MINISTÉRIO DA DEFESA
COMANDO DA AERONÁUTICA

QUALIDADE

ICA 800-2

VALIDAÇÃO DE SISTEMAS DE SOFTWARE DE COMUNICAÇÃO, NAVEGAÇÃO E VIGILÂNCIA PARA O SISCEAB

2011

MINISTÉRIO DA DEFESA
COMANDO DA AERONÁUTICA
DEPARTAMENTO DE CONTROLE DO ESPAÇO AÉREO

QUALIDADE

ICA 800-2

VALIDAÇÃO DE SISTEMAS DE SOFTWARE DE COMUNICAÇÃO, NAVEGAÇÃO E VIGILÂNCIA PARA O SISCEAB

2011

MINISTÉRIO DA DEFESA
COMANDO DA AERONÁUTICA
DEPARTAMENTO DE CONTROLE DO ESPAÇO AÉREO

PORTARIA DECEA Nº 80/DGCEA, DE 18 DE MAIO DE 2011.

Aprova a edição da Instrução Técnica para Validação de Sistemas de *Software* de Comunicação, Navegação e Vigilância para o Sistema de Controle do Espaço Aéreo.

O DIRETOR-GERAL DO DEPARTAMENTO DE CONTROLE DO ESPAÇO AÉREO, no uso das atribuições que lhe confere o art. 195, inciso IV, do Regimento Interno do Comando da Aeronáutica, aprovado pela Portaria nº 1049/GC3, de 11 de novembro de 2009, e o art. 10, inciso IV, do Regulamento do DECEA, aprovado pela Portaria nº 369/GC3, de 9 de junho de 2010, resolve:

Art. 1º Aprovar a edição da ICA 800-2 "Validação de Sistemas de *Software* de Comunicação, Navegação e Vigilância para o SISCEAB", que com esta baixa.

Art. 2º Esta Instrução entra em vigor na data de sua publicação.

Ten Brig Ar RAMON BORGES CARDOSO
Diretor-Geral do DECEA

(Publicado no BCA nº 113, de 14 de junho de 2011).

SUMÁRIO

1 DISPOSIÇÕES PRELIMINARES 9
1.1 ÂMBITO 9
1.2 ESCOPO 9
1.3 RELAÇÃO COM OUTROS DOCUMENTOS 9
1.4 COMO USAR ESTE DOCUMENTO 10
1.5 VISÃO GERAL DO DOCUMENTO 10

2 CONCEITUAÇÕES E ABREVIATURAS 12
2.1 CONCEITUAÇÕES 12
2.2 ABREVIATURAS 19

3 ASPECTOS DE SISTEMA RELACIONADOS AO DESENVOLVIMENTO DO *SOFTWARE* 20
3.1 PROCESSOS DE CICLO DE VIDA DO SISTEMA E DO *SOFTWARE* 20
3.2 INFORMAÇÕES DO PROCESSO DO SISTEMA 21
3.3 INFORMAÇÕES DO PROCESSO DO *SOFTWARE* 21
3.4 CONDIÇÃO DE FALHA E NÍVEL DE GARANTIA DO *SOFTWARE* 22
3.5 CONSIDERAÇÕES SOBRE ARQUITETURA DO SISTEMA 25
3.6 *SOFTWARE* MODIFICÁVEL E *SOFTWARE* COTS 28
3.7 CONSIDERAÇÕES DE PROJETO PARA *FIELD-LOADABLE SOFTWARES* 29
3.8 CONSIDERAÇÕES SOBRE OS REQUISITOS PARA VERIFICAÇÃO DO *SOFTWARE* 30
3.9 CONSIDERAÇÕES SOBRE O *SOFTWARE* NA VERIFICAÇÃO DO SISTEMA 30

4 CICLO DE VIDA DE *SOFTWARE* 31
4.1 PROCESSOS DE CICLO DE VIDA DO *SOFTWARE* 31
4.2 DEFINIÇÃO DE CICLO DE VIDA DO *SOFTWARE* 31
4.3 CRITÉRIO DE TRANSIÇÃO ENTRE PROCESSOS 33

5 PROCESSO DE PLANEJAMENTO DO *SOFTWARE* 34
5.1 OBJETIVOS DO PROCESSO DE PLANEJAMENTO DO *SOFTWARE* 34
5.2 ATIVIDADES DO PROCESSO DE PLANEJAMENTO DO *SOFTWARE* 34
5.3 PLANOS DO *SOFTWARE* 35
5.4 PLANEJAMENTO DO AMBIENTE DO CICLO DE VIDA DO *SOFTWARE* 36
5.5 PADRÕES DE DESENVOLVIMENTO DO *SOFTWARE* 38
5.6 PROCESSO DE REVISÃO E AVALIAÇÃO DO PLANEJAMENTO DO *SOFTWARE* 39

6 PROCESSO DE DESENVOLVIMENTO DO *SOFTWARE* 40
6.1 GENERALIDADES 40
6.2 PROCESSO DE REQUISITOS DO *SOFTWARE* 40
6.3 PROCESSO DE PROJETO DO *SOFTWARE* 41
6.4 PROCESSO DE CODIFICAÇÃO DO *SOFTWARE* 43
6.5 PROCESSO DE INTEGRAÇÃO 44

7 PROCESSO DE VERIFICAÇÃO DO *SOFTWARE* 46
7.1 GENERALIDADES 46
7.2 OBJETIVOS DO PROCESSO DE VERIFICAÇÃO DO *SOFTWARE* 46
7.3 ATIVIDADES DO PROCESSO DE VERIFICAÇÃO DO *SOFTWARE* 47
7.4 REVISÕES E ANÁLISES 48

7.5 ATIVIDADES DE TESTES 51

8 PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE* 57
8.1 OBJETIVOS DO PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE (SCM)* 57
8.2 ATIVIDADES DO PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE* 57
8.3 CATEGORIAS DE CONTROLE DE EVIDÊNCIAS 62

9 PROCESSO DE GARANTIA DA QUALIDADE DO *SOFTWARE* 64
9.1 OBJETIVOS DO PROCESSO DE GARANTIA DA QUALIDADE DO *SOFTWARE* 64
9.2 REVISÃO DE CONFORMIDADE 65

10 PROCESSO DE VALIDAÇÃO DO *SOFTWARE* 66
10.1 GENERALIDADES 66
10.2 MEIOS DE CONFORMIDADE E PLANEJAMENTO 66
10.3 SUBSTANCIAÇÃO DA CONFORMIDADE 66
10.4 EVIDÊNCIAS MÍNIMAS SUBMETIDAS À AUTORIDADE CERTIFICADORA 66
10.5 EVIDÊNCIAS RELACIONADAS AO PROJETO 67

11 EVIDÊNCIAS DO CICLO DE VIDA DO *SOFTWARE* 68
11.1 GENERALIDADES 68
11.2 PLANO PARA VALIDAÇÃO DO *SOFTWARE* 69
11.3 PLANO DE DESENVOLVIMENTO DO *SOFTWARE* 70
11.4 PLANO DE VERIFICAÇÃO DO *SOFTWARE* 70
11.5 PLANO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE* 71
11.6 PLANO DE GARANTIA DA QUALIDADE DO *SOFTWARE* 72
11.7 PADRÕES DE REQUISITOS DO *SOFTWARE* 73
11.8 PADRÕES DE PROJETO DO *SOFTWARE* 73
11.9 PADRÕES DE CODIFICAÇÃO DO *SOFTWARE* 74
11.10 REQUISITOS DO *SOFTWARE* 74
11.11 DESCRIÇÃO DO PROJETO 75
11.12 CÓDIGO-FONTE 75
11.13 CÓDIGO-EXECUTÁVEL 76
11.14 CASOS E PROCEDIMENTOS DE VERIFICAÇÃO DO *SOFTWARE* 76
11.15 RESULTADOS DA VERIFICAÇÃO DO *SOFTWARE* 76
11.16 ÍNDICE DE CONFIGURAÇÃO DO AMBIENTE DO CICLO DE VIDA DO *SOFTWARE* 76
11.17 ÍNDICE DE CONFIGURAÇÃO DO *SOFTWARE* 77
11.18 RELATÓRIO DE PROBLEMAS 77
11.19 REGISTROS DO GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE* 78
11.20 REGISTROS DE GARANTIA DA QUALIDADE DO *SOFTWARE* 78
11.21 SUMÁRIO DE REALIZAÇÕES DO *SOFTWARE* 78
11.22 DOCUMENTAÇÃO DA ADAPTAÇÃO 79

12 CONSIDERAÇÕES ADICIONAIS 80
12.1 USO DE *SOFTWARE* ANTERIORMENTE DESENVOLVIDO 80
12.2 *SOFTWARE* COMERCIAL (COTS) 82
12.3 PROCESSO DE ADAPTAÇÃO 90
12.4 QUALIFICAÇÃO DE FERRAMENTAS 92
12.5 MÉTODOS ALTERNATIVOS DE CONFORMIDADE 95

13 VISÃO GERAL DA CERTIFICAÇÃO 102
13.1 BASE DE CERTIFICAÇÃO 102
13.2 VALIDAÇÃO DO *SOFTWARE* 102
13.3 DETERMINAÇÃO DE CONFORMIDADE 102

14 DISPOSIÇÕES FINAIS 103

REFERÊNCIAS 104

Anexo A – Objetivos dos Processos e Saídas por Nível de *Software* 105

Anexo B – Objetivos Específicos aos COTS 115

1 DISPOSIÇÕES PRELIMINARES

Este documento fornece instruções para a produção de *software* utilizado em sistemas e equipamentos CNS/ATM, de forma que este *software* atenda aos requisitos de validação e cumpra a sua função com níveis previamente estabelecidos de confiabilidade e de segurança. Além disso, está baseado em normas internacionais de certificação de sistemas de *software* aeronáutico e corresponde às normas: RTCA/DO-178B, "*Software Considerations in Airborne Systems and Equipment Certification*" e RTCA DO-278, "*Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM)*". As instruções contidas no texto estão sob a forma de:

a) objetivos para os processos relacionados com o ciclo de vida de um *software*;

b) descrições de atividades e considerações de projeto visando alcançar os objetivos supramencionados; e

c) descrições de evidências que indiquem quais objetivos foram satisfeitos.

1.1 ÂMBITO

A presente Instrução aplica-se a todos os Elos do SISCEAB envolvidos nos processos de certificação de produto em conformidade ao MCA 63-4, "Procedimentos Administrativos para Homologação, Ativação e Desativação de Auxílios, Equipamentos, Sistemas e Órgãos Operacionais no Âmbito do SISCEAB".

1.2 ESCOPO

1.2.1 Este documento discute os aspectos de certificação relacionados à produção de *software* para sistemas e equipamentos CNS/ATM. O ciclo de vida do sistema e seu relacionamento com o ciclo de vida do *software* são descritos para auxiliar o entendimento do processo de validação. Não é foco deste documento uma descrição completa dos processos de ciclo de vida dos sistemas, incluindo a sua avaliação de segurança e os respectivos processos de validação, bem como a descrição dos processos de certificação de equipamentos. Esses detalhamentos podem ser obtidos na ICA 400-31, "Gerenciamento do Ciclo de Vida de Sistemas e Materiais do SISCEAB".

1.2.2 Uma vez que as questões sobre validação são discutidas somente em relação ao ciclo de vida do *software*, os aspectos atinentes à operação do *software* resultante não são discutidos.

1.2.3 Este documento não dispõe sobre a estrutura organizacional interna de entidades desenvolvedoras de produtos ou serviços que almejam tê-los validados nem sobre as relações comerciais existentes entre tais organizações e seus fornecedores, tampouco sobre a divisão contratual de responsabilidades entre elas.

1.2.4 Também estão fora do escopo deste documento os critérios para a qualificação de pessoal.

1.3 RELAÇÃO COM OUTROS DOCUMENTOS

Além dos requisitos de validação contratados, padrões nacionais e internacionais podem ser aplicáveis aos projetos, cuja conformidade poderá ser requerida. Entretanto, está fora do escopo deste documento invocar padrões nacionais ou internacionais

específicos e adequados a um dado contexto ou projeto, ou propor meios pelos quais tais padrões possam ser usados como alternativa ou suplemento a este documento.

1.4 COMO USAR ESTE DOCUMENTO

1.4.1 Os pontos a seguir devem ser considerados quando este documento for utilizado:

a) o capítulo 3 fornece as informações necessárias para entender a interação entre o ciclo de vida de um sistema e o ciclo de vida de um *software*. De forma similar, o capítulo 4 consiste em uma descrição do ciclo de vida do *software* e o capítulo 13, em uma visão geral do processo de certificação de um produto;

b) este documento é um guia para validação de *softwares* com a finalidade de compor um processo de certificação de sistema ou equipamento CNS/ATM usados em solo. Cabe notar que o processo ou metodologia para a produção de *software* sugerida por este documento não é mandatória, uma vez que se reconhece a possibilidade de haver métodos alternativos para, igualmente, alcançar-se conformidade com requisitos de segurança;

c) este documento estabelece objetivos de segurança de acordo com os níveis de garantia de *software* estabelecidos no item 3.4.4. O Anexo A orienta possíveis adequações de procedimentos, visando ao atendimento de objetivos de segurança de acordo com o nível de garantia do *software*;

d) o capítulo 11 mostra quais evidências são normalmente produzidas com vistas a auxiliar a validação de um *software*. Os nomes de tais evidências são descritos da seguinte forma: a primeira letra de cada palavra do nome em letra maiúscula. Exemplo: Código-Fonte;

e) o capítulo 12 versa sobre algumas considerações adicionais, incluindo-se instruções para o uso de *softwares* já desenvolvidos, para o processo de qualificação de ferramentas, e para o uso de métodos alternativos de verificação àqueles descritos do capítulo 3 ao 11. O capítulo 12 pode não se aplicar a todo o processo de certificação ou de homologação (MCA 63-4, de 2010, item 1.2.8); e

f) nos casos onde exemplos são usados para indicar como as instruções devem ser aplicadas, tanto graficamente quanto na forma de narrativa, tais exemplos não devem ser interpretados como o método preferencial.

1.4.2 Uma lista de itens não implica que a lista possui abordagem completa.

1.4.3 Notas são usadas ao longo deste documento como material explanatório, enfatizando um ponto ou chamando a atenção para itens relatados que não estão inteiramente dentro do contexto. Estas notas não são instruções.

1.5 VISÃO GERAL DO DOCUMENTO

A Figura 1 ilustra uma visão geral dos capítulos deste documento e o relacionamento entre elas.

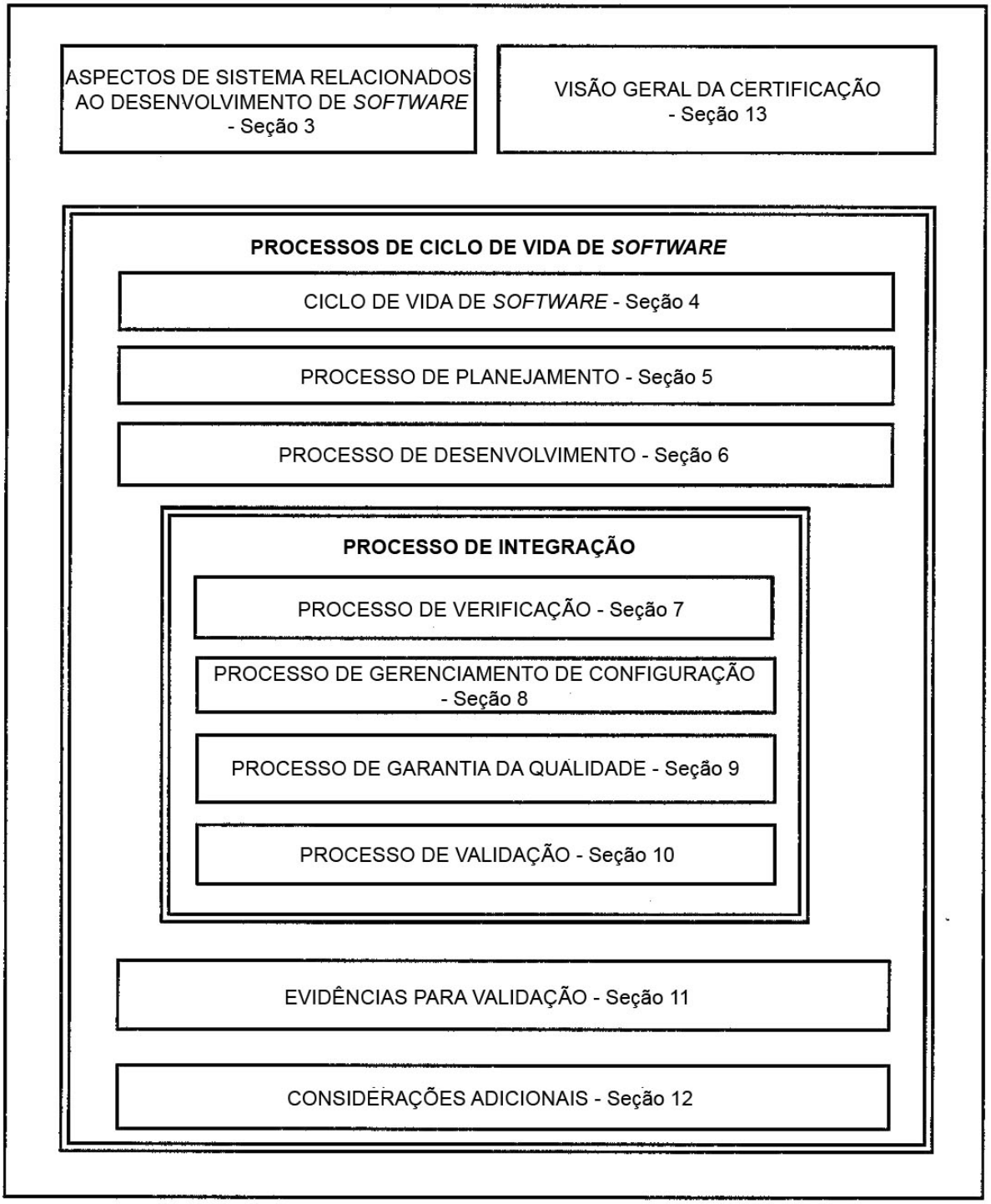

Figura 1 - Visão Geral do Documento

2 CONCEITUAÇÕES E ABREVIATURAS

2.1 CONCEITUAÇÕES

2.1.1 Acoplamento de controle: maneira ou grau pelo qual um componente de *software* influencia a execução de outro componente.

2.1.2 Acoplamento de dados: dependência do componente de *software* de dados que não estão sob seu controle exclusivo.

2.1.3 Algoritmo: conjunto finito de regras bem definidas que dá a sequência das operações para realizar uma tarefa específica.

2.1.4 Analisador estático: ferramenta de *software* que ajuda a revelar certas propriedades do programa sem a necessidade de executá-lo.

2.1.5 Análise de cobertura: processo de determinação do grau pelo qual o *software* proposto satisfaz aos objetivos do processo de verificação.

2.1.6 Arquitetura de sistema: estrutura e distribuição do *hardware* e dos componentes do *software* selecionados para atender aos requisitos do sistema.

2.1.7 Arquitetura de *software*: estrutura de *software* selecionada para atender aos requisitos de *software*.

2.1.8 Auditoria: exame independente dos processos e resultados do ciclo de vida de *software* ou de sistema para confirmar os atributos requeridos.

2.1.9 Autoridade certificadora: organização ou pessoa responsável pelo certificado de conformidade aos requisitos.

2.1.10 Avaliação de segurança: avaliação sistemática e abrangente do sistema proposto para mostrar que os requisitos de segurança relevantes foram satisfeitos.

2.1.11 Banco de dados: conjunto de dados, podendo ser parte de outro conjunto, consistindo de no mínimo um arquivo que é suficiente para um propósito ou para um sistema de processamento.

2.1.12 Base: configuração aprovada e guardada de um ou mais itens de configuração, que servirá como fundação para desenvolvimentos posteriores, e que é mudada por meio de procedimentos de controle de mudanças.

2.1.13 Biblioteca de *software*: repositório controlado contendo uma coleção de *software* e dados e documentos relacionados para ajudar no desenvolvimento, uso ou modificação de *software*. Exemplos incluem bibliotecas de desenvolvimento de *software*, bibliotecas mestre, bibliotecas de produção, bibliotecas de programas e repositório de *software*.

2.1.14 Caso de teste: conjunto de entradas, condições de execução e resultados esperados de testes desenvolvidos para um objetivo particular, bem como para exercitar um caminho particular de um programa ou verificar conformidade com um requisito específico.

2.1.15 Certificação: reconhecimento legal pela autoridade certificadora competente de que o produto, serviço, organização ou pessoa atende a todos os requisitos mínimos de desempenho para a função em tela. A certificação é composta pelas atividades de checagem técnica do produto, serviço, organização ou pessoa, e pelo reconhecimento formal, de conformidade com os requisitos aplicáveis pela emissão de certificado, licença, validação/homologação ou outro documento como requisito legal. Em particular, a certificação de produtos envolve: (a) o processo de avaliação do projeto do produto para assegurar que ele está em conformidade com um conjunto de padrões aplicáveis para aquele tipo de produto para a demonstração de um nível aceitável de segurança; (b) o processo de avaliação individual do produto para assegurar que ele está em conformidade com o certificado emitido; (c) emissão de um certificado requerido pelas leis nacionais, declarando que foi atingido o nível de conformidade esperado, de acordo com os padrões exigidos para o caso, após terem sido executados os processos previstos nos itens (a) e (b) acima.

2.1.16 Ciclo de vida de *software*: (1) coleção ordenada de processos determinados pela organização suficiente e adequada para projetar, produzir, operar, manter, substituir e descartar um produto de *software*; (2) período do tempo que começa com a decisão de produzir ou modificar um produto de *software* e termina quando o produto é retirado de serviço.

2.1.17 Classe de equivalência: subconjunto do domínio de entrada de um programa tal que o teste de um valor represente uma classe equivalente ao teste de outros valores da classe.

2.1.18 Cobertura de condição/decisão: cada ponto de entrada e saída no programa foi chamado uma vez pelo menos, cada condição numa decisão do programa foi tomada em todas as possibilidades de resultado pelo menos uma vez e cada decisão no programa foi tomada em todos os resultados possíveis pelo menos uma vez.

2.1.19 Cobertura de condição/decisão modificada: cada ponto de entrada e saída no programa foi invocado pelo menos uma vez, cada condição numa decisão no programa foi levada a todos os resultados possíveis pelo menos uma vez, cada decisão no programa foi levada a todos os resultados possíveis pelo menos uma vez e cada condição na decisão mostrou afetar de forma independente um resultado de decisão. Uma condição mostra afetar de forma independente o resultado da decisão pela variação da condição enquanto mantêm fixas todas as outras condições possíveis.

2.1.20 Cobertura de decisão: cada ponto de entrada e saída de um programa foi invocado pelo menos uma vez e cada decisão no programa foi levada a todos os resultados possíveis pelo menos uma vez.

2.1.21 Cobertura de linhas: cada linha do programa foi invocada pelo menos uma vez.

2.1.22 Código desativado: Código-Executável (ou dados) que por projeto: (a) não é passível de execução (código) ou de uso (dados), por exemplo, uma parte do componente de *software* previamente desenvolvido, ou (b) é somente executado (código) ou usado (dados) em certas configurações de ambiente do computador, como, por exemplo, código que é ativado pela seleção de um pino de *hardware* ou opção de *software* programada.

2.1.23 Código fonte: código escrito em linguagem, como a linguagem *assembly* ou outra linguagem de alto nível, na forma de entrada para um compilador ou montador.

2.1.24 Código morto: Código-Executável (ou dados) que como resultado de um erro de projeto não pode ser executado (código) ou usado (dados) na configuração operacional do ambiente do computador e não é rastreado ao sistema ou requisito de *software*. Uma exceção são os identificadores.

2.1.25 Código-Objeto: uma representação de baixo nível de um programa de computador geralmente não está na forma utilizável pelo computador, mas na forma que inclui informação de realocação e instruções para o processador.

2.1.26 Código: a implementação de um programa de computador na forma simbólica, como código fonte, Código-Objeto ou código de máquina.

2.1.27 Compilador: programa que traduz linhas do código fonte de linguagem de alto nível, como FORTRAN ou Pascal, em Código-Objeto.

2.1.28 Componente: parte ou combinação de partes, unidades ou submontagens, que realiza uma função distinta do sistema.

2.1.29 Comportamento anômalo: comportamento que não é consistente com os requisitos especificados.

2.1.30 Condição de falha: os efeitos diretos e consequentes, ou por contribuição, de uma ou mais falhas, considerando as condições adversas operacionais e ambientais. Uma condição de falha é classificada de acordo com a severidade de seus efeitos definidos nesta norma.

2.1.31 Condição: expressão booleana contendo operadores booleanos.

2.1.32 Controle de mudanças: (1) o processo de gravação, avaliação, aprovação ou reprovação e mudanças coordenadas para itens de configuração depois da formalização da identificação dos itens de configuração ou do estabelecimento de uma base; (2) avaliação sistemática, coordenação, aprovação ou reprovação e implementação de mudanças aprovadas na configuração de um item de configuração depois da formalização da identificação dos itens de configuração ou do estabelecimento de uma base. Nota: este termo pode ser chamado de controle de configuração em outros padrões.

2.1.33 COTS: *software* sob avaliação a ser utilizado em sistema ou equipamento CNS/ATM que pode não ter ou ter parcialmente evidências de conformidade com os objetivos dos processos de *software* estabelecidos por esta norma.

