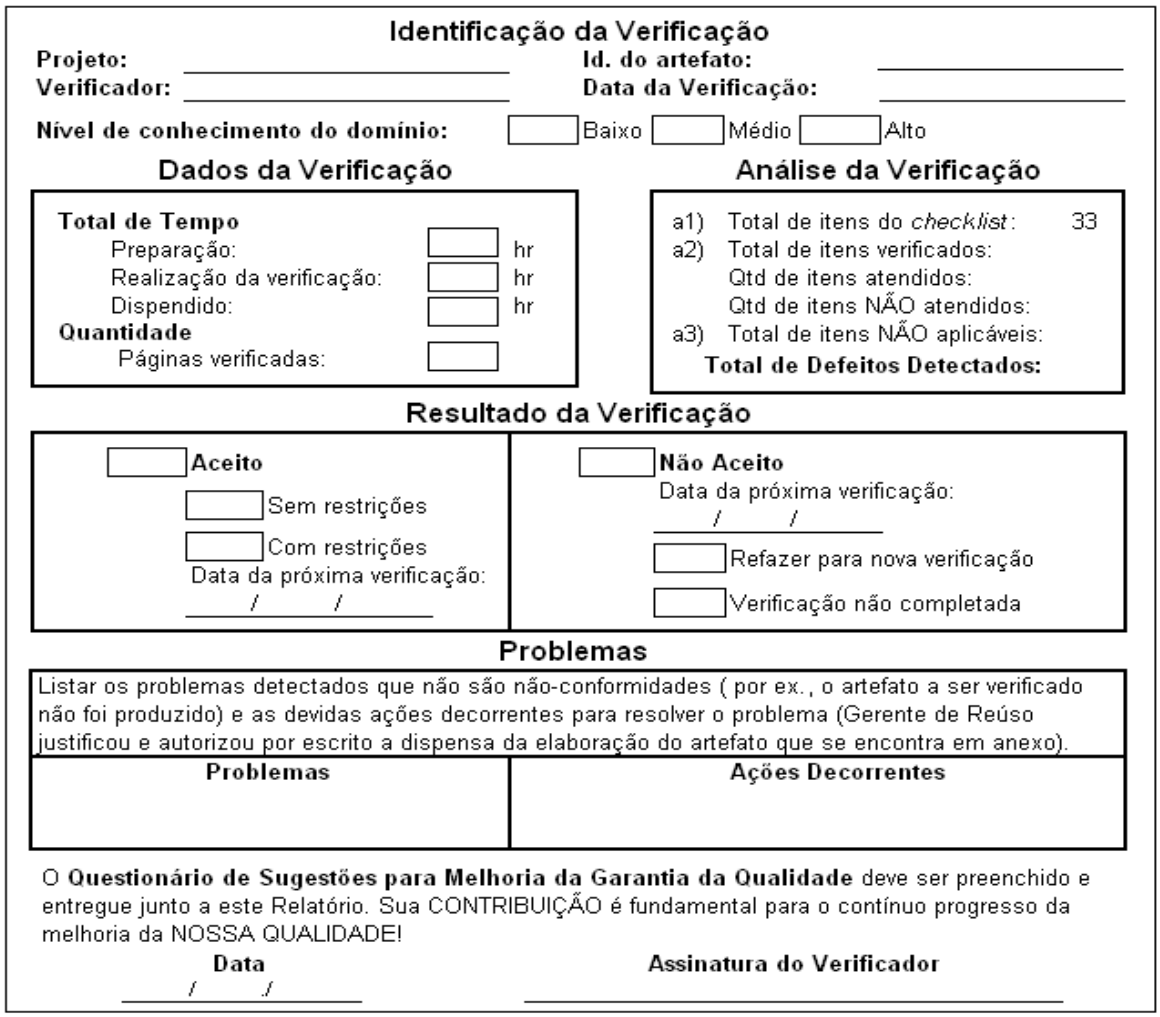


FIG. 4.2.2.1-1: (e) Relatório de Verificação para Necessidades do Cliente e Glossário.

4.2.2.2 VERIFICAÇÃO

A Verificação foi realizada da seguinte forma:

- **Técnica Utilizada –** As listas de conferências (*checklists*) elaboradas no Planejamento da Verificação baseadas em aspectos para os artefatos Necessidades do Cliente e Glossário, Arquitetura de Domínio, Código-fonte e Casos de Testes de Qualificação e os relatórios correspondentes conforme descrito na seção 3.2.1.2.1.
- **Tempo Despendido –** Em média uma hora para cada artefato no decorrer de duas semanas.
- **Fatores de Influência –** A execução da aplicação dos *checklists* foi facilitada pela disponibilidade de um grupo acadêmico de pós-graduação em Ciência da Computação.

- **Avaliação dos Resultados e do Tempo -** Apesar da falta de conhecimento por parte dos verificadores do domínio do projeto piloto, Controlador Tático, estes não tiveram dificuldades na realização da verificação e expressaram certo conforto quanto à utilização dos *checklists*. Também, foi verificado o grau de conhecimento de cada verificador, por exemplo, conhecimento em OO e *design patterns*, para atribuir o artefato apropriado ao conhecimento do verificador.

Na **FIG. 4.2.2.2-1 (a)** é apresentada parte da aplicação do *checklist* baseado em aspecto, em **(b)** as anotações das não-conformidades detectadas para Necessidades do Cliente e Glossário, e em **(c)** o respectivo Relatório da Verificação.

***Checklist*: Necessidades do Cliente e Glossário**

Artefato Verificado: CTNC001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

	1. Aspecto: Apresentação do Documento Este aspecto deve analisar as questões gerais de apresentação do documento.	P	C	I
1.1	O documento está de acordo com o *template* padrão?	s	n	1
1.2	O documento teve ortografia e gramática checada?	s	s	
1.3	O documento está livre de erros de *layout* ?	s	s	
1.4	O glossário dispõe de todas as definições e explicações que o revisor necessita para o completo entendimento das necessidades e características desejadas?	s	s	
1.5	Os números das linhas do texto do documento estão impressos para facilitar a referência de localização específica durante a verificação?	s	s	

Total de itens do aspecto verificados:	5
Total de não-conformidades:	1

	2. Aspecto: Qualidade Este aspecto descreve a qualidade que as necessidades, isto é, as funcionalidades e as características desejadas, devem apresentar no documento.	P	C	I
2.1	As necessidades apresentam nível de detalhamento adequado?	s	s	
2.2	Todas as características de desempenho estão corretamente especificadas?	s	s	
2.3	Toda as considerações de segurança estão corretamente especificadas?	s	s	
2.4	Outras características desejadas de qualidade pertinentes estão explicitamente documentadas e quantificadas de acordo com o objetivo especificado?	s	s	

Total de itens do aspecto verificados:	4
Total de não-conformidades:	0

	3. Aspecto: Completude Este aspecto descreve o que o documento deve apresentar com relação à consistência das necessidades e a completude destes documentos.	P	C	I
3.1	As funcionalidades e características desejadas provêem uma base adequada para análise de requisitos da engenharia de domínio?	s	s	
3.2	O documento Necessidades do Cliente inclui todas as funcionalidades do sistema?	s	s	
3.3	O documento Necessidades do Cliente inclui todas as características desejadas que o sistema precisa ter?	s	s	
3.4	Todas as referências internas das necessidades estão corretas?	s	s	
3.5	As necessidades fornecem uma base adequada para o projeto?	s	s	
3.6	A prioridade de implementação das necessidades está definida?	s	s	
3.7	Todo *hardware* , *software* e interfaces de comunicação estão definidos?	s	n	2
3.8	Os algoritmos intrínsecos para as necessidades funcionais estão definidos?	s	s	
3.9	A necessidade inclui todo o conhecimento do cliente ou necessidades de sistema?	s	s	
3.10	O comportamento esperado está documentado para todas as condições de erro?	s	s	
3.11	Todas as funções críticas de tempo estão identificadas e seus critérios de tempo especificados?	s	s	

Total de itens do aspecto verificados:	11
Total de não-conformidades:	1

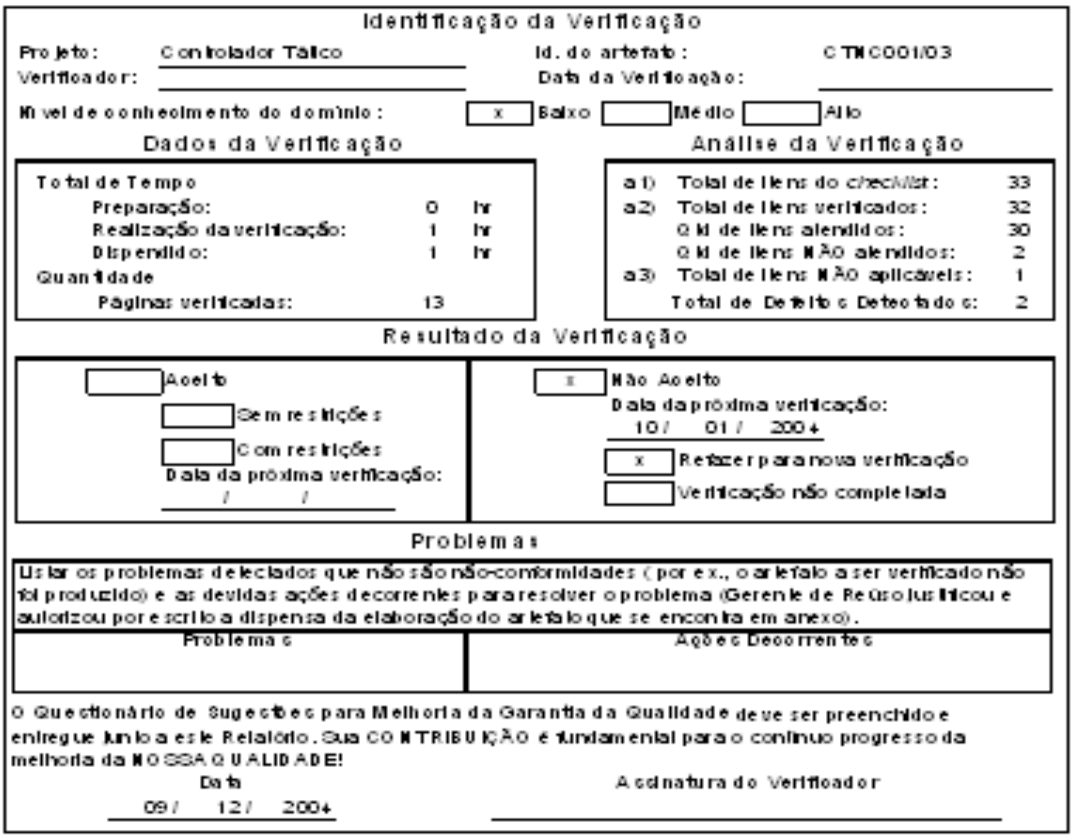
FIG. 4.2.2.2-1: (a) Itens do *Checklist* de Necessidades do Cliente e Glossário.

Não - Conformidades Detectadas
Artefato Verificado: CTNC001/03

Identificar e comentar brevemente os defeitos detectados. O número de identificação (Id) deve ser o mesmo fornecido na coluna I (Identificação) do *checklist*.

Id	Comentários
1	Os itens 5.8, 5.9, 5.11, 6.5.5 e 6.5.6 estão em desacordo em Necessidades do Cliente.
2	Falta definição dos tipos de monitores a serem usados.

FIG. 4.2.2.2-1: (b) Comentários dos itens do *Checklist* de Necessidades do Cliente e Glossário que não estão em conformidades.



Identificação da Verificação

Projeto: Controlador Tático — Id. do artefato: CTNC001/03

Verificador: — Data da Verificação:

Nível de conhecimento do domínio: [x] Baixo [ ] Médio [ ] Alto

Dados da Verificação

Total de Tempo
Preparação: 0 hr
Realização da verificação: 1 hr
Dispendido: 1 hr
Quantidade
Páginas verificadas: 13

Análise da Verificação

a1) Total de itens do checklist: 33
a2) Total de itens verificados: 32
Qtd de itens atendidos: 30
Qtd de itens NÃO atendidos: 2
a3) Total de itens NÃO aplicáveis: 1
Total de Defeitos Detectados: 2

Resultado da Verificação

[ ] Aceito
[ ] Sem restrições
[ ] Com restrições
Data da próxima verificação: / /

[x] Não Aceito
Data da próxima verificação: 10 / 01 / 2004
[x] Refazer para nova verificação
[ ] Verificação não completada

Problemas

Listar os problemas detectados que não são não-conformidades (por ex., o artefato a ser verificado não foi produzido) e as devidas ações decorrentes para resolver o problema (Gerente de Projeto justificou e autorizou por escrito a dispensa da elaboração do artefato que se encontra em anexo).

Problemas	Ações Decorrentes

O Questionário de Sugestões para Melhoria da Garantia da Qualidade deve ser preenchido e entregue junto a este Relatório. Sua CONTRIBUIÇÃO é fundamental para o contínuo progresso da melhoria da NOSSA QUALIDADE!

Data: 09 / 12 / 2004

Assinatura do Verificador: _____

FIG. 4.2.2.2-1: (c) Relatório de Verificação de Necessidades do Cliente e Glossário.

4.2.3 ETAPA: SUBPROCESSO DE REVISÃO

4.2.3.1 PLANEJAMENTO DA REVISÃO

O Planejamento da Revisão foi realizada da seguinte forma:

- **Técnica Utilizada –** Os modelos definidos na seção 4.2.1 e os procedimentos propostos em SPR-A1 da seção 3.2.1.2.2, que inclui a definição dos procedimentos para a Técnica de Leitura Horizontal e Vertical para os artefatos Modelo de Contexto do Domínio, Modelo Conceitual do Domínio, Modelo de Casos de Uso e Cenários, Modelo de Características, Modelo Tipo do Negócio, Interfaces da Camada de Negócio, Arquitetura de Domínio, Especificação de Componentes do Domínio e os relatórios correspondentes.
- **Tempo Despendido –** Cinco semanas.
- **Fatores de Influência –** A utilização de um editor de texto MS Word atendeu ao propósito da construção dos procedimentos das técnicas de leitura.
- **Avaliação dos Resultados e do Tempo –** Na elaboração dos procedimentos das técnicas de leitura (horizontal e vertical) ocorreram algumas dificuldades. Estas dificuldades foram quanto à elaboração da sistemática de passos para validar o Modelo de Característica, que por se tratar de um modelo abrangente, contendo todas os requisitos, os funcionais e não-funcionais, necessitando de um especialista do domínio. Também foi sentida a necessidade de uma verificação quanto aos aspectos da viabilidade e padrões de projeto e interfaces e clareza para Arquitetura de Domínio.

Como as inspeções individuais seguiram a proposta apresentada no Capítulo 3, então foram preenchidos para cada artefato a ser revisado os modelos PGQSP, Cronograma e Histórico de Versões. Além disso, foram elaborados as Técnicas de Leitura Horizontais e Verticais de acordo com o artefato a ser inspecionado (veja seções 7.4.2 e 7.4.3 do APÊNDICE 4) e o modelo de Relatório da Revisão. Na **FIG. 4.2.3.1-1 (a)** é apresentado um dos modelos da Técnica de Leitura Vertical (TLV1) e em **(b)** o modelo de Relatório da Revisão.

1 - Caso encontre, verifique se todos os fluxos de dados do domínio com esta entidade externa estão capturados. Realce em amarelo os fluxos de dados e certifique que esta entidade externa com os fluxos de dados esteja devidamente documentada na descrição do Modelo de Contexto do Domínio. Caso contrário, preencha a Tabela de Discrepância, falta ou está incompleta fluxo de dados no modelo de contexto ou sua descrição.

2 - Para cada fluxo de dado identificado, verificar se o nome está no padrão (todo em minúsculo) e é o mesmo utilizado pela entidade externa. Caso contrário, preencha a Tabela de Discrepância, nome do fluxo de dados não está adequado.

B. Verifique se todas as necessidades estão sendo encapsuladas dentro do escopo do domínio.

1 - Certifique que o mesmo conjunto de necessidades esteja presente na descrição do contexto do domínio. Caso contrário, preencha a Tabela de Discrepância, pois informação está presente em um documento, mas ausente no outro.

II. Reveja o Modelo Conceitual do Domínio para assegurar que todas as entidades externas e seus relacionamentos estão sendo levados em consideração.

ENTRADAS: Entidades externas marcadas com asterisco azul
Relacionamentos realçados em amarelo

SAÍDAS: Relatório de Discrepâncias

A. Reveja as entidades externas e seus relacionamentos para ter certeza que todo o conjunto de entidades e seus relacionamentos aparecem descritos no documento de Necessidades do Cliente e Glossário.

1 - Certifique que não exista nenhuma entidade externa sem asterisco e relacionamento sem estar realçado em amarelo. Caso exista algum, preencha a Tabela de Discrepância, pois uma entidade externa ou relacionamento no Modelo de Contexto do Domínio não está presente no documento Necessidades do Cliente e Glossário.

FIG. 4.2.3.1-1: (a) Técnica de Leitura Vertical (TLV1).

Relatório da Revisão

Id. Projeto: **Técnicas de Leitura:**
Data da revisão: **Início (hr):** **Término (hr):**
Revisores: **Duração da revisão:** hr

Papel	Nome	Rubrica	Nível de Conhecimento (0-baixo/1-médio/2-alto)
Líder			
Expositor			
Secretário			
Autor(es)			
Revisor(es)			
		Média do conhencimento:	

Nome e tipo dos documentos revisados: Doc1:
Doc2:
Total de discrepâncias:
Resultado da Revisão

Aceito Sem restrições Com restrições Data da próxima revisão: / /	**Não Aceito** Data da próxima revisão: / / Refazer para nova revisão Revisão não completada

Problemas

Listar os problemas detectados que não são não-conformidades (por exemplo, o artefato a ser verificado não foi produzido) e as devidas ações decorrentes para resolver o problema (o Gerente de Reúso justificou e autorizou por escrito a dispensa da elaboração do artefato que se encontra em

Problemas	Ações Decorrentes

O Questionário de Sugestões para Melhoria da Garantia da Qualidade deve ser preenchido e entregue junto a este Relatório. Sua CONTRIBUIÇÃO é fundamental para o contínuo progresso da melhoria da NOSSA QUALIDADE!

FIG. 4.2.3.1-1: (b) Relatório da Revisão.

4.2.3.2 REVISÃO

A Revisão foi realizada da seguinte forma:

- **Técnica Utilizada** – As técnicas de leitura elaboradas no Planejamento da Revisão baseadas nas Técnicas de Leitura OO para os artefatos produzidos na Engenharia de Domínio (veja ANEXO 2) conforme apresentadas na **FIG. 3.1.2.2-1**.
- **Tempo Despendido** – Uma média de três horas por cada reunião no decorrer de três semanas.
- **Fatores de Influência** – A falta de conhecimento do domínio do projeto piloto dificultou o entendimento por parte dos revisores para realização da inspeção individual, requerendo, assim, mais tempo de treinamento para explicar mais detalhadamente o domínio do Controlador Tático. Por outro lado, como o Consultor de Qualidade já tem alguma experiência, mesmo que academicamente, com este tipo de reunião (descrita na seção 3.2.1.2.2) foi um fator positivo na integração e condução das reuniões, exercendo o papel de líder – moderador.
- **Avaliação dos Resultados e do Tempo -** Foi observado, inicialmente, várias dúvidas e insegurança por parte dos revisores individuais na realização da Técnica de Leitura. As dúvidas eram relativas ao domínio da aplicação e a insegurança devido à utilização de um procedimento novo. Isto resultou na necessidade de um treinamento mais específico, ou seja, realização em grupo – grupo de revisores – de uma experiência prática com um artefato e também uma explicação mais detalhada sobre o domínio do problema – Controlador Tático.

Na **FIG. 4.2.3.2-1** é apresentada a Tabela de Discrepância para Técnica de Leitura Vertical em **(a)** e em **(b)** a Técnica de Leitura Horizontal, assim como o Detalhamento de Discrepância para Técnica de Leitura em **(c**) e em **(d)** o QSMGQ (Questionário de Sugestões para Melhoria da Garantia da Qualidade) preenchido pelo revisor de TLH1, após a realização da inspeção individual para TLV1 e TLH1 respectivamente.

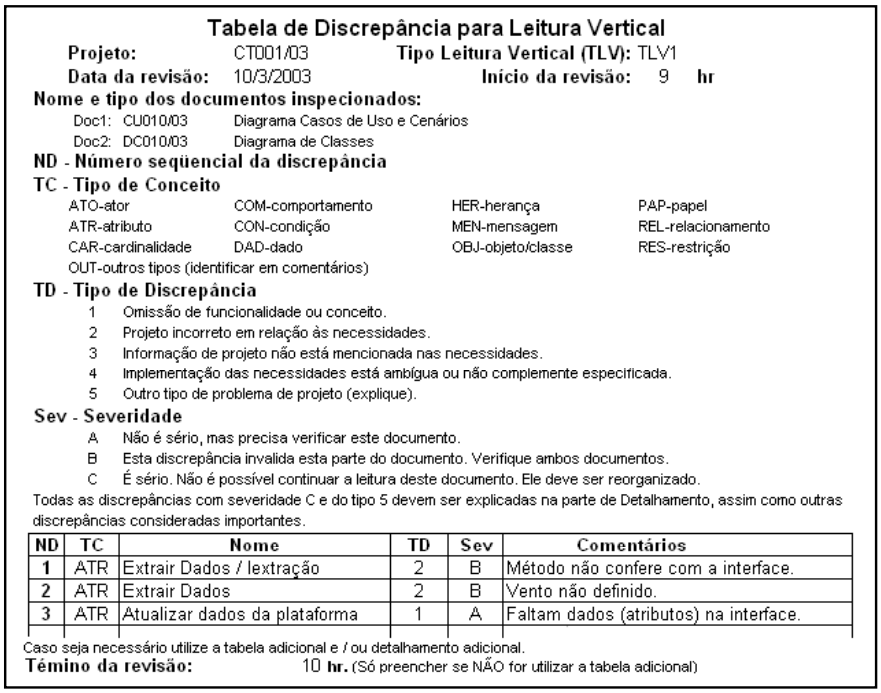


Tabela de Discrepância para Leitura Vertical

Projeto: CT001/03 **Tipo Leitura Vertical (TLV):** TLV1
Data da revisão: 10/3/2003 **Início da revisão:** 9 **hr**

Nome e tipo dos documentos inspecionados:
Doc1: CU010/03 Diagrama Casos de Uso e Cenários
Doc2: DC010/03 Diagrama de Classes

ND - Número seqüencial da discrepância

TC - Tipo de Conceito

ATO-ator	COM-comportamento	HER-herança	PAP-papel
ATR-atributo	CON-condição	MEN-mensagem	REL-relacionamento
CAR-cardinalidade	DAD-dado	OBJ-objeto/classe	RES-restrição

OUT-outros tipos (identificar em comentários)

TD - Tipo de Discrepância
1 Omissão de funcionalidade ou conceito.
2 Projeto incorreto em relação às necessidades.
3 Informação de projeto não está mencionada nas necessidades.
4 Implementação das necessidades está ambígua ou não complemente especificada.
5 Outro tipo de problema de projeto (explique).

Sev - Severidade
A Não é sério, mas precisa verificar este documento.
B Esta discrepância invalida esta parte do documento. Verifique ambos documentos.
C É sério. Não é possível continuar a leitura deste documento. Ele deve ser reorganizado.

Todas as discrepâncias com severidade C e do tipo 5 devem ser explicadas na parte de Detalhamento, assim como outras discrepâncias consideradas importantes.

ND	TC	Nome	TD	Sev	Comentários
1	ATR	Extrair Dados / lextração	2	B	Método não confere com a interface.
2	ATR	Extrair Dados	2	B	Vento não definido.
3	ATR	Atualizar dados da plataforma	1	A	Faltam dados (atributos) na interface.

Caso seja necessário utilize a tabela adicional e / ou detalhamento adicional.
Témino da revisão: 10 **hr.** (Só preencher se NÃO for utilizar a tabela adicional)

FIG. 4.2.3.2-1: (a) Tabela de Discrepância para Técnica de Leitura Vertical.

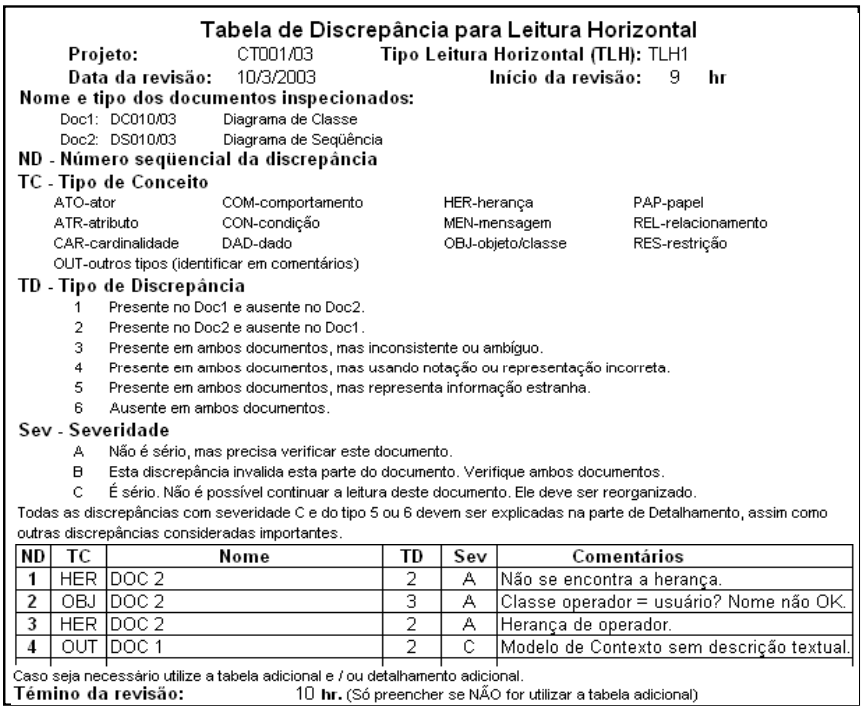


Tabela de Discrepância para Leitura Horizontal

Projeto: CT001/03 **Tipo Leitura Horizontal (TLH):** TLH1
Data da revisão: 10/3/2003 **Início da revisão:** 9 **hr**

Nome e tipo dos documentos inspecionados:
Doc1: DC010/03 Diagrama de Classe
Doc2: DS010/03 Diagrama de Seqüência

ND - Número seqüencial da discrepância

TC - Tipo de Conceito

ATO-ator	COM-comportamento	HER-herança	PAP-papel
ATR-atributo	CON-condição	MEN-mensagem	REL-relacionamento
CAR-cardinalidade	DAD-dado	OBJ-objeto/classe	RES-restrição

OUT-outros tipos (identificar em comentários)

TD - Tipo de Discrepância
1 Presente no Doc1 e ausente no Doc2.
2 Presente no Doc2 e ausente no Doc1.
3 Presente em ambos documentos, mas inconsistente ou ambíguo.
4 Presente em ambos documentos, mas usando notação ou representação incorreta.
5 Presente em ambos documentos, mas representa informação estranha.
6 Ausente em ambos documentos.

Sev - Severidade
A Não é sério, mas precisa verificar este documento.
B Esta discrepância invalida esta parte do documento. Verifique ambos documentos.
C É sério. Não é possível continuar a leitura deste documento. Ele deve ser reorganizado.

Todas as discrepâncias com severidade C e do tipo 5 ou 6 devem ser explicadas na parte de Detalhamento, assim como outras discrepâncias consideradas importantes.

ND	TC	Nome	TD	Sev	Comentários
1	HER	DOC 2	2	A	Não se encontra a herança.
2	OBJ	DOC 2	3	A	Classe operador = usuário? Nome não OK.
3	HER	DOC 2	2	A	Herança de operador.
4	OUT	DOC 1	2	C	Modelo de Contexto sem descrição textual.

Caso seja necessário utilize a tabela adicional e / ou detalhamento adicional.
Témino da revisão: 10 **hr.** (Só preencher se NÃO for utilizar a tabela adicional)

FIG. 4.2.3.2-1: (b) Tabela de Discrepância para Técnica de Leitura Horizontal.

Detalhamento de Discrepância para Leitura Horizontal

Projeto: CT001/03 **TLH: TLH1**

Data-revisão: 10/3/2003

Detalhar as discrepâncias sérias e dos tipos 5 e 6 conforme a estrutura a seguir:

ND	Descrição Detalhada
4	Sem descrição textual do Modelo Contexto dificulta o entendimento.

FIG. 4.2.3.2-1: (c) Detalhamento de Discrepância para Técnica de Leitura.

Garantia da Qualidade de *Software*
Questionário de Sugestões para Melhoria da Garantia da Qualidade — Versão:1.0
GQQS001/04 — Data da versão: 01/03/2004

QUESTIONÁRIO DE SUGESTÕES PARA MELHORIA DA GARANTIA DA QUALIDADE

O Questionário de Sugestões para Melhoria da Garantia da Qualidade tem como objetivo capturar lições aprendidas nas atividades de revisões e verificações dos artefatos e / ou *assets*. O preenchimento deste questionário não deve ultrapassar a quinze minutos, e deve ser realizado pelo verificador / líder da equipe de revisão. Lembre-se, sua contribuição é muito importante!

Questões	Respostas e Sugestões
1 - A verificação / revisão atingiu os objetivos dos consultores da qualidade? Se não, por quê?	Não, pois o Modelo de Contexto não possui uma descrição textual, fazendo com que não tenha dados suficientes para validação.
2 - A equipe de consultores da qualidade sente que eles contribuíram para melhorar a qualidade do artefato ou *asset* pela verificação / revisão?	Sim.
3 - Todos tiveram tempo suficiente para fazer preparação? Se não, quanto tempo precisa?	Não, pois a maioria não possui conhecimento suficiente sobre o mini-mundo – Controlador Tático.
4 - Há necessidade de conhecimento específico para utilizar o *checklist* / técnica de leitura para este tipo de artefato / *asset* durante a preparação?	Sim, deveria ter um conhecimento sobre o Controlador Tático.
5 - Foi útil para detectar não-conformidades?	Sim.
6 - O *checklist* / técnica de leitura podem ser melhorados? Se sim, como?	Sim, caso venha acompanhado de uma descrição dos modelos.
7 - No subprocesso de Revisão, todos os participantes estavam presentes? Se não, quem estava ausente? Indicar e justificar se não havia necessidade.	Sim.
8 - Os procedimentos para as verificações / revisões foram seguidos corretamente?	Sim, sendo que alguns não foram seguidos por falta de informação textual do modelo.
9 - No subprocesso de Revisão, como as reuniões conjuntas poderiam melhorar?	Na parte do treinamento, quanto a explicação sobre domínio da aplicação.
10 - Há necessidade de ajuda para realizar efetivamente *checklist* / técnica de leitura?	Sim.
11 - Há alguma outra sugestão para melhorar o Processo Garantia da Qualidade?	Não.

Data: 31 / 03 / 2004	Nome: Ana Carolina	Rubrica:

Restrito — Instituto Militar de Engenharia — Página: 1/1

FIG. 4.2.3.2-1: (d) Questionário de Sugestões para Melhoria da Garantia da Qualidade (QSMGQ) preenchido pelo revisor de TLH1.

4.2.4 ETAPA: SUBPROCESSO DE MEDIÇÃO E AVALIAÇÃO

4.2.4.1 PLANEJAMENTO DA MEDIÇÃO

O Planejamento do processo de Medição foi realizado da seguinte forma:

- **Técnica Utilizada -** Foi utilizado o modelo GQM (*Goal*/*Question*/*Metric*) de Basili (veja seção 2.1.3.2.1) para construção dos indicadores e, também, com base na Construção Mensurável do PSM (*Practical Software Measurement*) de McGarry et al. (veja seção

2.1.3.2.2) para elaboração da especificação dos indicadores, foram estabelecidos os itens chaves.

- **Tempo Despendido** - Duas semanas.
- **Fatores de Influência -** A utilização de um editor de texto MS Word atendeu ao propósito da elaboração dos formulários para especificação dos indicadores e implantação da metodologia GQM.
- **Avaliação dos Resultados e do Tempo –** A elaboração dos indicadores foi voltada para avaliar a qualidade dos artefatos dos processos de Engenharia de Domínio e Desenvolvimento de (GURGEL, 2004) que compõe os *assets* produzidos por estes processos, como também, avaliar o processo de controle de qualidade quanto às inspeções realizadas via *checklist* ou técnicas de leitura e sua produtividade.

Os objetivos apresentados são os *Goals*, que a partir destes foram levantadas as *Questions* para se chegar às métricas relativas as *Questions*. Desta forma, foram selecionadas as métricas mais relevantes, apresentadas acima junto aos *Goals* para a construção dos indicadores. Depois, baseado na Construção Mensurável que descreve como os atributos de qualidade de *software* são quantificados e convertidos para indicadores, foi estabelecida a especificação para cada indicador, como apresentado a seguir.

4.2.4.1.1 INDICADORES DE QUALIDADE DO PROCESSO DE VERIFICAÇÃO E REVISÃO (IQPVR)

Na **TAB. 4.2.4.1-1** é apresentado o modelo GQM (*Goal*/*Question*/*Metric*) aplicado as não-conformidades detectadas durante as atividades de verificações e revisão pelas listas de conferências (*checklists*) e técnicas de leitura verticais e horizontais respectivamente.