2.1.34 Crédito para validação: aceitação pela autoridade certificadora de que um processo, produto ou demonstração satisfaz aos requisitos de validação.

2.1.35 Critério de transição: condições mínimas, assim definidas pelo processo de planejamento do *software*, a serem satisfeitas para a entrada num processo.

2.1.36 Decisão: expressão booleana composta de condições e operadores booleanos. Uma decisão sem operador booleano é uma condição. Se a condição aparece mais de uma vez na decisão, cada ocorrência é uma condição distinta.

2.1.37 Dicionário de dados: detalhada descrição de dados, parâmetros, variáveis ou constantes usadas pelo sistema.

2.1.38 Dispositivo de memória: dispositivo de *hardware* capaz de armazenar programas de computador e dados associados.

2.1.39 Emulador: dispositivo, programa de computador ou sistema que aceita as mesmas entradas e produz as mesmas saídas que um dado sistema usando o mesmo Código-Objeto.

2.1.40 Engenharia reversa: método de extração de informação de projeto de *software* do código fonte.

2.1.41 Erro: com respeito ao *software*, um erro na escrita dos requisitos, do projeto ou do código fonte.

2.1.42 Estrutura: arranjo específico ou relação de partes para formar um todo.

2.1.43 Falha: inoperância do sistema ou do componente do sistema para realizar a função requerida dentro dos limites especificados. Uma falha pode ser produzida quando uma falta é encontrada.

2.1.44 Falta: manifestação de um erro no *software*. Uma falta, se ocorrer, pode causar uma falha.

2.1.45 Ferramenta de *software*: programa de computador usado para ajudar a desenvolver, testar, analisar, ou modificar outro programa ou sua documentação. Exemplos: ferramenta de automação de projeto, compiladores, ferramentas de testes e ferramentas de modificação.

2.1.46 Fornecedor: pessoa ou organização procurando aprovação da autoridade certificadora.

2.1.47 Garantia: ações sistemáticas e planejadas necessárias para prover confiança ou evidência de que o produto ou processo satisfaz aos requisitos.

2.1.48 Gerenciamento de configuração: (1) o processo de identificação e definição de itens de configuração de um sistema, controle de *release* e mudança de seus itens através do ciclo de vida do *software*, armazenamento, e relatório de status de itens de configuração e pedidos de mudanças e verificação de completeza e correção dos itens de configuração; (2) a disciplina aplicando direcionamento administrativo e técnico e supervisão para (a) identificar e armazenar características funcionais e físicas de um item de configuração, (b) controle de mudanças daquelas características e (c) armazenamento e relatório de controle de mudanças, incluindo processamento e status de implementação.

2.1.49 Histórico de serviço de produto: período contíguo de tempo durante o qual o *software* foi operado dentro de um ambiente conhecido e as falhas foram registradas.

2.1.50 Identificação de configuração: (1) processo de designação de itens de configuração no sistema e gravação de suas características; (2) documentação aprovada que define um item de configuração.

2.1.51 Independência: separação de responsabilidades que assegura o cumprimento do objetivo da avaliação; (1) para as atividades do processo de verificação, independência é atingida quando a atividade de verificação é realizada por pessoas diferentes das que desenvolveram os itens, ferramentas podem ser utilizadas para atingir equivalência à atividade manual de verificação; (2) para o processo de garantia da qualidade de *software*, independência inclui a autoridade para assegurar ações corretivas.

2.1.52 Integração de *software*: processo de combinação de componentes de *software*.

2.1.53 Integração *hardware/software*: o processo de combinar o *software* dentro de um computador.

2.1.54 Interrupção: suspensão de uma tarefa, como a execução de um programa de computador, causada por evento externo à tarefa e realizado de maneira que a tarefa possa vir a ser completada.

2.1.55 Item de configuração: (1) um ou mais componentes de *hardware* ou *software* tratados como uma unidade para fins de gerenciamento de configuração; (2) evidências do ciclo de vida tratadas como unidade para fins de gerenciamento de configuração.

2.1.56 *Linker*: programa de computador que combina um ou mais Códigos-Objetos em um único Código-Executável.

2.1.57 Meios de conformidade: método escolhido para ser usado pelo fornecedor para satisfazer aos requisitos da base de certificação do sistema ou equipamento. Exemplos: texto, desenhos, análises, cálculos, testes, simulações, inspeções e qualificação ambiental.

2.1.58 Métodos formais: notações descritivas ou métodos analíticos usados para construir, desenvolver e provar modelos matemáticos do comportamento de sistemas.

2.1.59 Mídia: dispositivos ou material que age como meio da transferência ou armazenamento de *software*, como, por exemplo, memórias programáveis somente leitura, fitas magnéticas ou discos, e papel.

2.1.60 Monitoramento: (1) [segurança] funcionalidade dentro de um sistema que é projetado para detectar comportamento anômalo; (2) [garantia da qualidade] ato de testemunhar ou inspecionar alguns testes, inspeções, ou outra atividade, ou registros de atividades, para assegurar que a atividade está sob controle e que os resultados reportados são representativos dos resultados esperados. O monitoramento é geralmente associado a atividades em que a ação presencial, durante todo período da atividade, é impraticável ou não necessária. O monitoramento permite assegurar que as atividades são realizadas conforme planejado.

2.1.61 Mudança de *software*: uma modificação no código fonte, Código-Objeto, Código-Executável ou documentação de sua base.

2.1.62 Número de peça (PN – *Part Number*): um conjunto de números, letras ou outros caracteres usados para identificar um item de configuração.

2.1.63 Padrão: regra ou base de comparação usada para prover guia e avaliação de desempenho de uma atividade ou conteúdo de um dado específico.

2.1.64 Partição de *software*: processo de separação, usualmente com o propósito de isolar um ou mais atributos do *software*, para prevenir interações específicas e interferência de acoplamento cruzado.

2.1.65 *Patch*: uma modificação no programa objeto, em que um ou mais passos planejados de recompilação, remontagem ou religação não são efetuados. Não inclui identificadores do código de *software*, por exemplo, número de peça ou *checksums*.

2.1.66 Procedimento de teste: instruções detalhadas para preparação e execução de um conjunto de casos de testes e instruções para avaliação dos resultados da execução dos casos de testes.

2.1.67 Processo de avaliação de segurança: atividades que demonstram conformidade com os requisitos de segurança e normas relacionadas. As principais atividades dentro desse processo são: avaliação de perigo funcional, avaliação preliminar de segurança de sistema e avaliação de segurança de sistema. O rigor das atividades depende da severidade, complexidade, inovação e experiência de serviço relevante do sistema.

2.1.68 Processo: coleção de atividades realizada no ciclo de vida do *software* para produzir uma saída definida ou produto.

2.1.69 Processos de integração: processo que assiste o desenvolvimento de *software* e a integração e por isso se mantém ativo durante todo o ciclo de vida do *software*. Os processos de integração são o processo de verificação, o processo de garantia da qualidade de *software*, o processo de gerenciamento de configuração e o processo de validação de *software*.

2.1.70 Produto de *software*: conjunto de programas de computador, e documentação e dados associados, designados para entrega ao usuário. No contexto deste documento, esse termo se refere ao *software* a ser utilizado nos sistemas e equipamentos CNS/ATM.

2.1.71 Programa de controle: programa de computador projetado para agendar e supervisionar a execução de programas em um sistema computacional.

2.1.72 Prova de correção: um argumento lógico de que o programa satisfaz aos requisitos.

2.1.73 Qualificação de ferramenta: processo necessário para obter crédito de validação para uma ferramenta de *software* dentro do contexto do sistema ou equipamento específico.

2.1.74 Qualificação: ato expressando opinião favorável ou sanção formal ou oficial.

2.1.75 Rastreabilidade: evidência da associação entre itens, como entre os resultados de processos, e entre a saída e o processo de origem, ou entre um requisito e sua implementação.

2.1.76 *Release*: ato de formalmente tornar disponível e autorizar o uso de um item de configuração recuperável.

2.1.77 Reporte de status de configuração: armazenamento e relatório de informações necessárias ao efetivo gerenciamento de configuração, incluindo a identificação de configuração aprovada, o status de mudanças propostas, a configuração e o status de implementação de mudanças aprovadas.

2.1.78 Requisito de *software*: descrição do que deve ser produzido pelo *software*, conhecidas as entradas e outras restrições. Requisitos de *software* incluem requisitos de alto nível e requisitos de baixo nível.

2.1.79 Requisitos de alto nível: requisitos de *software* desenvolvidos a partir da análise dos requisitos de sistema, requisitos de segurança e arquitetura de sistema.

2.1.80 Requisitos de baixo nível: requisitos de *software* derivados dos requisitos de alto nível, requisitos derivados e restrições de projeto do qual o código fonte pode ser diretamente implementado sem informações adicionais.

2.1.81 Requisitos derivados: requisitos adicionais resultantes do processo de desenvolvimento de *software*, que podem não estar diretamente rastreados aos requisitos de alto nível.

2.1.82 Robustez: extensão pela qual o *software* continua a operar corretamente, a despeito de entradas inválidas.

2.1.83 Simulador: dispositivo, programa de computador, ou sistema usado durante a verificação de *software*, que aceita as mesmas entradas e produz as mesmas saídas de um dado sistema, usando Código-Objeto que é derivado do Código-Objeto original.

2.1.84 Sistema: coleção de componentes de *hardware* e *software* organizados para cumprir uma função específica ou conjunto de funções.

2.1.85 *Software*: programa de computador, documentação associada e dados pertinentes à operação de um sistema de computador.

2.1.86 *Software* dissimilar múltipla-versão: conjunto de dois ou mais programas desenvolvidos separadamente para satisfazer aos mesmos requisitos funcionais. Erros específicos de uma versão são detectados pela comparação das saídas.

2.1.87 Tarefa: unidade básica de trabalho do ponto de vista de controle de programas.

2.1.88 Testes: atividade de exercício de um sistema ou componente para verificar se ele satisfaz aos requisitos especificados e detectar erros.

2.1.89 Tipo de dado: classe de dados, caracterizada pelos membros da classe e pelas operações que podem ser aplicadas. Exemplos são os tipos caractere e o tipo enumerado.

2.1.90 Tolerância a faltas: capacidade intrínseca do sistema de prover execução correta na presença de um número limitado de faltas de *software*.

2.1.91 Validação: processo para determinar se os requisitos contidos na base de certificação estão corretos e se são completos. O ciclo de vida do sistema pode usar os requisitos de *software* e requisitos derivados na validação do sistema.

2.1.92 Verificação: avaliação dos resultados de um processo para assegurar correção e consistência com respeito às entradas e aos padrões correspondentes.

2.1.93 *Watch Dog*: temporizador utilizado para reiniciar o sistema se o programa, devido à condição de erro, deixa de atualizá-lo.

2.2 ABREVIATURAS

AC	- *Advisory Circular* (Circular de Recomendação da FAA)
AL	- *Assurance Level* (Nível de Garantia de *Software* ou Nível de *Software*)
AMJ	- *Advisory Material Joint* (Material Consultivo Comum)
ATC	- *Air Traffic Control* (Controle de Tráfego Aéreo)
BIT	- *Built-In Test* (Teste Interno ao Equipamento)
CC1	- Categoria de controle 1
CC2	- Categoria de controle 2
CNS/ATM	- *Communications, Navigation and Surveillance Systems for Air Traffic Management* (Comunicação, Navegação e Vigilância para Gerenciamento do Controle de Tráfego Aéreo)
COTS	- *Commercial Off-The-Shelf* (*Software* Comercial de Prateleira)
DTCEA	- Destacamento de Controle do Espaço Aéreo
FAA	- *Federal Aviation Administration* (Agência Federal de Aviação)
FAR	- *Federal Aviation Regulations* (Normas Federais de Aviação)
FLS	- *Field-Loadable Software* (*Software* Carregável em Campo)
I/O	- *Input/Output* (Entrada/Saída)
JAA	- *Joint Aviation Authorities* (Autoridades de Aviação Comum)
JAR	- *Joint Aviation Requirements* (Requisitos de Aviação Comuns)
PN	- *Part Number* (Número/Código de Peça)
SCM	- *Software Configuration Management* (Gerenciamento de Configuração de *Software*)
SQA	- *Software Quality Assurance* (Garantia da Qualidade de *Software*)

3 ASPECTOS DE SISTEMA RELACIONADOS AO DESENVOLVIMENTO DO *SOFTWARE*

Este capítulo discute os aspectos necessários para o entendimento dos processos relacionados ao ciclo de vida do *software*. São discutidas:

a) a troca de dados entre os processos de ciclo de vida do sistema e do *software* (item 3.1);

b) a categorização das condições de falha, definição dos níveis de garantia do *software* e determinação do nível de garantia do *software* (item 3.4);

c) as considerações sobre a arquitetura do sistema (item 3.5);

d) as considerações de sistema para *software* modificável pelo usuário, *software* com opções selecionáveis e *software* COTS (item 3.6);

e) as considerações de projeto para FLS (item 3.7);

f) as considerações sobre os requisitos de sistema para verificação de *software* (item 3.8); e

g) as considerações de *software* na verificação do sistema (item 3.9).

3.1 PROCESSOS DE CICLO DE VIDA DO SISTEMA E DO *SOFTWARE*

A Figura 2 é uma visão geral dos aspectos de segurança do fluxo de informação entre os processos de ciclo de vida do sistema e os processos de ciclo de vida do *software*. Devido à interdependência entre o processo de avaliação de segurança do sistema e o processo de elaboração do sistema, o fluxo de informação descrito neste item é iterativo.

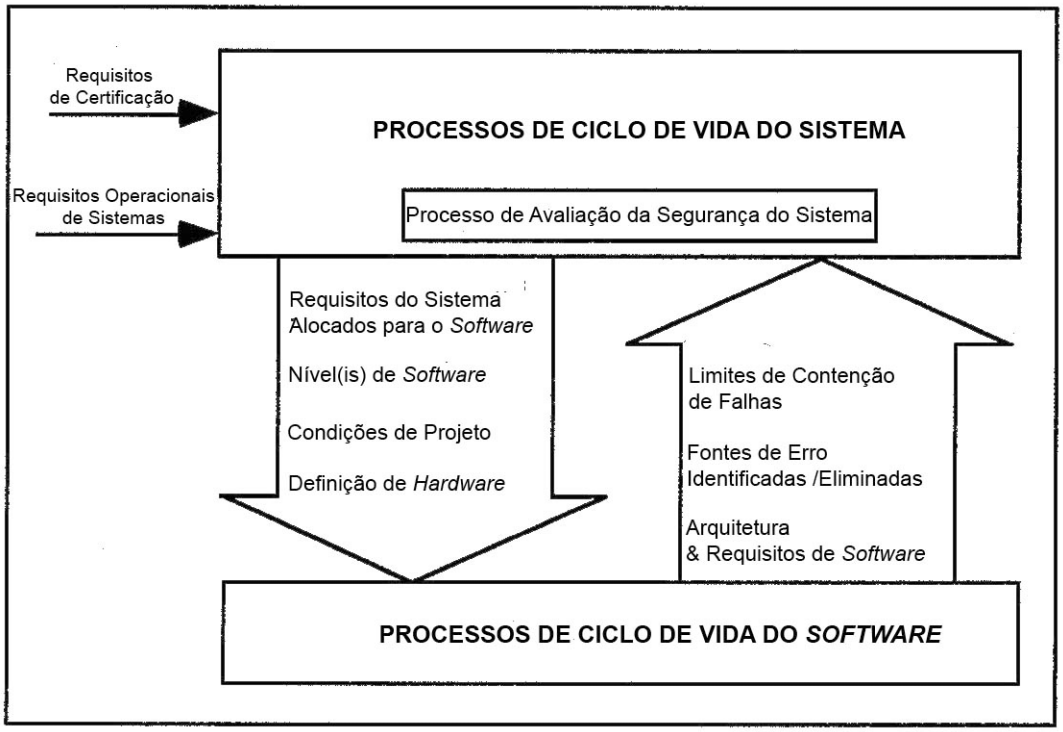

Figura 2 - Fluxo de Informação entre os Processos do Ciclo de Vida do *Software* e do Sistema, relativos à Segurança do Sistema

3.2 INFORMAÇÕES DO PROCESSO DO SISTEMA

3.2.1 O processo de avaliação de segurança do sistema identifica e classifica as condições de falha de um sistema. Dentro de uma avaliação de segurança, a análise do projeto do sistema define quais requisitos relativos à segurança especificam o nível de imunidade e as respostas desejadas para o sistema diante de determinadas condições de falha. Estes requisitos são definidos tanto para o *hardware* quanto para o *software,* visando prevenir ou limitar os efeitos das falhas e prover o sistema de capacidade de detecção e tolerância a falhas.

3.2.2 O processo de avaliação de segurança do sistema analisa se o projeto de sistema resultante satisfaz aos requisitos relacionados à segurança, à medida que as decisões de projeto são tomadas. O processo de avaliação de segurança do sistema inclui tanto o *hardware* quanto o *software*.

3.2.3 Os requisitos relativos à segurança são parte integrante dos requisitos do sistema, os quais são parâmetros de entrada dos processos de ciclo de vida do *software*. Os requisitos do sistema tipicamente fazem referência ou incluem:

a) a descrição do sistema e definições do *hardware*;

b) os requisitos de validação, incluindo as normas aplicáveis;

c) os requisitos de sistema destinados ao *software*, incluindo os requisitos funcionais, requisitos de desempenho e requisitos relacionados à segurança;

d) os níveis de garantia de *software*, condições de falha, categorias e funções alocadas ao *software*;

e) as estratégias de segurança e restrições do projeto, incluídos os métodos de projeto, tais como partição, dissimilaridade, redundância ou monitoramento de segurança; e

f) se o sistema é um componente de outro sistema, os requisitos relacionados à segurança e as condições de falha que se reportam àquele sistema maior.

3.2.4 Os processos de ciclo de vida do sistema podem especificar requisitos para os processos de ciclo de vida do *software* que auxiliam nas atividades de verificação do sistema.

3.3 INFORMAÇÕES DO PROCESSO DO *SOFTWARE*

3.3.1 Os processos de avaliação de segurança do sistema devem determinar o seu impacto no projeto e no desenvolvimento do *software*, com base nos processos de ciclo de vida do *software*. Estas informações devem incluir limites de retenção de falhas, requisitos do *software*, arquitetura do *software* e fontes de erro que possam ter sido detectadas ou eliminadas através da arquitetura do *software*, ou pelo uso de ferramentas, ou por outros métodos usados no processo para projetar o *software*. Rastrear a abrangência dos requisitos de sistema sobre as evidências de projeto do *software* é muito importante para o sucesso do processo de avaliação de segurança de um sistema.

3.3.2 Eventuais modificações na arquitetura ou no código fonte do *software* que possam afetar a segurança do sistema devem ser identificadas pelo processo de avaliação de segurança.

3.4 CONDIÇÃO DE FALHA E NÍVEL DE GARANTIA DO *SOFTWARE*

3.4.1 Este item versa sobre a definição de categorias para as condições de falha de um sistema, sobre a definição de níveis de garantia do *software*, sobre o relacionamento entre níveis de garantia do *software* e categorias das condições de falha e sobre como o nível de garantia do *software* é determinado.

3.4.2 A definição de uma categoria para uma condição de falha de um sistema é estabelecida a partir da determinação do impacto (grau de severidade) da falha sobre os usuários do sistema, decorrente de uma dada condição. Um erro no *software* pode implicar uma condição de falha para o sistema que o contém. Assim, o nível de integridade do *software* necessário e suficiente para operações seguras está relacionado às condições de falha do sistema.

3.4.3 CATEGORIZAÇÃO DA CONDIÇÃO DE FALHA

Como referência para categorização de condição de falha, utiliza-se a ICA 63-26 e as normas da FAA AC 25-1309-1A e AMJ 25-1309 da JAA. As categorias de condição de falhas listadas são definidas como:

a) catastrófica: condições de falha que levam aeronaves à colisão com outras aeronaves, com obstáculos ou com o terreno;

b) perigosa: condições de falha que levam à redução da separação entre aeronaves, de obstáculos ou com o terreno implicando erro operacional de alta severidade (Risco Crítico), ou perda total da capacidade ATC (ATC Zero);

c) maior: condições de falha que levam à redução da separação entre aeronaves, de obstáculos ou com o terreno implicando erro operacional de baixa ou moderada severidade (Risco Potencial), ou redução significativa da capacidade ATC;

d) menor: condições de falha que levam a leve redução da capacidade ATC, ou aumento significativo da carga de trabalho ATC; e

e) sem efeito: condições de falha que levam a um aumento leve na carga de trabalho ATC.

3.4.4 DEFINIÇÕES DE NÍVEL DE GARANTIA DE *SOFTWARE*

3.4.4.1 Os resultados do processo de avaliação de segurança devem ser usados para estabelecer os níveis apropriados de garantia do *software* para todos os elementos do sistema CNS/ATM. A descrição do processo de avaliação de segurança está fora do escopo deste documento, no entanto os níveis de garantia do *software* designados pelo processo são descritos abaixo:

a) Nível de Garantia 1 (AL1): *softwares* nos quais o comportamento anômalo, conforme descrição feita no processo de avaliação da segurança do sistema, causaria ou contribuiria para falhas de funções do sistema, resultando em uma condição de falha catastrófica para aeronaves;

b) Nível de Garantia 2 (AL2): *softwares* nos quais o comportamento anômalo, conforme descrição feita no processo de avaliação da segurança do sistema, causaria ou contribuiria para falhas de funções do sistema resultando em uma condição de falha perigosa para aeronaves;

c) Nível de Garantia 3 (AL3): *softwares* nos quais o comportamento anômalo, conforme descrição feita no processo de avaliação da segurança, causaria ou contribuiria para falhas nas funcionalidades do sistema resultando em uma condição de falha maior para aeronaves;

d) Nível de Garantia 4 (AL4): nível intermediário entre AL3 e AL5, não vinculado diretamente a uma condição de falha. Condição mais restrita que AL3 e menos permissiva que AL5;

e) Nível de Garantia 5 (AL5): *softwares* nos quais o comportamento anômalo, conforme descrição feita no processo de avaliação da segurança, causaria ou contribuiria para falhas nas funcionalidades do sistema resultando em uma condição de falha menor para sistema ou equipamento CNS/ATM; e

f) Nível de Garantia 6 (AL6): *softwares* nos quais o comportamento anômalo, conforme descrição feita no processo de avaliação da segurança, causaria ou contribuiria para falhas nas funcionalidades do sistema que não produziriam efeitos sobre a capacidade operacional de aeronaves ou à carga de trabalho de suas tripulações. Uma vez que o *software* tenha sido confirmado como nível AL6 pela autoridade certificadora, nenhuma das instruções deste documento é aplicável.

3.4.4.2 Existem alternativas para identificação de requisitos relacionados à segurança no processo de *software*. O usuário deste documento deve propor um processo ou metodologia para identificar os requisitos relacionados à segurança que podem afetar a garantia de *software* necessária e deve submetê-lo a aprovação da autoridade certificadora.

3.4.5 DETERMINAÇÃO DO NÍVEL DE GARANTIA DE *SOFTWARE*

3.4.5.1 Inicialmente, o processo de avaliação da segurança de um sistema determina os níveis apropriados para os componentes do *software* de um sistema particular sem considerar o projeto do sistema, após o que se verifica o impacto das falhas, para casos de perda de funcionalidades e/ou de mau funcionamento.

NOTA 1: O usuário deste documento pode considerar que a adição de funcionalidades durante futuros desenvolvimentos, assim como mudanças nos requisitos de sistema ou de *software,* pode resultar na mudança de categoria da condição de falha de um *software* para uma categoria mais severa e para um nível maior de garantia do *software*. Pode ser desejável que o desenvolvimento de um *software* seja feito para um nível maior do que aquele originalmente determinado pela avaliação de segurança do sistema. Isto se justificará sempre que se vislumbrar a possibilidade de aumento do nível de garantia de *software* em desenvolvimentos evolutivos posteriores e se verificar alguma dificuldade ou aumento de custos de desenvolvimento para a elevação posterior desse nível de garantia de *software*.

NOTA 2: Para sistemas e equipamentos CNS/ATM, cujo uso seja previsto por normas operacionais e que não afetem a segurança do controle de tráfego aéreo, o nível de garantia de *software* a ser estabelecido poderá ser menor.

3.4.5.2 Se o comportamento anômalo de um componente do *software* contribui para mais de uma condição de falha, então a categoria mais severa de condição de falha daquele componente determina o nível de garantia do *software* para aquele componente de *software*. Existem diversas estratégias de arquitetura, tais como aquelas descritas no item 3.5, as quais

durante a evolução do projeto do sistema podem resultar na revisão dos níveis de garantia do *software* em geral.

3.4.5.3 Uma funcionalidade do sistema pode ser alocada, em paralelo ou em série, para uma ou mais parcelas de componentes do *software*. Implementação paralela é aquela na qual uma função do sistema é implementada com múltiplos componentes de *software,* tais que o comportamento anômalo de mais do que um componente é requerido para produzir a condição de falha. Para uma implementação paralela, ao menos um componente de *software* terá o nível de garantia de *software* associado à categoria mais severa de condição de falha para aquela funcionalidade do sistema. O nível de garantia de *software* para os outros componentes são determinados usando a categoria de condição de falha associada com a perda daquela funcionalidade. Exemplos de tais implementações estão descritos nos itens 3.5.3, *Software* Dissimilar Múltipla-Versão, e 3.5.4, Monitoramento de segurança.