Abaixo, segue uma descrição mais detalhada dos quatro tipos de métricas apresentadas na **TAB. 4.2.4.1-1**, que tratam os estados das não-conformidades encontradas no processo de Verificação e Revisão, a saber:

- **Não-Conformidade Detectada**: o Grupo de Garantia de Qualidade de *Software* (GQS) encontra uma não-conformidade durante a verificação / revisão dos artefatos.
- **Não-Conformidade Resolvida Dentro do Prazo**: a equipe de desenvolvimento resolve a ocorrência de não-conformidade no prazo estabelecido pelo Grupo de GQS;

- **Não-Conformidade Resolvida Fora do Prazo**: a equipe de desenvolvimento resolve a ocorrência de não-conformidade com atraso. O prazo estabelecido pelo Grupo de GQS é estourado;
- **Não-Conformidade Não Resolvida:** o Grupo de GQS apontou a não-conformidade, no entanto a equipe de desenvolvimento relata que não vai resolver a não-conformidade (ação não realizada). Quando uma ocorrência de não-conformidade não é resolvida pela equipe de desenvolvimento no prazo estabelecido, o Grupo de GQS deve reportar a ocorrência à gerência de reuso. A gerência de reuso pode fornecer uma dispensa a equipe de desenvolvimento da solução da ocorrência de não-conformidade.

TAB. 4.2.4.1-1: Os principais objetivos, perguntas e métricas relativos aos estados das não-conformidades encontradas durante a verificação e revisão dos artefatos.

Objetivo: controlar as não–conformidades detectadas nas verificações / revisões dos artefatos inspecionados.	
Questões	**Métricas**
1 – Quantas não-conformidades detectadas?	• **Nº de não-conformidades detectadas**
2 – Quantas não-conformidades resolvidas?	• **Nº de não-conformidades resolvidas dentro do prazo** • **Nº de não-conformidades resolvidas fora do prazo**
3 – Quantas não-conformidades não resolvidas?	• **Nº de não-conformidades não resolvida**

4.2.4.1.2 INDICADORES DE PRODUTIVIDADE DO PROCESSO DE VERIFICAÇÃO E REVISÃO (IPPVR)

Na **TAB. 4.2.4.1-2** é apresentado o modelo GQM (*Goal*/*Question*/*Metric*) aplicado ao esforço associado as atividades de verificações e revisão na realização das listas de conferências (*checklists*) e técnicas de leitura verticais e horizontais respectivamente.

Abaixo, segue uma descrição mais detalhada dos três tipos de métricas apresentadas na **TAB. 4.2.4.1-2**, que tratam do esforço previsto e realizado pelo Grupo de GQS nas atividades de verificação / revisão do artefato, a saber:

- **Esforço Previsto:** o esforço previsto do Grupo de GQS para a realização da verificação e revisão individual em homem/hora;
- **Esforço Realizado:** o esforço despendido do Grupo de GQS para a realização da verificação e revisão individual em homem/hora;
- **Taxa de Esforço Realizado:** o esforço realizado pelo Grupo de GQS para execução da verificação / revisão individual em relação ao esforço do processo de desenvolvimento.

TAB. 4.2.4.1-2: Os principais objetivos, perguntas e métricas relativos à previsibilidade de esforço associado à verificação e revisão dos artefatos.

Objetivo: controlar o esforço associado aos processos de Verificação e Revisão.	
Questões	**Métricas**
1 – Quantidade de esforço estimado para a realização das atividades verificação e revisão?	• **Esforço previsto**
2 – Quantidade de esforço realizado para a execução das atividades verificação e revisão?	• **Esforço realizado**
3 – Qual a relação entre esforço previsto e esforço realizado?	• **Taxa de esforço realizado**

4.2.4.1.3 INDICADORES DE QUALIDADE DO PROJETO OO BASEADO EM REUSO (IQPOOR)

Foi utilizado o Modelo de Hierarquia para Avaliação da Qualidade de Projetos OO (BANSIYA et al., 2002) com a finalidade de medir a qualidade do projeto OO desenvolvido no processo baseado em reuso. Assim, foram estabelecidos seis objetivos do ponto de vista dos atributos de qualidade (**TAB. 2.2.3.1-1**).

TAB. 4.2.4.1-3: Os objetivos, perguntas e métricas relativos à qualidade do projeto OO.

(a) Compreensibilidade do projeto (CP).

Objetivo: controlar a capacidade de compreensão e entendimento do projeto.	
Questões	**Métricas**
1 – O quão fácil é de se aprender e entender?	• **Número médio de antecessores** • **Acoplamento direto da classe** • **Coesão entre métodos na classe** • **Número de métodos** • **Medida de acesso aos dados** • **Número de métodos polimorfos** • **Tamanho do projeto em classes**

(b) Efetividade do projeto (EfP).

Objetivo: controlar a capacidade efetiva do projeto.	
Questões	**Métricas**
1 – Quanto das funcionalidades e comportamentos desejados foram alcançados?	• **Número médio de antecessores** • **Medida de agregação** • **Medida de acesso aos dados** • **Medida de abstração funcional** • **Número de métodos polimorfos**

(c) Extensibilidade do projeto (EsP).

Objetivo: controlar a capacidade de inserção de novos requisitos no projeto.	
Questões	**Métricas**
1 – Quanto novos requisitos podem ser introduzidos no projeto?	• **Número médio de antecessores** • **Acoplamento direto da classe** • **Medida de abstração funcional** • **Número de métodos polimorfos**

(d) Flexibilidade do projeto (FxP).

Objetivo: controlar a capacidade do projeto de ser adaptado para fornecer funcionalmente uma capacidade específica.	
Questões	**Métricas**
1 – Qual a capacidade do projeto de adaptação a uma necessidade específica?	• **Acoplamento direto da classe** • **Medida de agregação** • **Medida de acesso aos dados** • **Número de métodos polimorfos**

(e) Funcionalidade do projeto (FuP).

Objetivo: controlar a capacidade funcional do projeto.	
Questões	**Métricas**
1 – Qual o percentual de serviços do projeto que estão disponíveis através de suas interfaces públicas?	• **Coesão entre métodos na classe** • **Número de hierarquia** • **Tamanho da interface da classe** • **Número de métodos polimorfos** • **Tamanho do projeto em classes**

(f) Reusabilidade do projeto (RP).

Objetivo: controlar a capacidade do projeto de ser reutilizado em outros domínios.	
Questões	**Métricas**
1 – Qual o percentual do projeto ser reaplicado para um novo domínio de problema?	• **Acoplamento direto da classe** • **Coesão entre métodos na classe** • **Tamanho da interface da classe** • **Tamanho do projeto em classes**

De acordo com a **TAB. 2.2.3.1-2** e **TAB. 2.2.3.1-3**, na **TAB. 4.2.4.1-3** são apresentadas as métricas que medem as propriedades de projeto que atuam diretamente nas funções de cálculo da medição para cada um dos objetivos, que se encontram definidas na **TAB. 4.2.4.1-4**.

A seguir uma descrição mais detalhada das métricas apresentadas na **TAB. 4.2.4.1-3**, a saber:

- **Número Médio de Antecessores (NMA):** o número médio de classes que herdam informação;

- **Acoplamento Direto da Classe (ADC):** o número de classes que uma classe se relaciona diretamente;
- **Coesão entre Métodos na Classe (CMC):** o número de intersecções dos parâmetros de um método com o conjunto máximo independente de todos os tipos de parâmetros na classe;
- **Número de Métodos (NM):** o número total de métodos definidos em uma classe;
- **Medida de Agregação (MAg):** o número de declarações de dados cujos tipos são classes definidas;
- **Medida de Acesso aos Dados (MAD):** a relação entre os atributos privados (protegidos) pelo número total de atributos declarados na classe;
- **Medida de Abstração Funcional (MAF):** a relação entre o número de métodos herdados por uma classe pelo número total dos métodos acessíveis por métodos de membro da classe;
- **Número de Hierarquia (NHq):** o número de classes hierárquicas no projeto;
- **Tamanho da Interface da Classe (TIC):** o número de métodos públicos em uma classe;
- **Número de Métodos Polimorfos (NMP):** o número total de métodos que podem ter comportamento polimorfo;
- **Tamanho do Projeto em Classes (TPC):** o número total de classes no projeto.

TAB. 4.2.4.1-4: Funções de cálculo das medições das qualidades do projeto OO com os índices definidos por (BANSIYA et al., 2002).

Funções de Medição
Compreensibilidade = 0, 33 (- NMA – ADC + CMC – NM + MAD – NMP – TPC)
Efetividade = 0,22 (NMA + MAg + MAD + MAF + NMP)
Extensibilidade = 0,5 (NMA – ADC + MAF + NMP)
Flexibilidade = 0,25 (- ADC + MAD) + 0,5 (MAg + NMP)
Funcionalidade = 0,12 * CMC + 0,22 (NHq + TIC + NMP + TPC)
Reusabilidade = 0,25 (- ADC + CMC) + 0,5 (TIC + TPC)

Baseados nos formulários de definição de Construção Mensurável proposto pelo PSM foram estabelecidos os seguintes formulários: Especificação de Indicadores (**TAB. 4.2.4.1-5**), Procedimento de Coleta de Dados (**TAB. 4.2.4.1-6**) e Procedimento para Análise de Dados (**TAB. 4.2.4.1-7**). Estes formulários foram aplicados nas especificações de indicadores para apoiar as decisões das atividades de acompanhamento e gestão da qualidade dos artefatos dos processos de Engenharia de Domínio e Desenvolvimento de (GURGEL, 2004). Nas tabelas

TAB. 4.2.4.1-5, TAB. 4.2.4.1-6 e **TAB. 4.2.4.1-7** são apresentados os formulários para especificação do Indicador de Qualidade do Processo de Verificação e Revisão.

TAB. 4.2.4.1-5: Especificação do Indicador de Qualidade do Processo de Verificação e Revisão.

ESPECIFICAÇÃO DO INDICADOR DE QUALIDADE DO PROCESSO DE VERIFICAÇÃO E REVISÃO		
Sigla: IQPVR	**Versão:** 1.3	**Data:** 10/05/2004
1 - Necessidade de Informação	Avaliar a qualidade dos artefatos para a produção do *asset*.	
2 - Categoria de Informação	Qualidade do Produto.	
3 - Conceito Mensurável	Artefato sem defeito.	
4 - Entidades Relevantes	▪ *Checklists* ▪ Relatórios de Revisão ▪ Relatórios de Resolução de Não-Conformidades	
5 - Atributos	▪ Número de não-conformidades detectadas ▪ Número de não-conformidades resolvidas dentro do prazo ▪ Severidade	
6 - Medidas Básicas	▪ Quantidade de não-conformidades detectadas ▪ Quantidade de não-conformidades resolvidas dentro do prazo ▪ Grau de severidade	
7 - Método de Medição	Somatório das não-conformidades detectadas pelas tarefas de inspeção realizadas multiplicadas pelo valor correspondente ao grau de severidade (1 para severidade do tipo A, 2 para tipo B e 3 para tipo C). Para as não-conformidades encontradas via *Checklist* , considerar grau de severidade 1.	
8 - Tipo de Método	Objetivo	
9 - Escala	Números reais positivos	
10 - Tipo de Escala	Razão	
11 - Unidade de Medida	▪ Não-conformidades ▪ Severidade	
12 - Medida Derivada	Percentagem de não-conformidades resolvidas no prazo (%)	
13 - Função de Medição	IQPVR = (Quantidade de não-conformidades resolvidas dentro do prazo / Quantidade de não-conformidades detectadas) * 100	
14 - Modelo de Análise	Os resultados do indicador devem ser analisados nos marcos definidos no plano do projeto após a entrega de uma versão.	
15 - Critério de Decisão	Caso o percentual esteja acima dos limites estabelecidos para o projeto, deve-se analisar a causa dos atrasos.	

TAB. 4.2.4.1-6: Formulário associado ao procedimento de Coleta de Dados.

COLETA DE DADOS	
1 - Medidas Básicas	▪ Quantidade de não-conformidades detectadas ▪ Quantidade de não-conformidades resolvidas dentro do prazo ▪ Grau de severidade
2 - Freqüência da Coleta de Dados	Nos marcos definidos no plano de projeto para entrega da versão.
3 - Responsável	Consultor de Qualidade
4 - Fase ou Atividade para Coleta	Entrega de uma nova versão
5 - Ferramentas usadas na Coleta	Excel
6 - Verificação e Validação	▪ Quantidade de não-conformidades detectadas deve ser a soma das quantidades de não-conformidades resolvidas dentro e fora do prazo Severidades devem ser do tipo A, B ou C
7 - Repositório para Dados Coletados	Banco de Dados de Medição do projeto

TAB. 4.2.4.1-7: Formulário associado ao procedimento para Análise dos Dados.

ANÁLISE DE DADOS	
1 - Indicador	Quantidade de não-conformidades resolvidas no prazo
2 - Freqüência do Reporte de Dados	Geração de uma nova versão do projeto
3 - Responsável	Analista de Medições
4 - Fase ou Atividade para Análise	Entrega de uma nova versão
5 - Fonte dos Dados para Análise	Banco de Dados de Medição do projeto
6 - Ferramentas usadas na Análise de Dados	Excel
7 - Relatório	Relatório de Medição e Avaliação para o gerente de reuso

4.2.4.2 MEDIÇÃO

A Medição foi realizada da seguinte forma:

- **Técnica Utilizada** – Os dados resultantes da realização das listas de conferências (*checklists*) e das revisões capturados conforme descrito na seção 3.2.1.2.3. E o cálculo das funções de medição relativo aos indicadores foi realizado conforme definidas no Planejamento da Medição.
- **Tempo Despendido** - Uma semana.
- **Fatores de Influência** – Como os artefatos Plano de Teste e Testes (Teste de Unidade, Teste de Integração, Teste de Interface Gráfica e Teste de Validação) não foram elaborados formalmente, estes não puderam ser avaliados pelo Grupo de GQS.
- **Avaliação dos Resultados e do Tempo** - Na aplicação do indicador IQPVR (Indicador de Qualidade do Processo de Verificação e Revisão) ocorreu uma dificuldade quanto à medição da quantidade de não-conformidades resolvidas no prazo, pois o cronograma do projeto piloto estava muito alto nível, de forma que não se pode distinguir quantidade de não-conformidades resolvidas no prazo ou fora do prazo.

Na **TAB. 4.2.4.2-1** é apresentada resumidamente os resultados da coleta de dados para os indicadores IQPVR e IPPVR.

TAB. 4.2.4.2-1: Resultados da Coleta de Dados dos *Checklits* e Técnicas de Leitura.

Checklists				
	Necessidades do Cliente e Glossário	Código-fonte	Arquitetura	Total
Σ quantidade de não-conformidade * severidade	7	7	7	21
Duração	45 minutos	40 minutos	35 minutos	120 minutos

Técnica de Leitura Horizontal						
	TLH1	TLH2	TLH3	TLH4	TLH5	Total
Σ quantidade de não-conformidade * severidade	6	11	4	2	0	23
Duração	40 minutos	45 minutos	65 minutos	35 minutos	40 minutos	225 minutos

Técnica de Leitura Vertical							
	TLV1	TLV2	TLV3	TLV4	TLV5	TLV6	Total
Σ quantidade de não-conformidade * severidade	3	13	3	5	8	13	45
Duração	40 minutos	50 minutos	65 minutos	65 minutos	40 minutos	45 minutos	305 minutos

Os dados coletados para os indicadores de IQPOOR são apresentados na **TAB. 4.2.4.2-2**.

TAB. 4.2.4.2-2: Resultados dos dados coletados para os indicadores de IQPOOR.

Métricas		**Dados**
Sigla	**Descrição**	
NMA	Número Médio de Antecessores	2
ADC	Acoplamento Direto da Classe	76
CMC	Coesão entre Métodos na Classe	1
NM	Número de Métodos	134
MAg	Medida de Agregação	2
MAD	Medida de Acesso aos Dados	1
MAF	Medida de Abstração Funcional	1
NHq	Número de Hierarquia	1
TIC	Tamanho da Interface da Classe	134
NMP	Número de Métodos Polimorfos	0
TPC	Tamanho do Projeto em Classes	30

Os dados coletados apresentados nas **TAB. 4.2.4.2-1** e **TAB. 4.2.4.2-2** foram aplicados nas funções de cálculo dos indicadores IQPVR, IPPVR e IQPOOR conforme especificado no Planejamento da Medição. Na **TAB. 4.2.4.2-3** é apresentado o resultado das medições realizadas para os indicadores estabelecidos no Planejamento da Medição.

TAB. 4.2.4.2-3: Resultado das medições para os indicadores estabelecidos no Planejamento da Medição.

Indicadores		Resultados
Sigla	**Descrição**	
IQPVR	Qualidade do Processo de Verificação e Revisão	0%
IPPVR	Produtividade do Processo de Verificação e Revisão	9,02%
ICP	Compreensibilidade do Projeto	- 12,74
IEfP	Efetividade do Projeto	1,32
IEsP	Extensibilidade do Projeto	0
IFxP	Flexibilidade do Projeto	0,5
IFuP	Funcionalidade do Projeto	8,04
IRP	Reusabilidade do Projeto	17

4.3 AVALIAÇÃO DO ESTUDO DE CASO

Através dos resultados obtidos na aplicação dos processos na execução do projeto piloto Controlador Tático, tanto a viabilidade dos processos e a hipótese formulada são válidas.

Os artefatos de qualidade levantaram não-conformidades que nos testes realizados para detecção de erros não foram capturados. Estas não-conformidades interferem não somente na padronização e corretude dos *assets* gerados para o projeto, mas para o entendimento claro e preciso dos *assets* quando estes forem reutilizados em outros domínios.

Os indicadores estabelecidos nesta proposta fornecem subsídios ao gerente de reuso verificar nos projetos futuros desenvolvidos dentro do contexto do Ministério da Defesa e de seus Comandos Subordinados via o processo baseado em reuso sistemático de (GURGEL, 2004) quanto à melhora da produtividade e do processo de Verificação e Revisão, e a qualidade do projeto desenvolvido sob a ótica da reusabilidade, extensibilidade e compreensibilidade.

5 CONCLUSÕES

O processo sistemático de Garantia da Qualidade de *Software* (GQS) desenvolvido neste trabalho atingiu seu objetivo de apoiar o Processo de Desenvolvimento de *Software* baseado em Reuso de (GURGEL, 2004) elaborado para o contexto da MD, visando capacitar os

Órgãos Descentralizados de Fornecimento de *Software* na aprovação dos artefatos a serem empacotados em *assets* para certificação, classificação e gerenciamento destes *assets* posteriormente em repositórios de artefatos reutilizáveis.

Este trabalho viabilizou a integração de diferentes técnicas de controle de qualidade de *software* em um processo de desenvolvimento do ciclo de vida de *software* baseado em *asset*s, visando a assegurar e melhorar a qualidade para reutilização.

A avaliação da qualidade dos artefatos gerados nas fases iniciais do processo de Engenharia de Domínio e Desenvolvimento possibilitou a realização de ações preventivas e / ou corretivas., como também, possibilitou a melhoria contínua na qualidade do desenvolvimento correto dos artefatos a serem reutilizados desde a primeira vez. Com isto, quantidades relevantes de retrabalho e de retestes futuros puderam ser evitadas (FALCONI, 1999).

As inspeções abordadas neste trabalho foram abrangidas pelas verificações e revisões. Estas inspeções de *software* possibilitaram a detecção e remoção de defeitos na fase em que foram gerados, inclusive na codificação. A combinação destas inspeções com as realizações dos testes foi necessária para a produção de *software* com qualidade. Através do estudo de caso, foi observado que investir exclusivamente em testes acarretaria em um retrabalho maior. Pois, os testes ao removerem defeitos ao final do ciclo de desenvolvimento não são capazes de identificar todos os defeitos do *software*, fazendo com que as equipes de desenvolvimento não prestem a devida atenção na qualidade dos artefatos intermediários produzidos (PERRY, 1999). As inspeções de *software* não substituíram os testes, mas ajudaram a manter a qualidade nos artefatos, principalmente os que compunham a estrutura do *asset* correspondente.

No trabalho aqui proposto foi apresentado um Processo de Medição e Avaliação que constituiu um caminho para o controle de projetos de *software* através de indicadores, que podem ser utilizados posteriormente para promover a melhoria contínua do Processo de Desenvolvimento de *Software* baseado em Reuso. Além disso, este trabalho forneceu dados e informações quantitativas para que o Gerente de Reuso possa decidir sobre que ações a serem tomadas.

As características da qualidade apresentadas neste trabalho foram os atributos de qualidade de projeto OO. Estes atributos foram relacionados aos requisitos da qualidade estabelecidos para o produto de *software* como a compreensibilidade, efetividade,

flexibilidade, extensibilidade, funcionalidade e reusabilidade. Estas qualidades[20] foram utilizadas como indicadores da qualidade do produto de *software* baseado em reuso.

Uma consideração importante observada durante a aplicação do processo proposto no estudo de caso foi que a assimilação de novas práticas dentro de uma abordagem totalmente diferente até então utilizadas pela equipe de desenvolvimento, trouxe um grande desconforto e gerou incertezas, por estar atuando em patamares desconhecidos. Por isso, a implantação de um programa de qualidade não é uma tarefa simples e requer habilidade para mostrar à gerência que os novos processos vão melhorar e agilizar planejamentos e acompanhamentos das atividades e, conseqüentemente, dos projetos. Para os técnicos, é importante evidenciar que o ganho com planejamentos e reusos aumenta o trabalho produtivo da equipe de desenvolvimento à medida que o retrabalho nas atividades tende a não ocorrer. Para tal, o processo de Medição e Avaliação é fundamental para controlar e avaliar os produtos de *software* e os processos envolvidos para geração destes produtos.

5.1 BENEFÍCIOS

O principal benefício deste trabalho é o desenvolvimento de um processo de garantir a qualidade de *software* que se integre plenamente ao processo de reuso sistemático proposto por (GURGEL,2004) para o Ministério da Defesa e de seus Comandos Subordinados. Com base neste processo de qualidade, a melhoria da qualidade dos artefatos produzido na especificação de análise e projeto de domínio e do próprio desenvolvimento do projeto possibilita um aumento no reuso destes artefatos.

Outro benefício reside no processo de medição, que quantifica a qualidade e a produtividade do processo de desenvolvimento baseado em reuso, fornecendo uma base quantitativa para a avaliação das iniciativas de melhoria do processo, e conseqüentemente, promovendo sua melhoria contínua.

Neste trabalho foram abordadas atividades específicas para assegurar a qualidade da Gerência da Qualidade, que estão em conformidade às normas de qualidade da ISO 9000:2000 (ABNT9000, 2001). Estas atividades são as atividades de *feedbacks* ao próprio processo de GQS realizadas através dos indicadores de qualidade e produtividade dos processos de Verificação e Revisão, como também, através do QSMGQ – Questionário de Sugestões para Melhoria da Garantia da Qualidade. Na literatura, foram poucos trabalhos

[20] Frutos de um estudo baseado na ISO 9126 por (BANSIYA, 2002).

encontrados que se propõem um tratamento explícito para garantir a qualidade da Gerência da Qualidade.

Além disso, com a disponibilidade das TLH e TLV às equipes de desenvolvimento conforme mencionado na seção 3.2.1.2.2, os procedimentos definidos nas Técnicas de Leitura podem contribuir aos desenvolvedores como suporte na elaboração dos artefatos das atividades de Análise e Projeto de Domínio do processo de Engenharia de Domínio de (GURGEL, 2004).

5.2 TRABALHOS FUTUROS

Os futuros trabalhos que podem ser realizados para melhoria do processo de GQS são:

- Implementar uma ferramenta de qualidade que gerencie o processo de medição integrado aos processos Verificação e Revisão para facilitar o trabalho de coleta de dados e evitar erros de computação e de armazenamento dos mesmos;
- Ampliar a estratégia proposta para tratar de aspectos humanos das reuniões das revisões quanto às abordagens para melhoria dos conflitos humanos – técnica de preparação para revisões técnicas (neste trabalho foi feita uma abordagem superficial);
- Aprofundar na validação da documentação, classificação e gerência de *assets* de acordo com as normas ISO 12207 e IEEE 1517. Portanto, deverão ser estabelecidas atividades de auditoria, padronização da documentação e gerência de *assets*;
- Realizar novos estudos de casos para verificar o grau de facilidade de aplicação da estratégia proposta.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ABNT9000 - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS: **ABNT/NBR ISO 9000** – Sistemas de gestão da qualidade, ABNT/CB-25 – Comitê Brasileiro da Qualidade, 2001.

ABNT9001 - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS: **ABNT/NBR ISO 9001** - Sistemas de gestão da qualidade – Requisitos, 2001.

ABNT12207 - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS: **ABNT/NBR ISO/IEC 12207** – Tecnologia da Informação – Processo de ciclo de vida de software, Out. 1998.

AGRESTI, W., EVANCO, W. Projecting software defects in analyzing Ada designs. **IEEE Transactions on Software Engineering**, v.18, n.11, 1992. p. 988–997.

ALMEIDA, K. M., STAA, A. V. Avaliação da qualidade em um modelo de ciclo de vida orientado a reuso. **III Simpósio de Desenvolvimento e Manutenção de Software**, Sessão Técnica VI, 2003. 10 p.

ARTHUR, L. J. **Improving software quality: an insider's guide to TQM**. New York: John Wiley & Sons, Jan.1993. 310 p.

BANSIYA, J. **A hierarchical model for quality assessment of object-oriented designs**, PhD Dissertation, University of Alabama in Huntsville, 1997.

BANSIYA, J., C. DAVIS, C. G. A hierarchical model for object-oriented design quality assessment, **IEEE Transactions on Software Engineering**, v. 28, n. 1, Jan. 2002. p. 4-18.

BARNES, B. H., BOLLINGER, T. B. Making reuse cost-effective**. IEEE Software**, v. 8, n. 1, 1991. p. 13-24.

BASILI, V. R., ROMBACH, D. H., BAILEY, J., DELIS, A. Ada reusability and measurement. **Computer Science Technical Report Series**, UMIACS-TR-90-73, CS-TR-2478, University of Maryland, May, 1990. Disponível: http://www.cs.umd.edu/ projects/SoftEng/ESEG/papers/84.67.pdf [capturado em agosto de 2003].

BASILI, V., MUSA, J. The future engineering of software: a management perspective. **IEEE Computer**, v. 24, n.9, Sep.1991. p. 90-96.

BASILI, V. R., CALDEIRA, G., CANTONE, G. A reference architecture for the component factory. **ACM Transactions on Software Engineering and Methodology**, v. 1, n. 1, Jan. 1992. p. 53-80.

BASILI, V. R., CALDIERA, G., ROMBACH, H. D. Goal question metric paradigm. In: **Encyclopedia of Software Engineering**, v. 1, John Wiley & Sons, 1994. p. 528-532.

BASILI, V. R., GREEN, S., LAITENBERGER, O., LANUBILE, F., SHULL, F., SOUNGÄRD, S., ZELKOWITZ, M.V. The empirical investigation of perspective-based reading. **Empirical Software Engineering Journal**, I, 1996. 38 p. Disponível: http://www.cs.umd.edu/~mvz/handouts/emp_pbr.pdf [capturado em junho de 2003].

BASILI, V. R., CALDIERA, G., ROMBACH, H. D. **The experience factory**. Maryland, EUA, Institute for Advanced Computer Studies Department of Computer Science, University of Maryland & FB Informatik Universität Kaiserlautern (German), 2000. 19 p. Disponível: http://wwwagse.informatik.uni-kl.de/pubs/repository/basili94c/encyclo.ef. pdf [capturado em abril de 2003].

BIEMAN, J., KARUNANITHI, S. Candidate Reuse Metrics for Object-Oriented and Ada Software. In: **Proceedings of IEEE-CS First International Software Metrics Symposium**, May, 1993. p. 120-128.

BOEHM, B. W., PAPACCIO, F. N. Understanding and controlling software costs. **IEEE Transactions on Software Engineering**, v.14, n. 10, Oct. 1988.

BRAGA, R. M. M., WERNER, C. M. L. **Processo de engenharia de domínio do ambiente Odyssey.** Relatório técnico do projeto Odyssey 4/99. Rio de Janeiro: COPPE/UFRJ, 1999. Disponível: http://www.cos.ufrj.br/~odyssey/pt/relatorios.htm [capturado em janeiro de 2004].

BRAUN, C. Reuse. In: **Encyclopedia of Software Engineering**, by Marciniak John J. (Editor-in-Chief), v. 1, John Wiley & Sons, 1994. p. 1055-1069.

BRIAND, L. C., FREIMUT, B., LAITENBERGER, O., RUBE, G., KLEIN, B. Quality assurance technologies for the EURO conversion – industrial experience at Allianz life assurance. In: 2nd **International Software Quality Week Europe**, Brussels, Belgium, 1998. p. 1-23. Disponível: http://www.visek.de/servlet/is/2228 [capturado em junho de 2003].

BRÖHL, A.P., DRÖSCHEL, W. **Das V-modell: der standard für die softwareentwicklung mit praxizleitfader**. Oldenbourg, 1995. 656 p.

CHENG, B., JEFFREY, R. Comparing inspection strategies for software requirements specifications. Proceedings of the 1996 **Australian Software Engineering Conference**, 1996. p. 203-211.

CHEESMAN, J., DANIELS, J. **UML Components: a simple process for specifying component-based software**. Boston: Addison Wesley, 2000. 176 p.

CHERNAK, Y. A statistical approach to the inspection checklist formal synthesis and improvement. **IEEE Transactions on Software Engineering**, v. 22, n. 12, 1996. p. 866-874.

CHIDAMBER, S. R., KEMERER, C. F. A metrics suite for object-oriented design, **IEEE Transactions on Software Engineering,** v. 20, n. 6, June 1994. p. 476-493.

CROSBY, P. B. **Quality is still free: making quality certain in uncertain times**. Mc Graw-Hill, 1996. 264 p.

DAVIS, T. The Reuse Capability Model: A Basis for Improving an Organization's Reuse Capability, **IEEE Transactions on Software Engineering,** v. 20, n. 6, 1993. p. 476-493.

DEMING, W. E**. Quality, productivity and competitive position**. Center for Advanced Engineering Study, Massachusetts Institute of Technology, Cambridge, MA, 1982.

DEMING, W. E. **Qualidade: a revolução da administração**. Editora Marques-Saraiva, 1990. ISBN: 85-85238-15-1.

DERBY, E. Projetando métricas úteis: utilizando observação, modelagem e mensuração na tomada de decisões. Newsletter do BFPUG, Janeiro 2001. Disponível: http://www.bfpug.com.br [capturado em janeiro de 2003].

DOOLEAN, E. P. Experiences with Fagan´s inspection method. **Software Practice and Experience**, v. 22, n. 2, 1992. p.173-182.

D´SOUZA, D.F., WILLS, A. C. **Objects, components and frameworks with UML – The Catalysis Approach**. Massachusetts, Addison-Wesley Professional, 1998. 816 p.

FAGAN, M. E. Advances in software inspections. **IEEE Transactions on Software Engineering**, v. 12, n. 7, July 1986. p. 744-751.

FALCONI, V. **TQC: controle da qualidade total**. 8ª Edição, São Paulo: EDG, 1999. 224 p. ISBN: 85-86948-14-4.

FAVARO, J. What price reusability? A case study. **Ada Lett** (Spring), 1991. p.115-124.

FAVARO, J. A Comparison of Approaches to Reuse Investment Analysis. 4th **International Conference of Software Reuse**, Orlando, April, 1996. p. 23-26.

FEIGENBAUM, A.V. **Total Quality Control**. 3rd Editon, Mc Graw-Hill, Jan. 1991. 896 p. ISBN: 00-70203-5-47.

FENTON, N. E. **Software metrics, a rigorous approach.** New York: Chapman & Hall, 1991. 360 p.

FICHMAN, R. G. Incentive compatibility and systematic software reuse. **Journal of Systems and Software**, New York, v. 57, iss. 1, April 27, 2001. 45 p.

FIORINI, S. T., LEITE, J. C. S. P., LUCENA, C. J. P. Reusing process patterns. In: **Proceedings of the Workshop on Learning Software Organizations**, Oulu, 2000. p. 15-33. Disponível: http://www.iese.fhg.de/LSOworkshop2000/LSO-Complete-2000.pdf [capturado em maio de 2003].

FIORINI, S. T., STAA, A., BAPTISTA, R. **Engenharia de software com CMM**. Rio de Janeiro: Brasport, 1998. 346 p.

FRAKES, W. B., NEJMEH, B.A. Software Reuse through Information Retrieval. In: **Proceedings of the Twentieth Annual Hawaii International Conference on Systems Sciences, Kona**, Jan., 1987. p. 530-535.

FRAKES, W., POLE, T. An empirical study of representation methods for reusable software components. **IEEE Transactions on Software Engineering**, v. 20, n. 8, Aug. 1994. p. 617-630.

FRAKES, W., FOX, C. Sixteen questions about software reuse. **Communication ACM**, v. 38, n. 6, 1995. p. 75-87.

FRAKESa, W. B., FOX, C. J. Quality improvement using software reuse failure modes model. **IEEE Transactions on Software Engineering**, v. 22, n. 4, April 1996. p. 274-279.

FRAKESb, W. B., TERRY, C. Software reuse metrics and models. **ACM Computing Surveys**, v. 28, n. 2, 1996. p. 415-435.

FUJIWARA, S., BOCHMAN, G. V., KHENDECK, F., AMALON, M., GHEDAMSI, A. Test selection based on finite state models. **IEEE Transactions on Software Engineering**, v. 17, n. 6, June 1991. p. 591-603.

FUSARO, P., LAMBILE, F., VISAGGIO, G. A replicated experiment to assess requirements inspections techniques. **Empirical Software Engineering: an International Journal**, v. 2, n.1, 1997. p. 39-57.

GAFFNEY, J. E. Jr., DUREK, T. A. Software reuse-key to enhanced productivity: some quantitative models. **Information Software Technology**, German, v.31, n. 5, 1989. p. 258-267.

GILB, T., GRAHAM, D. **Software inspection.** 1st Edition, Massachusetts: Addison-Wesley, 1993. 496 p.

GIORARDI, R. Main approaches to software classification and retrieval, em Cursos Complementares 1998: Ingeniería de Software y Reutilización: Aspectos Dinámicos y Generación Automática, Universidade de Vigo, Santiago, 1998.

GURGEL, M. S. **Estudo do desenvolvimento de software baseado em reuso no contexto do Ministério da Defesa e de seus comandos subordinados**. Dissertação de Mestrado, Instituto Militar de Engenharia - IME, 2004.

HARROLD, M. J. Testing: a roadmap. In: The Future of Software Engineering. Finkelstein, A (ed), 22nd **International Conference on Software Engineering**, 2000. 10 p.

HAZAN, C. **Metodologia para o uso de indicadores na gerência de projetos de desenvolvimento de software**. Tese de Mestrado, Instituto Militar de Engenharia - IME, 1999.

HEINEMAN, G. T., COUNCIL, W. T. **Component-based software-engineering: putting the pieces together**. 1st Edition, Addison Wesley, May 2001. 416 p.

HENNINGER, S. Using software process to support learning organizations. In: **Proceedings of the Workshop on Learning Software Organizations**, Kaiserslaurten, Germany, 1999.

HITZ, M., MONTAZERI, B. Chidamber and Kemerer's metrics suite: a measurement theory perspective, **IEEE Transactions on Software Engineering**, v. 22, n. 4, Apr. 1996. p. 267-271.

IEEE610.12 - INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS: IEEE Standard 610.12, Standard glossary of software engineering terminology, **IEEE Press**, 1990.

IEEE828 - INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS: IEEE/ANSI Standard 828-1990, Software configuration management plans, **IEEE Software Engineering Standard Collection**, 1990.

IEEE1517 - INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS: IEEE Standard 1517-1999, Software life cycle processes – reuse processes**, IEEE Standard for Information Technology**, 1999.

IFPUG - INTERNATIONAL FUNCION POINT USERS GROUP: **Counting practices manual**. Version 4.1.1, April 2000. Disponível: http://www.ifpug.org [capturado em 20 de junho de 2002].

ISO8402 - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: **ISO/IEC 8402**, Quality management and quality assurance – vocabulary, 2nd Edition, 1994.

ISO15504 - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: **ISO/IEC TR 15504**, Information technology – software process assessment, 1998. Disponível: http://www.sqi.gu.edu.au/spice [capturado em 10 de abril de 2003].

ISO9126 - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: **ISO/IEC 9126-1**, International standard, Information technology – Software product evaluation – Quality characteristic and guidelines for their use, 2000.

JACOBSON, I.,GRISS, M., JONSSON, P. **Software reuse: architecture, process and organization for business success**. 1st Edition, Boston: Addison Wesley, 1997. 528 p.

JACOBSON, I., BOOCH, G., RUMBAUGH, J. **The unified software development process**. 1st Edition, Boston: Addison Wesley, 1999. 463 p.

JONES, C. **Assessment and control of software risks**. 1st Edition, New York: Prentice Hall, 1994. 464 p.

JONES, C. **Applied software measurement, assuring productivity and quality**. 2nd Edition, New York: Prentice Hall, 1997. 618 p.

JURAN, J. M. **Planejando para a qualidade**. 2ª Edição, São Paulo: Editora Pioneira, 1990. 394 p.

KAN, S. H. **Metrics and models in software quality engineering**. Addison Wesley, 1995. 361 p.

KARLSSON, E. A. **Software reuse: a holistic approach**. 1st Edition, John Wiley & Sons, Jan. 1995. 528 p. ISBN: 04-71958-19-0.

KARNER, G. **Metrics for objectory**. Master Thesis Report, LiTH-IDA-Ex-9344, Linköping, Sweden, 1993.

KITCHENHAM, B., PICKARD, L. PFLEEGER, S. L. Case studies for method and tool evaluation. **IEEE Software**, Los Angeles, v. 12, n. 4, Jul. 1995. p. 52-62.

KOLTUN, P., HUDSON, A. A reuse maturity model. In: Fourth Annual **Workshop on Software Reuse**, Herndon, VA, 1991.

KRUEGER, C. W. Software reuse. **Computing Surveys**, v. 24, n. 2, June 1992. p. 131-183.

LAITENBERGER, O., DEBAUD, J. M. Perspective-based reading of code documents at Robert Bosch GmbH. **Information and Software Technology**, v.39, 1997. p. 781-791.

LAITENBERGER, O. **Cost-effective detection of software defects with perspective-based inspection**. PhD-Thesis, University of Kaiserslautern, ISBN 38-16755-83-6. 2000.

LAITENBERGER, O. A survey of software inspection technologies. Handbook on **Software Engineering and Knowledge Engineering**, v. 2, World Scientific Publishing, 2002.

LEWIS, W. E. **Software testing and continuous quality improvement.** 2nd Edition, Texas: Auerbach, 2000. 656 p.

LI, W., HENRY, S. Object-oriented metrics that predict maintainability. **The Journal of Systems and Software**, v. 23, Nov. 1993. p. 111-122.

MALDONADO, J. C. **Critérios potenciais – usos: uma contribuição ao teste estrutural de software**. Campinas, São Paulo, Tese de Doutorado, DCA/FEE/UNICAMP, July 1991.

MCCALL, J.A., RICHARDS, P.K., WALTERS, G.F. **Factors in software quality**, v. 1, 2 e 3, AD-A-049-014/015/055, Nat'l Tech. Information Service, Springfield, Va., 1977.

MCGREGOR, J. D., SYKES, D. A. **Object-oriented software development: engineering software for reuse.** New York: Van Nostrand Reinhold, 1992.

MCGARRY, J., CARD, D., JONES, C., LAYMAN, B., CLARK, E., DEAN, J., HALL, F. **Practical software measurement: objective information for decision makers.** 1st Edition, Boston: Addison Wesley, 2001. 304 p. ISBN: 02-01715-16-3.

MEYER, B. **Reusable Software: the base object-oriented component libraries.** Prentice Hall, 1994. 514 p. ISBN: 01-32454-99-8.

MILLER, J., ROPER, M., WOOD, M. Further experiences with scenarios and checklist. **Journal of Empirical Software Engineering**, v. 3 n. 1, 1998. p. 37-64.

MYERS, G. **The art of software testing.** 2nd Edition, Nova York: John Wiley & Sons, 1979. 192 p.

NIST - **NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY**: Glossary of software reuse terms. Special Publication 500-222, 1994.

OGUSH, M. Terms in transition: a software reuse lexicon. **Crosstalk**, v. 39, 1992. p. 41-45.

PAIVA, D. M. B., NUNES, M. G. V., FORTES, R. P. M. Qualidade de produto de software: avaliação prática de um sistema de auditoria hipermídia educacional. VII **Workshop de Qualidade de Software**, Out. 2001. p. 114-122.

PAGLIUSO, P. B. B., TAMBASCIA, C.A., VILLAS-BOAS, A. Melhoria da inspeção de requisitos segundo a técnica de leitura baseada em perspectiva. XI SEMINCO - Seminário de Computação 2002, Santa Catarina, Blumenau, Setembro de 2002. p.105-115. http://www.inf.furb.br/seminco/2002/artigos/Pagliuso-seminco2002-27.pdf [capturado em julho 2003].

PALADINI, E. P. **Qualidade total na prática: implantação e avaliação de sistemas de qualidade total.** 2ª Edição, São Paulo: Atlas, 1994.

PALMER. **Traceability: software requirements engineering**. R.H. Thayer and M. Dorfman, 1997. p. 364-174.

PARK, R. E. Software size measurement: a framework for counting source statements. Technical Report CMU/SEI-92-TR-20, ESC-TR-92-20, **Software Engineering Institute**, Carnegie Mellon University, Pittsburgh, Sep. 1992. 242 p. Disponível: http:// www.sei.cmu.edu/pub/documents/92.reports/pdf/tr20.92.pdf [capturado em julho de 2003].

PAULK, M. C., CURTIS, B., CHRISSIS, M. B., WEBER, C. V. Capability maturity model for software. Version. 1.1**,** Technical Report CMU/SEI-93-TR-24, ESC-TR-93-177, **Software Engineering Institute**, Carnegie Mellon University, Feb. 1993. Disponível: http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf [capturado em fevereiro de 2003].

PERRY, W. **Effective methods for software testing**. 2nd Edition, New York: John Wiley & Sons Inc., Oct. 1999. 704 p. ISBN: 04-71354-18-X.

PFLEEGER, S. L. **Software engineering: theory and practice**. 2nd Edition, Nova Jersey: Prentice-Hall, Feb. 2001. 659 p. ISBN: 01-30290-49-1.

PORTER, A. A., VOTTA, L. G., BASILI, V. R. Comparing detection methods for software requirements inspections: a replicated experiment. **IEEE Transactions on Software Engineering**, v.21, n.6, June 1995. p. 563-575.

PORTER, A. A., VOTTA, L. Comparing detection methods for software requirements inspection: a replication using professional subjects. **Journal of Empirical Software Engineering**, v.3, n.4, 1998. p. 355-378.

POULIN, J. S. **Measuring software reuse: principles, practices and economic models**, Massachusetts: Addison-Wesley, Jan. 1997. 224 p. ISBN: 02-01634-13-9.

PRESSMAN, R. S. **Software engineering: a practitioner's approach.** 5th Edition, New York: McGraw-Hill, Nov. 2001. 860 p. ISBN: 00-72496-68-1.

PRIETO-DIAZ, R., FRAKES, W. B. Advances in software reuse. Selected papers from the Second **International Workshop on Software Reusability**, Los Alamitos, California, March, IEEE Computer Society Press, 1993.

RATIONAL SOFTWARE CORPORATION. Reusable asset specification. 2001. Disponível: http://www.rational.com/rda/forms/preview_online_v2.jsp [capturado em 21 de janeiro de 2004].

REGNELL, B., RUNESON, P., THELIN, T. Are the perspectives really different? Technical Report CODEN: LUTEDX (tets-7172)/1-40/1999 & Local 4, **Department of Communication Systems**, Lund University, 1999. p.141-181.

RISING, L.. A Way to Reuse Expertise. **IEEE Communications Magazine**, v. 37, n. 4, April, 1999. Disponível: http://wwwagcs.com/supportv2/techpapers/expertise.htm [capturado em junho de 2003].

ROCHA, A. R. C., MALDONADO, J. C., WERBER, K. C. **Qualidade de software: teoria e prática.** 1ª Edição, São Paulo: Prentice-Hall, 2001. 303 p. ISBN: 85-87918-54-0

ROSENBERG, L., HYATT, L., HANNER, T., HUFFMAN, L., WILSON, W. Testing metrics for requirement quality. 11th **International Software Quality Week**, California, May 1998. 9 p. Disponível: http://satc.gsfc.nasa.gov/support/ISQW_MAY98/ qual_5_98.pdf [capturado em março de 2003].

ROSS, D. T. Applications and extensions of SADT. **IEEE Computer**, v. 18, n.4, April 1984. p. 25-35.

SAMETINGER, J. Reuse documentation and documentation reuse. In: **Proceedings of TOOLS**, Europe 96, Paris, February 1996. 12 pp. Disponível: http://www.swe.uni-linz.ac.at/publications/abstract/TR-SE-95.13.html [capturado em janeiro de 2004].

SANDAHL, K., BLOMKVIST, O., KARLSSON, J., KRYSANDER, C., LINDVALL, M., OHLSSON, N. An extended replication of an experiment for assessing methods for software requirements inspections. **Empirical Software Engineering**, v.3, n. 4, 1998. p. 327-354.

SEI - SOFTWARE ENGINEERING INSTITUTE. **CMMI: Capability maturity model integration**, v. 1.1 – CMU/SEI-2002-TR-012, Software Engineering Institute, Pittsburgh, 2002. p. 465-480.

SEPIN: SECRETARIA DA POLÍTICA DE INFORMÁTICA DO MINISTÉRIO DA CIÊNCIA DE TECNOLOGIA. Qualidade e produtividade no setor de software brasileiro. Brazilian Software, ISSN 1518-112X, 2002. Disponível: www.mct.gov.br/sepin.

SINGER, J., VINSON, N. Ethical issues in empirical studies of software engineering. **IEEE Transactions on Software Engineering**, v. 28, n. 12, Dec. 2002. p. 1171-1180. Disponível: http://iit-iti.nrc-cnrc.gc.ca/iit-publication-iti/docs/NRC-44912.pdf [capturado em agosto de 2003].

SOMMERVILLE, I. **Software engineering.** 6th Edition, Massachusetts: Addison Wesley, Aug. 2000. 693 p. ISBN: 02-01398-15-X.

STAA, A. Medição - Aspectos Conceituais. Rio de Janeiro, Pontifícia Universidade Católica do Rio de Janeiro, PUC-RJ, 10 Out. 2002. Aula ministrada aos alunos da pós-graduação.

TAVARES, G. Revisões técnicas formais. **Quarto Simpósio Internacional de Melhoria de Processo de Software**, São Paulo, Brasil, 2002.

TERRY, C. **Analysis and implementation of software reuse measurement.** Master´s Project and Report, Virginia Polytechnic Institute and State University, 1993.

TRAVASSOS, G. H., SHULL, F., CARVER, J., BASILI, V. R Reading techniques for OO design inspections. Greenbelt, EUA, 24th Annual **Software Engineering Workshop**, NASA / SEL, 1999. Disponível: http: //sel.gsfc.nasa.gov/website/sew/1999/topcs/ travassosSEW99paper.pdf [capturado em agosto de 2003].

TRAVASSOS, G. H., SHULL, F., CARVER, J., BASILI, V. R. **Reading techniques for OO design inspections.** Technical Report, 2003. http://www.cs.umd.edu/library/TRs/CS-TR-4353/CS-TR-4353.pdf . 56 p.

VOTTA JR., L.G. 1993 Does every inspection need a meeting? Proceedings of the **ACM SIGSOFT 1993 Symposium on Foundations of Software Engineering**, ACM Software Engineering Notes, v.18, n. 5, Dec. 1993. p. 107-114.

WOHLIN, C. RUNESON, P., HOST, M., OHLSSON, M. C., REGNELL, B., WESSLEN, A. **Experimentation in software engineering: an introduction**. Dordrecht: Kluwer Academic Publishers, Nov. 1999. 224 p.

WIEGERS, K. E. Improving quality through software inspections. **Software Development**, April 1995. p. 55-64.

ZAHRAN, S. **Software process improvement: practical guidelines for business success**. 1st Edition, Addison-Wesley, Feb. 1998. 480 p. ISBN: 02-01177-82-X.

7 APÊNDICES

7.1 APÊNDICE 1: ESTUDO SOBRE OS TIPOS DE REUSO

O reuso pode ser classificado em termos de:

- **Escopo de desenvolvimento**: refere-se se os componentes reusáveis são de fonte externa (público) ou interna (privado) ao projeto (FENTON, 1991).
  - **Reuso privado ou interno** é a reutilização de itens do produto dentro do próprio produto;
  - **Reuso público ou externo** é a porção de um produto que foi construído externamente.
- **Modificação:** refere-se à quantificação de modificação do *asset*, que pode ser caixa-branca, caixa-preta (*verbatim*) ou adaptativo.
  - **Reuso de caixa-preta** é o reuso de componentes de *software* sem qualquer modificação (PRIETO-DIAZ et al., 1993; BIEMAN et al., 1993), é o reuso "*as is*";
  - **Reuso de caixa-branca** é o reuso de componentes com modificação ou adaptação (PRIETO-DIAZ et al., 1993);
  - **Reuso adaptativo** é uma estratégia de reuso que usa grandes estruturas de software como variabilidade invariável e restritiva para locais isolados e de baixo nível (BARNES et al., 1991).
- **Abordagem**: refere-se ao método para implementação de reuso, que pode ser por gerador, composição, estreito, amplo, indireto, direto, transporte e influenciado.
  - **Reuso gerador** é reuso no nível de especificação com aplicação ou geradores de código – o reuso que oferece o mais alto investimento (PRIETO-DIAZ et al., 1993);
  - **Reuso de composição** é o reuso de componentes existentes como blocos de construção para novos sistemas (PRIETO-DIAZ et al., 1993). Um exemplo é a programação de linguagem alto nível (BARNES et al., 1991);
  - **Reuso estreito** de componentes que são dependentes no ambiente da aplicação para toda funcionalidade. Favaro afirma que reuso orientado a componente é reuso estreito (FAVARO, 1991);

- **Reuso amplo** é o uso em massa, pacotes completos como *spreadsheets* e sistema operacional (FAVARO, 1991);
- **Reuso indireto** é reuso por meio de uma entidade intermediária. O nível indireto é o número de entidades intermediárias entre o item de reuso e o item sendo reusado (BIEMAN et al., 1993);
- **Reuso direto** é reuso sem passar por uma entidade intermediária (BIEMAN et al., 1993);
- **Reuso *carry-over*** é quando uma versão de um componente de *software* é usado sem modificação ("*as is*") em uma versão subseqüente do mesmo sistema (OGUSH, 1992);
- **Reuso influenciado** é o reuso com modificações (BIEMAN et al., 1993).

▪ **Escopo de domínio:** refere-se se o reuso ocorre dentro de uma família de sistemas (vertical) ou entre famílias de sistemas (horizontal) (PRIETO-DIAZ et al., 1993).

- **Reuso vertical** (Inter) é o reuso dentro da mesma aplicação ou domínio;
- **Reuso horizontal** (Intra) é o reuso de partes genéricas em diferentes domínios de aplicação.

▪ **Gerência:** refere-se ao grau de sistematização de reuso, que pode ser sistemático (planejado) ou *ad-hoc* (PRIETO-DIAZ et al., 1993).

- **Reuso sistemático ou planejado** é a prática sistemática e formal de reuso como base nas indústrias de *software*;
- **Reuso *ad-hoc*** é a seleção de componentes que não são projetados para reuso de uma biblioteca geral, ou seja, o reuso é realizado por um indivíduo de maneira informal.

▪ **Entidade reusada:** refere-se ao tipo de objeto reusado, que pode ser customizado, genérico, código-fonte, de nível abstrato e nível de instância.

- **Reuso customizado** é o uso de herança de orientação a objeto para suportar desenvolvimento incremental. Uma nova aplicação pode herdar informação de uma classe existente, anular certos métodos e adicionar novos comportamentos (MCGREGOR et al., 1992);
- **Reuso genérico** é reuso de pacotes genéricos, como *templates* para pacotes ou subprogramas (BIEMAN et al., 1993);
- **Reuso de código-fonte** é a modificação de baixo nível de uma classe existente de orientação a objeto para mudar suas características de desempenho;

- **Reuso de nível abstrato** é o reuso de abstrações de níveis mais alto dentro da estrutura de herança de orientação a objeto como base para novas idéias ou esquemas adicionais de classificação (MCGREGOR et al., 1992);
- **Reuso de nível de instância** é a forma mais comum de reuso em um ambiente orientado a objeto. O reuso é definido como criar simplesmente uma instância de uma classe existente (MCGREGOR et al., 1992).

O reuso de uma organização pode ser definido por selecionar adequadamente pares ou grupos de termos. Por exemplo, uma organização pode optar por reuso sistemático de código interno, permitindo apenas a utilização da técnica de caixa preta como parte de uma abordagem de composição com reuso focado dentro de seus domínios (vertical).

7.2 APÊNDICE 2: RESUMO DOS PRINCIPAIS MODELOS E MÉTRICAS

TAB. 7.2-1: Resumo das principais métricas e modelos de reuso de *software*.

Métrica/Modelo	Fonte	Descrição
Análise de custo e benefício		
Modelo de Custo/ Produtividade	Gaffney, Durek (GAFFNEY, 1989)	Modelo simples: C = custo de desenvolvimento de *software*. R = proporção de código de reuso no produto. b = custo relativo a todo novo código de incorporar código reusado no produto. C = (b – 1) R + 1 para todo b >=1 e produtividade P = 1 / C Modelo de desenvolvimento de custo: E = custo de desenvolver um componente reusável relativo ao custo de produzir um componente que não seja reusado. n = número de uso em que o custo do código será amortizado. Custo C = (b + E / n – 1) R + 1 para todo n > 1
Qualidade de investimento	Barnes, Bollinger (BARNES et al., 1991)	Qualidade de investimento (Q) é a razão dos benefícios (B) sobre investimento de reuso (R): Q = B / R. Se Q < 1 então o esforço de reuso resultou em perda. Se Q > 1 então o investimento forneceu bom retorno.
Modelo esforço e custo	Jacobson (JACOBSON et al., 1997)	Custo: F_{uso} = Custo de reuso de um componente (varia de 0.10 a 0.25, em geral usa-se a média aproximada 0.2 como padrão) F_{criar} = Custo de criar e gerir um componente reusável ROI = Retorno de investimento com reuso de componentes ROI = (n * (1 - F_{uso}) - F_{criar}) / F_{criar} Esforço de sistema de aplicação com reuso: PF = ponto de função para tamanho do *software* Tempo de desenvolvimento = $PF^{0.4}$ mês Tamanho da equipe de desenvolvimento = PF/150 (inclui gerentes, testadores e documentadores) Esforço = $PF^{1.4}$ / 150 homem/mês Potencial de erros = $F^{1.25}$ Custo de desenvolvimento = Esforço * custo de homem/mês
Avaliação da maturidade		
Modelo de maturidade de reuso	Koltun, Hudson (KOLTUN et al., 1991)	Modelo de maturidade de reuso para nivelar procedimentos de uma organização por trabalhar para reuso de *software*. Consiste em uma tabela com 5 fases de maturidade de reuso (Inicial/Caótico; Monitorado; Coordenado; Planejado; Enraizado) por 10 dimensões (Motivação/cultura, Planejamento para reuso, Abrangência Responsabilidade de execução, Ponto de partida para reuso, *Assets* reusáveis, Esquema de classificação, Suporte tecnológico, Métricas, Considerações legais ou contratuais). No cruzamento da linha com a coluna da tabela dispõe da descrição dos aspectos que devem corresponder aos procedimentos da organização.
Modelo de capacidade de reuso	Consórcio de Produtividade-de *Software* (DAVIS, 1993)	O Consórcio de Produtividade de *Software* identifica 4 estágios no modelo de implementação da crescente redução de risco para reuso de *software*: Oportuno; Integrado; Influenciado e Adiantado. Cada estágio apresenta um conjunto de aspectos que, por meios destes, são avaliados os procedimentos de uma organização para trabalhar com e para reuso.
Quantidade de reuso		
Nível de reuso	Frakes, Terry (TERRY, 1993) (FRAKES et al., 1994)	Assume que um sistema consiste de partes onde itens de alto nível são compostos de itens de baixo nível. O nível de reuso interno de um item de alto nível é definido como o número de itens de baixo nível interno reusados dividido pelo número total de itens de baixo nível no item de alto nível. Também são definidos: nível de reuso externo, nível total de reuso, freqüência de reuso e complexidade.
Fração de reuso	Agresti, Evanco (AGRESTI, 1992)	A variável FNEMC é definida como a fração de unidades de *software* novas ou extensivamente modificadas. FNEMC é o

		número de novos componentes mais o número de componentes modificados dividido pelo número total de componentes. FNEMC é igual ao valor 1 menos a fração de reuso.
Modelos e métricas orientadas a objeto	Bieman, Karunanithi (BIEMAN et al., 1993)	Existem 3 perspectivas para reuso no ambiente de orientação a objeto: servidor, cliente e sistema. Atributos mensuráveis de reuso de sistemas orientados a objeto incluem percentagens de código e classes que são relacionamentos cliente/servidor específicos e novos ou derivados.
Análise de modos de fracasso	Frakes e Fox (FRAKESa et al, 1996)	Um método para medir e melhorar um processo de reuso baseado de como um processo pode falhar. Os modos de fracasso são: nenhuma tentativa para reuso, parte inexistente, parte indisponível, parte não encontrada, parte não entendida, parte inválida e parte não pode ser integrada.
Avaliação da reusabilidade		
Medição de reusabilidade para Ada	Basili, Rombach, Bailey, Delis (BASILI et al., 1990)	Dois estudos focalizam reuso para sistemas Ada. O primeiro mede ligações de dados para caracterizar e identificar componentes reusáveis. O segundo define uma medição abstrata de reusabilidade de componente de *software* por identificar *software* reusável e medir a distância daquele ideal.
Predições de reuso para objetos do ciclo de vida	Frakes, Fox (FRAKES et al., 1995)	Modelos permitem a predição de níveis de reuso para um objeto do ciclo de vida baseado em níveis de reuso para outros objetos do ciclo de vida.
Métricas de bibliotecas de reuso		
Indexação de custo, busca pela efetividade e eficiência	Frakes, Pole (FRAKES et al., 1994)	Indexar custos inclui o custo de criar, manter e atualizar o esquema de classificação de *asset*. Para encontrar medidas de efetividade incluem revogação, precisão, sobreposição e entendimento. Para encontrar medidas de eficiência incluem utilização, *overhead* de indexação e tempo de recuperação.
Métricas de qualidade de *asset*	Frakes, Nejmeh (FRAKES et al., 1987)	Indicadores da qualidade dos *assets* na biblioteca incluem: tempo de uso, reusos bem sucedidos, revisões de reuso, complexidade, inspeções e testes.
Uso de biblioteca de reuso	Lillie Frakes e Terry (FRAKESb et al., 1996)	Indicadores de utilização de biblioteca incluem: tempo *on-line* (sistema disponível), considerações demográficas, tipo de desempenho da função da biblioteca (busca, *browse*, extração), distribuição de *asset* (quantidade de subscritor), *assets* disponibilizados por tipo (operação, fornecedor de listas, componentes locais), número de extração por coleção (número de *assets* extraídos por coleção, número de *assets* extraídos por avaliação de nível), lista de *assets* por domínio, solicitação de serviços através de Telnet Logins, modem Logins, FTP, *World Wide Web*.

7.3 APÊNDICE 3: PRINCIPAIS NORMAS

Os padrões de processos mais utilizados atualmente pelas industrias e organizações são: CMM, CMMI, ISO 9000, ISO 12207, ISO 15504, *Extreme programming*. Estes padrões delineiam e justificam a organização, os papéis dos interessados (*stakeholders*), as principais atividades e tarefas, os tipos e pontos de controle da qualidade. Normalmente não especificam como deve ser realizado. Os processos devem ser utilizados em uma grande gama de projetos. Assim, eles devem ser instanciados para determinado domínio de tecnologia. Estas instâncias são restritas a projetos que estejam em conformidade com estes domínios. Finalmente, os

processos instanciados devem ser novamente instanciados para um projeto específico resultando no plano de desenvolvimento conforme ilustrado na **FIG. 7.3-1**.

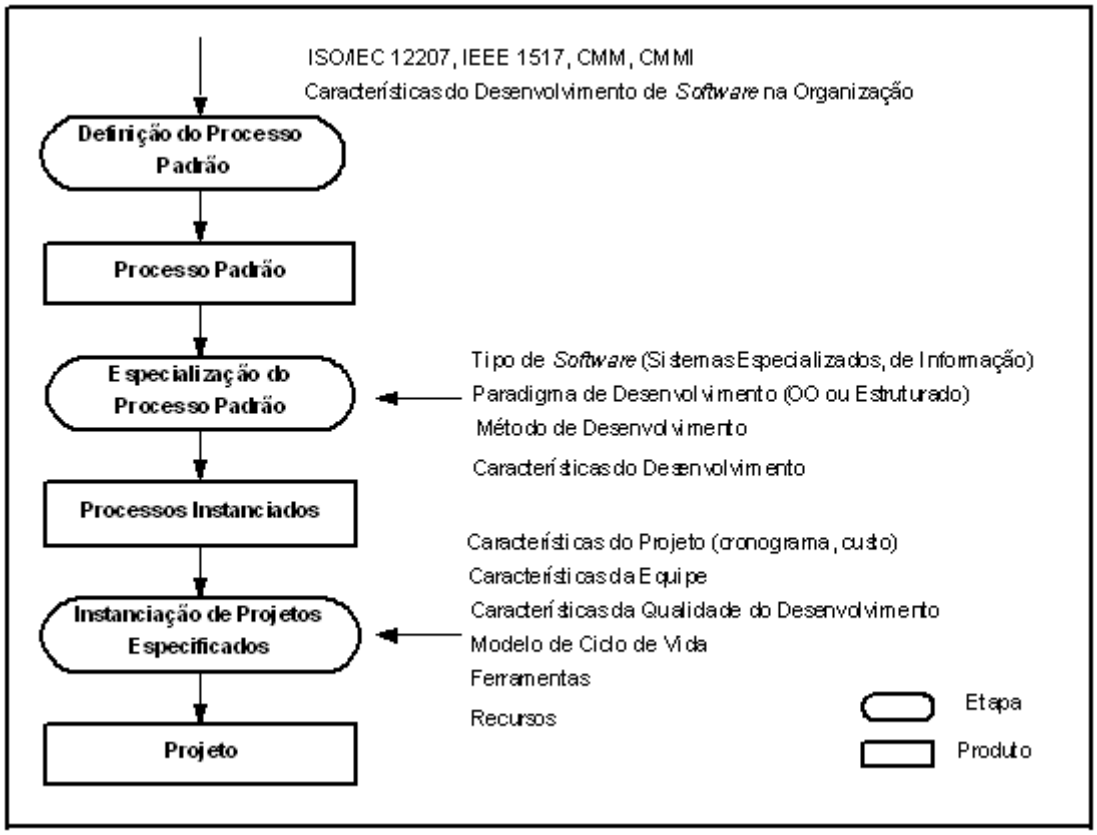


FIG. 7.3-1: Modelo para definição de processo de *software.*

Entre as normas, existe a família ISO/IEC 9000:2000 (ISO – *International Standart for Organizations*) (ABNT9000, 2001; ABNT9001, 2001) para garantia da qualidade de *software* e a ISO/IEC 12207 (ABNT12207, 1998) para estabelecimento de estrutura comum de processos de ciclo de vida de *software*. Dentre os modelos de melhoria de processos destacam o modelo CMM (*Capability Maturity Model*) (PAULK et al., 1993), CMMI (*Capability Maturity Model Integration*) (SEI, 2002) e ISO/IEC 15504 (SPICE - *Software Process Improvement and Capability dEtermination*) (ISO15504, 1998).

A versão ISO/IEC 9000:2000 (ABNT9000, 2001) apresenta como principal modificação no objetivo da garantia da qualidade, que anteriormente significava atendimento aos requisitos especificados para satisfação do cliente. A satisfação do cliente envolve tanto os requisitos explícitos como os implícitos, ou seja, a satisfação está diretamente relacionada com a qualidade dos produtos e serviços fornecidos.

A Norma ISO/IEC 12207 (ABNT12207, 1998) estabelece um modelo para definição de processos de *software*, onde as etapas relevantes são: definição do processo padrão, especificação do processo padrão e a instanciação para projetos. Ela oferece uma estrutura de processos baseada em *framework* para processos de ciclo de vida com terminologia bem definida: processos fundamentais, processos de apoio e processos organizacionais. Esta norma contém processos, atividades e tarefas que devem ser aplicadas à aquisição de sistemas que contém *software*, produtos de *software stand-alone* e serviços de *software*, durante o fornecimento, desenvolvimento, operação e manutenção de produtos de *software*.

A Norma ISO/IEC 9126 (ISO9126, 2000) define seis características de qualidade e métricas para um produto de *software*. Um *software* deve ter um conjunto de atributos que evidenciam as funcionalidades, a confiabilidade, a usabilidade, a eficiência, a manutenibilidade e a portabilidade. Cada uma dessas características subdivide-se em outras, que associadas a uma definição de métricas para correlacioná-las ao produto de *software*, permite uma avaliação da sua qualidade. É importante ressaltar que cada característica de qualidade depende da classe do produto de *software* e da ótica do usuário interno ou externo.

CMM (PAULK et al., 1993) classifica as organizações quanto ao nível de maturidade. Entende-se como nível de maturidade como os estágios bem definidos que evoluem para um processo de *software* maduro, ou seja, cada estágio compreende em um conjunto de objetivos, que quando alcançados estabiliza um componente do processo de *software*. Desta forma, uma organização madura possui uma capacidade organizacional para gerenciar o desenvolvimento e manutenção de *software*. Este modelo avalia as áreas de um processo (KPA – *key process areas*). Para cada KPA existem requisitos estabelecidos (metas, comprometimentos e capacidades) e orientações (práticas-chave) para o seu atendimento. Então, uma organização evolui de um nível para outro atendendo aos requisitos estabelecidos para as KPAs. Há 18 KPA´s de processo distribuídas nos níveis de maturidade, com exceção do Nível 1 (Inicial), como é apresentado na **FIG. 7.3-2**. Na **FIG. 7.3-2** também são apresentadas, de forma resumida para cada nível de maturidade, as características, os objetivos e as atuações para mudança de nível.