3.4.5.4 Implementação em série é aquela na qual múltiplos componentes de *software* são usados para uma funcionalidade do sistema, de modo que o comportamento anômalo de qualquer um destes componentes poderia produzir a condição de falha. Nessa implementação, todos os componentes de *software* terão o nível de garantia de *software* associado à categoria mais severa de condição de falha da funcionalidade do sistema.

3.4.5.5 O desenvolvimento de *software* para um nível de garantia de *software* não implica a atribuição de uma taxa de falhas para aquele *software*. Portanto, os níveis de garantia de *software* ou as taxas de confiabilidade do *software* baseadas nos níveis de garantia de *software* não podem ser usados para o processo de avaliação de segurança do sistema.

3.4.5.6 Cada item de um *software* conservará rigorosamente suas características funcionais e não funcionais até que seja propositadamente modificado ou atualizado. Assim, falhas de *software* redundam de erros cometidos ao longo do estabelecimento dos requisitos do projeto ou do sistema, da escrita do Código-Fonte, dos testes, da sua compilação e/ou na sua distribuição e, adicionalmente, fatores ambientais apenas interferem na confiabilidade de um componente de *software*, nos casos em que há interferência nos dados de entrada.

3.4.6 Abaixo é apresentada uma possível curva representativa da confiabilidade de *software*, segundo Pan (1999).

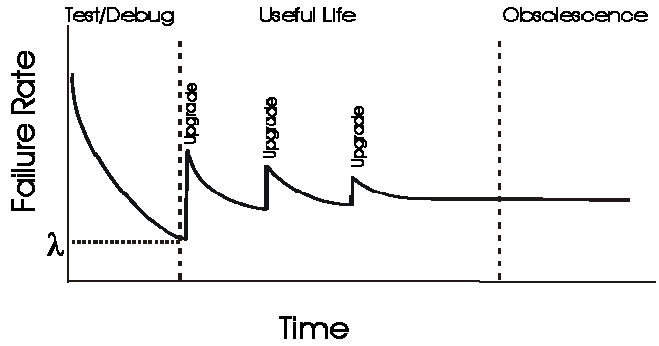

Figura 3 - Curva da banheira para confiabilidade de *software* (Fonte: Pan 1999)

3.4.7 Essa curva sugere que o *software* não tem uma taxa de falha crescente ao final de sua vida útil, e ao longo do seu ciclo de vida há picos de taxas de falha que vão diminuindo à medida que as atualizações vão sendo implementadas.

3.4.8 Por outro lado, as taxas de falha de *hardware* são definidas segundo a dedução matemática probabilística da confiabilidade, descrita abaixo:

$$R(t) = e^{-\Lambda(t)}$$

3.4.9 Essa formulação matemática constitui o modelo mais adequado para o uso em sistemas eletrônicos.

3.4.10 Da análise dessa equação, constata-se que a confiabilidade é função da variação temporal das taxas de falha Λ(t), representada pela "Curva da Banheira".

3.4.11 Na prática, os itens são considerados entregues ao cliente na condição apresentada pela Figura 4, no tempo t=0, em que Λ(t) é constante.

3.4.12 A curva representa o comportamento de confiabilidade de um item, em que a distribuição exponencial é aplicável no decorrer de sua fase operacional. A condição é que a variável aleatória Λ(t) permaneça constante até que o item falhe.

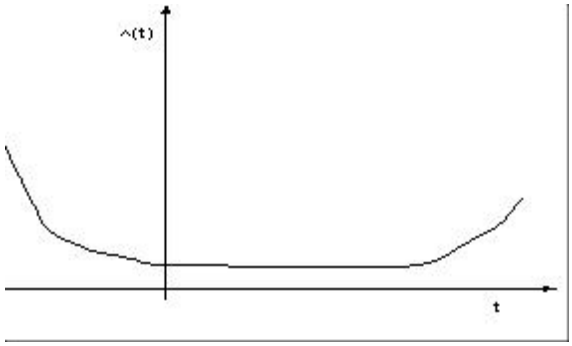

Figura 4 - Curva da banheira para itens de *hardware* (Fonte: Almeida 2001)

3.4.13 Dessa forma, o histórico de serviço deve ser fonte de consulta para definir nível de confiabilidade do *software* e contribuir para avaliação de segurança do sistema.

3.4.14 Estratégias que partam das instruções deste item (3.4.5) necessitam ser justificadas pelo processo de avaliação de segurança do sistema.

3.5 CONSIDERAÇÕES SOBRE ARQUITETURA DO SISTEMA

3.5.1 Se o processo de avaliação da segurança do sistema determina que a arquitetura do sistema evite comportamentos anômalos do *software* que contribuam para condições de falhas mais severas no sistema, então o nível do *software* é determinado pela categoria mais severa das condições de falhas remanescentes para as quais o comportamento do *software* possa contribuir. O processo de avaliação da segurança do sistema considera as decisões do projeto de arquitetura para determinar se eles afetam o nível ou a funcionalidade do *software*. Esta instrução provê várias estratégias de arquitetura que podem limitar o impacto de erros, ou detectar e prover respostas aceitáveis do sistema para controlar os erros. Estas técnicas de

arquitetura não possuem a intenção de ser interpretadas como soluções preferenciais ou requeridas.

3.5.2 PARTIÇÃO

3.5.2.1 Partição é a técnica usada para isolar componentes do *software* com funcionalidades independentes para conter e/ou isolar falhas e reduzir os esforços no processo de verificação do *software*. Se a proteção por partição é usada, o nível de garantia de *software* para cada componente com partição pode ser determinado usando a categoria de condição de falha mais severa associada àquele componente.

3.5.2.2 Instruções para partição:

a) os seguintes aspectos do sistema devem ser considerados para determinar o potencial de violação quando se opta pelo uso da proteção por partição:
   – recursos de *hardware*: processadores, dispositivos de memória, dispositivos de Entrada/Saída (I/O), interrupções e temporizadores;
   – acoplamento de controle: vulnerabilidade a acessos externos;
   – acoplamento de dados: dados compartilhados ou sobrepostos, incluindo registradores do processador e de pilha;
   – modos de falha dos dispositivos de *hardware* associados com os mecanismos de proteção;

b) os processos de ciclo de vida do *software* devem ser dirigidos às considerações do projeto de partição, incluindo a extensão e escopo de interações permitidas entre os componentes com partição, e se a proteção é implementada através de *hardware* ou por uma combinação de *hardware* e de *software*; e

c) se a proteção por partição envolve *software*, então aquele *software* deve ser associado ao nível de garantia de *software* correspondente ao nível mais alto dos componentes de *software* com partição.

3.5.3 *SOFTWARE* DISSIMILAR MÚLTIPLA-VERSÃO

3.5.3.1 *Software* Dissimilar Múltipla-Versão é uma técnica de projeto de sistema que envolve a produção de dois ou mais componentes de *software* que possuem a mesma função com o objetivo de evitar fontes de erros comuns. *Software* Dissimilar Múltipla-Versão é também conhecido como *software* de múltiplas versões, *softwares* dissimilares, programação de N versões ou diversidade de *software*.

3.5.3.2 Os produtos dos processos de ciclo de vida dos *software*s finalizados ou ativados antes da introdução da dissimilaridade precisam ser verificados. Os requisitos do sistema podem especificar configuração de *hardware* diverso que possibilite a execução de *Software* Dissimilar Múltipla-Versão.

3.5.3.3 O grau de dissimilaridade e, portanto, o grau de proteção, não é habitualmente mensurável. A probabilidade de perda de funcionalidades do sistema aumentará para fazer com que a monitoração de segurança associada a versões dissimilares do *software* detecte os erros atuais ou os transientes experimentados que excedam os limiares de comparação. Versões dissimilares do *software* são normalmente usadas, portanto, como um meio de prover proteção adicional após o processo de verificação dos objetivos do *software* para o nível de garantia de *software*, conforme descrito no capítulo 7. Métodos de verificação de

dissimilaridade do *software* podem ser reduzidos àqueles usados para verificar uma única versão do *software,* contanto que seja possível mostrar que a perda de funcionalidade é aceitável, de acordo com o processo de avaliação de segurança do sistema.

3.5.3.4 Verificação de dissimilaridade de múltiplas versões do *software* é discutida no item 12.5.7.

3.5.4 MONITORAMENTO DE SEGURANÇA

3.5.4.1 Monitoramento de segurança é um meio de proteção contra condições específicas de falhas. O monitoramento é realizado mediante a busca de falhas de uma função que contribuam para a condição de falha do sistema. O monitoramento de funções pode ser realizado em *hardware*, *software*, ou uma combinação destes.

3.5.4.2 O nível de garantia de *software* da função monitorada pode ser reduzido ao nível associado à perda da função por meio do uso de técnicas de monitoramento. Para permitir este nível de redução, existem três importantes atributos do monitor que devem ser determinados:

a) Nível de Garantia de *Software*: Monitoramento da segurança do *software* é atribuído ao nível de garantia de *software* associado à categoria de condição de falha mais severa para a função monitorada;

b) Cobertura de Falhas no Sistema: Avaliação da cobertura de falhas do sistema por um monitor, assegurando que o projeto e implementação deste monitor são tais que as falhas para as quais ele foi planejado para detectar serão detectadas de acordo com todas as condições necessárias; e

c) Independência de Função e Monitor: O monitor e os mecanismos de proteção não se tornam inoperantes pelas mesmas falhas que causam riscos.

3.5.5 SISTEMAS DE COMUNICAÇÕES

Assim como componentes de *hardware* e *software* podem se comunicar um com outro, falhas em um componente têm o potencial de se propagar para outros componentes. Análises de segurança devem considerar caminhos de dados através do sistema para assegurar que a função de um nível de garantia não corrompa os dados ou as funções associadas a outro nível de garantia. O desenvolvimento de sistemas CNS/ATM deve se preocupar com a propagação de falhas e com os caminhos de dados.

3.5.6 SEGURANÇA DA INFORMAÇÃO

Os sistemas CNS/ATM devem seguir os requisitos de segurança da informação. Conflitos potenciais entre segurança da informação e segurança operacional devem ser resolvidos no nível de sistema. Os requisitos resultantes alocados ao *software* devem ser definidos.

3.5.7 ADAPTAÇÃO

A modificação de sistemas CNS/ATM para locais específicos deve seguir as orientações constantes do processo de adaptação. O processo de adaptação é detalhado no item 12.3 deste documento.

3.5.8 *CUTOVER (HOT SWAPPING)*

3.5.8.1 Há casos em que os sistemas CNS/ATM são exigidos 24 horas por dia. Nestes casos, os sistemas necessitam de provisões para troca de componentes de *hardware* ou *software* enquanto o sistema está operacional. Isso é chamado correntemente de *cutover* ou *hot swapping*. Considerações sobre *cutover* incluem, mas não se limitam a:

a) mecanismos para instalação e *hot swapping* de *software* devem assegurar que níveis apropriados de disponibilidade e integridade sejam mantidos;

b) garantia de que os objetivos de segurança definidos no processo de avaliação de segurança não foram comprometidos durante ou depois da troca; e

c) depois da troca, a possibilidade de reversão para configuração anterior.

3.5.8.2 A compatibilidade com outros componentes e com a versão apropriada de *software* deve ser mantida.

3.6 *SOFTWARE* MODIFICÁVEL E *SOFTWARE* COTS

3.6.1 Os efeitos potenciais de modificações efetuadas pelo usuário são determinados pelo processo de avaliação da segurança do sistema e usados para desenvolver os requisitos de *software*, e desta forma as atividades do processo de verificação do *software*. O projeto de *software* modificável pelo usuário é discutido mais adiante, no item 6.3.3. Uma alteração que afete o *software* não modificável, sua proteção, ou seus limites é tratada como modificação de *software* e é discutida no item 12.1.1. Para este documento, um componente modificável é aquela parte do *software* que é planejada para ser alterada pelo usuário, e um componente não modificável é aquele que não é planejado para ser alterado pelo usuário.

3.6.2 Alguns sistemas e equipamentos podem incluir funções opcionais que podem ser selecionadas tanto pelas opções programadas pelo *software* quanto por pinos conectores do *hardware*. As funções do *software* selecionáveis por opções são usadas para escolher uma particular configuração. Veja o item 6.5.3 para instruções sobre código desativado.

3.6.3 Instruções para considerações do sistema para *software* modificável pelo usuário, *software* com opções selecionáveis e *software* COTS incluem:

a) *software* modificável pelo usuário: usuários podem modificar o *software* dentro das condições de modificação sem revisão das autoridades certificadoras, se os requisitos do sistema proporcionam modificação pelo usuário;

b) os requisitos do sistema devem especificar os mecanismos que previnam as modificações feitas pelo usuário que afetem a segurança do sistema estando ou não estas corretamente implementadas. O *software* que proporciona a proteção contra modificação de usuários deve estar no mesmo nível de garantia de *software* da função que a está protegendo de erros no componente modificável;

c) se os requisitos do sistema não incluem prescrições quanto à modificação pelos usuários, o *software* não deve ser modificado pelo usuário a menos que esteja em conformidade com a descrição deste documento para modificação;

d) no momento em que houver uma modificação pelo usuário, o mesmo deve assumir a responsabilidade sobre todos os aspectos quanto ao *software* modificado. Por exemplo, gerenciamento de configuração do *software*, garantia da qualidade do *software* e verificação do *software*;

e) *software* com opções selecionáveis: quando opções programadas do *software* estão inclusas, significa que devem ser fornecidas garantias que seleções inadvertidas não podem ser feitas;

f) *software* COTS: o *software* comercial incluso nos sistemas ou equipamentos CNS/ATM deve satisfazer aos objetivos deste documento; e

g) se deficiências existirem nas evidências de ciclo de vida dos *softwares* COTS, as evidências devem ser sanadas para satisfazer aos objetivos deste documento. As instruções nos itens 12.1.4 e 12.5.9 podem ser relevantes neste caso.

3.7 CONSIDERAÇÕES DE PROJETO PARA *FIELD-LOADABLE SOFTWARES*

3.7.1 Os FLS para sistemas CNS/ATM referem-se a *softwares* ou a tabelas de dados que podem ser carregados na memória sem a remoção do sistema ou equipamento do seu local de instalação. Os requisitos relativos à segurança associados com a função de carregamento dos dados do *software* são parte dos requisitos do sistema. Se uma habilitação inadvertida da função de carregamento de dados do *software* puder induzir ao sistema a condição de falha, um requisito relativo à segurança para a função de carregamento de dados do *software* deve ser especificado nos requisitos do sistema.

3.7.2 Considerações sobre segurança do sistema relativas à FLS incluem:

a) a detecção de *software* carregado parcialmente ou corrompido;

b) a determinação dos efeitos de carregamento do *software* de forma inapropriada;

c) a compatibilidade *hardware*/*software*;

d) a compatibilidade *software*/*software*;

e) a compatibilidade sistema/*software*;

f) o tratamento da habilitação inadvertida da função de carregamento do *software*; e

g) a perda ou a corrupção das informações na tela de identificação de configuração do *software*.

3.7.3 Instruções para FLS incluem:

a) a menos que seja justificado de outra maneira pelo processo de avaliação de segurança do sistema, para o mecanismo de detecção de *softwares* carregados parcialmente ou corrompidos deve ser designada a mesma condição de falha ou nível de garantia de *software* associada à função que utiliza o *software* carregado;

b) se um sistema possui um modo padrão (*default*) quando dados ou *software* inapropriados são carregados, então cada componente parcial do sistema deve possuir requisitos relacionados à segurança específicos para operação neste modo, os quais tratarão da condição de falha em potencial; e

c) a função de carregamento do *software*, incluindo sistemas de suporte e procedimentos, deve incluir um meio de detecção de combinações incorretas de *software* e/ou *hardware* e deve prover proteção apropriada para condições de falha dessa função.

3.8 CONSIDERAÇÕES SOBRE OS REQUISITOS PARA VERIFICAÇÃO DO *SOFTWARE*

Os requisitos do sistema são desenvolvidos a partir dos requisitos operacionais do sistema e dos requisitos relacionados à segurança, resultantes de um processo de avaliação da segurança do sistema. Estas considerações incluem:

a) os requisitos do sistema para os *softwares* de sistemas CNS/ATM, os quais estabelecem duas características do *software*:
   – o *software* executa funções específicas que são definidas pelos requisitos do sistema; e
   – o *software* não exibe comportamento(s) anômalo(s) específico(s) que esteja(m) determinado(s) no processo de segurança do sistema. Requisitos adicionais do sistema são gerados para eliminar comportamentos anômalos.

b) estes requisitos do sistema devem então ser desenvolvidos dentro dos requisitos de maior nível do *software*, os quais são verificados pelas atividades do processo de verificação do *software*.

3.9 CONSIDERAÇÕES SOBRE O *SOFTWARE* NA VERIFICAÇÃO DO SISTEMA

3.9.1 As instruções para verificação do sistema estão além do escopo deste documento Entretanto, os processos de ciclo de vida do *software* ajudam e interagem com os processos de verificação do sistema. Os detalhes do projeto do *software* dizem respeito às funcionalidades do sistema que necessitam ser disponibilizadas para auxiliar na verificação do sistema.

3.9.2 A verificação do sistema pode fornecer uma cobertura significativa da estrutura do código. A análise de cobertura dos testes de verificação do sistema pode ser usada para encontrar a abrangência dos objetivos de várias atividades de teste descritos com relação à verificação do *software*.

4 CICLO DE VIDA DE *SOFTWARE*

Este capítulo discute os processos e a definição de ciclo de vida do *software* e também os critérios de transição entre os processos de ciclo de vida do *software*. As instruções deste documento não prescrevem um ciclo de vida do *software* preferencial, mas descrevem os processos separados que abrangem a maioria dos ciclos de vida e suas interações. A separação dos processos não tem a intenção de sugerir uma estrutura para as organizações que as executam. Para cada produto de *software* o ciclo de vida do *software* é construído para incluir estes processos.

4.1 PROCESSOS DE CICLO DE VIDA DO *SOFTWARE*

4.1.1 Os processos de ciclo de vida do *software* são divididos em processo de planejamento, processos de desenvolvimento e processos de integração:

a) o processo de planejamento, descrito no capítulo 5, é o processo que define e coordena as atividades do desenvolvimento do *software* e as atividades dos processos de integração para o projeto;

b) os processos de desenvolvimento do *software*, descritos no capítulo 6, produzem o produto de *software*. São eles: processos de requisitos, de *projeto*, de codificação do *software* e o processo de integração; e

c) os processos de integração, descritos do capítulos 7 ao 10, asseguram a precisão, o controle e a confiabilidade dos processos de ciclo de vida do *software* e suas saídas. São denominados processos de integração: o processo de verificação do *software*, de gerenciamento da configuração do *software*, de garantia da qualidade do *software* e o processo de validação. É importante entender que os processos de integração são executados paralelamente aos processos de desenvolvimento *se* durante o ciclo de vida do *software*.

4.2 DEFINIÇÃO DE CICLO DE VIDA DO *SOFTWARE*

4.2.1 Um projeto define um ou mais ciclo(s) de vida do *software* a partir da escolha das atividades para cada processo, especificando uma sequência para as atividades e designando responsabilidades para estas atividades.

4.2.2 Para um projeto específico, a sequência destes processos é determinada pelos atributos do projeto, tais como funcionalidades e complexidade do sistema, complexidade e tamanho do *software*, requisitos de estabilidade, uso de resultados desenvolvidos previamente, estratégias de desenvolvimento e viabilidade de *hardware*. Os processos de desenvolvimento do *software* seguem a seguinte sequência: a elaboração dos requisitos, a definição do projeto, a codificação e a integração.

4.2.3 A Figura 5 ilustra a sequência dos processos de desenvolvimento do *software* para vários componentes de um único produto de *software* com diferentes ciclos de vida. O componente W implementa um conjunto de requisitos do sistema pelo desenvolvimento dos requisitos do *software*, usando esses requisitos para definir um projeto para o *software*, implementando esse projeto dentro de um Código-Fonte e, então, integrando o componente dentro de um *hardware*. O componente X ilustra o uso do *software* previamente desenvolvido usado em um equipamento certificado.O componente Y ilustra o uso de uma simples função com partição, que pode ser codificada diretamente a partir dos requisitos do *software*. O

componente Z ilustra o uso de uma estratégia de prototipagem. Normalmente, as metas de prototipagem são elaboradas para melhor entendimento dos requisitos do *software* e para mitigar os riscos técnicos e de desenvolvimento. Os requisitos iniciais são usados como as bases de implementação do protótipo. Este protótipo é avaliado em um ambiente representativo com a pretensão de testar o sistema em desenvolvimento. Os resultados desta avaliação são utilizados para refinar os requisitos.

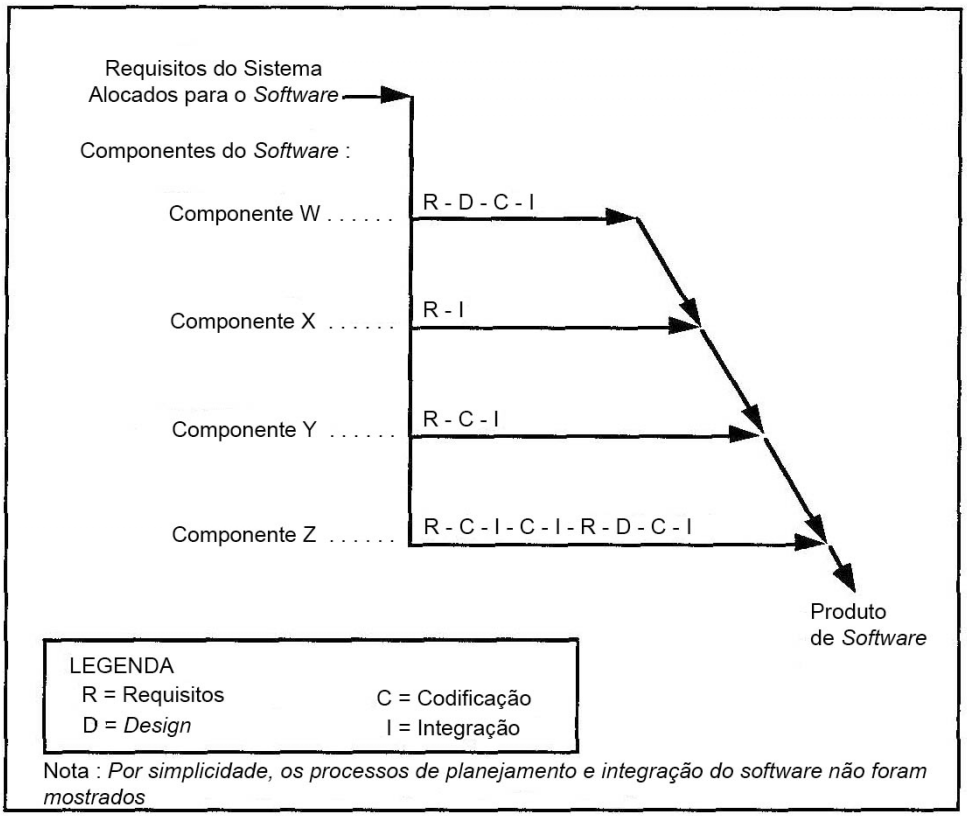

Figura 5 - Exemplo de um Projeto de Software usando Quatro Sequências diferentes de Desenvolvimento

4.2.4 Os processos de um ciclo de vida do *software* podem ser iterativos, isto é, de inserção e reinserção. A temporização e o grau das iterações variam de acordo com o desenvolvimento incremental das funções do sistema, complexidade, requisitos de desenvolvimento, viabilidade de *hardware*, realimentação de processos anteriores e outros atributos do projeto.

4.2.5 As diversas partes do ciclo de vida do *software* selecionado são vinculadas a uma combinação do processo de integração incremental e das atividades do processo de verificação do *software*.

4.3 CRITÉRIO DE TRANSIÇÃO ENTRE PROCESSOS

4.3.1 Os critérios de transição são usados para determinar se um processo pode ser iniciado ou reiniciado. Cada processo do ciclo de vida do *software* executa atividades sobre a entrada para gerar saídas. Um processo pode produzir uma realimentação para outros processos e receber a realimentação de outros processos. A definição de realimentação inclui como a informação é reconhecida, controlada e determinada na recepção. Um exemplo de definição de realimentação são os relatórios de problemas encontrados.

4.3.2 O critério de transição dependerá da sequência planejada para os processos de desenvolvimento do *software* e de integração, e poderá ser afetado pelo nível de *software*. Exemplos de critérios de transição, que podem ser escolhidos, são: revisões do processo de verificação do *software*; reconhecimento de item de configuração, ou análise de rastreamento completa para uma determinada entrada.

4.3.3 A entrada de um processo não necessita estar completa para que o processo seja iniciado, caso o critério de transição estabelecido para o processo seja satisfeito. Se um processo atua sobre entradas parciais, as entradas subsequentes do processo devem ser examinadas para determinar quais as saídas anteriores dos processos de desenvolvimento do *software* e verificação do *software* ainda são válidas.

5 PROCESSO DE PLANEJAMENTO DO *SOFTWARE*

Este capítulo discute os objetivos e atividades do processo de planejamento do *software*. Este processo produz os planos do *software* e os padrões que direcionam os processos de desenvolvimento e os processos de integração. A Tabela A-1 do Anexo A é um resumo dos objetivos e produtos do processo de planejamento do *software*, por nível de garantia do *software*.