A Norma 15504 (ISO15504, 1998) é o projeto SPICE que consiste em elaborar um padrão aplicável à melhoria de processos e à determinação de capacidade levando em consideração os métodos e normas já existentes – CMM/SEI, STD/Compita, Trillium/Bell, SQPA/HP, Bootstrap, SAM/BT, HelthCheck/BT, ISO 9001. Esta norma objetiva-se a ser mais abrangente que os modelos existentes e, além disso, ser mais específico que a norma ISO

9001 (ROCHA et al., 2001). Da ISO/IEC 12207 herdou a arquitetura dos processos do ciclo de vida do *software* e do CMM herdou o conceito de níveis de maturidade de processos.

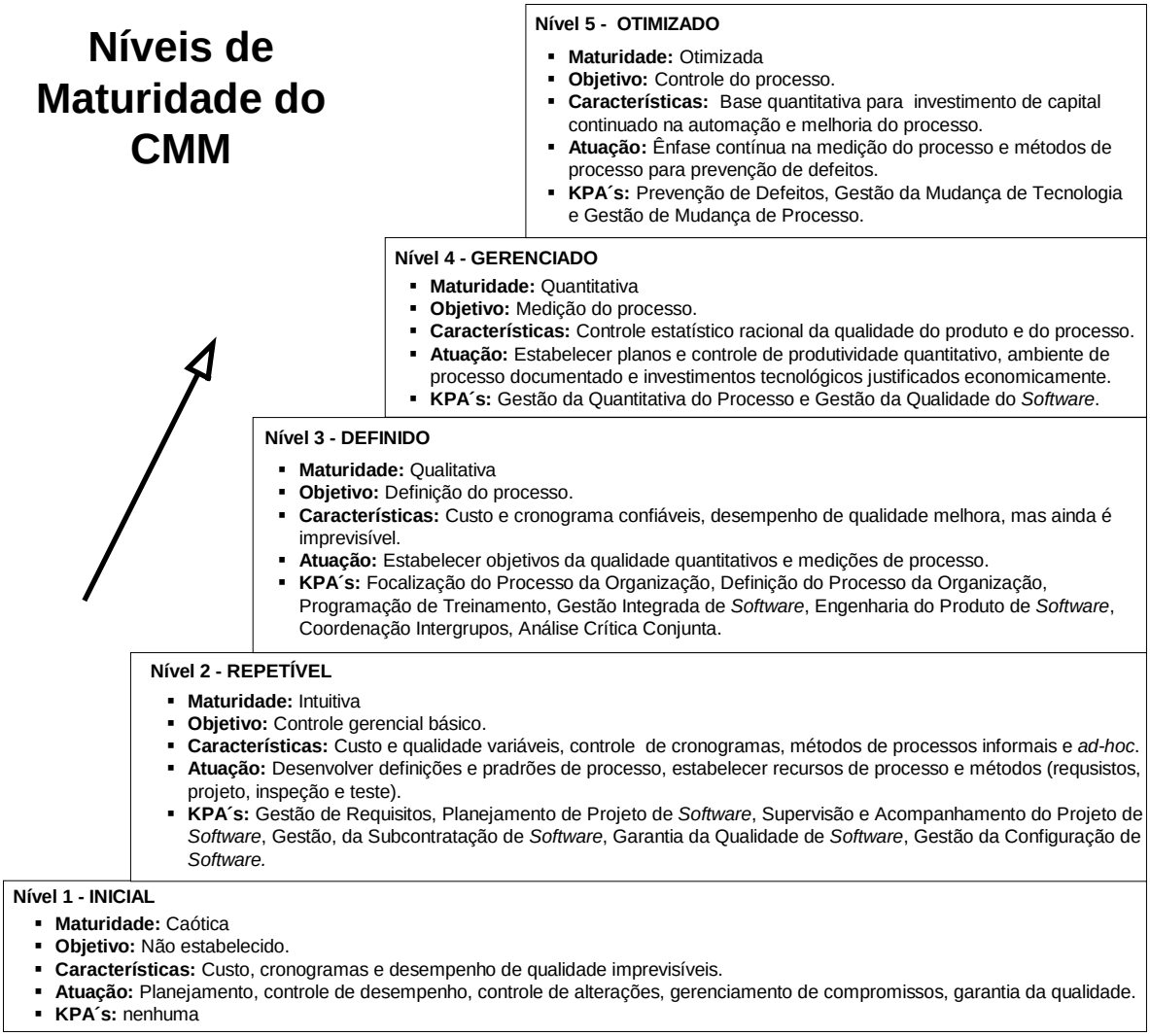


FIG. 7.3-2: Os níveis de maturidade do CMM.

CMMI (SEI, 2000) é uma variante da CMM (PAULK et al., 1993), que apresenta uma abordagem mais flexível em duas dimensões: Contínuo e Organizado. Os benefícios da representação por Contínuo consistem em: seleção da ordem de melhoria que melhor apresenta os objetivos do negócio da organização e redução das áreas de risco da organização; comparações entre organizações sobre uma área de processo pela comparação dos resultados; facilidade de comparação com o modelo de melhoria de processo da ISO 15504 – SPICE (ISO15504, 1998). A implantação do modelo por Organizado tem como benefícios: uma seqüência de melhorias, iniciando pelas práticas gerenciais básicas e progredindo para os níveis predefinidos sucessivos - cada nível serve como fundação para o próximo; possibilita

comparações entre organizações pelo uso dos níveis de maturidade; fácil migração do modelo CMM para o CMMI; fornece um único número que sumarize os resultados da avaliação e permita comparação entre organizações. Ambas representações oferecem resultados equivalentes, se utilizadas para melhoria do processo ou avaliações. A representação do Organizado organiza áreas de processo dentro de cinco níveis de maturidade para suportar e guiar o processo de melhoria: Inicial, Gerenciado, Definido, Quantitativamente Gerenciado e Otimização. Cada área de processo possui objetivos específicos, práticas específicas, objetivos genéricos e práticas genéricas. A representação do Organizado usa quatro características comuns para organizar as práticas genéricas, conforme são apresentadas na **FIG. 7.3-3** (SEI,2002).

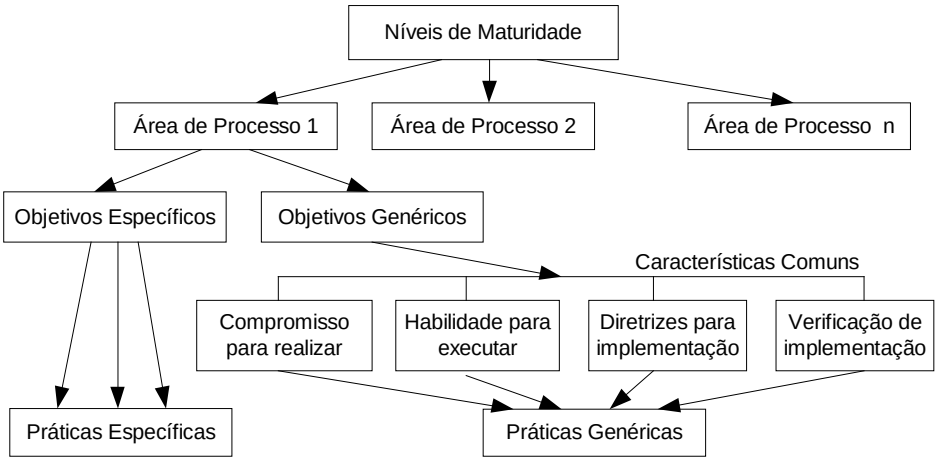


FIG. 7.3-3: Componentes do Modelo CMMI – representação Organizado.

7.3.1 INDICADORES DE PRODUTO DE *SOFTWARE*

Para os indicadores do produto de *software*, a norma ISO/IEC 9126 (ISO9126, 2000) representa a atual padronização mundial para a qualidade de produtos de *software*, definindo seis características de qualidade e métricas para um produto de *software*. Cada uma dessas características divide-se em subcaracterísticas, que associadas a uma definição de métricas para correlacioná-las ao produto de *software*, permite uma avaliação da sua qualidade. Além disso, cada característica de qualidade depende da classe do produto de *software* e da ótica do usuário (cliente, usuário final, equipe de desenvolvimento e equipe de suporte). Desta forma, um *software* deve ter um conjunto de atributos que evidenciam as características e suas

subcaracterísticas. Na **TAB. 7.3.1-1** são apresentados os indicadores de qualidade do produto da norma ISO/IEC 9126 (ISO9126, 2000).

TAB. 7.3.1-1: Indicadores de qualidade do produto da norma ISO/IEC 9126-1:2000.

1	Funcionalidade		
	Conjunto de atributos que evidencia a existência de um conjunto de funções, que satisfazem as necessidades explícitas ou implícitas dos usuários e suas propriedades especificadas.		
	1.1	Adequação	Evidencia a presença de um conjunto de funções e sua apropriação às tarefas especificadas.
	1.2	Acurácia	Evidencia a geração de resultados ou efeitos corretos ou conforme acordados.
	1.3	Interoperabilidade	Evidencia sua capacidade de interagir com outros sistemas especificados.
	1.4	Conformidade	Evidencia o quanto o *software* está de acordo com as normas, convenções ou regulamentações previstas em leis e descrições similares, relacionadas à funcionalidade da aplicação.
	1.5	Segurança de acesso	Evidencia sua capacidade de evitar acessos não autorizados, acidentais ou deliberados, a programas e dados.
2	Confiabilidade		
	Conjunto de atributos que evidenciam a capacidade do *software* de manter seu nível de desempenho, sob condições estabelecidas, durante um período de tempo estabelecido. No produto de *software*, falhas devido ao envelhecimento não ocorrem, as limitações na confiabilidade são decorrentes de defeitos na especificação dos requisitos, projeto e implementação. As falhas decorrentes destes defeitos dependem de como o produto de *software* é usado e das opções do programa selecionadas, e não do tempo decorrido.		
	2.1	Maturidade	Evidencia a freqüência de falhas causadas por defeitos no *software*.
	2.2	Tolerância a falhas	Evidencia sua capacidade em manter um nível de desempenho no caso das falhas no *software* ou de violação nas interfaces especificadas.
	2.3	Recuperabilidade	Evidencia sua capacidade de reestabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e o tempo e esforço necessário para tal.
	2.4	Conformidade	Evidencia o quanto o *software* está de acordo com as normas, convenções ou regulamentações relacionadas à confiabilidade da aplicação.
3	Usabilidade		
	Conjunto de atributos que evidenciam o esforço necessário ao uso, bem como a sua homologação individual por um conjunto de usuários estabelecido ou subentendido.		
	3.1	Inteligibilidade	Evidencia o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade.
	3.2	Apreensibilidade	Evidencia o esforço do usuário para aprender sua aplicação (por exemplo: controle de operação, entradas, saídas).
	3.3	Operacionalidade	Evidencia o esforço do usuário para sua operação e controle da sua operação.
	3.4	Atratividade	Evidencia sua capacidade de ser atrativo ao usuário, como uso de cores e gráficos ou interfaces gráficas.
	3.5	Conformidade	Evidencia o quanto o *software* está de acordo com as normas, convenções, guia de estilo ou regulamentações relacionadas à usabilidade da aplicação.
4	Eficiência		
	Conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do *software* e a quantidade de recursos utilizados, sob condições estabelecidas. Recursos podem incluir outros produtos de *software*, facilidades de hardware, materiais como papel de impressora, discos flexíveis e serviços de operação, manutenção e suporte.		
	4.1	Comportamento em relação ao tempo	Evidencia seu tempo de resposta, tempo de processamento e velocidade na execução de suas funções.
	4.2	Comportamento em relação aos recursos	Evidencia a quantidade de recursos usados e a duração de seu uso na execução de suas funções.
	4.3	Conformidade	Evidencia o quanto o *software* está de acordo com as normas ou convenções relacionadas à eficiência da aplicação.

5	Manutenibilidade		
	Conjunto de atributos que evidenciam o esforço necessário para fazer modificações especificadas no *software*. Modificações podem incluir correções, melhorias ou adaptações do *software*, mudanças no ambiente e nos requisitos e especificações funcionais.		
	5.1	Analisabilidade	Evidencia o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas.
	5.2	Modificabilidade	Evidencia o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais.
	5.3	Estabilidade	Evidencia o risco de efeitos inesperados ocasionados por modificações.
	5.4	Testabilidade	Evidencia o esforço necessário para validar o *software* modificado.
	5.5	Conformidade	Evidencia o quanto o *software* está de acordo com as normas ou convenções relacionadas à manutenibilidade da aplicação.
6	Portabilidade		
	Conjunto de atributos que evidenciam a facilidade do *software* ser transferido de um ambiente para outro. O ambiente pode incluir o ambiente organizacional, de *hardware* ou de *software*.		
	6.1	Adaptabilidade	Evidencia sua capacidade de ser adaptado a diferentes ambientes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo *software* considerado.
	6.2	Capacidade para ser instalado	Evidencia o esforço necessário para sua instalação em um ambiente especificado.
	6.3	Capacidade de coexistir	Evidencia sua capacidade de co-existir com outros *softwares* independentes em um ambiente comum compartilhando os mesmos recursos.
	6.4	Capacidade para substituir	Evidencia o esforço necessário de utilização do *software* desenvolvido no lugar de outro *software* e no ambiente estabelecido para esse outro *software* específico.
	6.5	Conformidade	Evidencia o quanto o *software* está de acordo com as normas ou convenções relacionadas à portabilidade da aplicação.

A norma ISO/IEC 9126 (ISO9126, 2000) enumera as características e as subcaracterísticas de um produto de *software*, mas não define como pontuar um *software* em cada um dos itens. Algumas características podem ser realmente medidas: tempo de execução de um programa, número de linhas de código, número de erros encontrados em uma sessão de teste ou o tempo médio entre falhas. Para estas características é possível utilizar uma ferramenta ou *software* para realizar as medições. Mas, existem características altamente subjetivas, necessitando formular um conjunto de perguntas do tipo “sim ou não”, onde as respostas “sim” indicam melhor pontuação para as características. Depois de avaliar o *software*, respondendo a cada pergunta, deve verificar o percentual de “sim” em relação à quantidade de perguntas. Pode-se melhorar atribuindo um número mínimo de perguntas para cada característica e para as perguntas mais importantes podem ter pesos maiores, e, ainda, para cada pergunta pode ter cinco opções de resposta, indicando um escore diferenciado. Existem outras formas diferentes de medir uma característica, baseada em conceitos do tipo: “não se aplica”, “não satisfaz”, “satisfaz parcialmente”, “satisfaz totalmente” e “excede os padrões”.

7.3.2 CICLO DE VIDA DO SOFTWARE NOS PADRÕES DA ISO 12207

A norma ISO/IEC 12207 (ABNT12207, 1998) estabelece um *framework* comum para o ciclo de vida de *software*, mas não especifica os detalhes de como implementar estes requisitos de processo. Na **TAB. 7.3.2-1** é apresentada uma lista completa e resumida dos processos desta norma. A versão brasileira da norma foi preparada pela ABNT (Associação Brasileira de Normas Técnicas) em outubro de 1998, NBR ISO/IEC 12207 (ABNT12207, 1998).

TAB. 7.3.2-1: Processos do Ciclo de Vida do *Software* nos padrões da ISO12207.

Processos	Descrição
1. Processos Fundamentais	Processos que atendem as partes principais (pessoa ou organização). Essas partes são responsáveis por iniciar ou executar o desenvolvimento, operação ou manutenção de *software* durante o seu ciclo de vida.
1.1 Aquisição	Atividades do adquirente do *software*. Inclui: definição da necessidade de adquirir um *software* (produto ou serviço), pedido de proposta, seleção de fornecedor, gerência da aquisição e aceitação do *software*.
1.2 Fornecimento	Atividades do fornecedor de *software*. Inclui: preparar uma proposta, assinatura de contrato, determinação de recursos necessários, planos de projeto e entrega do *software*.
1.3 Desenvolvimento	Atividades do desenvolvedor de *software*. Inclui: análise de requisitos, projeto de arquitetura, codificação, integração, testes, instalação e aceitação do *software*.
1.4 Operação	Atividades do operador de *software*. Inclui: teste operacional, operação de *software* e suporte operacional aos usuários.
1.5 Manutenção	Atividades do manutenedor de *software*. Inclui: análise do problema e da modificação, implementação da modificação, revisão e aceitação da manutenção, migração e descontinuação do *software*.
2 Processos de Apoio	Processos que provêem apoio a um outro processo como parte integrante, com propósito distinto e que contribuem para o sucesso e qualidade do projeto de *software*.
2.1 Documentação	Registro de informações produzidas por um processo ou atividade. Inclui: planejamento, projeto, desenvolvimento, produção, edição, distribuição e manutenção dos documentos necessários a gerentes, engenheiros e usuários do *software*.
2.2 Gerência de Configuração	Identificação e controle dos itens do *software*. Inclui: controle de armazenamento. Liberações, manipulação, distribuição e modificação de cada um dos itens que compõem o *software*.
2.3 Garantia da Qualidade	Garante que os processos e produtos de *software* estejam em conformidade com os requisitos e os planos estabelecidos.
2.4 Verificação	Determina se os produtos de *software* de uma atividade atendem completamente aos requisitos ou condições impostas a eles.
2.5 Validação	Determina se os requisitos e o produto final (sistema ou *software*) atendem ao uso específico proposto.
2.6 Revisão Conjunta	Define as atividades para avaliar a situação e produtos de uma atividade de um projeto, se apropriadas.
2.7 Auditoria	Determinar adequação aos requisitos, planos e contrato, quando apropriado.
2.8 Resolução de Problemas	Analisar e solucionar os problemas de qualquer natureza ou fonte, descobertos durante a execução do desenvolvimento, operação, manutenção ou outros processos.
3 Processos	Processos que estabelece e implementam uma estrutura subjacente constituída de processos de ciclo de vida e pessoal associados, melhorando continuamente a

Organizacionais		estrutura e os processos. São tipicamente empregados fora do domínio de projetos e contratos específicos.
	3.1 Gerência	Gerenciamento de processos.
	3.2 Infra-estrutura	Fornecimento de recursos para outros processos. Inclui: *hardware*, *software*, ferramentas, técnicas, padrões de desenvolvimento, operação ou manutenção.
	3.3 Melhoria	Atividades para estabelecer, avaliar, medir, controlar e melhorar um processo de ciclo de vida de *software*.
	3.4 Treinamento	Atividades para prover e manter pessoal treinado.

A ISO/IEC 12207 (ABNT12207, 1998) define como os processos podem ser usados de diferentes maneiras por diferentes organizações ou por parte destas, representando diversos pontos de vista para esta utilização. Existem cinco visões diferentes: contrato, gerenciamento, operação, engenharia e apoio. Cada uma destas visões representa a forma como uma organização emprega estes processos, agrupando-os de acordo com suas necessidades e objetivos. As visões têm o objetivo de organizar melhor a estrutura de uma empresa, para definir suas gerências e atividades alocadas às suas equipes.

7.3.3 PROCESSOS VOLTADOS A REUSO DE *ASSETS* PELA IEEE 1517

A norma IEEE 1517 (1999) consiste em:

- Estabelecer *um framework* comum para processos de reuso e como integrar a prática de reuso nos processos de ciclo de vida do *software*;
- Integrar processos de reuso, atividades e tarefas com os processos de ciclo de vida do *software* descritos em ISO/IEC Std 12207.0-1996 (ABNT12207, 1998);
- Definir os processos, atividades e tarefas que são necessários para executar e administrar a prática de reuso em um único projeto de *software*, por vários projetos de *software* e por uma organização;
- Facilitar a comunicação entre adquirentes de *software*, fornecedores, desenvolvedor, gerentes de programa de reuso, gerentes de *asset* e engenheiros de domínio ao prover um entendimento comum de reuso e por unificar terminologia de reuso.

O *framework* de processo de reuso da IEEE 1517 (IEEE1517, 1999) cobre ambos o ciclo de vida de um produto de *software* desenvolvido de *assets* e o ciclo de vida de um *asset*. Os processos de reuso abordam como os produtos de *software* são construídos com *assets* e como identificar, construir, manter e gerenciar estes *assets*.

Esta norma descreve processos de reuso como definido pela arquitetura de Basili (BASILI et al., 1992) para abstração de processo. Nesta arquitetura, a responsabilidade por

cada processo é atribuída a **agente**. O agente é o indivíduo ou organização responsável para executar um processo ou atividade. Entretanto, esta norma é limitada a:

- Descrever um *framework* de alto nível para processos de reuso, mas não os detalhes de como executar as atividades e tarefas incluídas nos processos;
- Descrever as responsabilidades inerentes a vários processos, mas não os dados detalhados ou relações de controle entre os processos;
- Especificar os processos de reuso do ciclo de vida de *software*, mas não prescreve um modelo específico de processo de ciclo de vida de *software* ou metodologia de *software*;
- Visualizar processos de reuso como uma tarefa de atribuir responsabilidades a agentes, mas não como umas séries de passos (procedimentos) para ser executado;
- Especificar provisões para adquirirem *assets*, mas não provisões para integração de *assets* comerciais *off-the-shelf* (COTS) que não são fornecidos em forma de código-fonte;
- Neste padrão, como em ISO/IEC 12207.0-1996 (ABNT12207, 1998), há um número de listas para as tarefas; nenhuma delas é exaustiva.

Cada processo de ciclo de vida é dividido em um conjunto de atividades e os requisitos de cada atividade são especificados em um conjunto de tarefas.

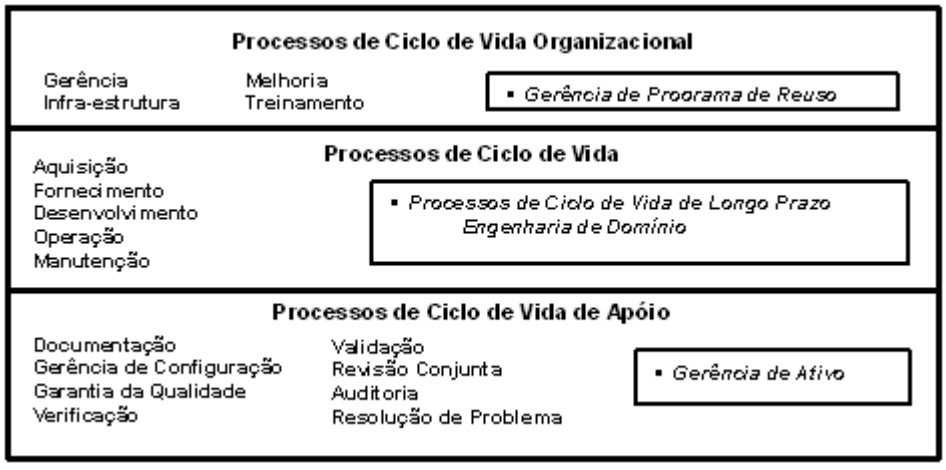


FIG. 7.3.3-1: Processos de ciclo de vida da ABNT 12207 com as extensões para processos de ciclo de vida de reuso da IEEE 1517.

Os processos de ciclo de vida do *software*, atividades e tarefas que são exigidos para praticar reuso são apresentados em quatro categorias de reuso (veja **FIG. 7.3.3-1**):

1 - **Desenvolvimento, operação e manutenção de produtos de *software* com *assets***: com a adição das atividades e tarefas de reuso, os processos de ciclo de vida fundamentais empregam modelos de domínio, arquiteturas de domínio e *assets* para desenvolver e manter

produtos de *software*. Agentes responsáveis pela execução dos Processos Fundamentais são: adquirente, fornecedor, desenvolvedor, operador e mantenedor.

2 - **Desenvolvimento e manutenção de *assets*:** o processo de Gerência de Domínio especifica um processo de ciclo de vida para um projeto de *asset*, fornecendo *asset* que pode ser usado por vários projetos. Este *asset* pode ser usado por processos do ciclo de vida fundamentais para adquirir, prover, desenvolver, operar e manter produtos de *software*. Agente responsável por desenvolver o Processo de Engenharia de Domínio é o engenheiro de domínio.

3 - **Gerência da prática de reuso**: o processo de Gerência de Programa de Reuso é acrescentado aos processos organizacionais da ISO/IEC 12207 (Administração, Infra-estrutura, Melhoria e Treinamento) para estabelecer um processo de estrutura de programa de reuso integrado e pessoal que podem ser administrados e melhorados. Agente responsável para executar o Processo de Gerência de Programa de Reuso é o gerente do programa de reuso.

4 - **Gerência de assets**: este processo especifica um processo que apóia outro processo como uma parte integrante com um propósito distinto e contribui ao sucesso e qualidade do projeto de *software*. A gerência de *asset* apóia outros processos de ciclo de vida, como Desenvolvimento, Manutenção e Engenharia de Domínio, especificando as atividades necessárias para administrar certificação de *asset*, classificação, armazenamento, recuperação, controle de versão e controle de alteração. O Processo de Gerência de *Asset* é acrescentado ao conjunto de Processos de Apóio definidos em ISO/IEC 12207 (ABNT12207, 1998). Os processos de apóio como Documentação, Gerência de Configuração e Revisão Conjunta são empregados para assegurar execução de processos de reuso como Engenharia de Domínio e Gerência de Programa de Reuso.Agente responsável para executar o Processo de Gerência de *Asset* é o gerente de *asset*.

7.4 APÊNDICE 4: *CHECKLISTS* E TÉCNICAS DE LEITURA

7.4.1 ARTEFATOS: *CHECKLISTS*

Garantia da Qualdade de Software
Checklist: Necessidades do Cliente e Glossário — Versão: 1.5
VENC001/03 — Data da versão: 01/03/2004

Checklist: Necessidades do Cliente e Glossário

Artefato Verificado: CTNC001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

1. Aspecto: Apresentação do Documento

Este aspecto deve analisar as questões gerais de apresentação do documento.	P	C	I
1.1 O documento está de acordo com o *template* padrão?	S		
1.2 O documento teve ortografia e gramática checada?	S		
1.3 O documento está livre de erros de *layout*?	S		
1.4 O glossário dispõe de todas as definições e explicações que o revisor necessita para o completo entendimento das necessidades e características desejadas?	S		
1.5 Os números das linhas do texto do documento estão impressos para facilitar a referência de localização específica durante a verificação?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

2. Aspecto: Qualidade

Este aspecto descreve a qualidade que as necessidades, isto é, as funcionalidades e as características desejadas, devem apresentar no documento.	P	C	I
2.1 As necessidades apresentam nível de detalhamento adequado?	S		
2.2 Todas as características de desempenho estão corretamente especificadas?	S		
2.3 Toda as considerações de segurança estão corretamente especificadas?	S		
2.4 Outras características desejadas de qualidade pertinentes estão explicitamente documentadas e quantificadas de acordo com o objetivo especificado?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

3. Aspecto: Completude

Este aspecto descreve o que o documento deve apresentar com relação à consistência das necessidades e a completude destes documentos.	P	C	I
3.1 As funcionalidades e características desejadas provêem uma base adequada para análise de requisitos da engenharia de domínio?	S		
3.2 O documento Necessidades do Cliente inclui todas as funcionalidades do sistema?	S		
3.3 O documento Necessidades do Cliente inclui todas as características desejadas que o sistema precisa ter?	S		
3.4 Todas as referências internas das necessidades estão corretas?	S		
3.5 As necessidades fornecem uma base adequada para o projeto?	S		
3.6 A prioridade de implementação das necessidades está definida?	S		
3.7 Todo *hardware*, *software* e interfaces de comunicação estão definidos?	S		
3.8 Os algoritmos intrínsecos para as necessidades funcionais estão definidos?	S		
3.9 A necessidade inclui todo o conhecimento do cliente ou necessidades de sistema?	S		
3.10 O comportamento esperado está documentado para todas as condições de erro?	S		
3.11 Todas as funções críticas de tempo estão identificadas e seus critérios de tempo especificados?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

Restrito — Instituto Militar de Engenharia — Página: 1/4

Checklist: Necessidades do Cliente e Glossário

Artefato Verificado: CTNC001/03

A coluna P (Previsão) ccontém a resposta correta desejada e na coluna C (Confirmação) o revidor deve preencher com a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados identificados e comentados na parte de Não-Conformidades.

4. Aspecto: Rastreabilidade

Este aspecto deve analisar a rastreabilidade das funcionalidades e características desejadas.

		P	C	I
4.1	Cada necessidade é exclusivamente e corretamente identificada?	S		
4.2	Cada funcionalidade pode ser rastreada nos requisitos de *software* de mais alto nível, por exemplo, requisitos de sistema, casos de uso?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

5. Aspecto: Corretude

Este aspecto deve analisar a corretude das funcionalidades e características desejadas.

		P	C	I
5.1	Cada necessidade é verificável por teste de qualificação, inspeção ou revisão?	S		
5.2	Cada característica desejada é verificável por teste de qualificação, inspeção ou revisão?	S		
5.3	As funcionalidades evitam conflitos entre si e com todas as características desejadas?	S		
5.4	As características desejadas evitam conflitos entre si?	S		
5.5	As necessidades estão escritas em uma linguagem simples e concisa, sem ambigüidade, possibilitando o completo entendimento?	S		
5.6	Cada necessidade está dentro do escopo do projeto?	S		
5.7	Cada necessidade está livre de erros de conteúdo e gramaticais?	S		
5.8	Toda informação necessária para as funcionalidades e características desejadas estão devidamente especificadas? Em caso negativo, identificar no verso?	S		
5.9	Todas as necessidades podem ser implementadas dentro das restrições conhecidas?	S		
5.10	Todas as funcionalidades são de fato funcionalidades e não soluções de projeto ou de implementação?	S		
5.11	As mensagens de erros especificadas são únicas e significativas?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist : Plano de Testes de Qualificação

Artefato Verificado: CTPT001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

1. Aspecto: Planejamento de teste

Este aspecto deve analisar o planejamento de teste.	P	C	I
1.1 É viável a realização de teste?	s		
1.2 Os objetivos de testes estão definidos?	s		
1.3 Todas as dependências de testes estão referenciadas, por exemplo, função de *driver, hardware* ?	s		
1.4 O ambiente de teste está completamente especificado?	s		
1.5 As condições de parada e reinício de teste estão definidas?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

2. Aspecto: Padrões e Rastreabilidade

Este aspecto deve analisar os padrões e a rastreabilidade dos requisitos nos casos de teste.	P	C	I
2.1 Todos os padrões especificados no plano de teste estão sendo seguidos?	s		
2.2 O plano de teste estabelece todas as especificações, padrões e documentos necessários para realização do teste?	s		
2.3 Cada requisito especificado na fase de análise de requisito está coberto pelo plano de teste de aceitação?	s		
2.4 Todos os casos de teste podem ser localizados pelo requisito especificado na fase de análise de requisito?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

3. Aspecto: Corretude e Completude

Este aspecto deve analisar a corretude e completude dos testes.	P	C	I
3.1 O teste de cobertura é suficiente para fornecer confiabilidade de que a função que é testada opera corretamente dentro de seu ambiente planejado?	s		
3.2 O teste de integração testa cada interface dos documentos de projeto correspondentes?	s		
3.3 A descrição da função de teste documentada no plano de teste está complete e precisa?	s		
3.4 Todos os critérios de entrada e saída de teste são suficientes e adequados ao ambiente planejado?	s		
3.5 Todos os itens excluídos do teste estão devidamente documentados?	s		
3.6 O plano de teste está completo, correto e sem ambigüidade?	s		
3.7 Os níveis desejados de exigências e cobertura de código estão quantitativamente especificados?	s		
3.8 As condições de entradas válidas e inválidas são testadas?	s		
3.9 Todos os critérios de omissão e falhas estão definidos?	s		
3.10 O plano de teste esboça os níveis de aceitabilidade para tolerância a falhas?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

4. Aspecto: Teste de Regressão

Este aspecto deve analisar o retestar.	P	C	I
4.1 Os casos de teste adequados e necessários para retestar funções anteriormente testadas são identificados?	s		
4.2 Todas as alterações de código são suficientemente testadas principalmente as alterações de interfaces?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

5. Aspecto: Recursos e Escalabilidade

Este aspecto deve analisar os recursos envolvidos para ajudar na realização do teste.	P	C	I
5.1 Todos os recursos humanos e de *hardware* e *software* estão sendo considerados?	s		
5.2 Os recursos, como desenvolvimento ou obtenção de utilitários e ferramentas facilitadores de testes estabelecidos, têm sido escalados no tempo apropriado?	s		
5.3 Estão estabelecidos os papéis e as responsabilidades das pessoas envolvidas nas atividades de teste?	s		
5.4 Todas as possibilidades de contenção de recurso ou restrições de disponibilidade estão sendo consideradas?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist : Arquitetura

Artefato Verificado: CTAR001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

1. Aspecto: Estrutura

Este aspecto deve verificar a estrutura do diagrama arquitetural do domínio.	P	C	I
1.1 A arquitetura permite implementação de todas as necessidades (funcionalidades e características desejadas)?	S		
1.2 A arquitetura foi decomposta adequadamente?	S		
1.3 As funções da aplicação foram alocadas adequadamente aos componentes?	S		
1.4 A arquitetura provê uma base adequada para fase seguinte do projeto?	S		
1.5 A arquitetura é possível de ser implementada?	S		
1.6 O conjunto de programa pode ser integrado e testado incrementalmente?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

2. Aspecto: Corretude e Qualidade

Este aspecto deve descrever a qualidade que as necessidades (funcionalidades e características desejadas) devem apresentar no diagarma e a adequação a estas necessidades.	P	C	I
2.1 A arquitetura evita redundância desnecessária?	S		
2.2 Todas as características desejadas de desempenho foram aplicadas?	S		
2.3 Todas as considerações de segurança foram aplicadas?	S		
2.4 A arquitetura considera todas as restrições?	S		
2.5 Toda a estrutura de dados necessária está definida?	S		
2.6 A arquitetura proposta vai satisfazer todos os atributos de qualidade especificados e os objetivos de desempenho?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

3. Aspecto: Padrões e Rastreabilidade

Este aspecto deve verificar as questões de adequação aos padrões de design e arquiteturais, e rastreabilidade das necessidades e características desejadas.	P	C	I
3.1 Todos os padrões de arquitetura foram seguidos?	S		
3.2 De qualquer parte da arquitetura as necessidades podem ser rastreadas?	S		
3.3 Os padrões de componentes estão sendo usados sempre que possíveis?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

4. Aspecto: Lógica

Este aspecto descreve o que o documento deve apresentar com relação à consistência das necessidades e a completude destes documentos.	P	C	I
4.1 Há alguma lógica perdida ou incompleta?	S		
4.2 Todos os possíveis casos estão sendo considerados?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

5. Aspecto: Interface e Clareza

Este aspecto deve verificar as interfaces de cada componente das camadas e inter camadas, e a objetividade e clareza do diagrama arquitetural.	P	C	I
5.1 Todas as interfaces estão claras e bem definidas?	S		
5.2 Os mínimos dados são passados a cada interface?	S		
5.3 Os mínimos dados globais de sistema são incluídos?	S		
5.4 A arquitetura está claramente representada, inclusive o fluxo de dados, controle de dados e interfaces?	S		
5.5 As representações do projeto estão consistentes entre si?	S		
5.6 Todas as decisões, dependências e suposições para este projeto estão devidamente documentadas?	S		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist: Código-Fonte em JAVA
Artefato Verificado (pacote): CTCF001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

1. Aspecto: Defeito de Declaração de Constantes, Variável e Atributo

Este aspecto deve verificar defeitos em declarações de constantes, variáveis e atributos.	P	C	I
1.1 Os nomes das variáveis e constantes estão de acordo com a convenção de nomes?	s		
1.2 Todas as variáveis e atributos estão com nomes distintos?	s		
1.3 Todas as variáveis e atributos estão digitados corretamente?	s		
1.4 Todas variáveis e atributos estão inicializados corretamente, se for o caso?	s		
1.5 Todas as variáveis locais estão declaradas dentro do seu escopo, ou seja, não existe variáveis não locais que podem ser locais?	s		
1.6 Todas as variáveis de controle de *loop* do comando *for* estão declaradas no escopo do método?	s		
1.7 Todos os atributos têm modificadores de acesso apropriados (*private, protected, public*)?	s		
1.8 Existem constantes literais que devem ser nomeadas constantes?	n		
1.9 Existem variáveis ou atributos que devem ser constantes?	n		
1.10 Existem atributos que devem ser variáveis locais ao método?	n		
1.11 Existem atributos estáticos que devem ser não estáticos ou vice-versa?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

2. Aspecto: Defeito de Definição de Método

Este aspecto deve verificar os defeitos de definição de método.	P	C	I
2.1 Os nomes dos métodos estão de acordo com a convenção de nomes?	s		
2.2 Todos os valores de parâmetro dos métodos foram checados antes de serem usados?	s		
2.3 Todos os valores de parâmetros são usados no cálculo ou manipulados?	s		
2.4 Para todos os métodos que implementam função têm o valor correto de ponto de retorno?	s		
2.5 Todos os métodos têm modificadores de acesso apropriados (*private* , *protected* , *public*)?	s		
2.6 Existem métodos estáticos que devem ser não estáticos ou vice-versa?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

3. Aspecto: Defeito de Definição de Classe

Este aspecto deve verificar os defeitos de definição de classe.	P	C	I
3.1 Cada classe tem seu construtor apropriado?	s		
3.2 Todas as subclasses têm membros comuns que devem estar na superclasse?	n		
3.3 A hierarquia de herança de classe pode ser simplificada?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

Verificador: _____
Nível de conhecimento do domínio: ____
(0 - baixo / 1 - médio / 2 - alto)
Data da Verificação: ____ / ____ / ____

Tempo de preparação: _____ Hr
Tempo de realização da verificação: _____ Hr
Qtde linhas de código não comentário: _____
Assinatura: _____

Checklist : Código-Fonte em JAVA
Artefato Verificado (pacote): CTCF001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

4. Aspecto: Defeito de Referência de Dados
Este aspecto deve verificar os defeitos de referência de dados.

		P	C	I
4.1	Em todos os objetos ou referências de *array*, o valor é diferente de nulo?	s		
4.2	Em todas as referências de *array*, cada valor está dentro dos limites definidos?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

5. Aspecto: Defeito de Operadores Relacionais
Este aspecto deve verificar os defeitos de expressões lógicas.

		P	C	I
5.1	Em todos os testes booleanos, a condição correta é checada?	s		
5.2	Os operadores de comparação estão corretos?	s		
5.3	Cada expressão booleana foi simplificada resultante de procedimento de negação?	s		
5.4	Cada expressão booleana está correta?	s		
5.5	Existem efeitos colaterais impróprios de uma comparação?	n		
5.6	Têm um "&" inadvertidamente trocado por um "&&" ou um "| " por um "||"?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

6. Aspecto: Defeito de Fluxo de Controle
Este aspecto deve verificar os defeitos de controle de estruturas de laço de repetição e condicionais.

		P	C	I
6.1	Cada *loop* utiliza a melhor escolha de laço de repetição?	s		
6.2	Todos *loops* terminam?	s		
6.3	Quando há várias saídas de um *loop*, cada saída é necessária e controlada corretamente?	s		
6.4	Cada estrutura *switch-case* tem um caso *default*?	s		
6.5	A falta da declaração *break* na estrutura *switch-case* está correta e comentada?	s		
6.6	As declarações *break* desviam o controle para o local correto?	s		
6.7	O aninhamento de várias estruturas *if-else* está correto?	s		
6.8	As estruturas nulas de controle estão corretas e comentadas?	s		
6.9	Todas as exceções são controladas adequadamente?	s		
6.10	Todos os métodos terminam?	s		
6.11	O aninhamento de estruturas *if-else* pode ser convertido em uma estrutura *switch-case*?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

Verificador: _____
Nível de conhecimento do domínio: _____
(0 - baixo / 1 - médio / 2 - alto)
Data da Verificação: _____ / _____ / _____

Tempo de preparação: _____ Hr
Tempo de realização da verificação: _____ Hr
Qtde linhas de código não comentário: _____
Assinatura: _____

Checklist : Código-Fonte em JAVA

Artefato Verificado (pacote): CTCF001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

7. Aspecto: Defeito de Cálculo Numérico

Este aspecto deve verificar os defeitos de computação.	P	C	I
7.1 Na expressão com mais de um operador, a ordem de precedência das operações está correta?	s		
7.2 Os parênteses usados são para evitar ambigüidade?	s		
7.3 Existe algum cálculo que está com diferentes tipos de dados?	n		
7.4 Existe algum cálculo que os mesmos tipos de dados estão em diferentes unidades de dados?	n		
7.5 Há possibilidade de ocorrer *overflow* ou *underflow* são durante um cálculo?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

8. Aspecto: Defeito de *Input-Output*

Este aspecto deve verificar os defeitos de entrada e saída de dados.	P	C	I
8.1 Todos os arquivos são abertos antes de serem usados?	s		
8.2 Os atributos de entrada do objeto são consistentes com o arquivo em uso?	s		
8.3 Todos os arquivos são fechados após serem usados?	s		
8.4 Todas as exceções de I/O são controladas?	s		
8.5 Existem erros gramaticais e de sintaxe em qualquer texto impresso?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

9. Aspecto: Defeito de Empacotamento

Este aspecto deve verificar os defeitos de empacotamento e sua apresentação	P	C	I
9.1 A endentação padrão e o formato do *layout* são usados constantemente?	s		
9.2 Cada método tem aproximadamente menos do que 60 linhas?	s		
9.3 Cada pacote tem aproximadamente menos do que 600 linhas?	s		
9.4 Existe um baixo nível de acoplamento entre os pacotes (métodos e classes)?	s		
9.5 Existe um alto nível de coesão dentro de cada pacote (métodos ou classe)?	s		
9.6 As bibliotecas de Java são usadas onde e quando adequadamente?	s		
9.7 Existe código repetitivo que pode ser substituído por uma chamada a um método que provê o comportamento do código repetitivo?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

10. Aspecto: Defeito de Armazenamento

Este aspecto deve verificar os defeitos de utilização de armazenamento.	P	C	I
10.1 Os objetos e os conjuntos de referências de *array* são nulos uma vez que os objetos ou *arrays* não são mais necessários?	s		
10.2 Os *arrays* são suficientemente grandes?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

Verificador: _____
Nível de conhecimento do domínio: _____
(0 - baixo / 1 - médio / 2 - alto)
Data da Verificação: ____ / ____ / _____

Tempo de preparação: _____ Hr
Tempo de realização da verificação: _____ Hr
Qtde linhas de código não comentário: _____
Assinatura: _____

Checklist : Código-Fonte em JAVA

Artefato Verificado (pacote): CTCF001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

11. Aspecto: Defeito de Interface

Este aspecto deve verificar os defeitos de assinatura de método.	P	C	I
11.1 O número, a ordem, os tipos e os valores de parâmetros de todas as chamadas dos métodos estão de acordo com as suas declarações de chamdas dos métodos?	s		
11.2 Os valores estão na mesma unidade, por exemplo, jardas?	s		
11.3 Se um objeto ou *array* passado como parâmetro é alterado, a alteração está correta para o método chamador?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

12. Aspecto: Defeito de Comentário

Este aspecto deve verificar os defeitos de comentário.	P	C	I
12.1 Todo método, classe e arquivo têm um comentário de cabeçalho adequado?	s		
12.2 Toda declaração de constante, atributo e variável têm um comentário?	s		
12.3 O comportamento de cada método e classe está comentado em uma linguagem clara e objetiva?	s		
12.4 O comentário de cabeçalho para cada método e classe está consistente com o comportamento do método ou classe?	s		
12.5 Os comentários estão de acordo com o código?	s		
12.6 Os comentários ajudam no entendimento do código?	s		
12.7 Existem suficientes comentários no código?	s		
12.8 Os comentários atendem as diretivas do aplicativo de documentação utilizado?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

13. Aspecto: Defeito de Performance

Este aspecto deve verificar os defeitos de desempenho.	P	C	I
13.1 A organização dos testes lógicos está de forma que os testes mais baratos e freqüentemente bem sucedidos precedem aos mais caros e menos sucedidos?	s		
13.2 Todo resultado que é calculado e armazenado é realmente usado?	s		
13.3 As estruturas de dados usadas podem ser melhoradas ou os algoritmos usados serem mais eficientes?	n		
13.4 O custo de recalcular um valor pode ser reduzido por um único cálculo e armazenamento dos resultados?	n		
13.5 Existe algum cálculo que pode ser retirado de dentro de um *loop* ?	n		
13.6 Existem testes dentro de uma *loop* que não precisam ser feitos?	n		
13.7 Um pequeno *loop* pode ser evitado?	n		
13.8 Existem dois *loops* sobre os mesmos dados que podem ser combinados em um?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

Verificador: _____
Nível de conhecimento do domínio: ____
(0 - baixo / 1 - médio / 2 - alto)
Data da Verificação: _____ **/** _____ **/** _____

Tempo de preparação: _____ **Hr**
Tempo de realização da verificação: _____ **Hr**
Qtde linhas de código não comentário: _____
Assinatura: _____

Checklist: Teste de Unidade

Artefato Verificado: CTTU001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

Os testes de unidade têm como objetivo verificar a lógica e a implementação da menor unidade de implementação. Neste checklist, quanto mais respostas negativas forem dadas às perguntas, mais alto é o risco do teste de unidade para alcançar seu devido propósito.

1. Aspecto: Corretude

Este aspecto deve verificar a adequação dos parâmetros aos atributos dos argumentos e a correta implementação.

		P	C	I
1.1	O número de entrada de parâmetros é igual ao número de argumentos?	s		
1.2	Os atributos do parâmetro são adequados aos atributos do argumento?	s		
1.3	As unidades do parâmetro são iguais às unidades do argumento?	s		
1.4	O número de argumentos que é transmitido para a chamada dos módulos é igual a número de parâmetros?	s		
1.5	Os atributos de argumentos que são transmitidos para a chamada de módulos são iguais aos atributos de parâmetros?	s		
1.6	O sistema de unidade dos argumentos que são transmitidos para a chamada de módulos é igual ao sistema de unidades dos parâmetros?	s		
1.7	O número de atributos e a ordem dos argumentos para as funções embutidas estão corretos?	s		
1.8	Os parâmetros por referência não estão associados aos atuais pontos de entrada?	s		
1.9	Os argumentos de entrada foram alterados?	s		
1.10	As declarações de variáveis globais são consistentes entre os módulos?	s		
1.11	Restrições são passadas como argumentos?	s		
1.12	Quando um módulo executa I/O externo são realizados testes de interface?	s		
1.13	Os atributos de arquivo estão corretos?	s		
1.14	As declarações OPEN/CLOSE estão corretas?	s		
1.15	A especificação de formato está adequado a declaração de I/O?	s		
1.16	O tamanho do buffer está adequado ao tamanho dos registros?	s		
1.17	Os arquivos foram abertos antes de serem usados?	s		
1.18	As condições de fim-de-arquivo são controladas?	s		
1.19	Os erros de I/O são controlados?	s		
1.20	Há ocorrência de erro textual nas informações de saída?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

2. Aspecto: Padrões

Este aspecto deve analisar nomes, tipos e exceções de dados e variáveis.

		P	C	I
2.1	Ocorrência de datilografia imprópria ou incompatível?	n		
2.2	Ocorrência de inicialização errônea ou valores *default?*	n		
2.3	Ocorrência de nomes de variáveis incorretos (mal escritos ou truncados)?	n		
2.4	Ocorrência de tipos de dados incompatíveis?	n		
2.5	Ocorrência de *underflow*, *overflow* e exceções?	n		

Total de itens do aspecto verificados:
Total de não-conformidades:

3. Aspecto: Completude

Este aspecto deve analisar os recursos envolvidos para ajudar na realização do teste.

		P	C	I
3.1	A interface de componente foi testada completamente?	s		
3.2	Os dados locais foram testados nos seus limites?	s		
3.3	Todos os caminhos básicos independentes foram testados?	s		
3.4	Todos os loops foram testados adequadamente?	s		
3.5	Todo o fluxo de dados foi testado?	s		
3.6	Todos os possíveis tipos de erro foram testados?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist : Teste de Integração

Artefato Verificado: CTTI001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

Os testes de integração têm como objetivo verificar a integração entre os componentes. Neste *checklist* , quanto mais respostas negativas forem dadas às perguntas, mais alto é o risco do teste de integração para alcançar seu devido propósito.

		P	C	I
1	A arquitetura de *software* está completamente definida no documento do projeto?	s		
2	A estrutura de dados globais está identificada?	s		
3	O projeto de componente está completo para todos os módulos dentro do sistema?	s		
4	Uma seqüência de integração foi estabelecida?	s		
5	Os *drivers* e *stubs* foram definidos e desenvolvidos?	s		
6	Uma estratégia de integração apropriada foi escolhida baseada nas necessidades do cliente?	s		
7	Se uma estratégia *top-down* foi escolhida, os *stubs* estão disponíveis de forma que os módulos de mais alto nível possam ser testados adequadamente? Ou, se uma estratégia *bottom-up* foi escolhida, os *drivers* estão disponíveis de forma que os *clusters* dos módulos de mais baixo nível possam ser adequadamente testados?	s		
8	O teste de regressão desenvolvido como novos módulos estão integrados?	s		
9	Os componentes foram integrados de forma a demonstrar logo a sua devida funcionalidade?	s		
10	Existem mecanismos de registro e de correção dos erros verificados?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist: Teste de Interface Gráfica

Artefato Verificado: CTTG001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

1. Aspecto: Corretude e Completude
Este aspecto deve verificar a presença de um conjunto de funções e sua adequação às tarefas específicas, obtendo resultados e efeitos corretos.

		P	C	I
1.1	A janela abre corretamente conforme o tipo selecionado ou comandos de menu?	s		
1.2	Todos os dados estão corretamente dentro da janela efetiva com o *mouse* , teclas de função, setas direcionais e teclado disponíveis?	s		
1.3	Todas as funções relacionadas à janela estão operacionais?	s		
1.4	Todos os menus *pull-down* , barras de ferramenta, barras de rolagem, caixas de diálogo e botões, ícones, e outros controles disponíveis estão corretamente exibidos na janela?	s		
1.5	A janela ativa é realçada corretamente?	s		
1.6	Na operação de multi-tarefa, todas as janelas são atualizadas no tempo devido?	s		
1.7	A janela fecha corretamente?	s		
1.8	A barra de menu é exibida adequadamente ao contexto?	s		
1.9	A barra de menu da aplicação retrata corretamente os requisitos relativos ao sistema?	s		
1.10	As operações de *pull-down* são realizadas corretamente?	s		
1.11	Os menus e as barras de ferramenta são realizados corretamente?	s		
1.12	Todas as funções de menu e as subfunções do menu *pull-down* são corretamente listadas?	s		
1.13	Todas as funções de menu são corretamente tratadas pelo *mouse*?	s		
1.14	O tipo, tamanho e formato dos textos apresentados estão corretos?	s		
1.15	As funções de menu realçadas estão coerente com o contexto das operações atuais da janela?	s		
1.16	Cada função do menu é executada conforme a especificação?	s		
1.17	As operações com *mouse* são corretamente reconhecidas pelas interações?	s		
1.18	Se várias seleções são requeridas, elas são corretamente reconhecidas?	s		
1.19	Se o *mouse* tiver vários botões, eles são corretamente reconhecidos?	s		
1.20	O cursor muda corretamente quando outra operação for requerida?	s		
1.21	Os dados de entrada alfanuméricos são corretamente introduzidos no sistema e apresentados?	s		
1.22	Todas as funções relacionadas à janela estão disponibilizadas quando necessárias?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist : Teste de Interface Gráfica

Artefato Verificado: CTTG 001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

2. Aspecto: Padronização
Este aspecto deve verificar as questões de adequação ao padrão estabelecido na especificação da apresentação.

		P	C	I
2.1	Se várias janelas são exibidas, o nome da janela é corretamente apresentado?	s		
2.2	As cores e/ou som dentro da janela ou quando uma seqüência de operações com janelas for apresentada está de acordo com a especificação?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

3. Aspecto: Confiabilidade
Este aspecto deve verificar o nível de desempenho mediante a ocorrência de restrições do *software* ou de defeitos na especificação de requisitos, projeto e implementação.

		P	C	I
3.1	A janela que sofreu atualização é reescrita e apresentada com os novos dados?	s		
3.2	Se seleções indevidas com o *mouse* forem realizadas dentro da janela, estas causam efeitos colaterais inesperados?	n		
3.3	Os dados inválidos são corretamente identificados?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

4. Aspecto: Usabilidade
Este aspecto deve verificar a legibilidade, atratividade e facilidade operativa e de aprendizado do *software* pelo usuário.

		P	C	I
4.1	A janela pode ter tamanho reajustado, ser movida ou rolar?	s		
4.2	Cada item de menu está disponível no Ajuda (*help on-line*)?	s		
4.3	Para cada item do menu está implementado o contexto sensitivo?	s		
4.4	Para cada função do menu, existe um comando alternativo?	s		
4.5	Os nomes das funções de menu são claros e auto explicativos?	s		
4.6	As mensagens de entrada de dados estão legíveis?	s		

Total de itens do aspecto verificados:
Total de não-conformidades:

Checklist: Teste de Validação

Artefato Verificado: CTTI001/03

As questões devem ser respondidas após um breve contato com o documento. A coluna P (Previsão) contém a resposta correta desejada e na coluna C (Confirmação) o revisor deve fornecer a resposta adequada: S (Sim) / N (Não) / NA (Não se aplica). Caso a resposta da Coluna C diferencie da coluna P, exceto quando não se aplica (NA), a coluna I (Identificação) deve ser preenchida com número seqüencial para identificação do erro detectado. Todos os erros detectados devem ser identificados e comentados na parte de Não-Conformidades.

Os testes de validação têm como objetivo verificar os requisitos do *software*. Neste *checklist*, quanto mais respostas negativas forem dadas às perguntas, mais alto é o risco do teste de validação para alcançar seu devido propósito.

		P	C	I
1	Uma especificação completa dos requisitos de *software* está disponível?	s		
2	Existe limitação para os requisitos?	s		
3	Classes equivalentes foram definidas para testar entradas?	s		
4	Foram realizados testes de limite para verificar os limites do *software*?	s		
5	Os *scripts* de teste foram desenvolvidos para validar cada função do *software*?	s		
6	Os *scripts* de teste foram desenvolvidos para validar todas as estruturas de dados?	s		
7	Os *scripts* de teste foram desenvolvidos para avaliar o desempenho do *software*?	s		
8	Os *scripts* de teste foram desenvolvidos para testar o comportamento do *software*?	s		
9	Os *scripts* de teste foram desenvolvidos para exercitar completamente a interface do usuário?	s		
10	Os *scripts* de teste foram desenvolvidos para verificar todo o erro controlado?	s		
11	Os casos de uso disponibilizados são para realizar cenários de teste?	s		
12	Testes estatísticos são utilizados como um elemento de validação?	s		
13	Foram desenvolvidos testes para verificar os procedimentos definidos no Manual do Usuário e no Manual de Instalação do software?	s		
14	Existem mecanismos de registro e de correção dos erros verificados?	s		
15	Uma relação de erros detectados foi produzida?	s		

Total de itens do aspecto verificados:	
Total de não-conformidades:	

7.4.2 ARTEFATOS: TÉCNICAS DE LEITURA HORIZONTAL

Garantia da Qualidade de *Software*
Leitura Horizontal 1 Versão:1.2
RELH101/03 Data da versão: 03/03/2004

TLH 1 – Modelo de Contexto do Domínio X Modelo Conceitual do Domínio

Objetivo: Verificar se um modelo conceitual de domínio representado por um diagrama de classes da UML captura todos os conceitos e identifica todos os relacionamentos especificados no Modelo de Contexto do Domínio, mantendo o sentido semântico dos mesmos. O modelo conceitual não é um modelo de *software*, mas um modelo de informação inerente ao domínio do problema.
Pré-requisito: Nenhum.
Entrada para o processo:

1. Modelo de Contexto do Domínio representado pelo diagrama de contexto baseado no método FORM e sua descrição textual.
2. Modelo Conceitual do Domínio representado pelo diagrama de classes.

Procedimentos:

I. Leia a descrição do Modelo de Contexto do Domínio e analise o diagrama de contexto para entender o escopo do domínio do problema e as entidades externas, e identificar seus relacionamentos.

ENTRADAS: Modelo de Contexto do Domínio
Descrição textual do Modelo de Contexto do Domínio
Modelo Conceitual do Domínio

SAÍDAS: Relatório de Discrepâncias

Para cada funcionalidade ou restrição na descrição textual do Modelo de Contexto do Domínio execute os seguintes passos:

A. Encontre a classe correspondente. Marque com um asterisco azul a classe correspondente no diagrama de classes quando encontrar.
   1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois uma funcionalidade não está presente no modelo de domínio.

B. Verifique se o nome da classe é inerente ao domínio do problema e se ela está utilizando o nível adequado de abstração.
   1 - Baseado no conhecimento fornecido pela descrição do Modelo de Contexto do Domínio certifique se pode entender o propósito desta classe. Caso contrário, preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.
   2 - Certifique que o nome da classe inicie sempre por letra maiúscula. Caso contrário, preencha a Tabela de Discrepância, pois o nome da classe está fora do padrão.

C. Verifique se todos os possíveis atributos explicitamente descritos nas funcionalidades estejam descritos na classe.
   1 - Certifique que o mesmo conjunto de atributos esteja presente em ambos os documentos. Caso contrário, preencha a Tabela de Discrepância, pois informação está presente em um documento, mas ausente no outro.
   2 - Certifique que os nomes dos atributos nas classes iniciem sempre por letra minúscula. Caso contrário, preencha a Tabela de Discrepância, pois o nome da classe está fora do padrão.

D. Verifique se existi associação da classe que represente a funcionalidade ou restrição considerada. Realce em amarelo a associação correspondente.
   1 - Certifique que o mesmo conjunto de associação e restrições esteja presente em ambos documentos. Caso contrário, há informação em um documento que não está presente em outro, ou há inconsistente entre os dois.

E. Se o diagrama de classe do modelo conceitual de domínio especifica algum mecanismo de herança para esta classe, verifique se eles estão corretamente descritos.
   1 - Certifique que o relacionamento de herança esteja incluído na descrição do Modelo de Contexto do Domínio. Caso contrário, preencha a Tabela de Discrepância, pois informação no Modelo Conceitual de Domínio não está na descrição textual do Modelo de Contexto do Domínio.

Restrito Instituto Militar de Engenharia Página: 1/2

1 - Utilize a hierarquia de classes para encontrar os pais desta classe. Verifique semanticamente se o nome da classe é do tipo da classe pai e se faz sentido ter esta classe neste ponto de hierarquia. Caso contrário, a hierarquia não deveria ser definida desta maneira, então preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.

B. Verifique se todos os relacionamentos das classes (associação, agregação e composição) estejam corretamente descritos com respeito às indicações de multiplicidade.

1 - Para cada relacionamento certifique semanticamente que o relacionamento faz sentido dado o papel e os objetos relacionados. Caso contrário, o relacionamento não deveria ser definido desta maneira, então preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.

II. Reveja o Modelo Conceitual do Domínio para assegurar que todas as classes e suas associações estão sendo levadas em consideração.

ENTRADAS: Classes marcadas com asterisco azul
Associações realçadas em amarelo

SAÍDAS: Relatório de Discrepâncias

A. Reveja as classes e seus relacionamentos para ter certeza que todo o conjunto de classe e seus relacionamentos aparecem na descrição textual do Modelo de Contexto do Domínio.

1 - Certifique que não exista nenhuma classe sem asterisco e relacionamento sem estar realçado em amarelo. Caso exista algum, preencha a Tabela de Discrepância, pois uma classe ou relacionamento no Modelo Conceitual do Domínio não está presente no Modelo de Contexto do Domínio.

TLH 2 – Modelo Conceitual do Domínio X Modelo de Casos de Uso do Domínio e Cenários

Objetivo: Verificar se um modelo de interação de usuário com o sistema (Modelo de Casos de Uso) juntamente com a descrição dos cenários projetam todas as necessidades do sistema em termos de interações que devem ocorrer através dos limites do sistema.
Pré-requisito: Nenhum.
Entrada para o processo:

1. Modelo Conceitual do Domínio representado pelo diagrama de classes
2. Modelo de Casos de Uso do Domínio
3. Descrição dos Cenários dos Casos de Uso do Domínio

Procedimentos:

I. Analise o Modelo Conceitual do Domínio para entender as classes conceituais e seus relacionamentos.

ENTRADAS: Modelo Conceitual do Domínio
Modelo de Casos de Uso do Domínio
Descrição dos Cenários dos Casos de Uso do Domínio

SAÍDAS: Relatório de Discrepâncias

Para cada classe do diagrama do Modelo Conceitual do Domínio execute os seguintes passos:

A. Encontre o objeto correspondente, ator ou caso de uso. Marque com um asterisco azul o objeto correspondente no Modelo de Casos de Uso do Domínio quando encontrar. Se for necessário, consulte na Descrição dos Cenários dos Casos de Uso do Domínio para ajudar a encontrar o caso de uso correspondente.
   1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois uma classe conceitual não está presente no Modelo de Casos de Uso e Cenários.

B. Verifique se o nome da classe conceitual é o mesmo utilizado nos objetos ator ou caso de uso, assim como na descrição dos cenários correspondentes.
   1 - Caso não tenha o mesmo nome, preencha a Tabela de Discrepância, pois a descrição pode estar ambígua, relatando a necessidade de conhecimento adicional para compreensão.

C. Se o objeto correspondente não for ator, então verifique se todos os atributos da classe conceitual estão sendo referenciados na descrição do cenário correspondente nos cursos normal ou alternativo.
   1 - Certifique que o mesmo conjunto de atributos esteja presente em ambos os documentos. Caso contrário, preencha a Tabela de Discrepância, pois informação está presente em um documento, mas ausente no outro.

D. Se o objeto correspondente não for ator, verifique se os relacionamentos da classe conceitual são capturados pelo caso de uso correspondente. Se o objeto correspondente for ator, verifique se o ator está interagindo com caso de uso correspondente a classe conceitual. Realce em amarelo a iteração ou associação de extensão ou inclusão correspondente.
   1 - Certifique que o mesmo relacionamento e iteração estejam presentes em ambos documentos. Caso contrário, há informação em um documento que não está presente em outro, ou há inconsistente entre os dois.

E. Se o Modelo Conceitual de Domínio especifica algum mecanismo de herança para esta classe, verifique se eles estão corretamente representados no Modelo de Casos de Uso.
   1 - Caso não encontre, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistente entre os dois.

A. Verifique se o objeto correspondente à classe conceitual está utilizando o nível adequado de abstração
   1 - Certifique que pode entender o propósito desta classe. Caso contrário, preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.

B. Verifique se todos os relacionamentos e restrições das classes conceituais estejam corretamente capturados nas descrições dos cenários correspondentes.
   1 - Certifique semanticamente que os relacionamentos fazem sentido pelo papel e objetos relacionados. Caso contrário, os relacionamentos não deveriam ser definidos desta maneira, então preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.

II. Reveja o Modelo de Casos de Uso do Domínio para assegurar que todos os casos de uso e atores estão sendo levados em consideração.

ENTRADAS: Casos de uso marcados com asterisco azul
Iterações e associações de extensão ou inclusão realçadas em amarelo

SAÍDAS: Relatório de Discrepâncias

A. Reveja os casos de uso e suas iterações e associações para ter certeza que todo o conjunto de caso de uso e suas iterações e associações aparecem representados no Modelo Conceitual de Domínio.
   1 - Certifique que não exista nenhum caso de uso sem asterisco, e iteração e associação sem estar realçado em amarelo. Caso exista algum, preencha a Tabela de Discrepância, pois um caso de uso ou iteração ou associação no Modelo de Casos de Uso do Domínio não está presente no Modelo Conceitual do Domínio.

TLH 3 – Modelo Tipo do Negócio X Interface da Camada de Negócio

Objetivo: Verificar se a Interface da Camada de Negócio contém todos os tipos de interface do negócio, seus atributos com os tipos básicos, suas regras de associação, suas assinaturas e *datas types* inerentes à modelagem do negócio representada pelo Modelo Tipo do Negócio.

Entrada para o processo:

1. Modelo Tipo do Negócio representado pelo diagrama de classes.
2. Interface da Camada de Negócio representado pelo diagrama de classes.

Procedimentos:

I. Analise o Modelo Tipo do Negócio para entender as classes de negócio e seus relacionamentos.

ENTRADAS: Modelo Tipo do Negócio
Interface da Camada de Negócio

SAÍDAS: Relatório de Discrepâncias

Para cada classe do diagrama do Modelo Tipo do Negócio execute os seguintes passos:

A. Encontre a classe de negócio correspondente na Interface da Camada de Negócio e marque com um asterisco azul esta classe.
   1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois uma classe de negócio não está presente na Interface da Camada de Negócio.
   2 - Certifique que o nome da classe de negócio é o mesmo utilizado na classe correspondente da Interface da Camada de Negócio. Caso não tenha o mesmo nome, preencha a Tabela de Discrepância, pois pode haver ambigüidade, relatando a necessidade de conhecimento adicional para compreensão.
   3 - Certifique que o estereótipo da classe esteja referenciado corretamente como “core”. Caso contrário, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistência entre os dois.

B. Verifique se no Modelo Tipo do Negócio especifica algum mecanismo de herança para esta classe e verifique ainda se ele está corretamente representado na Interface da Camada de Negócio.
   1 - Caso contrário, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistência entre os dois.

C. Verifique para cada atributo se existe o mesmo atributo especificado com seu tipo básico na classe correspondente. Sublinhe em azul o atributo correspondente na classe da Interface da Camada de Negócio.
   1 - Caso não consiga encontrar, verifique se existe um *data type* correspondente ao atributo. Caso encontre, marque com asterisco azul a classe *data type*, e sublinhe em azul os atributos correspondentes com o mesmo tipo básico da classe de negócio em questão na classe *data type* e também sublinhe em azul o atributo correspondente ao *data type* na classe “core” correspondente. Caso não exista, preencha a Tabela de Discrepância, pois um atributo da classe de negócio não está presente na classe correspondente da Interface da Camada de Negócio.
   2 - Certifique que o nome do atributo utilizado na classe correspondente da Interface da Camada de Negócio é o mesmo da classe de negócio. Caso não tenha o mesmo nome, preencha a Tabela de Discrepância, pois a descrição pode estar ambígua, relatando a necessidade de conhecimento adicional para compreensão.