5.1 OBJETIVOS DO PROCESSO DE PLANEJAMENTO DO *SOFTWARE*

O objetivo do processo de planejamento do *software* é definir os meios de se produzir o *software* que irá satisfazer aos requisitos do sistema e proverá o nível de confiabilidade que seja consistente com os requisitos de validação. Os objetivos do processo de planejamento de *software* são que:

a) as atividades dos processos de desenvolvimento de *software* e dos processos de integração do ciclo de vida do *software* irão tratar dos requisitos do sistema e que os níveis de garantia do *software* estejam definidos (item 3.4);

b) o(s) ciclo(s) de vida do *software*, incluindo as relações entre os processos, seus sequenciamentos, mecanismos de realimentação e critérios de transição estejam determinados (capítulo 4);

c) o ambiente do ciclo de vida do *software*, incluindo os métodos e ferramentas que serão utilizados nas atividades de cada processo do ciclo de vida, esteja selecionado (item 5.4);

d) as considerações adicionais, como as discutidas no capítulo 12, tenham sido tratadas, caso necessário;

e) os padrões de desenvolvimento de *software*, consistentes com os objetivos de segurança do sistema para o *software* a ser produzido, estejam definidos (item 5.5);

f) os planos do *software* que estejam em conformidade com o item 5.3 e que as evidências do capítulo 11 tenham sido produzidas; e

g) o desenvolvimento e a revisão dos planos do *software* estejam coordenados (item 5.3).

5.2 ATIVIDADES DO PROCESSO DE PLANEJAMENTO DO *SOFTWARE*

5.2.1 O planejamento efetivo é o fator determinante na produção do *software* que satisfaz as instruções deste documento. A orientação para o processo de planejamento do *software* inclui:

a) os planos do *software* devem ser desenvolvidos em um dado momento do ciclo de vida do *software* que forneça direção ao grupo executor dos processos de desenvolvimento e dos processos de integração. Veja também as instruções do item 10.2;

b) os padrões de desenvolvimento do *software* a serem utilizados para o projeto devem estar definidos ou selecionados;

c) os métodos e ferramentas devem ser escolhidos de forma a prevenir erros nos processos de desenvolvimento do *software*;

d) o processo de planejamento do *software* deve coordenar os processos de desenvolvimento e de integração, para prover consistência entre as estratégias dos planos do *software*;

e) o processo de planejamento do *software* deve incluir meio para revisar os planos do *software* enquanto o projeto progride;

f) quando um *software* dissimilar múltipla-versão for utilizado no sistema, o processo de planejamento deve escolher os métodos e ferramentas para atingir os níveis de detecção e prevenção de erros necessários para satisfazer aos objetivos de segurança do sistema;

g) para o processo de planejamento do *software* ser considerado completo os planos do *software* e os padrões de desenvolvimento devem estar sob controle de mudanças, e as revisões dos mesmos devem estar completas (item 5.6);

h) se um código desativado estiver planejado (item 3.6), o processo de planejamento de *software* deve descrever como o código desativado (opções selecionadas, testes de voo) será definido, verificado e tratado para atingir os objetivos de segurança do sistema; e

i) se algum código modificável pelo usuário estiver planejado, o processo, as ferramentas, o ambiente e as evidências de cumprimento das instruções do item 6.3.3 devem estar especificados nos planos do *software* e padrões.

5.2.2 Outros processos do ciclo de vida do *software* podem começar antes do processo de planejamento do *software* estar completo se os planos e procedimentos para a atividade do processo específico estiverem disponíveis.

5.3 PLANOS DO *SOFTWARE*

O propósito dos planos do *software* é definir os meios para satisfazer aos objetivos deste documento. Eles especificam as organizações que executarão essas atividades. Os planos do *software* são os seguintes:

a) o Plano para Validação do *Software* (item 11.2) serve como o meio primário para informar os métodos de desenvolvimento propostos para a autoridade certificadora, para concordância da mesma, e define a forma como esse plano está em conformidade com este documento;

b) o Plano de Desenvolvimento do *Software* (item 11.3) define o ciclo de vida do *software* e o ambiente de desenvolvimento do *software*;

c) o Plano de Verificação do *Software* (item 11.4) define o meio pelo qual os objetivos do processo de verificação do *software* serão satisfeitos;

d) o Plano de Gerenciamento da Configuração do *Software* (item 11.5) define o meio pelo qual os objetivos do processo de gerenciamento da configuração do *software* serão satisfeitos;

e) o Plano de Garantia da Qualidade do *Software* (item 11.6) define o meio pelo qual os objetivos do processo de garantia da qualidade do *software* serão satisfeitos; e

f) a orientação para os planos do *software* inclui:
   - os planos do *software* devem seguir as instruções deste documento;
   - os planos do *software* devem definir critérios de transição entre os processos de ciclo de vida do *software*, especificando:
   - as entradas do processo, incluindo a realimentação vinda de outros processos;
   - quaisquer atividades de processos de integração que possam ser requeridas para atuar sobre estas etapas;
   - a disponibilidade de ferramentas, métodos, planos e procedimentos; e
   - os planos do *software* devem especificar os procedimentos a serem usados para implementar alterações no *software* antes de usá-lo em um equipamento certificado. Tais alterações podem ser advindas da realimentação de outros processos e podem ocasionar mudança nos planos do *software*.

5.4 PLANEJAMENTO DO AMBIENTE DO CICLO DE VIDA DO *SOFTWARE*

5.4.1 O propósito do planejamento para o ambiente do ciclo de vida do *software* é definir os métodos, ferramentas, procedimentos, linguagens de programação e *hardware* que serão usados para desenvolver, verificar, controlar e produzir as evidências do ciclo de vida do *software* (capítulo 11) e produto de *software*. São exemplos de como o ambiente de *software* escolhido pode produzir efeitos benéficos sobre o *software*: a obrigatoriedade do uso de padrões, a detecção de erros e a implementação de prevenção de erros e métodos de tolerância a falhas. O ambiente do ciclo de vida do *software* é uma fonte de erros em potencial que pode contribuir para as condições de falha. A composição deste ambiente de ciclo de vida do *software* pode ser influenciada pelos requisitos relativos à segurança, determinados pelo processo de avaliação da segurança do sistema.

5.4.2 O objetivo dos métodos de prevenção de erro é o de evitar erros durante os processos de desenvolvimento do *software* que possam contribuir para uma condição de falha. O princípio básico é escolher os requisitos dos métodos de desenvolvimento e *projeto*, ferramentas e linguagens de programação que limitem a possibilidade de introdução de erros e os métodos de verificação que garantam que os erros introduzidos sejam detectados. O objetivo dos métodos de tolerância a falhas é incluir características de segurança no *projeto* do *software* ou no Código-Fonte para garantir que o *software* responderá corretamente aos erros das evidências de entrada com prevenção das respostas na saída e controle de erros. A necessidade da prevenção de erros ou métodos de tolerância a falhas é determinada pelos requisitos do sistema e pelo processo de avaliação da segurança do sistema.

5.4.3 As considerações apresentadas acima podem afetar:

a) os métodos e notações usadas no processo de requisitos do *software* e no processo de *projeto* do *software*;

b) as linguagens de programação e métodos usados no processo de codificação do *software*;

c) o ambiente de ferramentas de desenvolvimento do *software*;

d) as ferramentas de gerenciamento de configuração e verificação do *software*; e

e) a necessidade de uma qualificação da ferramenta (item 12.4).

5.4.4 AMBIENTE DE DESENVOLVIMENTO DO *SOFTWARE*

5.4.5 O ambiente de desenvolvimento do *software* constitui um fator preponderante para a produção de um *software* de alta qualidade. O ambiente de desenvolvimento do *software* pode afetar de forma adversa na produção do *software* de diversas maneiras. Por exemplo, um compilador poderia introduzir erros que acarretem em uma geração de respostas corrompidas, ou um *linker* poderia falhar para exibir um erro de alocação de memória que esteja presente. A orientação para a seleção dos métodos e ferramentas do ambiente de desenvolvimento do *software* inclui:

a) durante o processo de planejamento do *software*, o ambiente de desenvolvimento do *software* deve ser escolhido de forma a minimizar seu potencial de risco para a sua versão final;

b) o uso de ferramentas qualificadas ou combinações de ferramentas e partes do ambiente de desenvolvimento do *software* devem ser escolhidos para alcançar o nível necessário de confiabilidade, de forma que um erro introduzido por alguma parte possa ser detectado por outra. Um ambiente aceitável é produzido quando as partes são usadas juntas de forma consistente;

c) as atividades do processo de verificação do *software* ou os padrões de desenvolvimento do *software*, os quais incluem a consideração sobre o nível do *software*, devem ser definidos para minimizar os erros relacionados ao ambiente de desenvolvimento do *software*;

d) para a validação é necessário, quando do uso das ferramentas, a sequência de operação dessas seja especificada no plano apropriado; e

e) se características opcionais das ferramentas de desenvolvimento do *software* são escolhidas para o uso em um projeto, os efeitos das opções devem ser examinados e especificados no plano apropriado.

NOTA: É especialmente importante saber em que etapa a ferramenta gera diretamente a parte do produto de *software*. Neste contexto, compiladores são provavelmente as ferramentas mais importantes a serem consideradas.

5.4.6 CONSIDERAÇÕES QUANTO AO COMPILADOR E A LINGUAGEM

Em uma verificação bem sucedida do produto de *software*, o compilador é considerado aceitável para aquele produto. Para isso ser válido, as atividades do processo de verificação do *software* necessitam considerar características particulares da linguagem de programação e do compilador. O processo de planejamento do *software* considera estas características quando escolhe uma linguagem de programação e o planejamento para verificação. Esta orientação inclui:

a) alguns compiladores têm características planejadas para aperfeiçoar o desempenho do código-objeto. Se os casos de teste fornecem uma cobertura consistente com o nível de garantia do *software*, a precisão da otimização não necessita ser verificada. Por outro lado, o impacto destas características sobre a análise da cobertura estrutural deve ser determinado seguindo as instruções do item 7.5.7.2;

b) para implementar certas características, os compiladores de algumas linguagens podem produzir códigos-objeto que não são diretamente rastreáveis ao Código-Fonte, por exemplo, na inicialização, na detecção de erro integrado ou no tratamento de exceções (item 7.5.7.2, alínea "b"). O processo de planejamento do *software* deve fornecer meios de detectar este código-objeto, garantir a cobertura de verificação e definir os meios no plano apropriado; e

c) se um novo compilador, editor de ligação ou versão de carregamento na memória é introduzido, ou opções do compilador são alteradas durante o ciclo de vida do *software*, os testes ou análises de cobertura anteriores a estas alterações deverão ser revalidados. O planejamento de verificação deve prover um meio de reverificação que seja consistente com as instruções do capítulo 7 e item 12.1.3.

5.4.7 AMBIENTE DE TESTES DO *SOFTWARE*

O propósito do planejamento de teste do ambiente do *software* é definir os métodos, ferramentas, procedimentos e *hardware* que serão usados para testar a saída do processo de integração. Os testes podem ser realizados usando um computador objeto, um emulador do computador objeto ou um simulador do computador *host*. Esta orientação inclui:

a) o emulador ou simulador pode necessitar ser qualificado, conforme descrito no item 12.4; e

b) as diferenças entre o computador objeto e o emulador ou simulador, e os efeitos dessas diferenças sobre a habilidade de detectar erros e verificar a funcionalidade, devem ser consideradas. A detecção destes erros deve ser obtida mediante outras atividades do processo de verificação do *software* e especificada no Plano de Verificação do *Software*.

5.5 PADRÕES DE DESENVOLVIMENTO DO *SOFTWARE*

O propósito dos padrões de desenvolvimento do *software* é definir as regras e limitações para os processos de desenvolvimento do *software*. Os padrões de desenvolvimento do *software* incluem os Padrões de Requisitos do *Software*, os Padrões de Projeto do *Software* e os Padrões de Codificação de *Software*. O processo de verificação do *software* utiliza esses padrões como referência para avaliar a conformidade das respostas de um processo com as respostas esperadas. A orientação para padrões de desenvolvimento do *software* inclui:

a) os padrões de desenvolvimento do *software* devem estar de acordo com o capítulo 11;

b) os padrões de desenvolvimento do *software* devem habilitar componentes do *software* de um determinado produto do *software* ou associar o conjunto de produtos para ser uniformemente implementado e ter um projeto uniforme; e

c) os padrões de desenvolvimento do *software* devem proibir o uso de construtores ou métodos que produzam respostas não verificáveis, ou que não sejam compatíveis com os requisitos relacionados à segurança.

NOTA: Durante o desenvolvimento de padrões, considerações podem ser feitas de acordo com experiências anteriores. Limitações e regras dentro dos

métodos de desenvolvimento, projeto e codificação podem ser incluídas para controlar a complexidade. Práticas de programação defensiva podem ser consideradas para melhora da robustez.

5.6 PROCESSO DE REVISÃO E AVALIAÇÃO DO PLANEJAMENTO DO *SOFTWARE*

O processo de revisão e avaliação do planejamento do *software* é conduzido para garantir que os planos do *software* e os padrões de desenvolvimento do *software* estão de acordo com as instruções deste documento e com os meios que são fornecidos para executá-los. Esta orientação demanda ainda que:

a) a escolha de métodos que permitirão que os objetivos deste documento sejam alcançados;

b) os processos do ciclo de vida do *software* podem ser aplicados de forma consistente;

c) cada processo produz evidências de que suas respostas podem ser associadas a sua atividade e entradas, mostrando o grau de dependência entre a atividade, o ambiente e os métodos a serem usados; e

d) as respostas do processo de planejamento do sistema são consistentes e estão de acordo com o capítulo 11.

6 PROCESSO DE DESENVOLVIMENTO DO *SOFTWARE*

6.1 GENERALIDADES

6.1.1 Este capítulo discute os objetivos e atividades dos processos de desenvolvimento de *software*. Os processos de desenvolvimento de *software* são aplicados como definidos pelo processo de planejamento de *software* (capítulo 5) e o Plano de Desenvolvimento do *Software* (item 11.3). A Tabela A-2 do Anexo A é um resumo dos objetivos e resultados dos processos de desenvolvimento, por nível de garantia de *software*. Os processos de desenvolvimento são:

a) Processo de Requisitos de *Software;*

b) Processo de Projeto de *Software*;

c) Processo de Codificação de *Software*; e

d) Processo de Integração de *Software*.

6.1.2 Os processos de desenvolvimento de *software* produzem um ou mais níveis de requisitos de *software*. Requisitos de alto nível são produzidos diretamente a partir da análise da arquitetura e dos requisitos do sistema. Normalmente, estes requisitos de alto nível são mais bem desenvolvidos durante o processo de projeto do *software*, produzindo então um ou mais níveis sucessivos de requisitos de nível mais baixo. Entretanto, se o Código-Fonte é gerado diretamente dos requisitos de alto nível, então os requisitos de alto nível são também considerados requisitos de baixo nível, e as instruções para os requisitos de baixo nível também se aplicam.

6.1.3 O desenvolvimento de uma arquitetura de *software* envolve decisões tomadas sobre a sua estrutura. Durante o processo de projeto de *software*, a arquitetura de *software* é definida e os requisitos de baixo nível são desenvolvidos. Os requisitos de baixo nível são requisitos dos quais o Código-Fonte pode ser diretamente implementado sem informação adicional.

6.1.4 Cada um dos processos de desenvolvimento de *software* pode produzir requisitos derivados. Requisitos derivados são requisitos que não são diretamente rastreáveis para requisitos de nível mais alto. Um exemplo de requisito derivado é a necessidade do tratamento de interrupções a ser desenvolvido para o computador escolhido. Os requisitos de alto nível e os requisitos de baixo nível podem incluir requisitos derivados. Os efeitos dos requisitos derivados em requisitos relacionados à segurança são determinados pelo processo de avaliação da segurança do sistema.

6.2 PROCESSO DE REQUISITOS DO *SOFTWARE*

O processo de requisitos de *software* utiliza os resultados do processo de ciclo de vida do sistema para desenvolver os requisitos de alto nível de garantia de *software*. Estes requisitos de alto nível incluem requisitos funcionais, de desempenho, interface e relacionados à segurança.

6.2.1 OBJETIVOS DO PROCESSO DE REQUISITOS DO *SOFTWARE*

Os objetivos do processo de requisitos do *software* são garantir que:

a) os requisitos de alto nível estão desenvolvidos; e

b) os requisitos de alto nível derivados estão indicados para o processo de avaliação da segurança do sistema.

6.2.2 ATIVIDADES DO PROCESSO DE REQUISITOS DO *SOFTWARE*

6.2.2.1 As entradas para o processo de requisitos de *software* são provenientes do ciclo de vida do sistema, do Plano de Desenvolvimento do *Software* e dos Padrões de Requisitos de *Software* do processo de planejamento e incluem os requisitos do sistema, as interfaces de *hardware* e a arquitetura do sistema. Quando o critério de transição planejado para o projeto for satisfeito, estes requisitos são usados como entrada para desenvolver os requisitos de alto nível.

6.2.2.2 O resultado deste processo é o documento "Requisitos do *Software"* (item 11.10).

6.2.2.3 O processo de requisitos estará completo quando seus objetivos e os objetivos dos processos de integração associados a ele forem satisfeitos. Para a execução desse processo estão incluídas as seguintes instruções:

a) os requisitos funcionais e de interface que estão alocados ao *software* devem ser analisados para evitar ambiguidades, inconsistências e condições não definidas;

b) as entradas do processo de requisitos de *software* detectadas como não adequadas ou incorretas devem ser retornadas aos processos de origem para clarificação ou correção;

c) cada requisito do sistema que é alocado ao *software* deve estar especificado nos requisitos de alto nível;

d) os requisitos de alto nível devem estar em conformidade com os Padrões de Requisitos de *Software* e ser verificáveis e consistentes;

e) os requisitos de alto nível devem conter termos quantitativos com tolerâncias, quando aplicável;

f) os requisitos de alto nível não devem descrever detalhes de projeto ou verificação, exceto para fatores limitantes de projeto especificados e justificados;

g) cada um dos requisitos de sistema alocados ao *software* deve ser rastreável para um ou mais requisitos de *software* alto nível;

h) cada um dos requisitos de alto nível deve ser rastreado para um ou mais requisitos do sistema, exceto para requisitos derivados; e

i) requisitos de alto nível derivados devem ser disponibilizados para o processo de avaliação da segurança do sistema.

6.3 PROCESSO DE PROJETO DO *SOFTWARE*

Os requisitos de alto nível de garantia de *software* são refinados por meio de uma ou mais iterações no processo de projeto para desenvolver a arquitetura e os requisitos de baixo nível que podem ser utilizados para escrever o Código-Fonte.

6.3.1 OBJETIVOS DO PROCESSO DE PROJETO DO *SOFTWARE*

Os objetivos do processo de projeto do *software* se prestam para assegurar que:

a) a arquitetura de *software* e os requisitos de baixo nível sejam desenvolvidos a partir dos requisitos de alto nível; e

b) os requisitos de baixo nível, derivados dos primeiros, também sejam disponibilizados para o processo de avaliação da segurança do sistema.

6.3.2 ATIVIDADES DO PROCESSO DE PROJETO DO *SOFTWARE*

6.3.2.1 São entradas para o Processo de Projeto de *Software*: os Requisitos do *Software*, o Plano de Desenvolvimento do *Software* e os Padrões de Projeto do *Software*. Quando o critério de transição planejado for satisfeito, os requisitos de alto nível são utilizados no processo de projeto para desenvolver a arquitetura e os requisitos de baixo nível. Isto pode resultar em uma ou mais iterações para a produção de requisitos de nível suficientemente baixo.

6.3.2.2 A saída para este processo é a Descrição do Projeto (item 11.11) que, por sua vez, inclui a arquitetura de *software* e os requisitos de baixo nível.

6.3.2.3 O processo de projeto do *software* estará completo sempre que os seus objetivos e os objetivos dos processos de integração associados a ele forem satisfeitos. As instruções para este processo incluem:

a) os requisitos de baixo nível e a arquitetura desenvolvida durante o processo de projeto devem estar em conformidade com os Padrões de Projeto de *Software* que devem ser rastreáveis, verificáveis e consistentes;

b) requisitos derivados devem estar definidos e devem ser analisados para garantir que não tenha havido comprometimento dos requisitos de alto nível;

c) as atividades do processo de projeto podem introduzir possíveis modos de falha no *software* ou obstruir outros. O uso de partição ou outra arquitetura no projeto do *software* pode alterar o nível de garantia de *software* atribuído a alguns componentes do *software*. Nestes casos, dados adicionais devem ser definidos como requisitos derivados e devem ser disponibilizados para o processo de avaliação da segurança do sistema;

d) o fluxo de controle e o fluxo de dados devem ser monitorados quando os requisitos de segurança assim o exigirem. Por exemplo, devem ser monitorados temporizadores de *watchdog*, verificações de razoabilidade e comparações entre canais;

e) as respostas às condições de falhas devem estar consistentes com os requisitos de segurança; e

f) as entradas inadequadas ou incorretas, detectadas durante o processo de projeto do *software*, devem ser disponibilizadas tanto para o Processo de Ciclo de Vida do Sistema quanto para os processos de requisitos de sistemas e o de Planejamento do *Software*, bem como para clarificação ou correção.

NOTA: O estado corrente da engenharia de *software* não permite uma correlação quantitativa entre complexidade e o cumprimento dos objetivos de segurança. Enquanto nenhuma instrução objetiva pode ser disponibilizada, o Processo de Projeto de *Software* deve evitar introduzir complexidade, pois à medida que a complexidade do *software* aumenta, ela torna mais difícil a verificação do projeto e a demonstração da satisfação dos objetivos de segurança do *software*.

6.3.3 PROJETO PARA *SOFTWARE* MODIFICÁVEL PELO USUÁRIO

6.3.3.1 As instruções desse item se aplicam para o desenvolvimento de *software* modificável pelos usuários. Componente modificável é a parte do *software* que é feita para ser modificada pelo usuário. *Software* modificável pelo usuário pode possuir diferentes níveis de complexidade. Exemplos: um bit de memória usado para selecionar uma de duas opções de equipamento, uma tabela de mensagens, ou uma área de memória que pode ser programada, compilada e ligada para funções de manutenção. *Software*s de qualquer nível podem incluir um componente modificável.

6.3.3.2 As instruções para desenvolver *software* modificável pelo usuário devem garantir que:

a) o componente não modificável deve ser protegido do modificável para prevenir interferências na operação segura do componente não modificável. Esta proteção pode ser reforçada por *hardware*, *software*, ferramentas utilizadas para realizar a mudança, ou por uma combinação dos três métodos; e

b) deve haver evidências de que o meio disponibilizado pelo solicitante para efetuar a mudança é o único meio pela qual a mudança pode ser feita.

6.4 PROCESSO DE CODIFICAÇÃO DO *SOFTWARE*

No processo de codificação do *software*, o Código-Fonte é implementado a partir da arquitetura de *software* e dos requisitos de baixo nível.

6.4.1 OBJETIVOS DO PROCESSO DE CODIFICAÇÃO DO *SOFTWARE*

O objetivo do processo de codificação do *software* é que o Código-Fonte seja desenvolvido de forma rastreável, verificável, consistente e implementando corretamente os requisitos de baixo nível.

6.4.2 ATIVIDADES DO PROCESSO DE CODIFICAÇÃO DO *SOFTWARE*

6.4.2.1 As entradas do processo de codificação são: os requisitos de baixo nível e a arquitetura de *software,* advindos do processo de projeto de *software*, do Plano de Desenvolvimento do *Software* e dos Padrões de Codificação do *Software*. O processo de codificação do *software* pode ser iniciado ou reiniciado quando o critério de transição planejado for satisfeito. O Código-Fonte deve ser produzido através deste processo com base na arquitetura e nos requisitos de baixo nível do sistema.

6.4.2.2 Os principais resultados deste processo são o Código-Fonte (item 11.12) e o Código-Objeto.

6.4.2.3 O processo de codificação do *software* está completo quando seus objetivos e os objetivos dos processos de integração associados a ele estiverem satisfeitos. As instruções para este processo incluem as seguintes condições:

a) o Código-Fonte deve implementar os requisitos de baixo nível e de acordo com a arquitetura de *software*;

b) o Código-Fonte deve estar de acordo com os Padrões de Codificação do *Software* (item 11.9);

c) o Código-Fonte deve ser rastreado à Descrição do Projeto (item 11.11); e

d) as entradas inadequadas ou incorretas, detectadas durante o processo de codificação do *software*, devem ser disponibilizadas para o processo de requisitos do *software*, o processo de projeto do *software* e o processo de planejamento de *software*, para clarificação ou correção.

6.5 PROCESSO DE INTEGRAÇÃO

O computador, o Código-Fonte e o Código-Objeto do processo de codificação são usados com os dados de ligação e carregamento (item 11.17) no processo de integração para desenvolver o sistema ou equipamento CNS/ATM.

6.5.1 OBJETIVOS DO PROCESSO DE INTEGRAÇÃO

O objetivo do processo de integração é que Código-Executável é carregado no *hardware* para integração do *hardware*/*software*.

6.5.2 ATIVIDADES DO PROCESSO DE INTEGRAÇÃO

6.5.2.1 O Processo de Integração consiste na integração do *software* e na integração do *hardware*/*software*.

6.5.2.2 O Processo de Integração pode ser iniciado ou reiniciado quando o critério de transição planejado for satisfeito. As entradas para o Processo de Integração são a arquitetura do *software* (advinda do processo de projeto) e o Código-Fonte juntamente com o Código-Objeto (advindos do processo de codificação).