D. Para cada relacionamento e restrição da classe de negócio verifique na classe correspondente da Interface da Camada de Negócio o relacionamento e restrição equivalente. Realce em amarelo o relacionamento equivalente na classe correspondente da Interface da Camada de Negócio.
   1 - Caso contrário, os relacionamentos ou restrições não deveriam ser definidos desta maneira, então preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.

A. Para cada relacionamento da classe de negócio verifique se a navegabilidade está correta na classe correspondente da Interface da Camada de Negócio.
   1 - Caso contrário, preencha a Tabela de Discrepância, pois a navegabilidade pode estar ambígua, relatando a necessidade de conhecimento adicional para compreensão.

B. Verifique se existe uma classe de interface, cujo estereótipo "interface", que tenha um relacionamento de composição com a classe de negócio correspondente. Marque com asterisco verde a classe de interface. Caso contrário, preencha a Tabela de Discrepância, pois uma interface não está presente na Interface da Camada de Negócio.
   1 - Caso contrário, preencha a Tabela de Discrepância, pois um método de uma interface não está presente na Interface da Camada de Negócio.
   2 - Certifique que o nome da classe de interface do negócio inicia pela vogal em maiúscula "I" concatenado com o nome da classe do negócio. Caso não tenha o padrão, preencha a Tabela de Discrepância, pois pode haver falta de padrão ou ambigüidade, relatando a necessidade de conhecimento adicional para compreensão.

C. Para cada atributo da classe de negócio verifique se na classe de interface existe um método público correspondente para acessá-lo (leitura e escrita).
   1 - Caso contrário, preencha a Tabela de Discrepância, pois um método de uma interface não está presente na Interface da Camada de Negócio.

II. Reveja a Interface da Camada de Negócio para assegurar que todas as classes e seus relacionamentos e atributos estão sendo levados em consideração.

ENTRADAS: Interface da Camada de Negócio marcados com asterisco azul e verde, sublinhado em azul e realçado em amarelo

SAÍDAS: Relatório de Discrepâncias

A. Reveja as classes, seus atributos e relacionamentos para ter certeza que todas as classes de negócio estão corretamente capturadas na classe correspondente da Interface da Camada de Negócio.
   1 - Certifique que não existe nenhuma classe do tipo "core" sem asterisco azul. Caso exista alguma, preencha a Tabela de Discrepância, pois uma classe na Interface da Camada de Negócio não está presente no Modelo Tipo do Negócio.
   2 - Certifique que não exista nenhuma classe do tipo "interface" sem asterisco verde. Caso exista alguma, preencha a Tabela de Discrepância, pois uma classe de interface não deveria ser definida desta maneira, relatando a necessidade de conhecimento adicional para compreensão.
   3 - Certifique que não existe nenhum *data type* sem asterisco azul e nenhum atributo sem estar sublinhado em azul. Caso exista algum *data type* ou atributo, preencha a Tabela de Discrepância, pois um *data type* na Interface da Camada de Negócio não está sendo referenciado, relatando que um atributo de classe da Interface da Camada de Negócio não está presente na classe correspondente do Modelo Tipo de Negócio.
   4 - Certifique que não exista nenhum relacionamento sem estar realçado em amarelo. Caso exista algum, preencha a Tabela de Discrepância, pois um relacionamento na Interface da Camada de Negócio não está presente no Modelo Tipo do Negócio.

TLH 4 – Interfaces da Camada de Negócio e de Aplicação X Diagrama de Interações

Objetivo: Verificar se o diagrama de interações descreve a dinâmica dos serviços das classes de interfaces da Camada de Negócio e da Camada de Aplicação de forma que os comportamentos especificados no Diagrama de Interação estão capturados corretamente.

Entrada para o processo:

1. Interfaces da Camada de Aplicação representada por diagrama de classes.
2. Interfaces da Camada de Negócio representada por diagrama de classes.
3. Diagrama de Interações representado por diagrama de seqüência.

Procedimentos:

I. Leia o Diagrama de Interações para entender que serviços estão descritos e como estes deveriam ser implementados.

ENTRADAS: Diagrama de Interações
SAÍDAS: Relatório de Discrepâncias

Para cada diagrama de seqüência, execute os seguintes passos:

A. Encontre os atores, objetos e classes, e realce-os de azul.

B. Realce de verde a informação trocada entre os objetos (as setas horizontais). Considere se esta informação representa mensagens ou serviços. Se a informação trocada está muito detalhada, para um nível de mensagens, devem-se abstrair várias mensagens juntas para entender que serviços elas estão fornecendo em conjunto. Anote no diagrama de seqüência descrevendo estes serviços e realce-os de verde.

C. Circule de amarelo quaisquer restrições nas mensagens e serviços, tais como, restrições no número de classes/objetos que uma mensagem poderia enviar, restrições nos valores globais de um atributo, dependências entre dados, ou restrições de tempo que podem afetar o estado de um objeto. Também circule de amarelo quaisquer condições que determinam sob que circunstâncias uma mensagem pode ser enviada.

II. Identifique e inspecione os diagramas de classes (Interfaces da Camada de Negócio e de Aplicação) relacionados para identificar se os objetos correspondentes estão precisamente descritos.

ENTRADAS: Interfaces da Camada de Negócio
Interfaces da Camada de Aplicação
Diagrama de Interações com objetos, serviços e restrições marcados

SAÍDAS: Relatório de Discrepâncias

A. Verifique que todo objeto, classe e ator usado no diagrama de seqüência estão representados por uma classe concreta nos diagramas de classes. Para as classes e atores, encontre o nome nos diagramas de classes. Para os objetos, encontre a nome da classe da qual o objeto foi instanciado. Verifique as seguintes discrepâncias e, caso encontre-as, preencha a Tabela de Discrepância:

1 - Se uma classe ou objeto não pode ser encontrada nos diagramas de classes, isto significa que a informação está inconsistente entre os diagramas de classes e o diagrama de seqüência, está presente em um e ausente em outro.

2 - Se um ator não pode ser encontrado, determine se o ator precisa ser apresentado como uma classe para executar algum comportamento necessário. Se sim, então informação que está presente no diagrama de seqüência está faltando nos diagramas de classes.

B. Verifique se para todo serviço ou mensagem marcado em verde no diagrama de seqüência, existe um comportamento correspondente no diagrama de classes. Verifique se existem comportamentos de classes nos diagramas de classes que encapsulam os serviços de mais alto nível fornecidos pelo diagrama de seqüência. Ou seja, esteja certo que a classe ou objeto que recebe a mensagem no diagrama de seqüência, ou que deveria ser responsável pelo serviço, possui um comportamento associado nos diagramas de classes. Também esteja certo que existe algum tipo de associação (nos diagramas de classes) entre as classes que a mensagem conecta (no diagrama de seqüência). Lembre que em ambos os casos, pode-se precisar procurar na árvore de herança a que classe pertence para encontrar as características necessárias.

Verifique ainda, que para cada serviço, as mensagens descritas pelo diagrama de seqüência são suficientes para executar aquele serviço. Verifique as seguintes discrepâncias e, caso encontre-as, preencha a Tabela de Discrepância:

1 - Esteja certo que para cada mensagem no diagrama de seqüência a classe recebedora contém um comportamento apropriado no diagrama de classe correspondente. Caso contrário isto significa que existe uma inconsistência entre os diagramas. Um comportamento está presente no diagrama de seqüência , mas faltando no diagrama de classes.
2 - Esteja certo que existem comportamentos apropriados aos serviços. Se não, existe um serviço presente no diagrama de seqüência que não está representado no diagrama de classe correspondente.
3 - Esteja certo que existe uma associação no diagrama de classes correspondente entre as duas classes as quais trocam mensagens. Caso contrário, uma associação está presente no diagrama de seqüência por causa da troca de mensagem, mas não está presente no diagrama de classes.
4 - Esteja certo que não estão faltando comportamento, os quais poderiam evitar que algum serviço seja executado. Se existem, isto significa que algo está faltando no diagrama de seqüência.

B. Verifique que as restrições identificadas no diagrama de seqüência podem ser atendidas no diagrama de classes correspondente. Verifique as seguintes discrepâncias e preencha a Tabela de discrepância, se quaisquer das declarações seguintes não são verdadeiras, ou seja, informação no diagrama de seqüência não foi representada nos diagramas de classes:

1 - Se o diagrama de seqüência descreve restrições no número de objetos que podem receber uma mensagem, esteja certo que a restrição aparece como uma informação de cardinalidade na associação apropriada do diagrama de classe correspondente.
2 - Se o diagrama de seqüência especifica uma faixa de valores permitidos para os dados, esteja certo que uma restrição aparece como uma faixa de valores no atributo do diagrama de classes correspondente.
3 - Se o diagrama de seqüência contém informação relacionada às dependências entre os dados ou objetos, esteja certa que esta informação está incluída no diagrama de classes correspondente, via uma restrição na classe ou restrições de cardinalidade no relacionamento ou pela relação no diagrama de classes.
4 - Se o diagrama de seqüência contém restrições de tempo que poderiam afetar o estado de um objeto, esteja certa que esta informação está incluída como uma restrição em uma classe ou relação do diagrama de classes correspondente.

C. Para cada classe, mensagem e dado identificado acima, analise com base em experiências prévias se este Diagrama de Interações é viável. Por exemplo, se os atributos de qualidade do projeto como coesão (todos os comportamentos e atributos de uma classe realmente pertencem a esta classe), acoplamento (os relacionamentos entre as classes são apropriados). Verifique as seguintes discrepâncias e, caso encontre-as, preencha a Tabela de Discrepância:

1 - Esteja certo que é lógico para a classe receber esta mensagem com estes dados.
2 - Esteja certo que se pode verificar que as restrições são viáveis.
3 - Esteja certo que todos os atributos necessários estão definidos. Se não, os diagramas podem conter fatos incorretos.
4 - Para as classes especificadas no diagrama de seqüência, esteja certo que os comportamentos e atributos especificados para ela no diagrama de classes correspondente fazem sentido.
5 - Esteja certo que o nome da classe é apropriado para o domínio, e para seus atributos e comportamentos.
6 - Esteja certo que os relacionamentos com outras classes são apropriados.
7 - Esteja certo que os relacionamentos são do tipo correto, ou seja, um relacionamento de composição é utilizado ao invés de uma associação. Se não, encontrou um fato incorreto que contradiz o conhecimento do domínio.

TLH 4 – Interfaces da Camada de Negócio e de Aplicação X Diagrama de Interações

Objetivo: Verificar se o diagrama de interações descreve a dinâmica dos serviços das classes de interfaces da Camada de Negócio e da Camada de Aplicação de forma que os comportamentos especificados no Diagrama de Interação estão capturados corretamente.

Entrada para o processo:

1. Interfaces da Camada de Aplicação representada por diagrama de classes.
2. Interfaces da Camada de Negócio representada por diagrama de classes.
3. Diagrama de Interações representado por diagrama de seqüência.

Procedimentos:

I. Leia o Diagrama de Interações para entender que serviços estão descritos e como estes deveriam ser implementados.

ENTRADAS: Diagrama de Interações
SAÍDAS: Relatório de Discrepâncias
Para cada diagrama de seqüência, execute os seguintes passos:

A. Encontre os atores, objetos e classes, e realce-os de azul.
B. Realce de verde a informação trocada entre os objetos (as setas horizontais). Considere se esta informação representa mensagens ou serviços. Se a informação trocada está muito detalhada, para um nível de mensagens, devem-se abstrair várias mensagens juntas para entender que serviços elas estão fornecendo em conjunto. Anote no diagrama de seqüência descrevendo estes serviços e realce-os de verde.
C. Circule de amarelo quaisquer restrições nas mensagens e serviços, tais como, restrições no número de classes/objetos que uma mensagem poderia enviar, restrições nos valores globais de um atributo, dependências entre dados, ou restrições de tempo que podem afetar o estado de um objeto. Também circule de amarelo quaisquer condições que determinam sob que circunstâncias uma mensagem pode ser enviada.

II. Identifique e inspecione os diagramas de classes (Interfaces da Camada de Negócio e de Aplicação) relacionados para identificar se os objetos correspondentes estão precisamente descritos.

ENTRADAS: Interfaces da Camada de Negócio
Interfaces da Camada de Aplicação
Diagrama de Interações com objetos, serviços e restrições marcados
SAÍDAS: Relatório de Discrepâncias

A. Verifique que todo objeto, classe e ator usado no diagrama de seqüência estão representados por uma classe concreta nos diagramas de classes. Para as classes e atores, encontre o nome nos diagramas de classes. Para os objetos, encontre a nome da classe da qual o objeto foi instanciado. Verifique as seguintes discrepâncias e, caso encontre-as, preencha a Tabela de Discrepância:
   1 - Se uma classe ou objeto não pode ser encontrada nos diagramas de classes, isto significa que a informação está inconsistente entre os diagramas de classes e o diagrama de seqüência, está presente em um e ausente em outro.
   2 - Se um ator não pode ser encontrado, determine se o ator precisa ser apresentado como uma classe para executar algum comportamento necessário. Se sim, então informação que está presente no diagrama de seqüência está faltando nos diagramas de classes.
B. Verifique se para todo serviço ou mensagem marcado em verde no diagrama de seqüência, existe um comportamento correspondente no diagrama de classes. Verifique se existem comportamentos de classes nos diagramas de classes que encapsulam os serviços de mais alto nível fornecidos pelo diagrama de seqüência. Ou seja, esteja certo que a classe ou objeto que recebe a mensagem no diagrama de seqüência, ou que deveria ser responsável pelo serviço, possui um comportamento associado nos diagramas de classes. Também esteja certo que existe algum tipo de associação (nos diagramas de classes) entre as classes que a mensagem conecta (no diagrama de seqüência). Lembre que em ambos os casos, pode-se precisar procurar na árvore de herança a que classe pertence para encontrar as características necessárias.

A. Encontre o componente correspondente na Arquitetura de Domínio e marque o com asterisco verde. A especificação do componente deve estar representada por uma classe com estereótipo "comp esp" com notação *lollipop* para interface do componente. O nome da interface do componente deve ser o mesmo nome da classe de interface de sistema da especificação das Interfaces da Camada de Aplicação. O nome da classe "comp esp" do componente deve iniciar por letra em maiúscula e ter o mesmo nome da classe de interface sem a vogal maiúscula inicial "I".
   1 - Caso não consiga encontrar o componente correspondente, preencha a Tabela de Discrepância, pois há informação em um diagrama que não está presente em outro, ou há inconsistência entre os dois.
   2 - Caso consiga encontrar o componente corresponde verifique se o componente está especificado na Camada de Aplicação da Arquitetura de Domínio. Caso não esteja na Camada de Aplicação, preencha a Tabela de Discrepância, pois há inconsistência entre os dois diagrama.

B. Encontre no Diagrama de Interações de Componente o diagrama correspondente ao componente identificado no passo anterior 1. O nome do componente deve ser similar ou sinônimo do nome da interação correspondente.
   1 - Caso não consiga encontrar o diagrama de interação correspondente, preencha a Tabela de Discrepância, pois há informação em um diagrama que não está presente em outro, ou há inconsistência entre os dois.
   2 - Caso consiga encontrar o diagrama de interação correspondente, verifique se as interações entre as classes deste diagrama estão sendo representadas por linhas pontilhadas como dependências entre componentes. Caso contrário, preencha a Tabela de Discrepância, pois há informação em um diagrama que não está presente em outro, ou há inconsistência entre os dois.

III. Reveja a Arquitetura de Domínio com marcação de asterisco azul e verde e realce amarelo para assegurar que todos os componentes e suas dependências estão sendo levados em consideração.

ENTRADAS: Diagrama das Interações dos Componentes marcado com asterisco azul e verde, e dependências realçadas em amarelo

SAÍDAS: Relatório de Discrepâncias

C. Reveja na Camada de Negócio da Arquitetura de Domínio se todos componentes de negócio estão especificados.
   1 - Certifique que não existe nenhuma especificação de componentes sem asterisco azul. Caso exista algum, preencha a Tabela de Discrepância, pois um serviço de interface não deveria ser definido desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

D. Reveja na Camada de Aplicação da Arquitetura de Domínio se todos componentes da aplicação estão especificados.
   1 - Certifique que não existe nenhuma especificação de componentes sem asterisco verde. Caso exista algum, preencha a Tabela de Discrepância, pois um serviço de interface não deveria ser definido desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

E. Reveja na Arquitetura de Domínio cada componente para ter certeza que todas as suas dependências aparecem.
   1 - Certifique que não existe nenhuma interação sem estar realçada em amarelo. Caso exista alguma, preencha a Tabela de Discrepância, pois uma dependência não deveria ser definida desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

7.4.3 ARTEFATOS: TÉCNICAS DE LEITURA VERTICAL

Garantia da Qualidade de *Software*
Leitura Vertical 1 Versão:1.2
RELV101/03 Data da versão: 03/03/2004

TLV 1 - Necessidades do Cliente e Glossário X Modelo de Contexto

Objetivo: Verificar se um Modelo de Contexto do Domínio e sua descrição estão capturando todos os relacionamentos entre o domínio do problema com os outros domínios externos, de forma a estabelecer as entidades externas, suas interfaces de aplicação e requisitos de dados, o ambiente de operação e as restrições ao domínio do problema.
Pré-requisito: Esta leitura requer que o documento Necessidades do Cliente tenha sido aprovado pelo Processo Controle de Qualidade.
Entrada para o processo:

1. Descrição das Necessidades do Cliente e Glossário com as descrições detalhadas das funcionalidades.
2. Modelo de Contexto do Domínio representado pelo diagrama de contexto baseado no método FORM.
3. Descrição textual do Modelo de Contexto do Domínio.

Procedimentos:

I. Leia o documento Necessidades do Cliente e Glossário para entender o contexto do problema através das funcionalidades e características desejadas do cliente.
ENTRADAS: Necessidades do Cliente e Glossário
Modelo de Contexto do Domínio
Descrição textual do Modelo de Contexto do Domínio
SAÍDAS: Relatório de Discrepâncias
Para cada funcionalidade ou restrição em Necessidades do Cliente execute os seguintes passos:
A. Encontre a entidade externa que se deve relacionar e o fluxo de dados do domínio com esta entidade. Marque com um asterisco azul a entidade externa correspondente no diagrama de contexto quando encontrar.
1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois uma entidade externa não está presente no modelo de contexto.
2 - Caso encontre, baseado no conhecimento dos domínios existentes para reúso, verifique se o nome da entidade externa é inerente ao domínio externo correspondente e se ela está utilizando o nível adequado de abstração. Caso contrário, preencha a Tabela de Discrepância, nome da entidade externa não está adequado.
3 - Caso encontre, verifique se todos os fluxos de dados do domínio com esta entidade externa estão capturados. Realce em amarelo os fluxos de dados e certifique que esta entidade externa com os fluxos de dados esteja devidamente documentada na descrição do Modelo de Contexto do Domínio. Caso contrário, preencha a Tabela de Discrepância, falta ou está incompleta fluxo de dados no modelo de contexto ou sua descrição.
4 - Para cada fluxo de dado identificado, verificar se o nome está no padrão (todo em minúsculo) e é o mesmo utilizado pela entidade externa. Caso contrário, preencha a Tabela de Discrepância, nome do fluxo de dados não está adequado.
B. Verifique se todas as necessidades estão sendo encapsuladas dentro do escopo do domínio.
1 - Certifique que o mesmo conjunto de necessidades esteja presente na descrição do contexto do domínio. Caso contrário, preencha a Tabela de Discrepância, pois informação está presente em um documento, mas ausente no outro.

II. Reveja o Modelo Conceitual do Domínio para assegurar que todas as entidades externas e seus relacionamentos estão sendo levados em consideração.
ENTRADAS: Entidades externas marcadas com asterisco azul
Relacionamentos realçados em amarelo
SAÍDAS: Relatório de Discrepâncias
A. Reveja as entidades externas e seus relacionamentos para ter certeza que todo o conjunto de entidades e seus relacionamentos aparecem descritos no documento de Necessidades do Cliente e Glossário.
1 - Certifique que não exista nenhuma entidade externa sem asterisco e relacionamento sem estar realçado em amarelo. Caso exista algum, preencha a Tabela de Discrepância, pois uma entidade externa ou relacionamento no Modelo de Contexto do Domínio não está presente no documento Necessidades do Cliente e Glossário.

Restrito Instituto Militar de Engenharia Página: 1/1

TLV 2 – Necessidades do Cliente e Glossário X Modelo de Características

Objetivo: Verificar se todas as características explícitas comuns e variáveis do domínio do problema através das funcionalidades e características desejadas estão capturadas no Modelo de Características baseado no método FORM para estabelecer as possibilidades de seleção destas características mediante uma tomada de decisão. Este modelo deve estar representado em quatro camadas: de capacidades (identifica serviços, operações, funções, apresentações ou performances de uma aplicação de um dado domínio que pode ser funcional ou não-funcional), ambiente de operação (identifica ambiente operacional, plataforma operativa, sistema de rede de comunicação, banco de dados), tecnologia de domínio (detalha a implementação de mais baixo nível que mapeia um conjunto específico de técnica de implementação) e técnica de implementação (identifica algoritmos, tipos abstratos de dados).

Pré-requisito: Esta leitura requer que o documento Necessidades do Cliente tenha sido aprovado pelo Processo Controle de Qualidade.

Entrada para o processo:

1. Descrição das Necessidades do Cliente e Glossário.
2. Modelo de Características representado pelo diagrama de características em camada de categorias baseado no método FORM.

Procedimentos:

I. Leia o documento Necessidades do Cliente e Glossário para entender as funcionalidades e características desejadas do cliente.

ENTRADAS: Necessidades do Cliente e Glossário
Modelo Conceitual do Domínio

SAÍDAS: Relatório de Discrepâncias

Para cada funcionalidade e característica desejada em Necessidades do Cliente execute os seguintes passos:

A. Identifique a característica correspondente e em que categoria ela pertence: capacidade, ambiente de operação, tecnologia de domínio e técnica de implementação.

B. Encontre a característica correspondente e marque-a com um asterisco azul no Modelo de Característica.

1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois uma funcionalidade ou característica desejada não está presente no Modelo de Características.

2 - Caso encontre, verifique se a característica está na camada adequada conforme o método FORM e se segue o padrão de representação do método FORM. Caso contrário, preencha a Tabela de Discrepância, pois uma característica está fora do padrão do método FORM.

C. Verifique se o nome da característica é inerente ao domínio do problema e se ela está utilizando o nível adequado de abstração.

1 - Baseado no conhecimento fornecido pelo documento Necessidades do Cliente e Glossário, certifique se pode entender o propósito desta característica. Caso contrário, preencha a Tabela de Discrepância relatando a necessidade de conhecimento adicional para compreensão.

TLV 3 – Modelo Conceitual do Domínio X Modelo Tipo do Negócio

Objetivo: Verificar se um modelo de especificação (Modelo Tipo do Negócio) representado pelo diagrama de classes da UML modela efetivamente toda a informação pertinente ao escopo do negócio.
Pré-requisito: Esta leitura requer que o diagrama do Modelo Conceitual de Domínio e Modelo de Casos de Uso do Domínio tenham sido aprovado pelo Processo Controle de Qualidade.
Entrada para o processo:

1. Modelo Conceitual do Domínio representado pelo diagrama de classes.
2. Modelo de Casos de Uso do Domínio.
3. Descrição dos Cenários dos Casos de Uso do Domínio.
4. Modelo Tipo do Negócio representado pelo diagrama de classes.

Procedimentos:

I. Analise o Modelo Conceitual do Domínio para entender as classes conceituais e seus relacionamentos.
ENTRADAS: Modelo Conceitual do Domínio
Modelo de Casos de Uso do Domínio
Modelo Tipo do Negócio
SAÍDAS: Relatório de Discrepâncias
Para cada classe do diagrama do Modelo Conceitual do Domínio execute os seguintes passos:
A. Para cada relacionamento da classe tente encontrar uma associação, agregação ou composição no Modelo Tipo do Negócio. Realce em amarelo o relacionamento correspondente no Modelo Tipo do Negócio quando encontrar.
1 - Caso não consiga encontrar, certifique no Modelo de Casos de Uso do Domínio que o relacionamento pode ser proveniente de uma classe do Modelo Conceitual do Domínio que representa o conceito de uma interação de ator / sistema com o domínio do negócio. Caso contrário, preencha a Tabela de Discrepância, pois um relacionamento não está presente no Modelo Tipo do Negócio.
2 - Certifique que a cardinalidade esteja representada no relacionamento no Modelo Tipo do Negócio de acordo com a representação correspondente no Modelo Conceitual do Domínio.
B. Se o Modelo Conceitual de Domínio especifica algum mecanismo de herança para esta classe, verifique se eles estão corretamente representados no Modelo Tipo do Negócio.
1 - Caso não consiga encontrar, certifique no Modelo de Casos de Uso do Domínio que a classe pode ser proveniente de uma classe do Modelo Conceitual do Domínio que representa o conceito de uma interação de ator / sistema com o domínio do negócio. Caso contrário, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistente entre os dois.

II. Analise a Descrição dos Cenários dos Casos de Uso do Domínio para verificar se os atributos das classes do Modelo Tipo do Negócio foram capturadas adequadamente.
ENTRADAS: Descrição dos Cenários dos Casos de Uso do Domínio
Modelo Tipo do Negócio
SAÍDAS: Relatório de Discrepâncias
Para cada cenário da Descrição dos Cenários dos Casos de Uso do Domínio execute os seguintes passos:
A. Leia atentamente os cursos normal e alternativos para identificar os dados a serem armazenados. Sublinhe em azul os dados provenientes da interação e de verde os dados calculados pelo sistema.
B. Encontre a(s) classe(s) correspondentes no Modelo Tipo de Negócio e marque com asterisco azul. Verifique para cada dado sublinhado em verde se tem atributo correspondente nesta(s) classe(s) e sublinhe em verde o atributo correspondente.
1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois ocorre ausência de atributo(s) na(s) classe(s) correspondente(s) do Modelo Tipo do Negócio.
2 - Certifique que cada atributo tenha o mesmo nome do dado correspondente na Descrição do Cenário e seu tipo básico definido. Caso contrário, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistente entre os dois.

A. Encontre a(s) classe(s) correspondentes no Modelo Tipo de Negócio. Verifique para cada dado sublinhado em azul se tem atributo correspondente nesta(s) classe(s) e sublinhe em azul o atributo correspondente.
   1 - Caso não consiga encontrar, certifique nas descrições dos outros cenários a necessidade de armazenar o dado obtido pela interação. Caso haja necessidade, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ocorrendo ausência de atributo(s) na(s) classe(s) correspondente(s) do Modelo Tipo do Negócio.
   2 - Certifique que cada atributo tenha o mesmo nome do dado correspondente na Descrição do Cenário e seu tipo básico definido. Caso contrário, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistente entre os dois.

II. Compare os diagramas de classes para verificar se as informações foram capturadas adequadamente.
ENTRADAS: Modelo Conceitual do Domínio
Modelo Tipo do Negócio
SAÍDAS: Relatório de Discrepâncias
Para cada classe do diagrama do Modelo Tipo do Negócio execute os seguintes passos:
A. Verifique se o nome da classe existe no Modelo Conceitual do Domínio e se ela está utilizando o nível adequado de abstração.
   1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, relatando a necessidade de conhecimento adicional para compreensão.

III. Reveja o Modelo Tipo do Negócio para assegurar que todos os atributos, relacionamentos e classes estão sendo levados em consideração.
ENTRADAS: Modelo Tipo do Negócio marcados com asterisco azul, realçado em amarelo e sublinhado em azul e verde
SAÍDAS: Relatório de Discrepâncias
A. Reveja no Modelo Tipo do Negócio as classes, seus atributos e relacionamentos para ter certeza que todo o diagrama de classe represente integralmente os Modelos Conceitual de Domínio e Casos de Uso e Descrição dos Cenários dos Casos de Uso do Domínio.
   1 - Certifique que não exista nenhuma classe sem asterisco, relacionamento sem estar realçado em amarelo e atributo sem estar sublinhado. Caso exista algum, preencha a Tabela de Discrepância, pois uma classe ou relacionamento no Modelo Tipo do Negócio não está presente no Modelo Conceitual do Domínio, ou por um atributo no Modelo Tipo do Negócio não está presente na Descrição dos Cenários dos Casos de Uso do Domínio.

TLV 4 – Modelo de Casos de Uso do Domínio e Cenários X Interface da Camada da Aplicação

Objetivo: Verificar se a especificação da Interface da Camada da Aplicação projeta todas as interfaces necessárias do sistema com todos os tipos de interface, seus atributos com os tipos básicos e suas assinaturas de operação inerentes à modelagem do sistema representada pelo Modelo de Casos de Uso do Domínio.
Pré-requisito: Esta leitura requer que o diagrama do Modelo de Casos de Uso do Domínio e Descrição dos Cenários dos Casos de Uso do domínio tenham sido aprovado pelo Processo Controle de Qualidade.
Entrada para o processo:

1. Modelo de Casos de Uso do Domínio.
2. Descrição dos Cenários dos Casos de Uso do Domínio.
3. Interface da Camada da Aplicação representada pelo diagrama de classes.

Procedimentos:

I. Leia o Modelo de Casos de Uso do Domínio e Cenários para entender os casos de uso e suas iterações.
ENTRADAS: Modelo de Casos de Uso do Domínio
Descrição de Cenários do Domínio
SAÍDAS: Serviços candidatos marcados em verde na Descrição de Cenários do Domínio
A. Para cada caso de uso leia atentamente a descrição do Cenário do Domínio correspondente tanto o curso normal quanto os alternativos.
B. Encontre os verbos ou descrição de ações, que são candidatos a serem serviços ou comportamento no sistema. Sublinhe em verde os verbos ou descrição de ações.
C. Procure por dados necessários de entrada e saída nos verbos identificados no passo B. Sublinhe em vermelho os dados de entrada em azul e os dados de saída.

II. Compare os casos de uso às interfaces do sistema para verificar se as interfaces foram capturadas adequadamente.
ENTRADAS: Modelo de Casos de Uso do Domínio
Interface da Camada da Aplicação
SAÍDAS: Relatório de Discrepâncias
Para cada caso de uso do Modelo de Casos de Uso do Domínio execute os seguintes passos:
A. Encontre a classe correspondente na especificação da Interface da Camada da Aplicação e marque esta classe com um asterisco azul.
1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois uma classe de sistema não está presente na especificação da Interface da Camada da Aplicação.
2 - Certifique que o nome da classe de interface do sistema inicia pela vogal em maiúscula "I" concatenado com o nome do caso de uso, quando possível. Caso não tenha o padrão, preencha a Tabela de Discrepância, pois pode haver falta de padrão ou ambigüidade, relatando a necessidade de conhecimento adicional para compreensão.
3 - Certifique que o estereótipo da classe esteja referenciado corretamente como "interface". Caso contrário, preencha a Tabela de Discrepância, pois pode haver falta de padrão ou ambigüidade, relatando a necessidade de conhecimento adicional para compreensão.

III. Compare as Descrições dos Cenários do Domínio às assinaturas das interfaces do sistema correspondente para verificar se os serviços das interfaces foram capturados adequadamente com os dados necessários de entrada e saída.
ENTRADAS: Descrição dos Cenários dos Casos de Uso do Domínio sublinhado em verde
Interface da Camada da Aplicação
SAÍDAS: Relatório de Discrepâncias
Para cada descrição de caso de uso do Modelo de Casos de Uso do Domínio execute os seguintes passos:
A. Encontre a classe de interface correspondente na especificação da Interface da Camada da Aplicação.
B. Para cada descrição de ação sublinhada em verde na descrição do cenário, tente encontrar um comportamento associado ou combinação de comportamentos nas assinaturas da classe de interface correspondente identificada no passo anterior A. Utilize dicas sintáticas, ou seja, nome de comportamento que é similar ou sinônimo para uma descrição de ação para ajudar na busca. Quando encontrado,

marque com um asterisco verde ambos os nomes do serviço na especificação das Interfaces da Camada da Aplicação e do cenário na Descrição dos Cenários dos Casos de Uso do Domínio.

1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois um serviço não está presente na classe correspondente da especificação da Interface da Camada da Aplicação.

2 - Certifique que o nome do serviço utilizado na classe correspondente da especificação da Camada da Aplicação é similar ou sinônimo à descrição da ação sublinhada em verde e com método de acesso público. Caso não tenha um nome similar, preencha a Tabela de Discrepância, pois a descrição pode estar ambígua, relatando a necessidade de conhecimento adicional para compreensão.

3 - Para cada dado sublinhado em azul correspondente a ação sublinhada em verde na descrição do cenário, certifique que tenha um argumento de entrada correspondente com seu tipo básico na assinatura do método identificado no passo 2. Quando encontrado, sublinhe o argumento de Discrepância, pois um parâmetro de entrada não está presente na assinatura do serviço da classe correspondente da especificação da Interface da Camada da Aplicação.

4 - Para cada dado sublinhado em vermelho correspondente a ação sublinhada em verde na descrição do cenário, certifique que a assinatura do serviço da classe correspondente identificada retorna o dado com o mesmo tipo básico. Quando encontrado, sublinhe em vermelho o nome da função e seu tipo básico. Caso contrário, preencha a Tabela de Discrepância, pois um parâmetro de saída não está presente na assinatura do serviço da classe correspondente da especificação da Interface da Camada da Aplicação.