6.5.2.3 O resultado do Processo de Integração é o Código-Executável, como definido no item 11.13. O Processo de Codificação do *Software* está completo quando seus objetivos e os objetivos dos processos de integração associados a ele estão satisfeitos. As instruções para este processo incluem as seguintes condições:

a) o Código-Executável deve ser gerado a partir do Código-Fonte e dos dados de ligação e carregamento;

b) o *software* deve estar carregado no computador para a integração de *hardware*/*software*; e

c) as entradas inadequadas ou incorretas, detectadas durante o processo de integração, devem ser disponibilizadas para o processo de requisitos do *software*, o Processo de Projeto do *Software*, o Processo de Codificação do *Software* ou o Processo de Planejamento para clarificação ou correção.

6.5.3 CONSIDERAÇÕES DE INTEGRAÇÃO

6.5.3.1 As considerações a seguir são para código desativado ou *patches*. Um sistema ou equipamento CNS/ATM pode ser desenvolvido com várias configurações disponíveis, porém nem todas as configurações são aplicáveis a qualquer situação. Isto pode levar à desativação de Código-Fonte, ou a não utilização de dados. Código desativado é diferente de código morto, que está definido no glossário e discutido no item 7.5.7.3. As instruções para código desativado e *patches* incluem as seguintes condições:

a) deve haver evidências de que o código desativado não está ativo para os ambientes para os quais não foi projetado. A ativação não intencional de código desativado devido a condições anormais do sistema representa o mesmo risco que a ativação não intencional de código ativo;

b) os métodos para tratar o código desativado devem estar em conformidade com os planos do *software*;

c) não devem ser utilizados *patches* em *softwares* embarcados em equipamentos certificados, quando for necessário implementar mudanças de requisitos ou de projeto, ou mudanças exigidas por um processo de verificação. Somente em algumas situações, como, por exemplo, para resolver deficiências conhecidas do ambiente de desenvolvimento como um problema de compilador, os *patches* podem ser aplicados; e

d) quando um *patch* é usado, os itens a seguir devem estar disponíveis:
   - confirmação de que o gerenciamento de configuração pode rastrear corretamente o *patch*;
   - análise de regressão para evidenciar que o *patch* satisfaz a todos os objetivos do *software* desenvolvido pelos métodos normais;
   - justificativa para o uso do *patch* no Sumário de Realizações de *Software*.

6.5.4 RASTREABILIDADE

6.5.4.1 As instruções para rastreabilidade incluem:

a) rastreabilidade entre os requisitos do sistema e os requisitos de *software* deve ser disponibilizada para permitir a verificação da completa implementação dos requisitos do sistema e dar visibilidade aos requisitos derivados;

b) rastreabilidade entre os requisitos de baixo nível e os requisitos de alto nível deve ser disponibilizada para dar visibilidade aos requisitos derivados e às decisões de projeto arquiteturais tomadas durante o processo de projeto de *software*, bem como permitir a verificação da implementação completa dos requisitos de alto nível; e

c) rastreabilidade entre o Código-Fonte e os requisitos de baixo nível deve ser disponibilizado para possibilitar a verificação da inexistência de Código-Fonte não documentado e verificação da implementação completa dos requisitos de baixo nível.

7 PROCESSO DE VERIFICAÇÃO DO *SOFTWARE*

7.1 GENERALIDADES

7.1.1 Este capítulo discute os objetivos e as atividades do processo de verificação de *software*. A verificação é a análise dos resultados dos processos de desenvolvimento e de verificação. O processo de verificação é aplicado como definido pelo processo de planejamento (capítulo 5) e pelo Plano de Verificação do *Software* (item 11.4).

7.1.2 Verificação não é simplesmente testar. Em geral, testar não pode mostrar ausência de erros, assim, o termo "verificar" é usado em vez de "testar", dessa maneira os objetivos do processo de verificação são discutidos tipicamente como a combinação de revisões, análises e testes.

7.1.3 As tabelas A-3 até A-7 do anexo A contém um resumo dos objetivos e resultados do processo de verificação por nível de garantia de segurança.

NOTA: Para níveis mais baixos de garantia de *software* uma ênfase menor pode ser dada nos seguintes itens relacionados a seguir:

a) verificação de requisitos de baixo nível;

b) verificação da arquitetura;

c) grau de cobertura dos testes;

d) controle de verificações;

e) independência nas atividades de verificação;

f) sobreposição de atividades de verificação (múltiplas atividades de verificação em paralelo), em que o mesmo tipo de erro pode ser detectado por mais de uma atividade;

g) testes de robustez; e

h) atividades de verificação de efeito indireto à prevenção ou à detecção, por exemplo, conformidade aos padrões de desenvolvimento.

7.2 OBJETIVOS DO PROCESSO DE VERIFICAÇÃO DO *SOFTWARE*

O propósito do processo de verificação é detectar e reportar erros que podem ser introduzidos através do processo de desenvolvimento. Remoção de erros é uma atividade do processo de desenvolvimento. Os objetivos gerais do processo de verificação são:

a) os requisitos alocados para o *software* podem ser desenvolvidos em requisitos de alto nível para satisfazer aos requisitos de sistemas;

b) os requisitos de alto nível podem ser desenvolvidos em uma arquitetura de *software* e requisitos de baixo nível que satisfaçam aos requisitos de alto nível. Se um ou mais níveis de requisitos são desenvolvidos entre alto e baixo nível, os sucessivos níveis de requisitos são desenvolvidos para que a camada de nível inferior satisfaça aos requisitos da camada superior. Se código é gerado diretamente dos requisitos de alto nível, esse objetivo não se aplica;

c) a arquitetura e os requisitos de baixo nível são desenvolvidos em código que satisfaz aos requisitos de baixo nível e à arquitetura de *software*;

d) o código executável satisfaz aos requisitos; e

e) os meios para se satisfazer a esses objetivos são tecnicamente corretos e completos para cada nível de garantia de *software*.

7.3 ATIVIDADES DO PROCESSO DE VERIFICAÇÃO DO *SOFTWARE*

7.3.1 Os objetivos do processo de verificação são satisfeitos através da combinação de revisões, análises, desenvolvimento de casos de testes e procedimentos, subsequente execução dos procedimentos de testes. Revisões e análises provêm uma verificação da acurácia, completeza e facilidade de verificação dos requisitos de *software*, arquitetura, e código. O desenvolvimento de casos de testes serve para prover meios para análise de consistência e completeza dos requisitos. A execução dos procedimentos de testes provê demonstração de conformidade aos requisitos.

7.3.2 As entradas do processo de verificação incluem os requisitos do sistema, os requisitos de *software* e arquitetura, dados de rastreabilidade de requisitos, Código-Fonte, Código-Executável e Plano para Verificação de *Software*.

7.3.3 As saídas do processo de verificação são consolidadas nos documentos: Procedimentos e Casos de Teste e Resultados da Verificação.

7.3.4 A necessidade de que os requisitos sejam verificáveis, uma vez que estejam implementados no *software*, por si mesma, gera requisitos adicionais e restrições ao processo de desenvolvimento.

7.3.5 O processo de verificação provê rastreabilidade entre a implementação dos requisitos de *software* e a verificação dos mesmos:

a) a rastreabilidade entre os requisitos de *software* e os casos de testes é obtida através de análise da cobertura dos requisitos; e

b) a rastreabilidade entre a estrutura de código e os casos de testes é obtida através da análise de cobertura estrutural.

7.3.6 As atividades de verificação incluem:

a) requisitos de alto nível e sua rastreabilidade para sua implementação;

b) resultados das análises de rastreabilidade, análises de cobertura de requisitos e cobertura estrutural devem mostrar que cada requisito de *software* é rastreado para o código que o implementa, o que é verificado por revisões, análises e casos de testes;

c) se o código testado não é idêntico ao *software* CNS/ATM carregado no sistema, as diferenças devem ser especificadas e justificadas;

d) quando não é possível verificar os requisitos de *software* num ambiente de testes, outros meios podem ser usados desde que justificados apropriadamente e formalmente descritos nos documentos: Plano para Verificação e Resultados da Verificação; e

e) deficiências ou erros descobertos durante o Processo de Verificação devem ser reportados para o processo de desenvolvimento para clarificação e correção.

7.4 REVISÕES E ANÁLISES

Revisões e Análises são aplicadas aos resultados do Processo de Desenvolvimento e Processo de Verificação. Há distinção entre revisões e análises: as análises provêm evidência repetitiva para correção e as revisões provêm uma avaliação da correção. Uma revisão pode consistir de uma inspeção da saída do processo guiada por um *checklist* ou ajuda similar. Uma análise pode examinar em detalhes a funcionalidade, performance, rastreabilidade de requisitos e implicações de segurança de um componente de *software*, assim como a sua relação com os outros componentes dentro do sistema ou do equipamento.

7.4.1 REVISÕES E ANÁLISES DE REQUISITOS DE ALTO NÍVEL

O objetivo dessas revisões e análises é detectar e reportar os erros nos requisitos introduzidos durante o processo de sua elaboração. As revisões e análises podem confirmar que os requisitos de alto nível satisfaçam aos objetivos listados e descritos a seguir:

a) conformidade com os requisitos de sistema: o objetivo é assegurar que as funções do sistema, realizadas pelo *software*, estejam definidas, que os requisitos funcionais, de desempenho e de segurança sejam satisfeitos pelo nível de garantia de *software*;

b) acurácia e consistência: o objetivo é assegurar que cada requisito de alto nível tenha acurácia, não seja ambíguo e seja suficientemente detalhado e que os requisitos em geral não sejam conflitantes;

c) compatibilidade com o computador: o objetivo é assegurar que não existam conflitos entre os requisitos de alto nível e as características de *hardware*/*software* do computador, de modo particular, assegurar compatibilidade entre os tempos de resposta e as entradas e saídas de *hardware* requeridas pelo sistema e as características do *hardware*;

d) verificabilidade: o objetivo é assegurar que os requisitos de alto nível possam ser verificados;

e) conformidade com os padrões: o objetivo é assegurar que os padrões de requisitos de *software* sejam seguidos durante o processo de requisitos de *software* e que os desvios do padrão sejam justificados;

f) rastreabilidade: o objetivo é assegurar que os requisitos funcionais, de performance e de segurança sejam desenvolvidos para obter os requisitos de *software* de alto nível; e

g) aspectos relacionados com os algoritmos: o objetivo é assegurar a acurácia e o comportamento dos algoritmos propostos em relação aos requisitos aprovados para o projeto. Observar, particularmente, os casos de descontinuidade.

7.4.2 REVISÕES E ANÁLISES DE REQUISITOS DE BAIXO NÍVEL

O objetivo dessas revisões e análises é detectar e reportar os erros nos requisitos que podem ser introduzidos durante o Processo de Projeto. As revisões e análises podem confirmar que os requisitos de baixo nível satisfaçam aos objetivos listados e descritos a seguir:

a) conformidade aos requisitos de alto nível: o objetivo é assegurar que os requisitos de baixo nível satisfaçam aos requisitos de *software* de alto nível e que a base de projeto, a arquitetura e os requisitos de mais baixo nível derivados sejam corretamente definidos;

b) acurácia e consistência: o objetivo é assegurar que cada requisito de baixo nível tenha acurácia e não seja ambíguo e que os requisitos em geral não sejam conflitantes;

c) compatibilidade com o computador: o objetivo é assegurar que não existam conflitos entre os requisitos de baixo nível e as características de *hardware*/*software* do computador, de modo particular, os aspectos relativos ao uso de recursos (como sobrecarga de barramento), aos tempos de resposta do sistema e às entradas e saídas de *hardware*;

d) verificabilidade: o objetivo é assegurar que os requisitos de baixo nível possam ser verificados;

e) conformidade com os padrões: o objetivo é assegurar que os padrões de projeto de *software* sejam seguidos durante o processo de projeto de *software* e que os desvios do padrão sejam justificados;

f) rastreabilidade: o objetivo é assegurar que os requisitos de alto nível e requisitos derivados sejam desenvolvidos para obter os requisitos de *software* de baixo nível; e

g) aspectos relacionados com os algoritmos: o objetivo é assegurar a acurácia e o comportamento dos algoritmos propostos em relação aos requisitos aprovados para o projeto. Observar, particularmente, os casos de descontinuidade.

7.4.3 REVISÕES E ANÁLISES DA ARQUITETURA DE *SOFTWARE*

O objetivo dessas revisões e análises é detectar e reportar os erros nos requisitos que podem ser introduzidos durante o processo de elaboração da arquitetura de *software*. As revisões e análises podem confirmar que os requisitos de baixo nível satisfaçam aos objetivos listados e descritos a seguir:

a) compatibilidade com os requisitos de alto nível: o objetivo é assegurar que a arquitetura de *software* não seja conflitante com os requisitos de *software* de alto nível. Especialmente as funções que garantam a integridade, por exemplo, os esquemas de partição;

b) consistência: o objetivo é assegurar que a relação existente entre os componentes da arquitetura de *software* esteja em conformidade com o projeto. Essa relação existe via diagramas fluxo de dados e fluxo de controle;

c) compatibilidade com o computador: o objetivo é assegurar que não existam conflitos, especialmente na inicialização, operação assíncrona, sincronização e interrupções, entre a arquitetura de *software* e as características de *hardware*/*software* do computador;

d) verificabilidade: o objetivo é assegurar que a arquitetura de *software* possa ser verificada para que, por exemplo, não haja algoritmos recursivos ilimitados;

e) conformidade com os padrões: o objetivo é assegurar que os padrões de projeto de *software* sejam seguidos durante o processo de projeto de *software* e que os desvios do padrão sejam justificados, especialmente quanto às restrições de complexidade e construções de projeto que não atendam aos objetivos de segurança; e

f) integridade da partição: o objetivo é assegurar que sejam evitadas brechas na partição.

7.4.4 REVISÕES E ANÁLISES DO CÓDIGO-FONTE

O objetivo dessas revisões e análises é detectar e reportar os erros nos requisitos que podem ser introduzidos durante o processo de codificação. As revisões e análises podem confirmar que os produtos desse processo são acurados, completos e podem ser verificados. A atenção é voltada para a compatibilidade dos requisitos com a arquitetura e conformidade com os padrões de codificação adotados. As revisões e análises se restringem ao código e podem incluir:

a) compatibilidade com os requisitos de baixo nível: o objetivo é assegurar que o Código-Fonte é acurado e completo com respeito aos requisitos de baixo nível e que o Código-Fonte não implementa uma função não documentada;

b) compatibilidade com a arquitetura: o objetivo é assegurar que o Código-Fonte corresponde ao fluxo de dados e fluxo de controle definidos na arquitetura de *software*;

c) verificabilidade: o objetivo é assegurar que o Código-Fonte não contém instruções e estruturas que não possam ser verificadas e que o código não precise ser alterado para realização de testes;

d) conformidade com os padrões: o objetivo é assegurar que os padrões de codificação do *software* foram seguidos durante o desenvolvimento de código, especialmente as restrições de complexidade e as checagens de código que devem ser consistentes com os objetivos de segurança. Complexidade inclui o grau de acoplamento entre os componentes de *software*, os níveis de estruturas aninhadas de controle e a complexidade de expressões lógicas ou numéricas. Essa análise também assegura que os desvios do padrão foram justificados;

e) rastreabilidade: o objetivo é assegurar que os requisitos de baixo nível foram desenvolvidos no Código-Fonte; e

f) acurácia e consistência: o objetivo é determinar a correção e consistência do Código-Fonte, incluindo o uso de pilha, resolução e *overflow* de aritmética de ponto-flutuante, contenção de recursos, tempo de execução do pior caso, tratamento de exceções, uso de variáveis ou constantes não inicializadas, corrupção de dados devido a conflito de tarefas ou interrupções.

7.4.5 REVISÕES E ANÁLISES DOS PRODUTOS DO PROCESSO DE INTEGRAÇÃO

O objetivo dessas revisões e análises é assegurar que o processo de integração é completo e correto. Isso pode ser feito através do exame detalhado com a ligação e carregamento de dados e do mapa de memória. Verificando:

a) endereçamento incorreto de *hardware;*

b) sobreposição de memória; e

c) falta de componentes de *software.*

7.4.6 REVISÕES E ANÁLISES DE PROCEDIMENTOS, CASOS E RESULTADOS DE TESTES

O objetivo dessas revisões e análises é assegurar que o teste do código é feito de modo acurado e completo. Isso pode incluir:

a) casos de testes: a verificação de casos de testes é apresentada no item 7.5.7;

b) procedimentos de testes: o objetivo é verificar que os casos de testes são precisamente desenvolvidos em procedimentos de testes e resultados esperados; e

c) resultados de testes: o objetivo é assegurar que os resultados dos testes são corretos e que as discrepâncias entre resultados obtidos e esperados sejam explicados.

7.5 ATIVIDADES DE TESTES

7.5.1 Os testes têm dois objetivos complementares. Um dos objetivos é demonstrar que o *software* satisfaz aos requisitos. O segundo é demonstrar com alto grau de confiabilidade que os erros que podem levar a condições inaceitáveis de segurança foram removidos.

7.5.2 A Figura 6 é o diagrama de atividades de testes. Os três tipos de testes e seus objetivos são:

a) Testes de Integração de *Hardware*/*Software*: para verificar a correta operação do *software* no computador;

b) Testes de Integração de *Software*: para verificar as relações entre os requisitos de *software* e os componentes e para verificar a implementação dos requisitos e componentes dentro da arquitetura de *software; e*

c) Testes de Baixo Nível: para verificar a implementação dos requisitos de baixo nível.

7.5.3 Para satisfazer aos objetivos de testes de *software*:

a) os casos de testes devem ser baseados primariamente em requisitos de *software;*

b) os casos de testes devem ser desenvolvidos para verificar a correta funcionalidade e para revelar os erros potenciais;

c) a análise da cobertura de requisitos deve determinar quais requisitos não foram testados; e

d) a análise da cobertura estrutural deve determinar quais estruturas não foram verificadas.

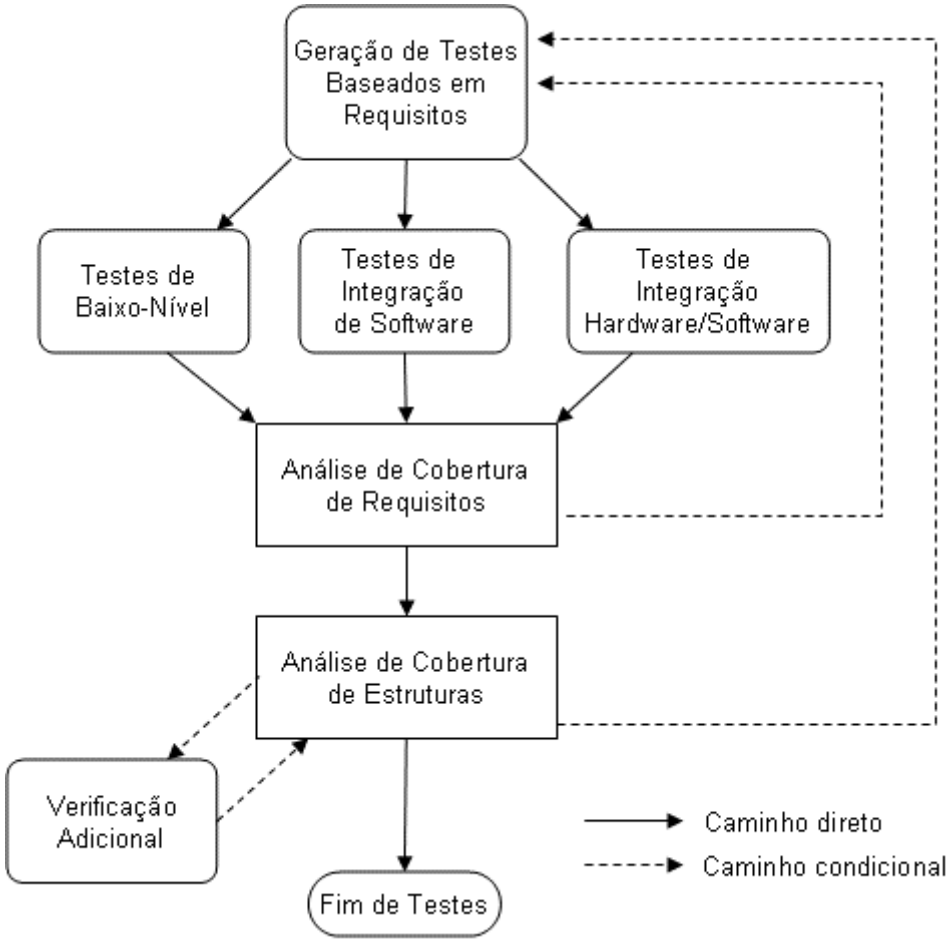

Figura 6 - Atividade de Testes

7.5.4 AMBIENTE DE TESTES

7.5.4.1 Mais de um ambiente de testes pode ser necessário para satisfazer aos objetivos de testes de *software*. Um bom ambiente de testes deve incluir o *software* que é carregado no equipamento e permitir testes em condições simulando o ambiente de operação.

NOTA: Em muitos casos, para se atingir a cobertura de requisitos e a cobertura estrutural é necessário controle mais preciso e monitoramento das entradas dos testes e da execução do código que geralmente só é possível num ambiente completamente integrado. Esses testes podem requerer um pequeno componente de *software* que funcione isoladamente dos outros componentes.

7.5.4.2 Para fins de validação podem ser utilizados testes usando emuladores ou simuladores. Isso inclui testes selecionados que devem ser realizados no ambiente integrado, pois certos erros só podem ser detectados nesse ambiente.

7.5.5 SELEÇÃO DE CASOS DE TESTES

Testes baseados em requisitos são enfatizados, pois essa estratégia é mais efetiva em revelar erros. A seleção deve:

a) efetivar os objetivos dos testes de *software*, com duas categorias de casos de testes: casos de variação normal e casos de variação para robustez (variação anormal); e

b) desenvolver casos de testes específicos a partir dos requisitos e das fontes mais comuns de erros do processo de desenvolvimento.

7.5.5.1 Casos de Testes com Variação Normal

O objetivo dos casos de testes com variação normal é demonstrar que o *software* responde as entradas em condições normais. Os testes com variação normal devem incluir as seguintes considerações:

a) variáveis reais e inteiras devem ser exigidas usando uma classe de equivalência dos valores limites;

b) para funções variantes no tempo, como filtros, integradores e temporizadores, múltiplas iterações do código devem ser realizadas para checar as características da função no seu contexto;

c) para transições de estado, casos de testes devem ser desenvolvidos para exercitar as possíveis transições durante a operação normal; e

d) para requisitos de *software* expressados por equações lógicas, a variação normal pode verificar o uso das variáveis e os operadores booleanos.

NOTA: Um método é testar todas as combinações das variáveis. Para expressões complexas, esse método é impraticável devido ao grande número de combinações necessário. Uma estratégia diferente que assegure a cobertura deve ser desenvolvida. Por exemplo, para o nível de segurança AL1 os operadores booleanos podem ser verificados por análise ou revisão, e para complementação dessa atividade, casos de testes podem ser escolhidos para prover cobertura de decisão/condição modificada (MC/DC).

7.5.5.2 Casos de Testes de Robustez

O objetivo dos casos de testes de robustez é demonstrar que o *software* responde as entradas em condições não normais. Os testes de robustez devem incluir os seguintes aspectos:

a) variáveis reais e inteiras devem ser exigidas usando uma classe de equivalência dos valores inválidos;

b) a inicialização do sistema deve ser exercitada fora das condições normais;

c) os possíveis modos de falha dos dados de entrada devem ser determinados, especialmente dados digitais complexos organizados e enviados em sequência (*strings* de dados digitais) vindos de sistemas externos;

d) para execuções sequenciais e repetitivas de comandos de *software* como parte de um algoritmo (*loops*) em que o contador do *loop* (variável que controla o número de vezes que a sequência de comandos será repetida) é um valor computado, os casos de testes devem ser desenvolvidos para tentar processar valores fora da variação normal e demonstrar a robustez do código do *loop*;

e) a checagem deve assegurar os mecanismos de proteção para tempos de resposta maiores que o esperado;

f) para funções temporais, como os filtros, integradores e temporizadores, os casos de testes devem desenvolver mecanismos de proteção de *overflow* aritmético (quando o resultado de um cálculo computacional aritmético extrapola a capacidade de armazenamento do número resultante na variável de memória); e

g) para transições de estados, os casos de testes devem provocar transições que não são permitidas pelos requisitos de *software*.

7.5.6 MÉTODOS PARA TESTES BASEADOS EM REQUISITOS

Os métodos para testes baseados em requisitos consistem nos métodos de testes de integração de *hardware* e *software*, de integração de *software* e de requisitos de baixo nível. Com exceção de testes de integração de *hardware* e *software*, os métodos prescrevem ambiente de teste ou estratégia para teste. As instruções incluem:

a) testes de integração de *hardware* e *software*: esse método deve concentrar-se nas fontes de erros associadas com a operação dentro do ambiente em que o *software* opera, e com alto grau de funcionalidade. O objetivo dos testes de integração de *hardware* e *software* é assegurar que o *software* no computador satisfaz aos requisitos de alto nível. Típicos erros revelados por esse método incluem:
   - incorreto tratamento de interrupções;
   - falha para satisfazer aos requisitos de tempo de execução;
   - resposta incorreta aos transientes ou falhas de *hardware*, como exemplo, sequência incorreta de partida, transientes nas entradas ou transientes de energia;
   - problemas no barramento de dados ou com contenção de recursos, por exemplo, mapeamento de memória;
   - problemas na detecção de falhas do BIT;
   - erros nas interfaces de *hardware*/*software;*
   - comportamento inadequado de *loops*;
   - controle incorreto do gerenciamento de memória ou outro dispositivo com controle em *software*;
   - estouro de pilha;
   - operação incorreta dos mecanismos usados para carregamento de *software* em campo; e
   - violações da partição;

b) testes de integração de *software*: esse método deve concentrar-se nos relacionamentos entre os requisitos de *software* e na implementação dos requisitos pela arquitetura de *software*. Os objetivos dos testes de integração de *software* é assegurar que os componentes interagem corretamente entre si. Típicos erros revelados por esse método incluem:
   - incorreta inicialização de variáveis e constantes;
   - erros na passagem de parâmetros;
   - corrupção de dados, especialmente os dados globais;
   - inadequada resolução numérica; e
   - incorreta sequência de eventos e operações;

c) testes de requisitos de baixo nível: esse método deve concentrar-se na demonstração de que cada componente é compatível com os requisitos de baixo nível. Típicos erros que devem ser levantados:
   - falha no algoritmo para satisfazer ao requisito;
   - operações incorretas em *loops*;
   - decisões lógicas incorretas;
   - falhas para processar combinações válidas de condições de entradas;
   - respostas incorretas com perda ou corrupção nos dados de entrada;
   - incorreto tratamento de exceções, como falhas aritméticas ou violação dos limites de matrizes*;*
   - incorreta computação de sequências; e
   - precisão, acurácia ou desempenho inadequados do algoritmo.

7.5.7 ANÁLISE DE COBERTURA

Análise de cobertura é um processo de dois passos, envolvendo análise da cobertura de requisitos e a análise da cobertura estrutural. O primeiro passo analisa os casos de testes com relação aos requisitos de *software* para confirmar que os casos selecionados satisfazem critérios específicos. O segundo passo confirma que os testes exercitam a estrutura de código. A análise da cobertura estrutural pode não satisfazer a um critério específico. Instruções adicionais são dadas para os casos de código desativado.

7.5.7.1 Análise da Cobertura de Requisitos

O objetivo dessa análise é determinar quão bem os testes de requisitos verificaram a implementação dos requisitos de *software*. Essa análise pode revelar a necessidade de casos de testes adicionais. A análise mostra:

a) a existência de casos de testes para cada requisito de *software; e*

b) a satisfação dos testes com critérios de variação normal e robustez, definidos no item 7.5.5.

7.5.7.2 Análise da Cobertura Estrutural

O objetivo dessa análise é determinar quais estruturas de código não foram exercitadas pelos procedimentos de testes. Os casos de testes podem não exercitar completamente a estrutura de código, assim verificações adicionais são produzidas para se alcançar à cobertura estrutural. A análise deve:

a) confirmar o grau de cobertura apropriado para o nível de segurança de *software*;

b) pode ser feita no Código-Fonte, a menos que o nível de segurança de *software* seja AL1 e que o compilador gere Código-Objeto que não é diretamente rastreado às linhas de código. Então, verificações adicionais devem ser feitas para avaliar o Código-Objeto de modo a estabelecer que este foi gerado correto; e

c) deve confirmar o acoplamento entre dados e controle existentes no código.

7.5.7.3 Resolução da Cobertura Estrutural

A análise de cobertura estrutural pode revelar que partes do código não foram exercitadas durante os testes. A resolução seria requerer verificações adicionais. O código não exercitado pode ser resultado de:

a) atalhos ou simplificações nos casos de testes: os casos de testes poderiam ser suplementados ou mudados para aumentar a cobertura. O método usado para análise de cobertura pode ser revisado;

b) inadequações nos requisitos de *software*: os requisitos devem ser modificados e casos de testes adicionais desenvolvidos e procedimentos de testes executados;

c) código morto: o código deve ser removido. Uma nova análise dever ser feita para verificar seus efeitos e avaliar a necessidade de nova verificação; e

d) código desativado: não é esperado que o código desativado seja executado em qualquer configuração usada. Uma combinação de análises e testes deve demonstrar que há no Código-Fonte dispositivos que garantam ou previnam que a parte desativada do código não seja inadvertidamente executada. Para código desativado que é executado somente quando são ativadas certas configurações em um ambiente computacional a configuração em operação necessária a execução normal desse código deve ser estabelecida e testes adicionais devem ser incluídos nos casos de teste e procedimentos de teste desenvolvidos para satisfazer aos objetivos de cobertura requeridos.

8 PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE*

Este capítulo estabelece os objetivos e as atividades do processo de gerenciamento de configuração de *software* (SCM). O processo SCM é aplicado segundo o processo de planejamento (capítulo 5) e o Plano de Gerenciamento de Configuração do *Software* (item 11.5). As saídas geradas pelo processo SCM são armazenadas nos registros do gerenciamento de configuração (item 11.19) ou em outra evidência do ciclo de vida do *software*. A Tabela A-8 do Anexo A é um resumo dos objetivos e das saídas do processo SCM.

8.1 OBJETIVOS DO PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE (SCM)*

8.1.1 O processo SCM, em conjunto com os outros processos de ciclo de vida do *software*, auxilia a satisfação dos seguintes objetivos gerais:

a) prover uma configuração de *software* definida e controlada através do ciclo de vida do *software*;

b) prover capacidade de replicar objeto de código executável para a fabricação de *software* ou de regenerá-lo em caso de necessidade de investigação ou modificação;

c) prover controle de entradas e saídas de processo durante o ciclo de vida do *software* que garante a consistência e a capacidade de reprodução das atividades do processo;

d) prover um ponto conhecido para revisar, reportar *status*, e controlar mudanças pelo controle de itens de configuração e o estabelecimento de bases;

e) prover controles para assegurar que os problemas recebam atenção e suas mudanças sejam registradas, aprovadas e implementadas;

f) prover evidências para validação do *software* através do controle das saídas dos processos do ciclo de vida do *software*;

g) auxiliar a avaliação da conformidade do produto de *software* com os requisitos; e

h) assegurar que o armazenamento, recuperação e o controle dos itens de configuração são seguros.

8.1.2 Os objetivos SCM são independentes do nível de *software*. No entanto, duas categorias de evidências do ciclo de vida são definidas (item 8.2.4).

8.2 ATIVIDADES DO PROCESSO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE*

O processo SCM inclui as atividades de identificação de configuração, controle de mudanças, estabelecimento de bases, armazenamento do produto de *software*, incluindo evidências relacionadas. As seguintes recomendações definem os objetivos para cada atividade do processo SCM. O processo SCM não termina quando o produto de *software* é aceito pela autoridade certificadora, mas continua durante todo o ciclo de vida do sistema ou equipamento.

8.2.1 IDENTIFICAÇÃO DA CONFIGURAÇÃO

O objetivo da atividade de identificação da configuração é rotular de forma inequívoca cada item de configuração e suas sucessivas versões. Desta forma, é possível se estabelecer uma linha de base, visando ao efetivo controle e objetivando criar uma referência para os itens de configuração. As recomendações incluem:

a) a identificação de itens de configuração deve ser estabelecida para registrar resultados, produtos e subprodutosas que são evidências do ciclo de vida do *software*;

b) a identificação de itens de configuração deve ser estabelecida considerando cada componente controlado como um item de configuração em separado e para as respectivas combinações dos itens de configuração que compõem um produto de *software;*

c) a identificação de itens de configuração deve ser feita antes da implementação do controle de mudanças e do registro da rastreabilidade;

d) um item de configuração deve ser identificado antes de ser utilizado por outros processos do ciclo de vida do *software*, de ser referenciado por outras evidências do ciclo de vida do *software*, ou de ser usado para manutenção de *software* ou para o carregamento de *software*; e

e) se a identificação de produto de *software* não puder ser feita por meio de exame físico (por exemplo, verificação de PN), então o Código-Executável deverá conter a identificação da configuração a qual pode ser acessada por outras partes do sistema ou equipamento. Isto pode ser aplicado para FLS. (item 3.7).

8.2.2 LINHA DE BASE E RASTREABILIDADE

O objetivo do estabelecimento de uma linha de base (*baseline*) é definir um ponto para continuar a atividade do processo do ciclo de vida e permitir referência, controle e rastreabilidade entre os itens de configuração. As recomendações incluem:

a) linhas de bases devem ser estabelecidas para itens de configuração usados para dar crédito a processos de certificação. Em paralelo, bases intermediárias podem ser estabelecidas para auxiliar no controle das atividades dos processos do ciclo de vida do *software*; e

b) uma linha de base de produto de *software* deve ser estabelecida para o produto de *software* e definida no Índice de Configuração de *Software* (item 11.17);

NOTA: Componentes de um *software* modificáveis pelo usuário não são incluídos na linha de base (*baseline*) do produto de *software*, exceto quando interferem na segurança operacional e se são considerados componentes de interface com outros sistemas de *software*. Assim, modificações podem ser feitas pelo usuário em um *software* modificável sem afetar a respectiva linha de base que identifica uma configuração de produto de *software*.

c) linhas de base deverão ser estabelecidas em bibliotecas de *software* controladas (bibliotecas físicas, eletrônicas, ou outras) para assegurar sua

integridade. Uma vez que uma linha de base é estabelecida, ela deve ser protegida de mudanças;

d) atividades de controle de mudanças devem ser seguidas para desenvolver uma linha de base derivada a partir de uma linha de base estabelecida;

e) se o processo de certificação solicitar evidências que comprovem procedimentos ou resultados decorrentes de atividades e processos do ciclo de vida do *software*, ou evidência associada ao desenvolvimento anterior, a linha de base deverá ser rastreada à linha de base da qual inicialmente foi derivada;

f) se for solicitada comprovação para fins de crédito de certificação para as atividades de processos de ciclo de vida de *software*, ou comprovação de evidência associada ao desenvolvimento anterior, o item de configuração deve ser rastreado ao item de configuração do qual foi derivado; e

g) uma linha de base ou item de configuração deverá poder ser rastreado até a saída do processo ou até o processo ao qual ele está associado.

8.2.3 REPORTE DE PROBLEMAS, MONITORAMENTO E AÇÕES CORRETIVAS

8.2.3.1 O objetivo da atividade de reporte de problemas, monitoramento e ações corretivas é registrar as não conformidades do processo com relação ao planejamento do *software* e aos padrões pré-estabelecidos, além de registrar deficiências das saídas dos processos de ciclo de vida do *software*, de registrar algum comportamento anômalo de produtos de *software* e de assegurar resolução desses problemas.

NOTA: Problemas em processos do ciclo de vida do *software* e problemas em produtos de *software* devem ser registrados em sistemas separados.

8.2.3.2 As recomendações incluem:

a) relatórios descrevendo problemas devem ser preparados de forma a descrever: não conformidades entre os processos executados e os planos aprovados; saídas deficientes ou comportamentos anômalos do *software*; e as ações corretivas executadas como definido no item 11.18;

b) relatórios descrevendo problemas devem informar seu *status*, sua aprovação e seu fechamento visando à identificação de configuração dos itens de configuração afetados ou à definição das atividades afetadas do processo; e

c) relatórios descrevendo problemas que requerem ações corretivas sobre o produto de *software* ou sobre as saídas dos processos de ciclo de vida do *software* devem ser submetidos à atividade de controle de mudanças.

NOTA: As atividades para relatar problemas e para controle de mudanças são interdependentes.

8.2.4 CONTROLE DE MUDANÇAS

8.2.4.1 O objetivo da atividade de controle de mudanças é o de fornecer registros, avaliação e aprovação das mudanças por meio do ciclo de vida do *software*. As recomendações incluem:

a) o controle de mudanças deve preservar a integridade dos itens de configuração e das linhas de base através do serviço de proteção contra mudanças;

b) o controle de mudanças deve assegurar que qualquer mudança em um item de configuração requeira mudança na sua identificação de configuração;

c) as mudanças de linhas de base e de itens de configuração sob controle de mudanças devem ser registradas, aprovadas e rastreadas. Os relatórios de problemas estão relacionados aos controles de mudanças, pois a resolução de um problema reportado pode resultar em mudanças nos itens de configuração ou nas linhas de base;

NOTA: A implementação antecipada do controle de mudanças auxilia o controle e o gerenciamento das atividades dos processos de ciclo de vida do *software*.

d) as mudanças de *software* devem ser rastreadas até a sua origem e até a origem dos processos de ciclo de vida repetidos a partir do ponto que a mudança afetou suas saídas. Por exemplo, um erro descoberto na integração *hardware/software*, devido a um erro de projeto, deve resultar na correção do projeto, na correção do Código-Fonte e na repetição das atividades do processo de integração associadas; e

e) as evidências do ciclo de vida afetadas por mudanças devem ser atualizadas e registradas pela atividade de controle de mudanças.

8.2.4.2 A atividade de controle de mudanças é auxiliada pela atividade de revisão de mudanças.

8.2.5 REVISÃO DE MUDANÇAS

O objetivo da atividade de revisão de mudanças é assegurar: que as mudanças são avaliadas antes de serem aprovadas; que foram efetivamente implementadas; e que os processos afetados sofram realimentação através do relatório de problemas e dos controles de mudança definidos durante o processo de planejamento. A atividade de revisão da mudança deve incluir:

a) a confirmação de que os itens de configuração afetados sofreram identificação;

b) a avaliação do impacto sobre os requisitos de segurança com retorno para o processo de avaliação de segurança do sistema;

c) a avaliação do problema ou da mudança, com decisões para as ações a serem tomadas; e

d) o retorno do relatório de problemas ou do impacto da mudança e das decisões para os processos afetados.

8.2.6 REPORTE DO *STATUS* DE CONFIGURAÇÃO

O objetivo da atividade de reporte de *status* é o de fornecer informação para o gerenciamento da configuração dos processos de ciclo de vida do *software* com relação à identificação de configuração, linhas de bases, relatórios de problemas e controle de mudanças. A atividade de reporte do *status* inclui:

a) relatórios sobre identificação de item de configuração, identificação de linhas de base, reporte de *status* de problema, histórico de mudanças e *status* da versão; e

b) definição dos dados a serem mantidos, e os meios de registro e relatório de *status* desses dados.

8.2.7 ARMAZENAMENTO, RECUPERAÇÃO E LIBERAÇÃO DE VERSÃO

O objetivo da atividade de armazenamento e de recuperação é assegurar que as evidências do ciclo de vida do *software* associado com o produto de *software* podem ser recuperadas em caso de ser necessária a sua duplicação, regeneração, "reteste" ou a modificação do produto de *software*. O objetivo da atividade de liberação de versão é assegurar que somente *software*s autorizados são usados, visando, especialmente, ao caso de *software* manufaturado, e também o arquivamento ou a consulta. A recomendação abrange:

a) as evidências do ciclo de vida do *software* associadas ao produto de *software* devem permitir consulta a partir de fonte aprovada (por exemplo, a partir da empresa desenvolvedora);

b) os procedimentos devem ser estabelecidos para assegurar a integridade dos dados armazenados (independentemente do meio de armazenamento), com o objetivo de:
   - assegurar que nenhuma mudança não autorizada possa ser feita;
   - selecionar mídia de armazenamento que minimize erros;
   - exercitar ou regravar dados armazenados em uma frequência compatível com a vida útil da mídia;
   - guardar cópias em locais fisicamente separados visando minimizar o risco de perda em caso de desastre;

c) um processo de duplicação deve ser implementado para produzir cópias precisas e, ainda, os procedimentos escolhidos devem ser documentados com o objetivo de assegurar cópias sem erros;

d) que o fornecedor e a autoridade responsável pela liberação devem identificar e liberar versões dos itens de configuração antes do uso. Os componentes do produto de *software* a serem carregados no sistema ou no equipamento (o que inclui o Código-Executável e pode também incluir mídia associada com o carregamento de *software*) devem ter versões liberadas; e

e) procedimentos de armazenamento de dados devem ser estabelecidos para satisfazer aos requisitos legais e permitir modificações de *software*.

NOTA: Considerações para a guarda de dados podem incluir necessidades de negócio e revisão futura da autoridade certificadora.

8.2.8 CONTROLE DE CARREGAMENTO DO *SOFTWARE*

O objetivo da atividade de carregamento do *software* é assegurar que o Código-Executável seja carregado no sistema ou no equipamento com as salvaguardas adequadas. O controle do carregamento de *software* refere-se ao processo no qual procedimentos e dados são transferidos a partir de um dispositivo de memória mestre para o sistema ou equipamento alvo. Por exemplo, a instalação de dispositivo de memória pré-programada em fábrica ou reprogramação do sistema ou equipamento em sítio usando um dispositivo de carregamento em campo. Independentemente do método utilizado, o controle de carregamento de *software* deve incluir:

a) procedimentos para associar o *Part Number* (PN) e a identificação da mídia às configurações de *software* aprovadas para o carregamento no sistema ou equipamento; e

b) se o *software* é entregue como produto final ou é entregue instalado no sistema ou equipamento, então registros que confirmem a compatibilidade do *software* com o sistema ou equipamento de *hardware* devem ser registrados.

NOTA: Recomendações adicionais para o controle de carregamento de *software* são fornecidas no item 3.7.

8.2.9 CONTROLE DE CICLO DE VIDA DO AMBIENTE DE *SOFTWARE*

O objetivo de controlar o ciclo de vida do ambiente de *software* é o de assegurar que as ferramentas usadas para produzir o *software* sejam identificadas, controladas e restauráveis. As ferramentas usadas durante o ciclo de vida de *software* são definidas pelo processo de planejamento e identificadas no Índice de Configuração do Ambiente do Ciclo de Vida do *Software* (item 11.16). As recomendações incluem:

a) a identificação de configuração deve ser estabelecida para o Código-Executável (ou equivalente) das ferramentas usadas para desenvolver, controlar, construir, verificar e carregar o *software*;

b) o processo SCM para controlar ferramentas qualificadas deve obedecer aos objetivos associados com a Categoria de Controle 1 ou 2 (item 8.3), como especificado no item 12.4.10, alínea "b"; e

c) exceto quando o item 8.2.9, alínea "b", é aplicado, o processo SCM para controlar o Código-Executável (ou equivalente) das ferramentas usadas para construir e carregar o *software* (por exemplo, compiladores, montadores, editores de ligação) deve obedecer, no mínimo, aos objetivos associados com a Categoria de Controle 2.

8.3 CATEGORIAS DE CONTROLE DE EVIDÊNCIAS

8.3.1 As evidências do ciclo de vida podem ser categorizadas em: Categoria de Controle 1 (CC1) e Categoria de Controle 2 (CC2). Estas categorias estão relacionadas com os controles de gerenciamento de configuração exercidos nos dados. A tabela 1 define um conjunto de objetivos para os processos associados a cada categoria de controle, em que uma marcação indica quais objetivos devem ser aplicados a cada categoria.

8.3.2 As tabelas do Anexo A especificam o controle de categoria para cada item de configuração em um ciclo de vida, por nível de *software*. As recomendações incluem:

a) os objetivos do processo SCM para evidências do ciclo de vida do *software* classificados como CC1 devem ser aplicados de acordo com a tabela seguinte; e

b) os objetivos do processo SCM para evidências do ciclo de vida do *software* classificados como CC2 devem ser, no mínimo, aplicados de acordo com a tabela a seguir.

Tabela 1 - Objetivos do Processo SCM associados às categorias CC1 e CC2

Objetivos do processo SCM	Referência	CC1	CC2
Identificação da Configuração	8.2.1	s	s
Linha de Base	8.2.2 a, b, c, d, e	s	
Rastreabilidade	8.2.2 f, g	s	s
Relatório de Problemas	8.2.3	s	
Controle de Mudanças – Integridade e identificação	8.2.4 a, b	s	s
Controle de Mudanças – Rastreabilidade	8.2.4 c, d, e	s	
Revisão de Mudanças	8.2.5	s	
Reporte do *Status* de Configuração	8.2.6	s	
Recuperação de Versão	8.2.7 a	s	s
Proteção contra Mudanças não Autorizadas	8.2.5 b	s	s
Seleção de Mídia, Atualização e Duplicação	8.2.7 b, c	s	
Liberação de Versão	8.2.7 d	s	
Armazenamento de Dados	8.2.7 e	s	

s – denota aplicação da categoria CC1 ou CC2

9 PROCESSO DE GARANTIA DA QUALIDADE DO *SOFTWARE*

Este capítulo discute os objetivos e atividades do processo de garantia da qualidade de *software* (SQA). O processo SQA é aplicado assim como definido pelo processo de Planejamento (capítulo 5) e pelo Plano de Garantia da Qualidade (item 11.6). Os resultados das atividades do processo SQA são armazenados nos Registros da Garantia da Qualidade de *Software* (item 11.20) e outras evidências do ciclo de vida. O processo SQA avalia os processos do ciclo de vida do *software* e os resultados para assegurar que os respectivos objetivos são satisfeitos, que as deficiências são detectadas, avaliadas, rastreadas e resolvidas, e que o produto de *software* e as outras evidências do ciclo de vida de *software* estão em conformidade com os requisitos de validação.

9.1 OBJETIVOS DO PROCESSO DE GARANTIA DA QUALIDADE DO *SOFTWARE*

9.1.1 Os objetivos do Processo de Garantia da Qualidade do *Software* asseguram que os processos do ciclo de vida do *software* produzem *software* em conformidade aos requisitos, pela avaliação de que esses processos seguem os planos e padrões aprovados.

9.1.2 O processo SQA tem os seguintes objetivos:

a) o processo SQA deve garantir que os processos de desenvolvimento e de integração estejam em conformidade com os planos e padrões aprovados;

b) o processo SQA deve garantir que os planos e padrões sejam desenvolvidos e revisados com consistência;

c) o processo SQA deve garantir que os processos do ciclo de vida estejam em conformidade com os planos e padrões aprovados;

d) o processo SQA deve incluir auditorias dos processos de desenvolvimento e de integração durante o ciclo de vida para assegurar que:
   - planos do *software* estejam disponíveis conforme especificado no item 4.2;
   - desvios dos planos do *software* e dos padrões sejam detectados, gravados, avaliados, rastreados e resolvidos;

   NOTA: A detecção prematura dos desvios dos processos ajuda a alcançar os objetivos dos processos.
   - desvios aprovados sejam documentados;
   - o ambiente de desenvolvimento esteja de acordo com os planos do *software;*
   - as atividades relacionadas ao reporte de problemas, seu rastreio e as ações corretivas estejam em conformidade com o Plano de Gerenciamento de Configuração; e
   - os apontamentos feitos pelos processos do ciclo de vida tenham sido observados na avaliação de riscos realizada.

   NOTA: O monitoramento das atividades dos processos do ciclo de vida pode ser realizado com o intuito de garantir que as atividades estejam sob controle.

e) o processo SQA deve garantir que os critérios de transição para os processos do ciclo de vida de *software* sejam satisfeitos em conformidade com os planos do *software*;

f) o processo SQA deve garantir que as evidências do ciclo de vida sejam controladas de acordo com as categorias de controle definidas no item 8.3 e com as tabelas constantes no Anexo A;

g) o processo SQA deve garantir que antes da entrega do *software*, submetido à validação, seja conduzida uma revisão de conformidade; e

h) o processo SQA deve garantir que as suas atividades (item 11.20) sejam registradas, incluindo auditoria de resultados e evidências do término da revisão de conformidade para cada produto de *software* submetido como parte da validação.

9.2 REVISÃO DE CONFORMIDADE

9.2.1 O objetivo da revisão de conformidade é assegurar, para um produto de *software* submetido como parte da validação, que sejam completos os processos e as evidências do ciclo de vida, e que os Código-Objeto e Código-Executável sejam controlados e possam ser reproduzidos.

9.2.2 A revisão deve determinar:

a) se as atividades do processo de ciclo de vida planejadas com o objetivo de dar crédito à validação, incluindo a geração de documentação, estão completas e armazenadas;

b) se existe rastreabilidade entre evidências coletadas ao longo do ciclo de vida e os requisitos (requisitos específicos do sistema, requisitos de segurança ou requisitos de *software*);

c) se evidências coletadas ao longo do ciclo de vida estão em conformidade com os planos e padrões e se o controle é feito de acordo com o plano SCM;

d) se o relatório de problemas está em conformidade com o plano SCM, se o mesmo foi avaliado e se seu *status* foi armazenado;

e) se os desvios do projeto em relação aos requisitos de *software* foram documentados e aprovados;

f) se os Código-Objeto e Código-Executável podem ser reproduzidos a partir do Código-Fonte armazenado;

g) se o *software* aprovado pode ser carregado através do uso das instruções;

h) se o relatório de problemas relativos aos contratempos que causaram atrasos na revisão de conformidade anterior foram reavaliados para determinar a condição dos mesmos; e

i) se, no caso de *software* legado examinado com o objetivo de dar crédito à validação, a linha de base corrente foi rastreada até a anterior e se foram verificadas as mudanças aprovadas.

NOTA: No caso de necessidade de mudanças pós-validação, pode ser realizado um subconjunto da revisão das atividades de conformidade, desde que tais mudanças sejam justificadas e consideradas significativas.

10 PROCESSO DE VALIDAÇÃO DO *SOFTWARE*

10.1 GENERALIDADES

10.1.1 O objetivo do processo de validação é estabelecer a comunicação e o entendimento entre o fornecedor e a autoridade certificadora durante o ciclo de vida do *software*.

10.1.2 O processo de validação é definido pelo Processo de Planejamento e pelo Plano para Validação do *Software*. Tabela A-10 contém o sumário dos objetivos e das saídas desse processo.