II. Reveja todas as especificações da Interface da Camada da Aplicação para assegurar que todas as classes e seus serviços estão sendo levados em consideração.

ENTRADAS: Interface da Camada da Aplicação marcada com asterisco azul e verde, sublinhada em azul, verde e vermelho

SAÍDAS: Relatório de Discrepâncias

A. Procure por classes de interfaces que tenham sido omitidas na especificação da Interface da Camada da Aplicação.

1 - Certifique que não exista nenhuma classe do tipo “interface” sem asterisco azul. Caso exista alguma, preencha a Tabela de Discrepância, pois uma classe de interface não deveria ser definida desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

B. Procure por descrição de serviços nos cenários que tenham sido omitidos na especificação da Interface da Camada da Aplicação.

1 - Certifique que não existem verbos não marcados com asterisco verde. Caso exista, verifique se ele deveria ser incluído na especificação da Interface da Camada da Aplicação ou não por estar descrito apenas para melhorar a legibilidade da função do cenário. Se devesse estar incluído, preencha a Tabela de Discrepância, pois serviço foi omitido na especificação da Interface da Camada da Aplicação.

C. Para cada classe de interface da especificação da Interface da Camada da Aplicação procure por serviços que não tenham sido referenciadas.

1 - Certifique que não existem assinaturas não marcadas com asterisco verde. Caso exista, preencha a Tabela de Discrepância, pois um serviço não deveria ser definido desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

2 - Certifique que não exista nenhum argumento do serviço sem estar sublinhado em azul ou vermelho. Caso exista alguma, preencha a Tabela de Discrepância, pois um argumento de entrada ou saída da classe de interface não deveria ser definido desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

TLV 5 – Modelo de Características X Arquitetura do Domínio

Objetivo: Verificar a generalidade arquitetural de domínio baseado em camadas para facilitar o reúso com o explícito propósito de atender ao maior número de seleções identificadas no Modelo de Características.
Pré-requisito: Esta leitura requer que o diagrama do Modelo de Características tenha sido aprovado pelo Processo Controle de Qualidade.
Entrada para o processo:

1. Modelo de Características baseado no método FORM representado pelo diagrama de características em camada de categorias.
2. Arquitetura do Domínio representado pelo diagrama de classes em camada.

Procedimentos:

I. Leia as características de cada categoria do Modelo de Características para verificar se a estrutura arquitetural do domínio atende ao maior número de possibilidades de seleção.

II. Leia os componentes de cada camada da Arquitetura de Domínio para identificar o padrão arquitetural definido e os padrões utilizados.

ENTRADAS: Modelo de Características
Arquitetura de Domínio

SAÍDAS: Relatório de Discrepâncias

Para cada possibilidade de seleção de característica do Modelo de Característica execute os seguintes passos:

A. Identifique a camada arquitetural de domínio adequada e encontre a(s) classe(s) correspondente(s) no diagrama de classes da Arquitetura de Domínio. Utilize dicas sintáticas, o nome da característica seja similar ao nome da classe, para ajudar na pesquisa, mas esteja certo que o significado semântico dos conceitos na característica e classe seja o mesmo. Marque com asterisco azul a(s) classe(s) correspondente(s) na Arquitetura do Domínio.

1 - Caso não consiga encontrar, preencha a Tabela de Discrepância, pois há característica que não está presente na Arquitetura de Domínio, ou há inconsistência entre os dois.

2 - Para cada característica identificada, certifique que suas relações representadas por linha contínua ou tracejada no Modelo de Características estejam sendo capturadas no diagrama arquitetural do domínio. Caso não contrário, preencha a Tabela de Discrepância, pois há relações entre características que não está presente na Arquitetura do Domínio, ou há inconsistência entre os dois.

III. Reveja na Arquitetura do Domínio se todos as classes de cada camada estão sendo levadas em consideração.

ENTRADAS: Arquitetura do Domínio marcada com asterisco azul

SAÍDAS: Relatório de Discrepâncias

A. Reveja se as classes estão corretamente capturadas na Arquitetura do Domínio.

1 - Certifique que não existe nenhuma classe sem asterisco azul. Caso exista alguma, verifique se está utilizando um padrão de projeto adequado. Caso contrário, preencha a Tabela de Discrepância, pois há classe que não deveria ser definido desta maneira, relatando a necessidade de conhecimento adicional para compreensão.

TLV 6 – Arquitetura do Domínio X Especificação de Componentes

Objetivo: Verificar se todas as classes do diagrama arquitetural do domínio e seus relacionamentos estão capturados na Especificação de Componentes.
Pré-requisito: Esta leitura requer que o diagrama Arquitetura do Domínio tenha sido aprovado pelo Processo Controle de Qualidade.
Entrada para o processo:

1. Arquitetura de Domínio representada por diagrama de classe em camada.
2. Especificação de Componentes representados por diagrama de classe com notação *lollipop*.

Procedimentos:

I. Analise na Arquitetura do Domínio as classes de cada camada para entender as especificações dos componentes.

ENTRADAS: Arquitetura do Domínio
Especificação de Componentes

SAÍDAS: Relatório de Discrepâncias

A. Para cada classe da Arquitetura do Domínio execute os seguintes passos:

1 - Encontre a especificação de componente correspondente e a marque com um asterisco azul. O nome do componente deve ser similar ou sinônimo ao nome da classe de especificação de componente. Caso não consiga encontrar a especificação do componente, preencha a Tabela de Discrepância, pois um componente não está presente na Especificação de Especificação de Componentes.

2 - Verifique se a classe tem algum relacionamento de associação, herança, agregação ou composição. Caso tenha, para cada relacionamento encontre a interface correspondente que deve estar representada na especificação do componente identificado no passo anterior em notação *lollipop*. Realce em verde o nome desta(s) interface(s) na especificação do componente. Caso não consiga encontrar a(s) dependência(s) desta(s) interface no componente em questão, preencha a Tabela de Discrepância, pois há informação em um documento que não está presente em outro, ou há inconsistência entre os dois.

II. Reveja as especificações dos componentes para assegurar que todas as classes e seus relacionamentos especificados no diagrama arquitetural do domínio estão sendo levados em consideração.

ENTRADAS: Especificação de Especificação de Componentes marcado com asterisco azul e realçado em verde Arquitetura do Domínio

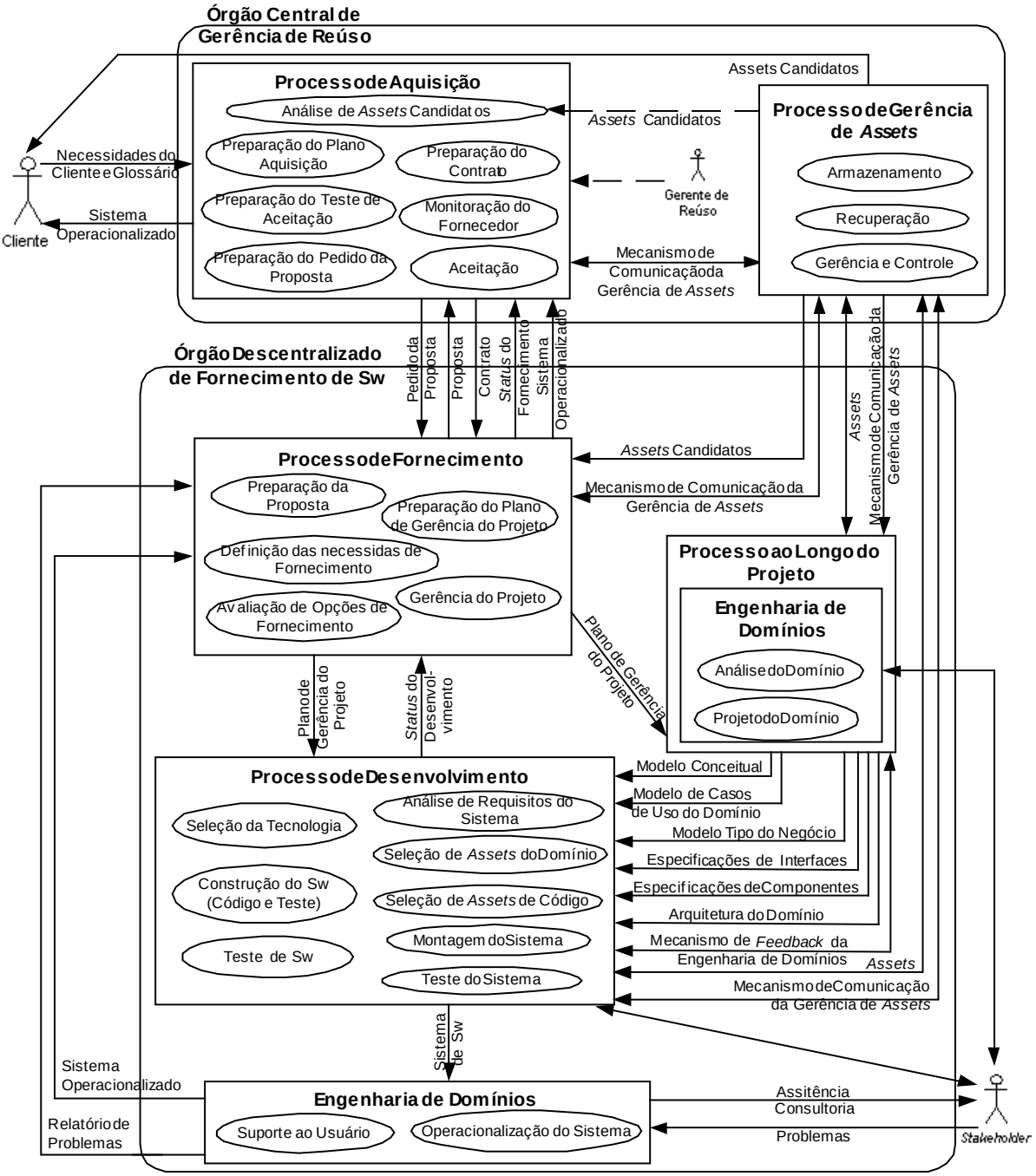
SAÍDAS: Relatório de Discrepâncias

A. Reveja os componentes para ter certeza que todas as classes e seus relacionamentos da Arquitetura do Domínio estão corretamente capturados na Especificação de Componente.

1 - Certifique que não existe nenhuma classe de especificação de componente sem asterisco azul. Caso exista alguma, preencha a Tabela de Discrepância, pois uma especificação de componente não está presente na Especificação de Especificação de Componentes.

8 ANEXOS

8.1 ANEXO 1: PROPOSTA DO PROCESSO DE CICLO DE VIDA DO SOFTWARE BASEADO EM REUSO NO CONTEXTO DO MD



8.2 ANEXO 2: ARTEFATOS DOS PROCESSOS DE ENGENHARIA DE DOMÍNIO E DESENVOLVIMENTO BASEADO EM REUSO NO CONTEXTO DO MD

Necessidades do Cliente

1. Objetivos

Auxiliar e automatizar algumas das tarefas de Compilação do Quadro Tático, a fim de aumentar os níveis de confiabilidade e disponibilidade das informações táticas e melhorar a visualização do cenário tático.

2. Domínio

Compilação do Quadro Tático, dentro do domínio de Sistemas de Comando, Controle e Navegação.

3. Referências

Glossário Controlador Tático 2.0

4. Visão Geral do Produto

O Controlador Tático deverá receber dados dos diversos sensores existentes na plataforma, ou de inserção manual, e ser capaz de tratar estes dados coletados, fornecendo informações táticas em tempo real a fim de auxiliar na compilação e avaliação do Quadro Tático.

5. Funcionalidade

5.1 Recepção de dados de sensores

O aplicativo deverá ser capaz de receber dados dos diversos sensores como giroscópicas, hodômetros, anemômetros, GPS, DGPS, radares e outros, além da possibilidade de inserção manual de valores dos mesmos.

5.2 Comunicação via Link YB/14

O aplicativo deverá ser capaz de receber dados das diversas plataformas que possuam sistemas automáticos de informações táticas e que as transmitam através do link YB.

5.3 Cálculo do Rumo e Velocidade reais da plataforma primária

Calcular rumo e velocidade reais da plataforma primária baseado nos dados de duas posições inseridos diretamente por um sensor ou inserção manual do operador.

5.4 Cálculo do Vento real

Calcular o vento real baseado nos dados de rumo e velocidade reais da plataforma primária e nos dados de direção e intensidade do vento relativo existentes no sistema.

5.5 Cálculo da Corrente

Calcular a corrente baseado nos dados de rumo e velocidade reais da plataforma primária e os fornecidos pela agulha giroscópica e hodômetro.

5.6 Abertura de Acompanhamento

Através da inserção de dados de posição de uma plataforma e da correlação destes com um acompanhamento.

5.7 Atualização de Acompanhamento

Quando da inserção de dados de posição de uma plataforma já acompanhada pelo sistema. Devem ser atualizados o rumo e velocidade da mesma e efetuado o cálculo dos dados táticos desta em relação à plataforma primária.

5.8 Extração de Dados do Acompanhamento

Através da seleção de um acompanhamento, deve ser calculada a posição estimada deste e apresentado os dados atualizados do mesmo.

5.9 Extração de Dados do Ambiente

Através da seleção para extração de dados do ambiente, o sistema apresenta os dados atualizados do ambiente (vento real e corrente).

5.10 Cálculo de Dados do Acompanhamento

Calcular, após cada atualização dos dados do acompanhamento, rumo, velocidade, rumo de passagem segura, ponto de maior aproximação.

5.11 Cálculo do Rumo e Velocidade da Plataforma

Calcular rumo e velocidade reais da plataforma baseado na posição atual e anterior do mesmo.

5.12 Cálculo do Ponto de Maior Aproximação do Acompanhamento

Calcular a posição e a hora de maior aproximação do acompanhamento à plataforma primária baseado na posição atual e anterior do mesmo.

5.13 Cálculo do Rumo para Passagem Segura

Calcular o rumo para a plataforma primária passar a uma distância pré-determinada do acompanhamento definido mantendo sua velocidade caso seu PMA esteja inferior à distância de passagem segura.

5.14 Alarme de Colisão

Apresenta o número do acompanhamento e hora prevista do ponto de maior aproximação caso este venha a passar a uma distância inferior à distância de passagem segura definida previamente.

5.15 Cancelamento de Acompanhamentos

Um acompanhamento é cancelado por inferência direta do operador ou caso se afaste para fora dos limites do Cenário Tático definido.

6. Características desejadas

6.1 Usabilidade

6.1.1 Tempo de treinamento do usuário padrão

Quatro horas. Uma pessoa com nível secundário de formação deve estar capacitada a operar o sistema com quatro horas de treinamento.

6.1.2 Tempo de treinamento do mantenedor do sistema

Oito horas. Uma pessoa com conhecimento básico de computação deve estar capacitada a detectar falhas e reiniciar o sistema com oito horas de treinamento.

6.2 Confiabilidade

6.2.1 Disponibilidade

O sistema deve poder permanecer em funcionamento 24 (vinte e quatro) horas por dia.

6.2.2 Tempo entre falhas (MTBF – Mean Time Between failures)

Não deve apresentar falhas em menos de sete dias.

6.2.3 Tempo para reparos (MTTR – Mean Time to repair)

O tempo para reiniciar/reparar o sistema não deve ser superior a dez minutos.

6.2.4 Persistência de dados

Os dados do acompanhamento devem ser persistidos de forma que uma queda de energia, ou a reinicialização do sistema não gere perda de dados.

6.2.5 Precisão

Tempo: 30 segundos.
Distância: 10 jardas ou 10 metros.
Marcação: 1º (um grau).

6.3 Performance

6.3.1 Tempo de resposta da transação (média/máximo)

Dois segundos/dez segundos, após a inserção de dados.

6.3.2 Transações por segundo

Cem transações/segundo.

6.3.3 Capacidade (usuários/transações simultâneas)

Três usuários/ cinqüenta transações.

6.3.4 Recursos

RAM: 256 MB.
Disco Rígido: 2GB
Comunicação: placa de conexão com o Link Yb/14, interface para GPS, interface para agulha giroscópica, interface para hodômetro e interface para anemômetro.

6.4 Suporte

6.4.1 Manual de Apoio

O sistema deve possuir um manual de apoio para utilização, glossário e procedimento de reinicialização..

6.5 Regras de tecnologia e limitações

6.5.1 Linguagem de software

Não determinada.

6.5.2 Plataforma de operação

Multiplataforma (linux, windows).

6.5.3 Plataforma de Midddleware

Não determinada.

6.5.4 Portabilidade

Sistema portável.

6.5.5 Sistemas legados

Não determinada.

6.5.6 Sistemas correlacionados

Sistema de Armas, Sistema de Navegação Satélite, Sistema Radar, sensores de navegação (agulha giroscópica, anemômetro, hodômetro).

Glossário

1. Introdução

Este documento é usado para definir a terminologia específica do domínio do problema, o Controlador Tático, cujo pleno entendimento se faz necessário para o desenvolvimento e operação do sistema a ser desenvolvido.

2. Definições

2.1 Acompanhamento

Visão virtual de uma plataforma para o sistema e que possui dados correlacionados. Passa a existir para o sistema após inserção direta do operador ou automaticamente por um dos sensores conectados ao sistema (Extrator radar ou link). Possui sempre um identificador numeral, o número de acompanhamento (NA).

2.2 Agulha Giroscópica

Sensor que fornece a indicação referencial de Norte para o sistema, o norte da giro.

2.3 Alarme de Colisão

Retorno fornecido pelo sistema quando o cálculo do PMA indicar a passagem de um contato a uma distância inferior a definida como segura. Pode ser uma indicação na tela ou um evento para um sistema externo.

2.4 Cálculos Táticos

Cálculos efetuados sobre os dados recebidos do cenário tático. Por exemplo: cálculo do rumo e velocidade de um acompanhamento, cálculo do ponto de maior aproximação de um acompanhamento, cálculo do rumo para passagem segura de um acompanhamento, cálculo do vento real, cálculo da corrente e cáculo da posição estimada da plataforma.

2.5 Cenário Tático

Conjunto de informações formado pelos meios disponíveis e acompanhamentos existentes dentro de um espaço real limitado pelo alcance dos sensores e pela impossibilidade de alteração destes em um curto espaço de tempo. Este espaço de tempo é definido pela velocidade e disponibilidade dos meios existentes.

2.6 Centro de Informações de Combate - CIC

Local onde são compilados os dados referentes ao Cenário Tático.

2.7 Controlador Tático

Sistema responsável por auxiliar no recebimento de dados referentes ao Cenário Tático, na realização de cálculos táticos necessários e na apresentação destes de uma forma amigável.

2.8 Dados históricos de posição da plataforma

Dados fornecidos pelos sensores de detecção, que servem de entrada para o cálculo dos dados táticos. Por exemplo: distância, marcação, posição em latitude e longitude.

2.9 Dados de Configuração

Dados fornecidos pelo operador, que definem a distância para passagem segura e o limite do cenário tático.

2.10 Dados de Ambiente

Dados fornecidos pelos sensores de navegação, que servem de entrada para o cálculo do vento real e corrente. Por exemplo: vento relativo e velocidade relativa da plataforma.

2.11 Differential Global Positioner System - DGPS

Um dos Sistemas de Posicionamento Satélite existente. Sua precisão da informação de posicionamento é de aproximadamente 15 metros.

2.12 Distância para Passagem Segura

Distância pré-definida para o cálculo do rumo para passagem segura.

2.13 Enlace Automático de Dados - EAD

Sistema de comunicação, que permite troca de informações entre plataformas sobre acompanhamentos.

2.14 Giro

O mesmo que agulha giroscópica.

2.15 Global Positioner System - GPS

Um dos Sistemas de Posicionamento Satélite existente. Sua precisão da informação de posicionamento é de aproximadamente 100 metros.

2.16 Hodômetro

Sensor que fornece a velocidade relativa da plataforma em relação ao meio na qual ela se encontra.

2.17 Link

Equipamento responsável pela conexão de uma plataforma a outras, através de um enlace automático de dados por radio transmissão. O Link pode ser considerado como um sensor de detecção para o sistema.

2.28 Sistema Externo

Sistema que não pertencem ao Controlador Tático, mas que interage com este fornecendo ou extraindo dados de acompanhamento e de ambiente.

2.29 Sistema de Posicionamento Satélite

Sistema formado por sensor que fornece, através da detecção de sinais enviados de satélites, informações sobre a plataforma tais como: latitude, longitude, rumo, velocidade e altitude em relação à superfície da terra.

2.30 Velocidade Relativa

Velocidade da plataforma em relação ao meio na qual se encontra. Para Navios, o mesmo que velocidade na superfície.

2.31 Velocidade

Velocidade absoluta ou relativa a superfície da terra. Para Navios, o mesmo que velocidade no fundo.

2.32 Vento Real ou Verdadeiro - VV

Rumo e direção do vento existente no local.

2.33 Vento Relativo - VR

Rumo e velocidade do vento em relação à plataforma primária.

Cenários do Domínio

Nome	inserir dados de ambiente	**N°1**
Ator	sensor de navegação	
Trigger	recebimento de dados de ambiente	
Pré-condição		
Curso Normal		
1. O sensor de detecção insere dados de ambiente (vento relativo, velocidade relativa)**.** 2. O sistema calcula o vento real e corrente a partir dos dados de ambiente inseridos e do rumo e velocidade da plataforma primária. 3. O sistema armazena dados calculados.		
Cursos Alternativos		

Nome	abrir acompanhamento	**N°2**
Ator	sensor de detecção	
Trigger	inserção de dados de um novo acompanhamento	
Pré-condição		
Curso Normal		
1. O sensor de detecção insere latitude e longitude da plataforma definida**.** 2. O sistema cria novo acompanhamento e atribui número de acompanhamento. 3. O sistema armazena os dados históricos de posição. 4. O sistema associa os dados ao novo acompanhamento.		
Cursos Alternativos		

Nome	atualizar dados da plataforma	**N°3**
Ator	sensor de detecção	
Trigger	recebimento de novos dados históricos de posição da plataforma	
Pré-condição	já existir a plataforma	
Curso Normal		
1. O sensor de detecção define a plataforma a ser atualizada. 2. O sensor de detecção insere latitude e longitude da plataforma definida. 3. O sistema armazena os dados históricos de posição. 4. O sistema calcula rumo, velocidade e PMA. 5. O sistema armazena os dados calculados.		
Cursos Alternativos		
Passo 4: **Trigger:** Os dados inseridos são da plataforma primária. a) O sistema calcula rumo e velocidade. b) O sistema retorna ao passo 5. **Passo 5:** **Trigger:** O PMA é inferior à distância de passagem segura inserida. a) Inclui *alarme de colisão*. b) O sistema calcula rumo para passagem segura. c) O sistema armazena os dados calculados.		

Nome	extrair dados	N°4
Ator	sistema externo	
Trigger	solicitação de dados	
Pré-condição		

Curso Normal

1. O sistema externo solicita dado a ser apresentado**.**
2. O sistema calcula a posição estimada da plataforma cujos dados foram solicitados.
3. O sistema fornece dado solicitado.

Cursos Alternativos

Passo 2:
Trigger: O dado solicitado é de ambiente (vento real e corrente).
a) O sistema retorna ao passo 3.
Passo 2:
Trigger: O dado solicitado é de configuração (limite do cenário e distância passagem segura).
a) O sistema retorna ao passo 3.
Passo 3:
Trigger: A posição estimada da plataforma se encontra fora do cenário tático.
a) O sistema cancela o acompanhamento da plataforma.
b) Encerra o caso de uso.

Nome	definir configuração	N°5
Ator	operador	
Trigger	alteração da configuração (distância padrão para passagem segura e limite cenário tático)	
Pré-condição		

Curso Normal

1. O operador insere distância para passagem segura e limite cenário tático**.**
2. O sistema armazena distância para passagem segura e limite cenário tático.

Cursos Alternativos

Nome	cancelar acompanhamento	N°6
Ator	operador	
Trigger	cancelamento de um acompanhamento	
Pré-condição	já existir o acompanhamento	

Curso Normal

1. O operador define o acompanhamento a ser cancelado**.**
2. O sistema cancela o acompanhamento.

Cursos Alternativos

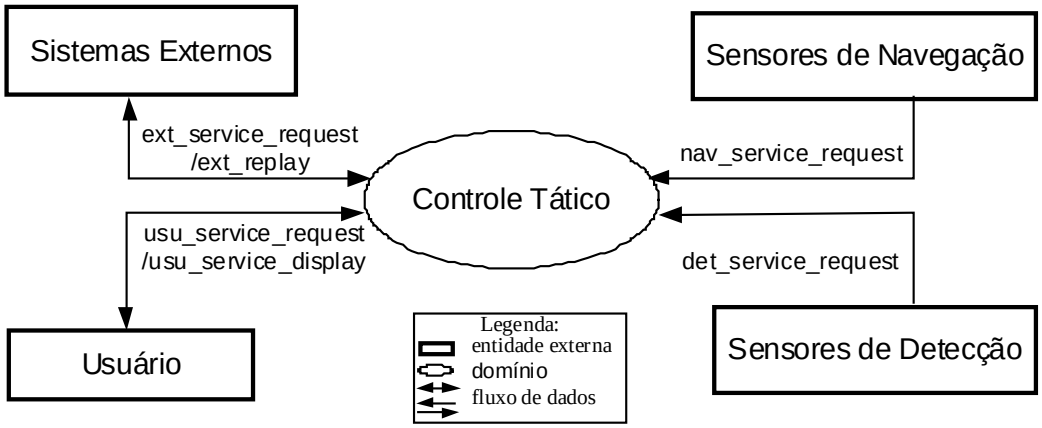
Nome	alarme de colisão	N°7
Ator	incluir	
Trigger	a distância calculada no PMA é inferior a definida como segura	
Pré-condição		

Curso Normal

1. O sistema apresenta mensagem de alarme de colisão com o número do acompanhamento e o PMA do mesmo**.**

Cursos Alternativos

Modelo de Contexto do Domínio do Controlador Tático



Modelo Conceitual do Controle Tático

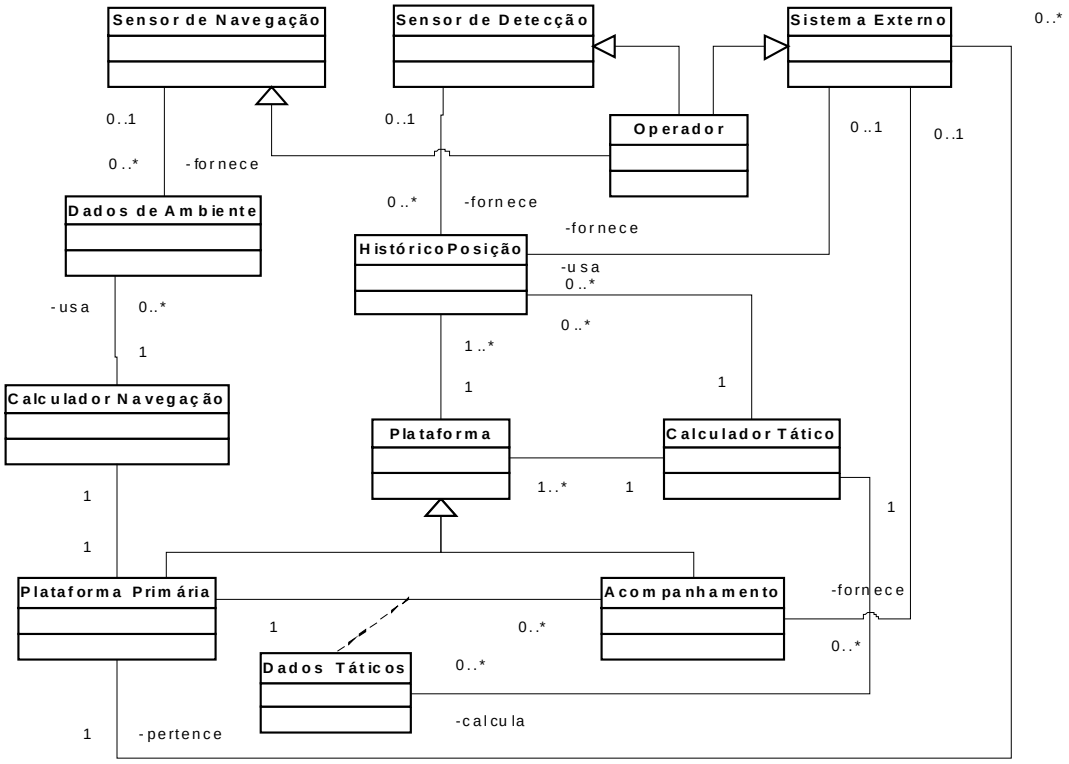
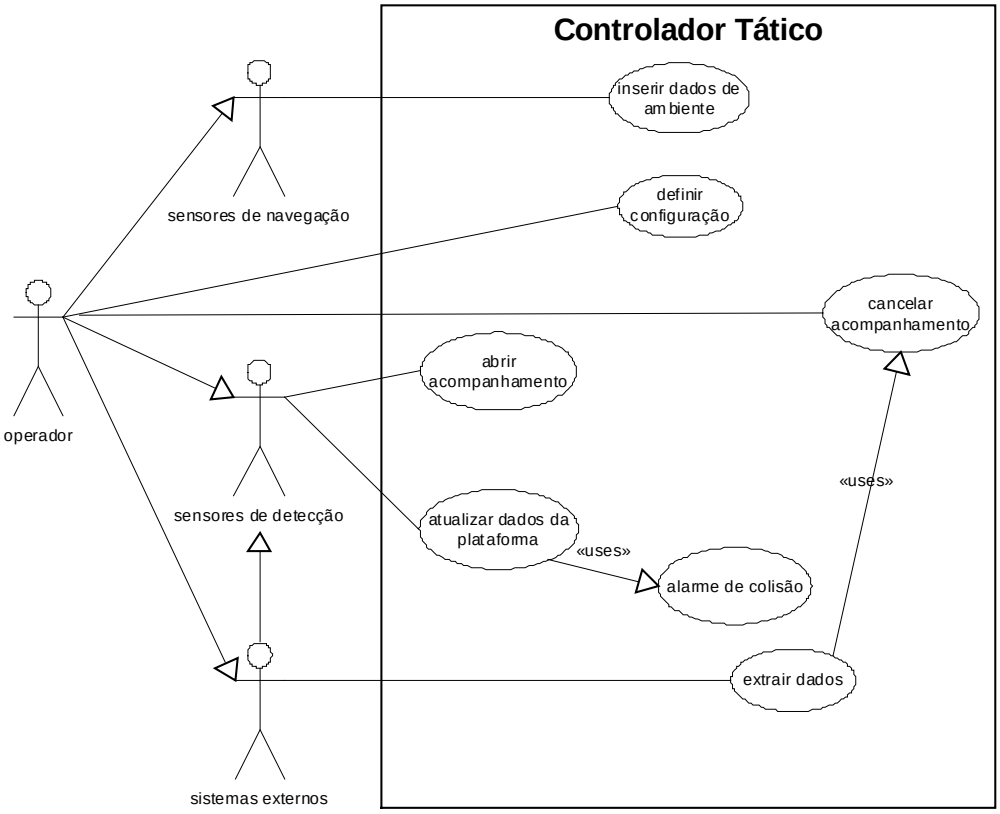
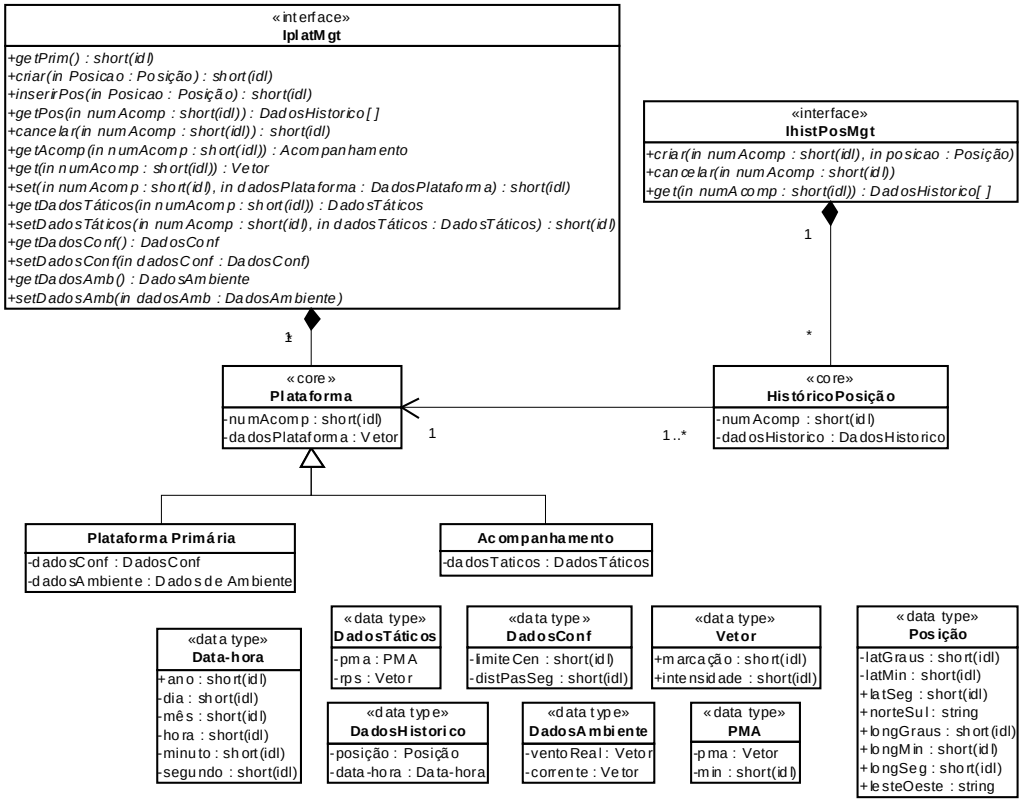


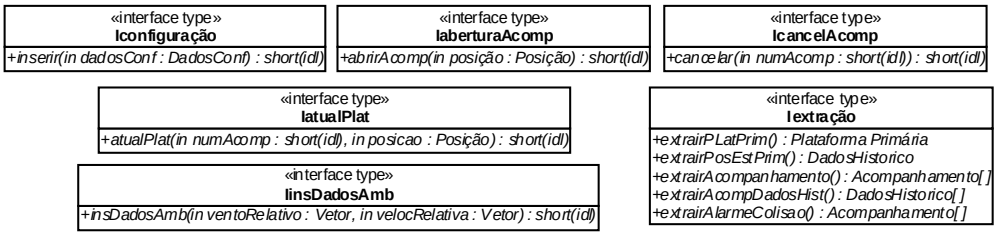
Diagrama de Casos de Uso do Domínio do Controle Tático



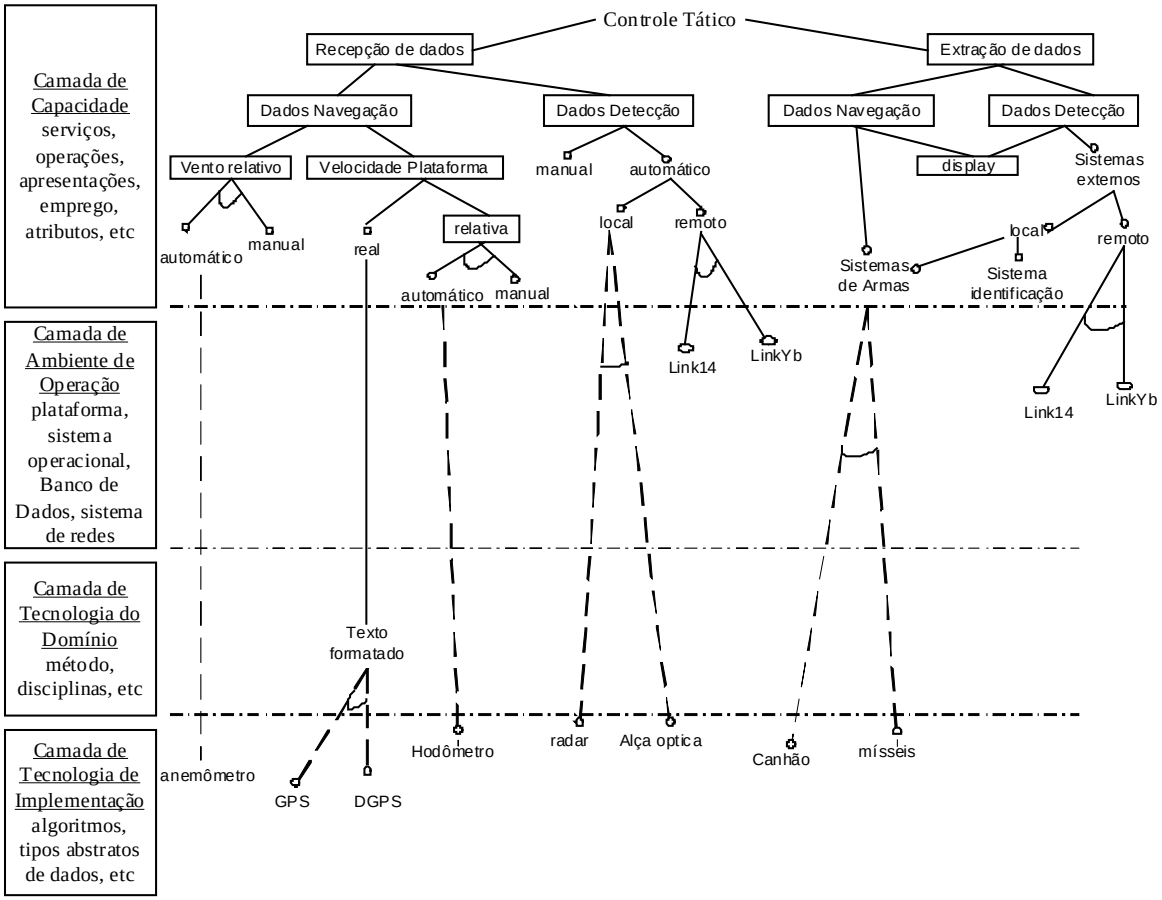
Interface da Camada de Negócio e Tipos de Dados do Controle Tático



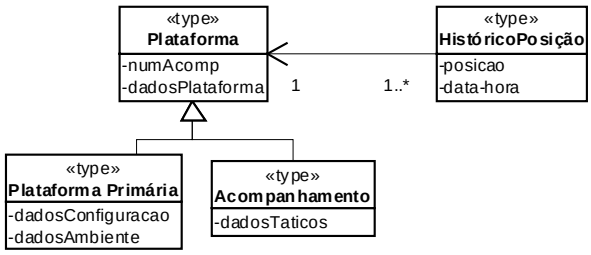
Interface da Camada de Aplicação do Controle Tático



Modelo de Característica do Controle Tático



Modelo Tipo de Negócio do Controle Tático



Arquitetura do Controle Tático

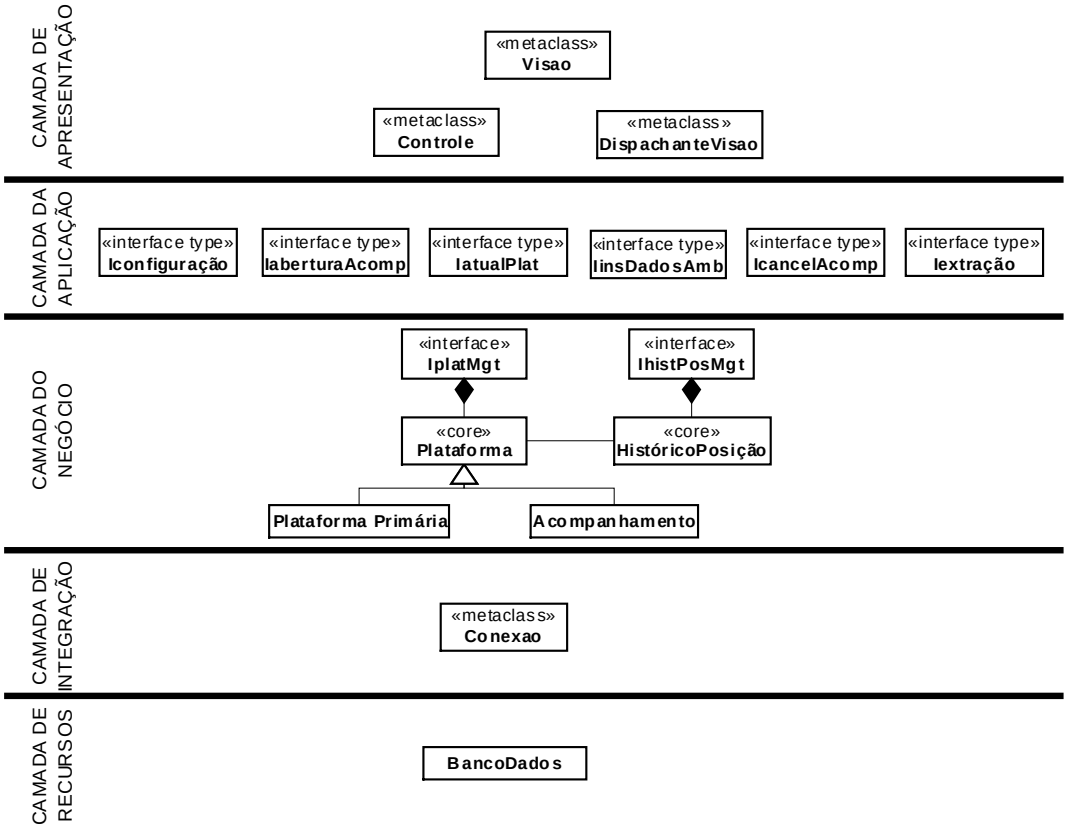
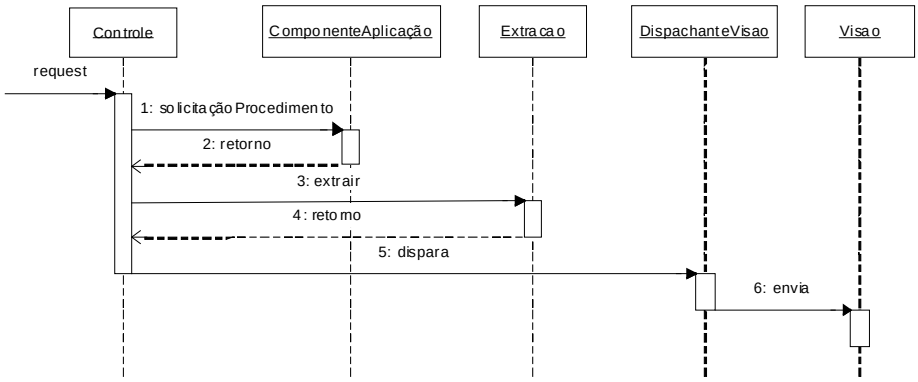


Diagrama de Seqüência / Colaboração do Controle Tático

Diagrama Seqüência Camada Aplicação



DiagramaSeqüênciaAbrirAcompanhamento

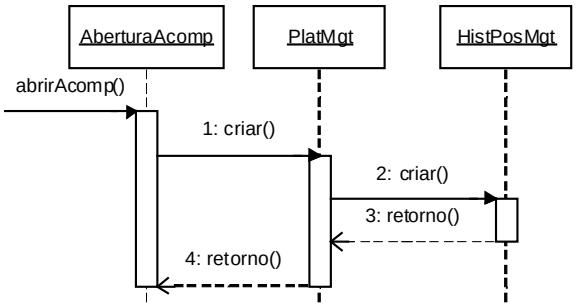


Diagrama Seqüência Configuração

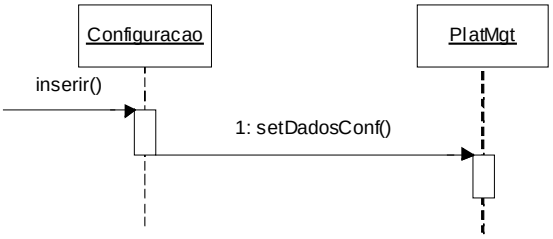


Diagrama de Seqüência Cancela Acompanhamento

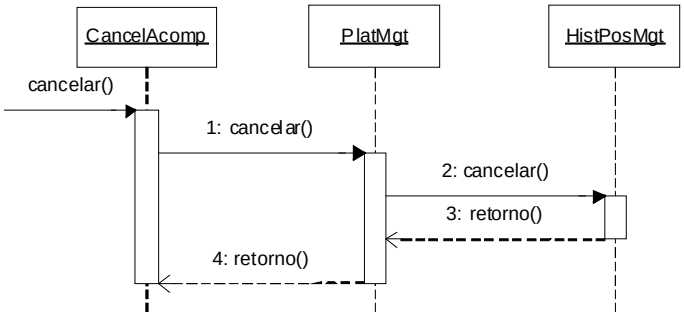


Diagrama de Seqüência Extração de Dados da Plataforma Primária

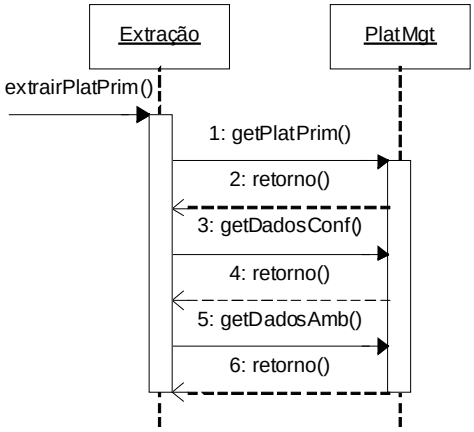


Diagrama de Seqüência Extração de Dados de posição da Plataforma Primária

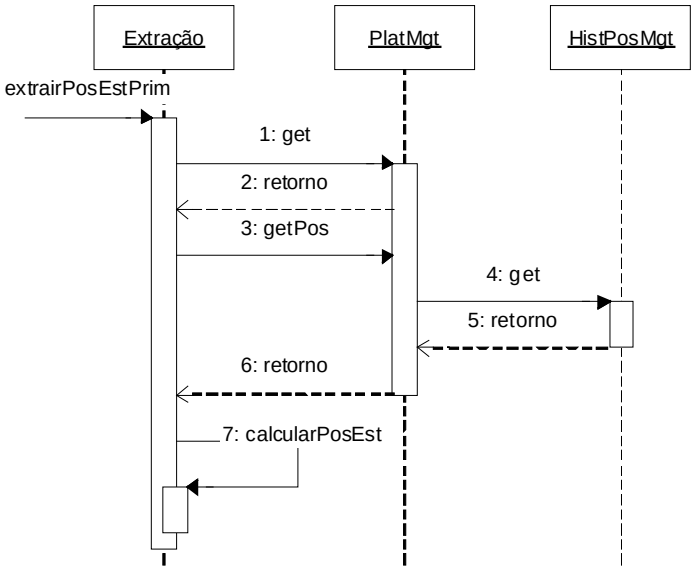


Diagrama de Seqüência Extração de Dados do Acompanhamento

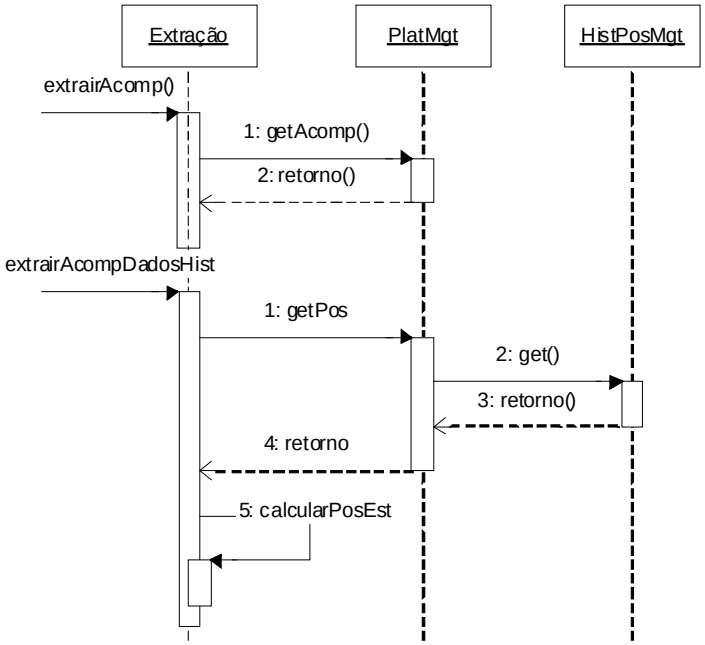
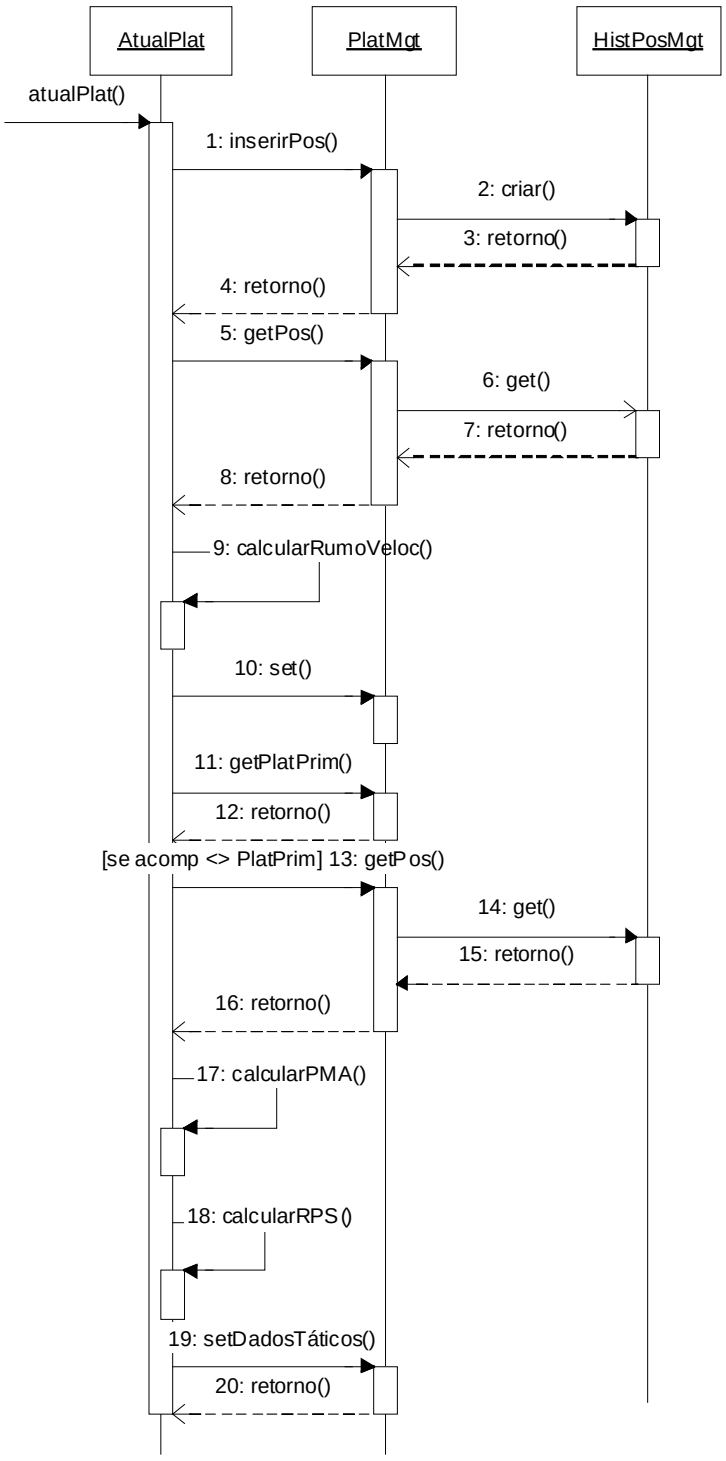
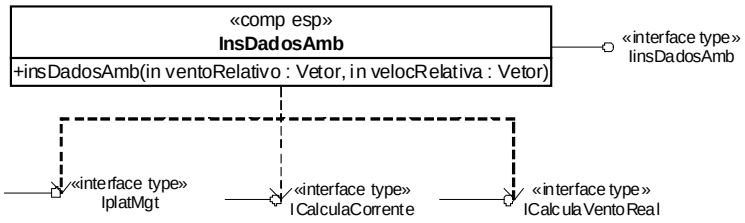


Diagrama de Seqüência Atualiza Acompanhamento

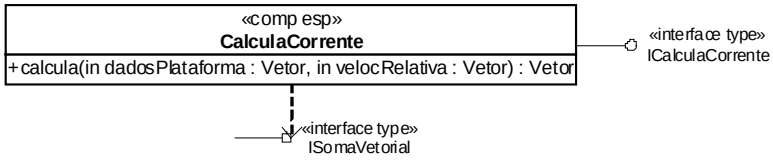


Especificação de Componentes

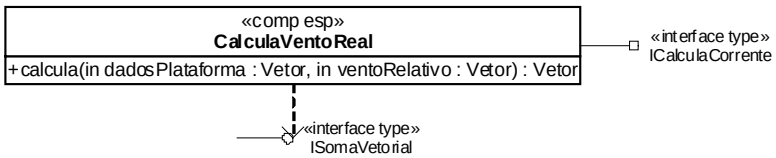
Componente InsDadosAmb com suas dependências



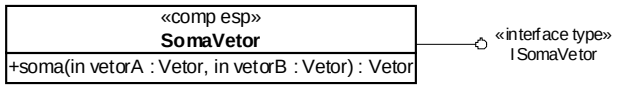
Componente CalculaCorrente com suas dependências



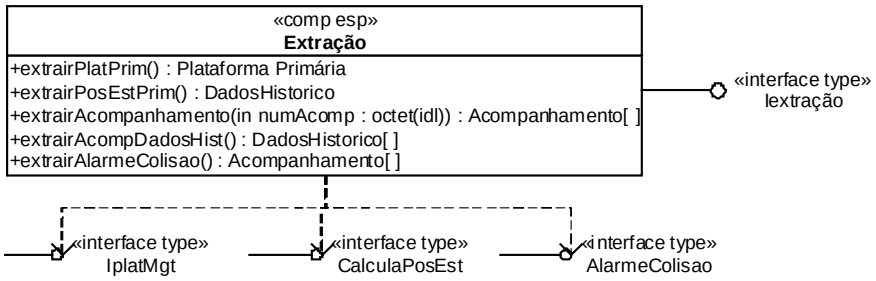
Componente CalculaVentoReal com suas dependências



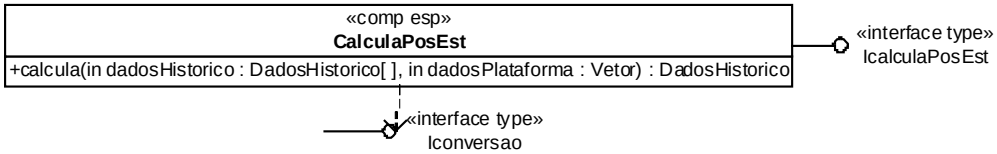
Componente SomaVetor



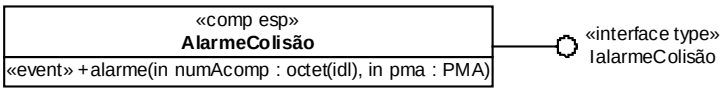
Componente Extração



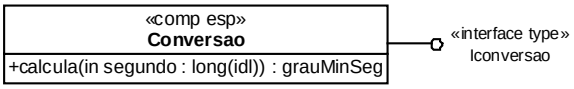
Componente Calcula Posição Estimada



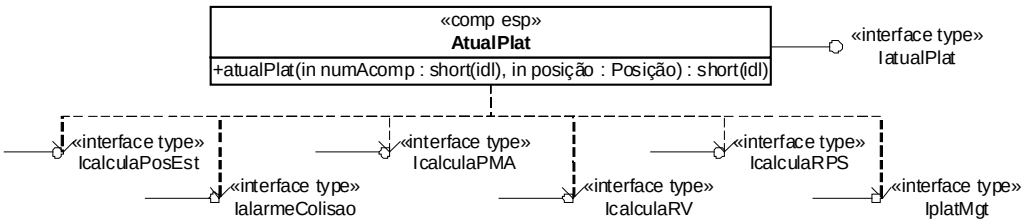
Componente Alarme de Colisão



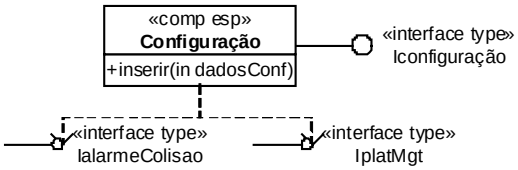
ComponenteConversão



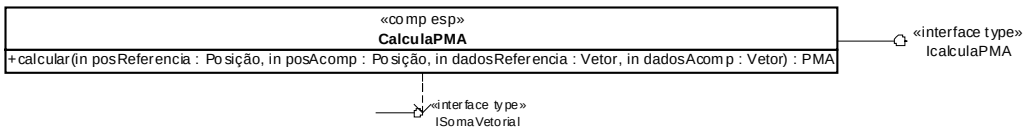
ComponenteAtualizaPlataforma



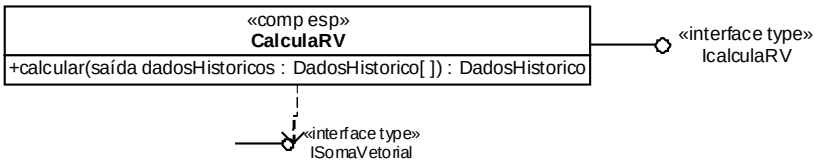
Componente Configuração



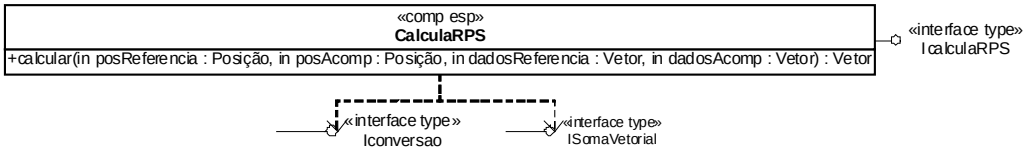
Componente Calcula PMA



Componente Calcula Rumo e Velocidade



Componente Calcula Rumo para Passagem Segura



9 GLOSSÁRIO DE TERMOS TÉCNICOS E EXPRESSÕES USADAS

- **ANÁLISE DE PONTOS POR FUNÇÃO.** Técnica de avaliação de um sistema, conhecida como FPA – *Function Point Analysis*, baseada na medição do valor das funções executadas pelos programas, ao invés de utilizar como base o volume ou a complexidade do código dos programas. A técnica está baseada na visão externa do usuário, sendo, portanto, independente da linguagem utilizada, permitindo calcular o esforço de programação e auxiliando o usuário final a melhorar o exame e avaliação de projetos. (IFPUG, 2000)
- **ARTEFATO**, s. m. Uma peça de informação tangível que é criada, alterada e usada durante o desenvolvimento das atividades; representa uma área de responsabilidade; e é passível de ser colocada separadamente sobre um controle de versões. Um artefato pode ser um modelo, um elemento de modelo ou um documento. (JACOBSON et al., 1999)
- ***ASSET***, s. m. *Asset* é um artefato ou conjunto de artefatos que tenham sido criados com um propósito explícito de ser reutilizado em outros esforços de desenvolvimento. Neste sentido, ele é a unidade que irá ser produzida, utilizada e gerenciada para permitir o reuso. (RATIONAL SOFTWARE CORPORATION, 2001)
- **CARACTERÍSTICA**, s. f. Um atributo de um sistema que expressa ou afeta diretamente o usuário, o desenvolvedor ou outra entidade que interage com o sistema. (NIST, 1994)
- **COMPONENTE**, s. m. Um pacote coerente de artefatos de *software* que podem ser desenvolvidos independentemente e entregues como uma unidade e que podem ser compostos, ou trocados, com outros componentes para formar algo maior. (D´SOUZA et al., 1998)
- **CONFORMIDADE**, s. f. Atendimento a um requisito. (ABNT9000, 2001)
- **DEFEITO**, s. m. Passo, processo ou definição de dados incorretos, por exemplo, um comando incorreto. (IEEE610.12, 1990)
- **DOMÍNIO**, s. m. O espaço do problema. (NIST, 1994)
- **EFICÁCIA**, s. f. O grau em que um sistema realiza o que dele se espera - “fazer a coisa certa”. (SEPIN, 2002)
- **EFICIÊNCIA**, s. f. O grau em que um sistema utilizou os recursos que deveria ter utilizado para atingir os objetivos ou realizar as atividades programadas – “fazer certo as coisas”. (SEPIN, 2002)

- **EFETIVIDADE**, s.f. A qualidade do que gera efeito real e resultados verdadeiros, que merece confiança, que se pode contar com segurança. (SEPIN, 2002)
- **ENGANO**, s. m. Ação humana que produz um resultado incorreto, por exemplo, uma ação incorreta tomada pelo programador. (IEEE610.12, 1990)
- **ENGENHARIA DE DOMÍNIO**. Uma linha baseada em reuso para definir o escopo (definição do domínio), especificar a estrutura (arquitetura do domínio), e construir os *assets* (por exemplo, projeto de requisitos, código de *software*, documentação) para uma classe de sistemas, subsistemas ou aplicações. Engenharia de domínio pode incluir as seguintes atividades: definição do domínio, análise de domínio, desenvolver a arquitetura de domínio e implementação do domínio. (NIST, 1994)
- **ENGENHEIROS DE DOMÍNIO**. Um grupo que desempenha atividades de engenharia de domínio (incluindo análise de domínio, projeto de domínio, construção de *asset* e manutenção de *asset*). (IEEE1517, 1999)
- **ERRO**, s. m. Diferença entre o valor obtido e o valor esperado, por exemplo, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro. (IEEE610.12, 1990)
- **FALHA**, s. m. Produção de uma saída incorreta com relação à especificação. (Veja também não-conformidade). (IEEE610.12, 1990)
- **GARANTIA DA QUALIDADE**. Conjunto de atividades planejadas e sistemáticas, implementadas no sistema da qualidade e demonstradas como necessárias para prover confiança adequada de que uma entidade atenderá os requisitos para a qualidade. (ISO8402, 1994)
- **MODELO DE DOMÍNIO**. Um produto da análise de domínio que prove uma representação dos requisitos do domínio. O modelo de domínio identifica e descreve a estrutura dos dados, fluxo de informações, funções, restrições e controles do domínio que são incluídos no sistema de *software* dentro do domínio. O modelo de domínio descreve as semelhanças e variações entre os requisitos dos sistemas de *software* no domínio. (NIST, 1994)
- **NÃO-CONFORMIDADE**. Não atendimento de um requisito. (ABNT9000, 2001)
- **PROCEDIMENTO**, s. m. Forma específica de executar uma atividade ou um processo. (ISO9000, 2001)
- **PROCESSO DE *SOFTWARE***. Conjunto de atividades, métodos, práticas e transformações que as pessoas empregam para desenvolver e manter *software* e os produtos

associados (por exemplo, planos de projeto, documentos de projeto, código, casos de teste, manual do usuário). (ISO8402, 1994)

- **QUALIDADE**, s. f. Grau no qual um conjunto de características inerentes satisfaz a requisitos. (ISO9000, 2001)
- **RASTREABILIDADE**, s. f. Capacidade de recuperar o histórico, a aplicação ou a localização daquilo que está sendo considerado. (ABNT9000, 2001)
- **REQUISITO**, s. m. Necessidade ou expectativa do cliente e de outra parte interessada, geralmente explicitada como condição de negócio no contrato com o fornecedor para resolver um problema ou alcançar um objetivo. É uma característica, tais como especificação técnica, prazo de entrega, garantia, que o cliente "requer" do produto. (ABNT9000, 2001)
- **RETRABALHO**, s. m. Ação sobre um produto não-conforme, afim de torná-lo conforme aos requisitos. (ABNT9000, 2001)
- **REUSABILIDADE**, s. f. (A) O grau que um *asset* pode ser utilizado em mais de um sistema de *software* ou na construção de outros *assets*. (B) Em uma biblioteca de reuso, a característica de um *asset* de ser utilizada em diferentes contextos, sistemas de software ou na construção de diferentes *assets*. (IEEE1517, 1999)
- **REUSO**, s. m. O uso de um *asset* na solução de diferentes problemas. (IEEE1517, 1999)
- **REUSO SISTEMÁTICO**. A prática de reuso de acordo com um processo bem definido e repetível. (IEEE1517, 1999)
- **SATISFAÇÃO DO CLIENTE**. Percepção do cliente do grau no qual os seus requisitos foram atendidos. Reclamações de cliente são indicadores usuais da baixa satisfação do cliente, porém sua ausência não implica, necessariamente, alta satisfação do cliente. Mesmo, que os requisitos tenham sido acordados com o cliente e atendidos, isto não garante, necessariamente, uma alta satisfação. (ABNT9000, 2001)
- **SISTEMA**. Uma composição integrada que consiste de um ou mais processos, *hardware*, *software*, facilidades e pessoas, que provêem uma capacidade de atender uma necessidade ou objetivo definido. (ABNT12207,1998)
- ***STAKEHOLDER***. Um grupo ou indivíduo que possui um grande interesse no sucesso do sistema de *software* que está sendo desenvolvido e possui alguma influência direta ou indireta nos requisitos do sistema. (HEINEMAN et al.,2001)

- ***TEMPLATE***. Um *asset* com parâmetros ou *slots* que podem ser usados para construir uma instância de *asset*. (IEEE1517,1999)
- **USABILIDADE**, s. f. Conjunto de atributos que evidenciam o esforço necessário para se poder utilizar o *software*, bem como o julgamento individual desse uso, por um conjunto explícito ou implícito de usuários. (ISO9126-1,2000)
- **VALIDAÇÃO**, s. f. Confirmação, por exame e fornecimento de evidência objetiva, de que os requisitos específicos para um uso pretendido são atendidos. Informações cuja veracidade pode ser comprovada com base em fatos obtidos através da observação, medição, ensaios ou outros meios constituem evidência objetiva. (ISO8402,1994)
- **VERIFICAÇÃO**, s. f. Confirmação, por exame e fornecimento de evidência objetiva, do atendimento aos requisitos especificados. Processo de avaliação de um sistema (ou componente) com o objetivo de determinar se o produto de uma dada fase do desenvolvimento satisfaz às condições impostas no início dessa fase. (ISO8402,1994)
- ***WALKTHROUGH***, s. f. Técnica de análise estática na qual um projetista ou programador apresenta aos membros do grupo de desenvolvimento e outros profissionais interessados uma parte de documentação ou código, e os participantes fazem perguntas e comentários sobre possíveis erros, violação de padrões de desenvolvimento ou sobre outros problemas. (IEEE828,1990)