10.2 MEIOS DE CONFORMIDADE E PLANEJAMENTO

O fornecedor pode propor como o desenvolvimento do sistema ou equipamento CNS/ATM irá satisfazer à linha de base de validação (meios de conformidade). O Plano para Validação do *Software* define que partes do sistema ou equipamento serão avaliadas, considerando os meios de conformidade propostos. Esse plano deve conter os níveis de segurança de *software* determinados pelo gerenciamento de riscos de segurança operacional. Para tanto o fornecedor deve:

a) submeter para revisão da autoridade certificadora o Plano para Validação do *Software* e outros dados requeridos em tempo de não afetar as restrições do projeto;

b) resolver as questões sobre o planejamento para a validação de *software* que forem propostas pela autoridade certificadora; e

c) obter um acordo com a autoridade certificadora para a aplicação do Plano para Validação de *Software* no projeto.

10.3 SUBSTANCIAÇÃO DA CONFORMIDADE

O fornecedor deve prover evidências de que os processos de ciclo de vida satisfazem aos planos. As revisões da autoridade certificadora podem acontecer nos estabelecimentos do fornecedor principal ou nos estabelecimentos de fornecedores terceirizados. O fornecedor organiza as revisões das atividades dos processos do ciclo de vida do *software* e faz com que as evidências se tornem disponíveis quando necessárias. O fornecedor deve:

a) resolver as questões levantadas pela autoridade certificadora e apresentar os resultados das revisões;

b) submeter o Sumário de Realizações e o Índice de Configuração à autoridade certificadora; e

c) submeter ou deixar disponível outros dados ou evidências de conformidade requeridas pela autoridade certificadora.

10.4 EVIDÊNCIAS MÍNIMAS SUBMETIDAS À AUTORIDADE CERTIFICADORA

O conjunto mínimo de evidências que deve ser submetido à autoridade certificadora é:

a) Plano para Validação de *Software;*

b) Índice de Configuração; e

c) Sumário de Realizações.

10.5 EVIDÊNCIAS RELACIONADAS AO PROJETO

A menos que modificações estejam contempladas em acordo formal com a autoridade certificadora, a norma aplica-se fundamentalmente aos documentos relacionados a seguir, com vistas a suportar o processo de validação das evidências de projeto:

a) Requisitos de *Software;*

b) Descrição de Projeto;

c) Código-Fonte;

d) Código-Executável;

e) Índice de Configuração; e

f) Sumário de Realizações.

11 EVIDÊNCIAS DO CICLO DE VIDA DO *SOFTWARE*

11.1 GENERALIDADES

11.1.1 Informações são produzidas durante o ciclo de vida do *software* para planejar, dirigir, explicar, definir, gravar ou prover evidências das atividades. Estes dados possibilitam a execução dos processos de ciclo de vida, a execução da validação do sistema ou equipamento, e das modificações pós-validação do *software*. Este capítulo discute as características, os controles de gerência de configuração e as evidências do ciclo de vida do *software*.

11.1.2 As características das evidências do ciclo de vida do *software* são:

a) sem ambiguidade: escrito em termos que permitem somente uma interpretação, acompanhadas de uma definição, se necessário;

b) completa: uma informação é completa quando inclui os requisitos relevantes e necessários e/ou material descritivo, as respostas estão definidas para o intervalo de valores de entrada válido, as figuras estão identificadas, e os termos e unidades de medida estão definidos;

c) verificável: uma informação é verificável se a sua correção pode ser verificada por uma pessoa ou ferramenta;

d) consistente: uma informação é consistente se não há conflitos nela;

e) modificável: uma informação é modificável se for estruturada e possuir um estilo de maneira tal que mudanças podem ser realizadas de forma completa, consistente e correta, mantendo a estrutura da informação;

f) rastreabilidade: uma informação tem rastreamento se a origem dos seus componentes pode ser determinada;

g) forma: a forma deve permitir a recuperação e a revisão das evidências do ciclo de vida do *software* durante todo o período de serviço do sistema ou equipamento. As evidências e suas formas devem ser especificadas no Plano para Validação do *Software*.

NOTA 1: As evidências do ciclo de vida do *software* podem tomar diversas formas. Por exemplo, pode ser em forma escrita em arquivos de computador armazenados em mídia magnética, ou em *displays* num terminal remoto. Podem ser apresentados em documentos individuais, combinados em documentos maiores, ou distribuídos entre vários documentos.

NOTA 2: O Plano para Validação do *Software* e o Sumário de Realizações de *Software* podem ser requeridos por autoridades de certificação na forma de documentos impressos separados.

11.1.3 As evidências do ciclo de vida do *software* podem ser colocadas em uma de duas categorias associadas com os controles de gerência de configuração de *software* aplicados: Controle de Categoria 1 (CC1) e Controle de Categoria 2 (CC2) (item 8.3). Esta distinção provê meios para gerenciar custos de desenvolvimento, em que controles menos severos podem ser aplicados sem redução na segurança. A categoria mínima de controle associada a cada item e suas variações pelo nível de *software* estão especificadas no Anexo A.

11.1.4 As descrições das evidências aqui constantes definem evidências que normalmente são produzidas para auxiliar o processo de validação. As descrições não têm a intenção de

descrever todas as evidências necessárias para desenvolver um *software*, tampouco sugerir um método de documentação particular ou organização.

11.1.5 Outras evidências podem ser produzidas para apoiar o processo de validação, além daquelas já descritas nestes itens.

a) controle: Se utilizado para este propósito, a evidência deve ser definida no Plano de *Software* que define o processo para o qual ela será produzida. Enquanto a natureza e o conteúdo das evidências podem variar, aplica-se no mínimo o controle CC2. Exemplos são os memorandos e as minutas de reunião.

11.2 PLANO PARA VALIDAÇÃO DO *SOFTWARE*

O propósito do Plano para Validação do *Software* é o meio pelo qual a autoridade certificadora determina se o fornecedor está propondo um ciclo de vida do *software* adequado ao rigor requerido para o nível de garantia de *software* desenvolvido.

a) visão geral do sistema: capítulo que provê uma visão geral do sistema, incluindo a descrição de suas funcionalidades e as suas alocações em *hardware* e *software*, a arquitetura, processadores utilizados, interfaces de *hardware*/*software* e características de segurança;

b) visão geral do *software*: capítulo que descreve resumidamente as funcionalidades de *software* com ênfase na segurança proposta e nos conceitos de partição, por exemplo, compartilhamento de recursos, redundância, *software* dissimilar com múltiplas-versões, tolerância a falhas, e estratégias de temporização e escalonamento;

c) considerações de validação: capítulo que provê um sumário da base de validação, incluindo os meios para conformidade relacionados à validação do *software*. Este capítulo também especifica os níveis de garantia de *software* proposto e resume a justificativa provida pelo processo de avaliação da segurança do sistema, incluindo as potenciais contribuições do *software* às condições de falha;

d) ciclo de vida do *software*: capítulo que define a forma do ciclo de vida do *software* a ser utilizado e inclui um resumo de cada ciclo de vida de *software* e seus processos para os quais informações detalhadas estão definidas em seus respectivos planos do *software*. O resumo explica como os objetivos de cada processo do ciclo de vida do *software* serão satisfeitos e especifica as organizações que deverão ser envolvidas, as responsabilidades organizacionais, e os processos do ciclo de vida do sistema e as responsabilidades sobre os processos de validação;

e) evidências do ciclo de vida do *software*: capítulo que especifica as evidências do ciclo de vida do *software* que serão produzidas e controladas pelo processo de ciclo de vida. Este capítulo também descreve a relação dos dados entre si ou com outros dados que definem o sistema, as evidências do ciclo de vida que deverão ser submetidas à autoridade certificadora, a forma dos dados e os meios pelos quais as evidências do ciclo de vida deverão ser disponibilizadas à autoridade de certificação;

f) planejamento: capítulo que descreve os meios que o solicitante vai utilizar para prover à autoridade de certificação a visibilidade das atividades dos processos do ciclo de vida do *software* para que revisões sejam planejadas; e

g) considerações adicionais: capítulo que descreve funcionalidades específicas que podem afetar o processo de validação, como, por exemplo, métodos alternativos de conformidade, qualificação de ferramentas, *software* legado, *software* com opções selecionáveis, *software* modificável pelo usuário, *software* COTS, *software* com carregamento em campo, *software* dissimilar múltipla-versão e histórico de serviço do produto.

11.3 PLANO DE DESENVOLVIMENTO DO *SOFTWARE*

O Plano de Desenvolvimento do *Software* inclui os objetivos, padrões e ciclo(s) de vida do *software* que serão utilizados no processo de desenvolvimento de *software*. Pode ser incluído no Plano para Validação do *Software*. Este plano inclui:

a) padrões: identificação dos Padrões de Requisitos de *Software*, Padrões de Projeto de *Software* e Padrões de Codificação do *Software* para o projeto, referências para os padrões do *software* legado, incluindo *software* COTS, se estes padrões são diferentes;

b) ciclo de vida do *software*: a descrição do processo do ciclo de vida do *software* a ser utilizado para formar os ciclos de vida de *software* específicos a serem utilizados no projeto, incluindo o critério de transição para o processo de desenvolvimento de *software*. Esta descrição é distinta do resumo colocado no Plano para Validação do *Software*, devendo prover os detalhes necessários para garantir a implementação apropriada dos processos de ciclo de vida do *software*; e

c) ambiente de desenvolvimento: uma declaração do ambiente de desenvolvimento escolhido, em termos de *hardware* e *software*, incluindo:
   - os métodos de desenvolvimento de requisitos escolhidos e as ferramentas que serão utilizadas;
   - os métodos de projeto escolhidos e as ferramentas que serão utilizadas;
   - as linguagens de programação, ferramentas de codificação, compiladores, editores de ligação e carregadores que serão utilizados; e
   - as plataformas de *hardware* para as ferramentas que serão utilizadas.

11.4 PLANO DE VERIFICAÇÃO DO *SOFTWARE*

O Plano de Verificação do *Software* é uma descrição dos procedimentos de verificação para satisfazer aos objetivos do processo de verificação de *software*. Estes procedimentos podem variar por nível de garantia de *software*, como definido nas tabelas do Anexo A. Este plano deve incluir:

a) organização: responsabilidades organizacionais dentro do processo de verificação de *software* e interfaces com os outros processos do ciclo de vida;

b) independência: uma descrição dos métodos para estabelecer independência da verificação, quando requerido;

c) métodos de verificação: uma descrição dos métodos de verificação a serem utilizados para cada atividade do processo de verificação de *software*;
   - métodos de revisão, incluindo *checklists* e outros auxílios;
   - métodos de Análise, incluindo rastreabilidade e análise de cobertura;
   - métodos de teste, incluindo instruções que estabelecem o processo de seleção de casos de teste, os procedimentos de testes a serem utilizados e os dados de teste a serem produzidos.

d) ambiente de verificação: uma descrição do equipamento para teste, as ferramentas de teste e análise, e as instruções para aplicar estas ferramentas e *hardware* do equipamento de teste (veja também o item 5.4.7, alínea "b" para instruções de como indicar o computador e as diferenças do simulador ou emulador);

e) critério de transição: o critério de transição para entrar no processo de verificação definido neste plano;

f) considerações de partição: se partição for utilizada, os métodos utilizados para verificar a integridade da partição;

g) suposições de compilador: uma descrição das suposições feitas pelo solicitante sobre compilador, editor de ligação ou carregador (item 4.4.2);

h) instruções de reverificação: para modificações de *software*, uma descrição dos métodos para identificar as áreas afetadas do *software* e as partes alteradas do Código-Executável. A reverificação deve garantir que os erros previamente apontados ou classes de erros foram eliminados;

i) *software* legado: para *software* legado, se a base inicial de conformidade para o processo de verificação não está em conformidade com este documento, uma descrição dos métodos para satisfazer aos objetivos deste documento; e

j) *software* dissimilar múltipla-versão: se *software* dissimilar múltipla-versão for utilizado, uma descrição das atividades do processo de verificação de *software* deverá ser explicitada (item 12.5.7).

11.5 PLANO DE GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE*

O Plano de Gerenciamento de Configuração do *Software* estabelece os métodos a serem utilizados para cumprir os objetivos do processo de gerenciamento de configuração do *software* (SCM) durante todo o ciclo de vida do *software*. Este plano deve conter os seguintes itens como abaixo descritos:

a) ambiente: descrição do ambiente SCM a ser utilizado, incluindo procedimentos, ferramentas, métodos, padrões, responsabilidades organizacionais e interfaces;

b) atividades: descrição das atividades dos processos SCM no ciclo de vida do *software* que satisfarão os objetivos de:
   - identificação da configuração: itens a serem identificados, quando eles serão identificados, os métodos de identificação para as evidências do ciclo de vida (por exemplo, PN) e a relação da identificação do *software* e o sistema CNS/ATM ou identificação do equipamento;

- bases e rastreabilidade: os meios de estabelecer linhas de base, quais linhas de base serão estabelecidas, quando estas linhas de base serão estabelecidas, os controles das bibliotecas de *software*, o item de configuração e a rastreabilidade da linha de base;
- relatórios de problemas: o conteúdo e a identificação dos relatórios de problemas para o produto e os processos de ciclo de vida, quando eles serão escritos, o método para fazer um comunicado de problema e a relação com a atividade de controle de mudanças;
- controle de mudanças: itens de configuração e linhas de base a serem controladas, quando elas serão controladas, as atividades de controle de problemas/mudanças que os controlam, os controles de pré-validação, controles de pós-validação, e os meios de preservar a integridade das linhas de bases e itens de configuração;
- revisão de mudanças: os métodos para tratar o retorno de e para os processos de ciclo de vida; os métodos para avaliar e priorizar os problemas, aprovar mudanças e tratar as implementações de mudança ou a resolução de problemas; e a relação destes métodos com o problema comunicado e as atividades de controle de mudanças;
- informações sobre o *status* de configuração: os dados a serem registrados para possibilitar a informação sobre o *status* do gerenciamento de configuração, definição de onde os dados serão mantidos, como eles serão recuperados e quando estarão disponíveis;
- arquivos, recuperação e liberação: os controles de integridade, os métodos de liberação e autoridades, e retenção de dados;
- controle do carregamento de *software*: uma descrição dos mecanismos de segurança dos controles de carregamento de *software* e os registros;
- controles do ambiente do ciclo de vida do *software*: controles para as ferramentas usadas para desenvolver, montar, verificar e carregar o *software*, endereçando os itens de 1 a 7 acima. Isso inclui controle das ferramentas a serem qualificadas;
- controles de evidências do ciclo de vida de *software*: controles associados com os dados do Controle Categoria 1 e do Controle Categoria 2;

c) critério de transição: o critério de transição para entrar no processo SCM;

d) dados SCM: a definição dos dados de ciclo de vida do *software* produzidos pelo processo SCM, incluindo os registros SCM, o Índice de Configuração de *Software* e o Índice de Configuração do Ambiente do Ciclo de Vida do *Software*; e

e) controle de fornecedores: os meios de aplicar os requisitos do processo SCM para os fornecedores.

11.6 PLANO DE GARANTIA DA QUALIDADE DO *SOFTWARE*

O Plano de Garantia da Qualidade do *Software* estabelece os métodos a serem utilizados para alcançar os objetivos do processo de garantia da qualidade do *software* (SQA). O Plano SQA pode incluir descrições do processo de melhorias, métricas e os métodos de gerenciamento progressivo. Este plano deve conter:

a) ambiente: descrição do ambiente de SQA, incluindo escopo, responsabilidades organizacionais e interfaces, padrões, procedimentos, ferramentas e métodos;

b) autoridade: declaração da autoridade de SQA, suas responsabilidades e autonomia, incluindo a autoridade de aprovação para produtos de *software*;

c) atividades: as atividades de SQA que devem ser executadas para cada processo de ciclo de vida do *software* e durante todo o ciclo de vida, incluindo:
   - métodos de SQA, como, por exemplo, as revisões, auditorias, relatórios, inspeções e monitoramento dos processos de ciclo de vida;
   - atividades relacionadas à comunicação de problemas, rastreabilidade e sistema de ações corretivas;
   - uma descrição das atividades de revisão de conformidade de *software*;

d) critérios de transição: os critérios de transição para entrar no processo SQA;

e) temporização: a temporização das atividades do processo SQA em relação às atividades dos processos de ciclo de vida;

f) registros SQA: definição dos registros a serem produzidos pelo processo SQA; e

g) controle de fornecedores: descrição dos meios de garantir que os processos dos fornecedores e suas saídas cumpram o Plano de SQA.

11.7 PADRÕES DE REQUISITOS DO *SOFTWARE*

O propósito dos Padrões de Requisitos do *Software* é definir os métodos, regras e ferramentas a serem utilizados para desenvolver os requisitos de alto nível. Estes padrões devem conter:

a) os métodos a serem utilizados para desenvolver os requisitos do *software*, tais como métodos estruturais;

b) as notações a serem utilizadas para expressar os requisitos, tais como os diagramas de fluxo de dados e linguagens de especificação formal;

c) as restrições na utilização das ferramentas de desenvolvimento de requisitos; e

d) o método a ser utilizado para prover os requisitos derivados ao processo do sistema.

11.8 PADRÕES DE PROJETO DO *SOFTWARE*

O propósito dos Padrões de Projeto do *Software* é definir os métodos, regras e ferramentas a serem utilizados para desenvolver a arquitetura do *software* e os requisitos de baixo nível. Estes padrões devem conter:

a) os métodos de descrição de projeto a serem utilizados;

b) as convenções de nomes a serem utilizadas;

c) as condições impostas sobre os métodos de projeto permitidos, como, por exemplo, escalonamento, e o uso de interrupções e arquiteturas orientadas a

eventos, tarefas dinâmicas, reentradas, dados globais, e o tratamento de exceções, e a base lógica para seu uso;

d) as restrições no uso de ferramentas de projeto;

e) as restrições sobre o projeto, como, por exemplo, exclusão de recursão, objetos dinâmicos; e

f) as restrições de complexidade, como, por exemplo, nível máximo de chamadas aninhadas ou estruturas condicionais, o uso de ramos incondicionais e o número de pontos de entrada/saída dos componentes do código.

11.9 PADRÕES DE CODIFICAÇÃO DO *SOFTWARE*

O propósito dos Padrões de Codificação do *Software* é definir as linguagens de programação, os métodos, as regras e as ferramentas a serem utilizadas para codificar o *software*. Estes padrões devem conter:

a) as linguagens de programação a serem utilizadas e/ou seus subconjuntos. Para uma linguagem de programação, referenciar informações que definem a sua sintaxe sem ambiguidades, o comportamento do controle, o comportamento dos dados e os efeitos colaterais da linguagem. Pode ser necessário limitar o uso de algumas funcionalidades de uma linguagem;

b) os padrões de apresentação do Código-Fonte, como, por exemplo, limitação no tamanho da linha, endentação, e uso de linhas vazias, e padrões de documentação do Código-Fonte, como, por exemplo, nome do autor, histórico de revisões, entradas e saídas, e dados globais afetados;

c) as convenções sobre nomes para os componentes, subprogramas, variáveis e constantes;

d) as condições e restrições impostas nas convenções de codificação permitidas, tais como o grau de acoplamento entre os componentes e a complexidade das expressões numéricas ou lógicas; e

e) as restrições no uso de ferramentas de codificação.

11.10 REQUISITOS DO *SOFTWARE*

Requisitos do *Software* são as definições dos requisitos de alto nível, incluindo os requisitos derivados. Estes dados devem conter:

a) a descrição da alocação dos requisitos do sistema para o *software*, com atenção aos requisitos relacionados à segurança e potenciais condições de falha;

b) os requisitos operacionais e funcionais sob cada modo de operação;

c) os critérios de desempenho, como, por exemplo, precisão e acurácia;

d) os requisitos de temporização e restrições;

e) as restrições de tamanho de memória;

f) as interfaces de *software* e *hardware*, como, por exemplo, protocolos, formatos, frequência das entradas e frequência das saídas;

g) os requisitos de Segurança da Informação;

h) a detecção de falhas e os requisitos de monitoramento de segurança; e

i) os requisitos de partição alocados ao *software*, como os componentes interagem uns com os outros, e os níveis de garantia de *software* para cada partição.

11.11 DESCRIÇÃO DO PROJETO

A Descrição do Projeto é uma definição da arquitetura do *software* e dos requisitos de baixo nível que irão satisfazer aos requisitos de alto nível. Estes devem incluir:

a) uma descrição detalhada de como o *software* satisfaz os requisitos de alto nível especificados, incluindo algoritmos, estruturas de dados e como os requisitos do *software* estão alocados aos processadores e tarefas;

b) a descrição da arquitetura do *software* definindo a estrutura do *software* para implementar os requisitos;

c) a descrição das entradas e saídas, como, por exemplo, um dicionário de dados, tanto internas quanto externas, por toda a arquitetura do sistema;

d) o fluxo de dados e fluxo de controle do projeto;

e) as limitações de recursos, a estratégia para gerenciar cada recurso e suas limitações, as margens e os métodos para medir as margens, como, por exemplo, temporização e memória;

f) os procedimentos de escalonamento e mecanismos de comunicação entre processos e entre tarefas, incluindo sequência rígido no tempo, escalonamento preemptivo, Ada *rendezvous* e interrupções;

g) os métodos de projeto e detalhes para a implementação, como, por exemplo, carregamento dos dados do *software*, *software* modificável pelo usuário, ou *software* dissimilar múltipla-versão;

h) os métodos de partição e meios de prevenção de brechas em cada partição;

i) as descrições dos componentes do *software*, se os mesmos são novos ou legados, e, se legados, referências para a base da qual eles foram tomados;

j) os requisitos derivados resultantes do processo de projeto do *software*;

k) se o sistema tiver código desativado, uma descrição dos meios utilizados para garantir que o código não pode ser habilitado no computador; e

l) a lógica para as decisões de projeto que são rastreáveis para requisitos de sistemas relacionados à segurança.

11.12 CÓDIGO-FONTE

Este elemento consiste em Código-Fonte escrito em linguagem fonte, as instruções de compilação para gerar o Código-Objeto do Código-Fonte, e dados de ligação e carregamento. Este elemento deve incluir a identificação do *software*, incluindo o nome e a data da revisão e/ou versão, tal como aplicável.

11.13 CÓDIGO-EXECUTÁVEL

O Código-Executável consiste em uma forma do Código-Fonte que é diretamente utilizável pela unidade central de processamento do computador e é, portanto, o *software* que é carregado no *hardware* ou sistema.

11.14 CASOS E PROCEDIMENTOS DE VERIFICAÇÃO DO *SOFTWARE*

Casos e Procedimentos de Verificação do *Software* detalham como as atividades do processo de verificação do *software* são implementadas e geram suas evidências. Estas evidências devem incluir as descrições dos:

a) procedimentos de análise e revisão: Detalhes, suplementares à descrição constante no Plano de Verificação do *Software*, que descrevem o escopo e a profundidade dos métodos de análise e revisão a serem utilizados;

b) casos de teste: O propósito de cada caso de teste, conjunto de entradas, condições, resultados esperados para atingir o critério de cobertura requerido e o critério de sucesso/falha; e

c) procedimentos de teste: As instruções passo a passo de como cada caso de teste deve ser preparado e executado, como os resultados do teste são avaliados e o ambiente de teste a ser utilizado.

11.15 RESULTADOS DA VERIFICAÇÃO DO *SOFTWARE*

Os Resultados da Verificação do *Software* são produzidos pelas atividades do processo de verificação do *software*. Os Resultados da Verificação do *Software* devem:

a) para cada revisão, análise e teste, indicar cada procedimento que passou ou falhou durante as atividades e os resultados finais de sucesso/falha;

b) identificar o item de configuração ou versão do *software* revisada, analisada ou testada; e

c) incluir os resultados dos testes, revisões e análises, incluindo as análises de cobertura e análises de rastreabilidade.

11.16 ÍNDICE DE CONFIGURAÇÃO DO AMBIENTE DO CICLO DE VIDA DO *SOFTWARE*

O Índice de Configuração do Ambiente do Ciclo de Vida do *Software* (SECI) identifica a configuração do ambiente do ciclo de vida do *software*. Este índice é escrito para auxiliar a reprodução do ambiente do ciclo de vida de *hardware* e *software*, para a regeneração do *software*, reverificação, ou modificação do *software*, e deve:

a) identificar o *hardware* do ambiente do ciclo de vida e o seu sistema operacional;

b) identificar as ferramentas de desenvolvimento de *software*, tais como compiladores, editores de ligação e carregadores, e ferramentas de integridade de dados (tais como ferramentas que calculam ou inserem o código de verificação ou verificação de redundância cíclica);

c) identificar o ambiente de teste utilizado para verificar o produto de *software*, como, por exemplo, as ferramentas de verificação de *software*; e

d) identificar as ferramentas qualificadas e os dados associados a sua qualificação.

NOTA: Estes dados podem ser incluídos no Índice de Configuração do *Software*.

11.17 ÍNDICE DE CONFIGURAÇÃO DO *SOFTWARE*

O Índice de Configuração do *Software* identifica a configuração do produto de *software*.

NOTA 1: O Índice de Configuração do *Software* pode conter um item ou um conjunto (hierarquia) de itens. O Índice de Configuração de *Software* pode conter os itens listados abaixo ou pode referenciar outro Índice de Configuração do *Software* ou outro item de configuração identificado que especifique os itens individuais e suas versões.

O Índice de Configuração do *Software* deve identificar:

a) o produto de *software*;

b) o Código-Executável;

c) o Código-Fonte de cada componente;

d) o *software* legado, se utilizado no produto;

e) as evidências do ciclo de vida do *software*;

f) as mídias e as versões aprovadas;

g) instrução para montagem do Código-Executável, incluindo instruções para compilar e ligar; e os procedimentos utilizados para restaurar o *software* para regeneração, teste ou modificação;

h) uma referência para o Índice de Configuração do Ambiente do Ciclo de Vida do *Software* (item 11.16), se estiver empacotado separadamente; e

i) as verificações de integridade dos dados para o Código-Executável, se utilizado.

NOTA 2: O Índice de Configuração do *Software* pode ser produzido para uma versão do produto ou pode ser estendida para conter dados para diversas alternativas ou versões sucessivas de produto de *software*.

11.18 RELATÓRIO DE PROBLEMAS

Relatório de Problemas é o meio de identificar e registrar a resolução para comportamentos anômalos do produto, as não conformidades do processo com os planos do *software* e padrões, e deficiências nas evidências do ciclo de vida. Relatórios de problemas devem conter:

a) identificação do item de configuração e/ou da atividade do processo de ciclo de vida do *software* no qual o problema foi observado;

b) identificação dos itens de configuração a serem modificados ou uma descrição do processo a ser modificado;

c) uma descrição do problema que possibilite o entendimento e a solução do mesmo; e

d) uma descrição da ação corretiva realizada para resolver o problema apontado.

11.19 REGISTROS DO GERENCIAMENTO DE CONFIGURAÇÃO DO *SOFTWARE*

Os resultados das atividades dos processos SCM são armazenados nos Registros SCM. Exemplos incluem listas de identificação de configuração, bases ou registros de bibliotecas de *software*, relatórios de histórico de mudanças, registros de arquivo e registros de liberação. Estes exemplos não implicam que registros destes tipos específicos precisam ser produzidos.

NOTA: Devido à natureza do processo SCM, seus resultados são frequentemente incluídos como partes de outras evidências do ciclo de vida.

11.20 REGISTROS DE GARANTIA DA QUALIDADE DO *SOFTWARE*

Os resultados das atividades dos processos SQA são armazenados nos Registros SQA. Eles podem conter revisões de SQA, relatórios de auditorias, minutas de reunião, registros de desvios autorizados do processo, ou registros de revisões de conformidade do *software*.

11.21 SUMÁRIO DE REALIZAÇÕES DO *SOFTWARE*

O Sumário de Realizações do *Software* é o meio principal utilizado para mostrar conformidade com o Plano para Validação do *Software*. Este sumário deve incluir:

a) visão geral do sistema: este capítulo provê uma visão geral do sistema, incluindo a descrição de suas funcionalidades e as suas alocações em *hardware* e *software*, a arquitetura, processadores utilizados, interfaces de *hardware*/*software* e características de segurança. Este capítulo também descreve quaisquer diferenças da visão geral do sistema constante no Plano para Validação do *Software*;

b) visão geral do *software*: este capítulo descreve resumidamente as funcionalidades do *software* com ênfase na segurança proposta e nos conceitos de partição utilizados, e explica diferenças da visão geral do *software* proposta no Plano para Validação do *Software*;

c) considerações de validação: este capítulo reforça as considerações de validação constantes no Plano para Validação do *Software* e descreve quaisquer diferenças;

d) características do *software*: este capítulo descreve o tamanho do Código-Executável, margens de temporização e memória, limitações de recurso e os meios de se medir cada característica;

e) ciclo de vida do *software*: este capítulo resume o real ciclo de vida do *software* e explica qualquer diferença do ciclo de vida e dos processos de ciclo de vida propostos no Plano para Validação do *Software*;

f) evidências do ciclo de vida do *software*: este capítulo referencia às evidências do ciclo de vida do *software* produzido pelos processos de desenvolvimento de *software* e o processo de integração. Ele descreve a

relação dos produtos entre si e com outros dados definidos no sistema, e os meios pelos quais as evidências do ciclo de vida do *software* vão ser disponibilizadas para a autoridade certificadora. Este capítulo também descreve qualquer diferença do Plano para Validação do *Software*;

g) considerações adicionais: este capítulo resume os problemas de validação que merecem a atenção da autoridade de certificação e referencia os itens aplicáveis a estes problemas, tais como documentos do problema ou condições especiais;

h) identificação do *software*: este capítulo identifica a configuração de *software* pelo número de peça e a versão;

i) histórico de mudanças: se aplicável, este capítulo inclui um resumo das mudanças de *software,* com atenção às mudanças realizadas devido a falhas afetando a segurança, e identifica mudanças dos processos de ciclo de vida do *software* desde a última validação;

j) status do *software*: este capítulo contém um resumo das comunicações de problemas não resolvidos no momento da validação, incluindo uma declaração das limitações funcionais; e

k) declaração de conformidade: este capítulo inclui uma declaração de conformidade com este documento e um resumo dos métodos utilizados para demonstrar a conformidade com os critérios especificados nos planos do *software*. Este capítulo também identifica regras adicionais e desvios dos planos do *software*, padrões e este documento.

11.22 DOCUMENTAÇÃO DA ADAPTAÇÃO

11.22.1 Documentação da Adaptação define os itens específicos que podem ser adaptados e que formam um elemento único de alto nível e seus requisitos de baixo nível associados. Os itens do ciclo de vida para adaptação devem conter:

11.22.2 Descrição dos dados no sistema que são adaptados e as relações ou dependências entre os mesmos. Um exemplo de relacionamento ou dependência é a latitude e longitude de uma estação radar. O processador radar e o dispositivo de apresentação precisam ter o mesmo valor e devem ser atualizados conjuntamente quando mudanças ocorrerem.

11.22.3 Descrição dos mecanismos para geração e modificação de dados para adaptação.

12 CONSIDERAÇÕES ADICIONAIS

Os capítulos anteriores do documento deram instruções para satisfação de requisitos de validação em que o fornecedor submete evidências para os processos de ciclo de vida. Este capítulo introduz considerações adicionais sobre os aspectos de validação em relação ao uso de *software* anteriormente desenvolvido, qualificação de ferramentas e métodos alternativos para alcançar os objetivos dos capítulos anteriores deste documento.

12.1 USO DE *SOFTWARE* ANTERIORMENTE DESENVOLVIDO

Este item aborda as questões relacionadas ao uso de *software* anteriormente desenvolvido, incluindo avaliação de modificações, efeito de mudanças nas instalações, no ambiente da aplicação, no ambiente de desenvolvimento, mudança de base de desenvolvimento, considerações de SCM e de SQA. O uso de *software* anteriormente desenvolvido deve estar claro no Plano para Validação do *Software*.

12.1.1 MODIFICAÇÕES A *SOFTWARE* ANTERIORMENTE DESENVOLVIDO

Este item diz respeito a modificações de *software* anteriormente desenvolvido em que as saídas dos processos de ciclo de vida do *software* estão em conformidade com esse documento. Modificação pode resultar de mudanças de requisitos, detecção de erros e/ou melhorias de *software*. Atividades de análise para as modificações propostas incluem:

a) revisar a avaliação de riscos considerando as mudanças propostas;

b) se o nível do *software* é revisado, as instruções para mudança de base citadas em 12.1.4 devem ser consideradas;

c) devem ser analisados ambos, impacto de mudanças de requisitos e impacto de mudanças de arquitetura, incluindo as consequências das mudanças de requisitos sobre outros requisitos e acoplamento entre os componentes que podem resultar em reverificação envolvendo modificação em mais de uma área ;

d) a área afetada pela mudança deve ser determinada. Isso pode ser realizado através de análise de fluxo de dados, análise fluxo de controle, análise temporal e análise de rastreabilidade; e

e) as áreas afetadas pela mudança devem ser reverificadas considerando as instruções do processo de verificação.

12.1.2 MODIFICAÇÃO DE INSTALAÇÃO

Sistemas ou equipamentos CNS/ATM contendo *software* anteriormente desenvolvido e certificado em um determinado nível de garantia de *software* e sobre uma base de validação específica pode ser usado para uma nova instalação. As seguintes instruções devem ser utilizadas nesse caso:

a) o processo de avaliação de riscos da nova instalação determina o nível de garantia de *software* e a linha de base de validação. Nenhum esforço adicional será necessário se na nova instalação o nível de garantia de *software* e linha de base de validação forem os mesmos das instalações anteriores;

b) se as mudanças funcionais são requisitos para a nova instalação, as instruções "Modificações para *Software* Anteriormente Desenvolvido" devem ser aplicadas; e

c) se as atividades de desenvolvimento anteriores não produziram os resultados requeridos para subsidiar os objetivos de segurança para a nova instalação, as instruções "Atualizando uma Linha de Base de Desenvolvimento" devem ser aplicadas.

12.1.3 MUDANÇA DE APLICAÇÃO OU AMBIENTE DE DESENVOLVIMENTO

12.1.3.1 Uso e modificação de *software* anteriormente desenvolvido pode envolver novo ambiente de desenvolvimento, novo computador ou *hardware*, ou a integração com outro *software* diferente do usado na versão original da aplicação.

12.1.3.2 Novo ambiente de desenvolvimento pode impactar (aumentar ou diminuir) algumas atividades dentro do ciclo de vida do *software*. Novo ambiente de aplicação pode requerer atividades adicionais para as atividades do processo de *software*. Instruções sobre a mudança de ambiente de aplicação ou do desenvolvimento incluem:

a) se o novo ambiente de desenvolvimento usa ferramentas de desenvolvimento, as instruções de "Qualificação de Ferramentas" podem ser aplicáveis;

b) o rigor da avaliação de uma mudança da aplicação deve considerar a complexidade e a sofisticação da linguagem de programação. Por exemplo, o rigor utilizado na avaliação de ADA *generics* (polimorfismo) deve ser maior se os parâmetros do *generics* são diferentes na nova aplicação;

c) se um compilador diferente ou conjunto diferente de opções de compilação são usados, resultando num Código-Objeto diferente, os resultados do processo de verificação anteriores a mudança podem não ser válidos e não podem ser usados para a nova aplicação. Nesse caso, os resultados dos testes anteriores podem não validar os critérios de cobertura estrutural da nova aplicação. Similarmente, as otimizações de compilação podem afetar da mesma maneira;

d) se um processador diferente é utilizado:
   - os resultados anteriores das atividades do processo de verificação diretamente ligados à interface *hardware*/*software* não devem ser utilizados para a nova aplicação;
   - os testes de integração de *hardware*/*software* devem ser executados para a nova aplicação;
   - revisões da compatibilidade *hardware*/*software* devem ser repetidas;
   - testes e revisões adicionais de integração *hardware*/*software* podem ser necessários; e

e) a verificação das interfaces de *software* deve ser conduzida quando a interface de *software* for diferente da interface do *software* anteriormente desenvolvido.

12.1.4 TROCA DA BASE DE DESENVOLVIMENTO

12.1.4.1 Instruções que se seguem são para as evidências do ciclo de vida de *software* do desenvolvimento da aplicação anterior que foram consideradas inadequadas ou que não satisfazem aos objetivos desse documento, devido aos objetivos de segurança da nova aplicação. As instruções têm o objetivo de auxiliar na aceitação de:

a) *software* COTS;

b) *software* desenvolvido seguindo outras instruções;

c) *software* desenvolvido antes da existência deste documento; e

d) *software* desenvolvido anteriormente num nível de garantia de *software* inferior.

12.1.4.2 Instruções para atualizar a base de desenvolvimento:

a) os objetivos deste documento devem ser satisfeitos, considerando-se as evidências de ciclo de vida de *software* do desenvolvimento anterior que satisfaçam aos objetivos da nova aplicação;

b) os aspectos de validação de *software* devem ser baseados nas condições de falha e níveis de garantia de *software* determinados pelo processo de avaliação de segurança. Comparações entre condições de falha da aplicação anterior determinarão as áreas que precisam ser melhoradas;

c) os produtos do ciclo de vida de *software* de desenvolvimentos anteriores devem ser avaliados para assegurar que os objetivos do nível de garantia de *software* são satisfeitos para a nova aplicação;

d) engenharia reversa pode ser utilizada para regerar produtos do ciclo de vida de *software* inadequados ou que não satisfaçam os objetivos desse documento. Além de gerar produtos de *software*, atividades adicionais podem ser necessárias para satisfação dos objetivos do processo de verificação;

e) se o uso de histórico de serviço é planejado para satisfazer aos objetivos desse documento na troca de base de desenvolvimento, as instruções do item “Histórico de Serviço” devem ser consideradas; e

f) o fornecedor deve especificar a estratégia para atingir o cumprimento do Plano para Validação de *Software*.

12.2 *SOFTWARE* COMERCIAL (COTS)

Software COTS engloba uma grande gama de *software*, incluindo *software* comprado, *software* extrínseco, *software* legado sem as considerações desta norma. Esse *software* pode ter sido aprovado através do processo de certificação. Podem estar disponíveis dados parciais ou não haver dados para demonstrar conformidade com os objetivos deste documento. Para o resto do capítulo, todo o *software* será referido como COTS.

12.2.1 INTRODUÇÃO

O uso de COTS pode significar que métodos alternativos foram usados para garantir que os objetivos apropriados foram satisfeitos. Esses métodos incluem, não tão somente, experiência de serviço do produto, garantia prévia, reconhecimento de processos,

engenharia reversa, restrição de funcionalidades, métodos formais, avaliações e inspeções. Revisões de conformidade de *software* podem sofrer adaptações com respeito ao método usado para a validação do COTS. Garantia de disponibilidade pode ser combinada com mais de um método para satisfazer aos objetivos.

12.2.2 ESCOPO DO COTS

12.2.2.1 Este item aplica-se somente aos COTS usados para aplicações CNS/ATM e não objetiva alterar ou substituir quaisquer objetivos aplicados a um *software* CNS/ATM, a menos que justificada pelo processo de avaliação de segurança e aceita pela autoridade certificadora.

12.2.2.2 Exemplos de *software* geralmente implementados por COTS são sistemas operacionais, *kernel* de tempo-real, interfaces gráficas de usuário, protocolos de comunicação ou telecomunicações, bibliotecas de tempo de execução, rotinas em nível de bit, e rotinas de manipulação de strings. *Softwares* COTS podem ser comprados separados ou em conjunto com *hardware*, como *workstations*, *mainframes*, equipamentos de comunicação e rede, ou itens de *hardware* (memória, dispositivo de armazenamento, dispositivos de *I/O*). Pode haver situações nas quais o uso de *software* COTS é inevitável, como, por exemplo, nos casos de bibliotecas de código associadas com certos compiladores.

12.2.2.3 Produtos COTS comprados podem variar conforme contrato com o fornecedor. Eles podem se estender aos direitos de licença, código executável, documentação ao usuário e treinamento ao conjunto completo de evidências do ciclo de vida do COTS, incluindo Código-Fonte resultante do desenvolvimento. Confidencialidade da informação COTS relaciona-se ao custo, proteção aos direitos intelectuais e questões legais (posse de *software*, patentes, obrigações e responsabilidade documental). Esses aspectos estão além do escopo deste documento, que se destina apenas a questões de segurança.

12.2.3 ASPECTOS DE SISTEMAS RELACIONADOS AOS COTS EM SISTEMAS CNS/ATM

12.2.3.1 Pode haver a necessidade de integrar *Software* COTS em sistemas ou equipamentos CNS/ATM de alta integridade, no entanto, quanto mais grave a classificação da condição de falha, mais severos são os requisitos de garantia para o sistema e para o *software*. Métodos alternativos podem ser utilizados para aumentar a segurança dos dados para *software* COTS num nível desejado à garantia de *software*. Quando COTS são usados em sistemas CNS/ATM, condições adicionais como planejamento, aquisição e verificação devem ser realizadas.

12.2.3.2 Técnicas de mitigação de riscos podem ser usadas para reduzir a dependência ao COTS. O objetivo dessas técnicas é manter classificação de falhas associada pela redução dos efeitos de comportamentos anômalos do COTS nas funções dos sistemas CNS/ATM. Técnicas de mitigação de riscos podem compreender a combinação de pessoas, procedimentos, equipamento e/ou arquitetura. Por exemplo, meios relacionados à arquitetura podem envolver partição, redundância, monitoramento de segurança, subconjuntos seguros COTS pelo uso de encapsulamento ou *wrappers*, e checagem de integridade de dados.
Referência ao Anexo A desse documento.

12.2.4 PROCESSO DE PLANEJAMENTO COTS

12.2.4.1 O propósito do processo de planejamento COTS é coordenar os processos do ciclo de vida específicos ao COTS e definir métodos e ferramentas necessárias à incorporação do COTS nos sistemas CNS/ATM. O processo de planejamento COTS é uma subatividade do processo de planejamento do fornecedor. A verificação do planejamento COTS tem o objetivo de assegurar que todos os aspectos foram considerados. O processo de planejamento do fornecedor deve documentar o uso de COTS se for previsto. O planejamento COTS inclui planejamento de aspectos COTS, aquisição, verificação, controle de configuração e garantia da qualidade.

12.2.4.2 Como parte do processo de validação, o célere envio dos resultados dos processos de avaliação e seleção COTS é recomendado pela autoridade certificadora.

12.2.4.3 Objetivos do Planejamento COTS

Os objetivos do planejamento COTS são:

a) definir as atividades de aquisição, integração, incluindo considerações adicionais, integração e manutenção;

b) definir dos Critérios de Transição para esses processos e critérios de transição com respeito ao ciclo de vida CNS/ATM; e

c) definir os planos para processos COTS, incluindo critérios de transição, consistentes com os planos do *software* do fornecedor.

12.2.4.4 Atividades do Processo de Planejamento COTS

a) as atividades relacionadas ao processo de planejamento COTS devem avaliar o nível de aplicabilidade dos produtos COTS aos requisitos CNS/ATM. As seguintes considerações devem ser incluídas na avaliação para determinar o nível de esforços envolvidos no uso de COTS;

b) verificar a disponibilidade do produto;

c) definir requisitos (mapeamento dos requisitos CNS/ATM às capacidades do COTS, referência ao item 12.2.5 deste documento);

d) verificar a disponibilidade e aplicabilidade das evidências do ciclo de vida;

e) verificar o nível de integração e extensão dos esforços adicionais;

f) definir as relações entre o planejamento COTS, aquisição COTS e integração COTS. Adicionalmente, relações entre os processos COTS e os processos CNS/ATM apropriados devem ser definidos. Todas as entradas dos processos não precisam estar completas para o processo se iniciar desde que os critérios de transição estejam satisfeitos; e

g) conduzir as revisões que:
   - processo de planejamento COTS e processo de planejamento CNS/ATM estejam consistentes;
   - critérios de transição de processos COTS estejam consistentes com os critérios de transição dos processos CNS/ATM;
   - critérios de transição sejam verificados de modo a assegurar que as saídas dos processos sejam suficientes para se começar o próximo processo;

NOTA: No uso de COTS pode haver a necessidade de código para adaptações, técnicas de mitigação arquiteturais, requisitos derivados e integração COTS específica. Qualquer *software* suplementar para integração do COTS ao sistema CNS/ATM deve ser considerado desenvolvimento de *software* para o sistema CNS/ATM para o qual os objetivos do Anexo A deste documento se aplicam.

12.2.5 PROCESSO DE AQUISIÇÃO COTS

O foco deste item é abordar os aspectos relacionados à aquisição COTS. Existem considerações adicionais não descritas neste documento. O processo de aquisição é composto de definição, avaliação e seleção.

a) definição de requisitos: o processo de definição de requisitos CNS/ATM identifica os requisitos que podem ser satisfeitos pelo *software* COTS que podem ter capacidades além dos requisitos necessários ao sistema CNS/ATM. A definição dessas capacidades pode estar disponível através do fornecedor COTS ou a partir de manuais do usuário, documentos técnicos, dados do produto etc. No modelo da Figura 7, os requisitos de *software* CNS/ATM são a interseção das capacidades COTS com os requisitos CNS/ATM;

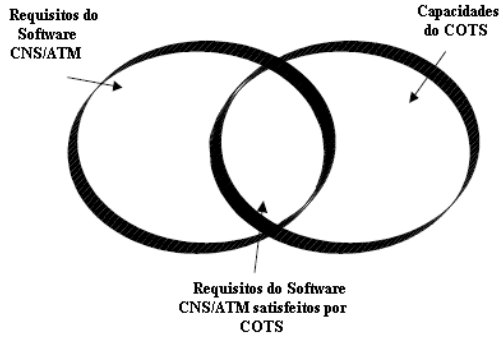

Figura 7 - Interseção dos Requisitos

NOTA 1: Devido a uso de COTS, pode haver requisitos derivados (ex.: requisitos dependentes de plataforma, tratamento de interrupções, tratamento da interface, requisitos de recursos, restrições de uso, tratamento de erros, partição) que devem ser adicionados aos requisitos CNS/ATM.

NOTA 2: Todos os requisitos CNS/ATM satisfeitos pelo *software* COTS e os requisitos derivados resultantes devem passar pelo processo de avaliação de segurança.

b) avaliação: produtos COTS candidatos devem ser avaliados quanto à implementação dos requisitos CNS/ATM, quanto ao efeito dos seus requisitos derivados e quanto ao seu suporte ao nível de garantia de segurança de *software*; e

c) seleção: a seleção é um processo iterativo baseado nos resultados do processo de avaliação e de comparações entre os fornecedores COTS (ex.: capacidades do fornecedor COTS de atender às demandas CNS/ATM, capacidade de controle de versão e manutenção do *software* COTS durante o tempo esperado de vida do sistema CNS/ATM, comprometimento do fornecedor COTS em manter o fornecedor CNS/ATM informado dos erros

detectados e prontidão do fornecedor em atender questões contratuais). Análises devem ser conduzidas para comparar as vantagens do uso do COTS em relação ao desenvolvimento de *software*.

12.2.5.1 Objetivos do Processo de Aquisição COTS

Os objetivos do processo de aquisição COTS são:

a) determinar o grau em que os requisitos CNS/ATM são satisfeitos pelas capacidades COTS;

b) determinar se os produtos do ciclo de vida disponíveis são adequados para os propósitos de segurança;

c) determinar os requisitos derivados, que consistem em:
   – requisitos impostos ao sistema CNS/ATM devido ao uso de COTS; e
   – requisitos para prevenir o uso de capacidades não necessárias de afetar adversamente ao sistema CNS/ATM.

d) garantir que existe compatibilidade entre o COTS e o *hardware* do computador.

12.2.5.2 Atividades do Processo de Aquisição COTS

As atividades do Processo de Aquisição COTS são:

a) analisar e examinar as capacidades do COTS contras os requisitos do sistema CNS/ATM. O propósito dessa análise é determinar quais requisitos CNS/ATM são satisfeitos pelos COTS, a fim de ajudar as comparações entre os produtos candidatos;

b) analisar documentação do *software* COTS. Uma análise de deficiências deve ser realizada contra os objetivos do Anexo A para o nível de garantia de *software* proposto. Essa análise ajuda a comparação dos produtos candidatos. Essa análise é usada para identificar os objetivos que são parcialmente ou totalmente satisfeitos e os que necessitam de métodos alternativos;

c) conduzir análise para identificar requisitos derivados. Essa análise deve incluir todas as capacidades COTS, necessárias ou não. Requisitos derivados podem ser classificados em:
   – requisitos para prevenir os efeitos adversos de qualquer função não necessária do COTS. Isso pode gerar requisitos de isolamento / partição, código *wrapper*, diretivas de código, customização etc.;
   – requisitos impostos pelo COTS selecionado ao sistema CNS/ATM para prevenir efeitos adversos de funções necessárias do COTS (ex.: formatação de entrada, ordem de chamada, inicialização, conversão de dados, recursos, checagem de limites). Isso pode gerar requisitos para o código de interface, diretivas de código, considerações de arquitetura, quantidade de recursos, *glue-code* etc.;

d) garantir que todos os requisitos CNS/ATM satisfeitos pelo *software* COTS, os requisitos derivados resultantes e qualquer documentação pertinente do fornecedor COTS sejam fornecidos ao processo de avaliação de segurança;

e) comprovar que o COTS selecionado tenha compatibilidade com o computador e interfaces do sistema.

12.2.6 PROCESSO DE VERIFICAÇÃO COTS

12.2.6.1 O Processo de Verificação COTS é uma extensão dos processos de produtos CNS/ATM. Em particular, o Processo de Aquisição COTS pode identificar os objetivos que não podem ser satisfeitos pelos meios tradicionais. Para esses objetivos de verificação em que a conformidade não pode ser demonstrada pela documentação do COTS disponível (ex.: projeto ou requisitos), atividades adicionais, incluindo métodos alternativos como engenharia reversa, podem ser usadas, depois do aceite da autoridade certificadora.

12.2.6.2 O uso de meios alternativos para conformidade deve observar as condições:

a) se o processo de avaliação de segurança aceita a justificativa do uso do meio alternativo; e

b) se há aprovação do meio alternativo pela autoridade certificadora.

12.2.6.3 Atividades usadas para especificar métodos alternativos ou combinação de métodos alternativos são consideradas caso a caso. Um exemplo de atividade associada é o uso de experiência de serviço, descrito no item 12.2.6.6.

12.2.6.4 Objetivos do Processo de Verificação COTS

Não há objetivos adicionais impostos ao sistema CNS/ATM devido ao uso de COTS.

12.2.6.5 Atividades do Processo de Verificação COTS

As atividades típicas de verificação para *software* COTS para atingir aos objetivos dos processos incluem:

a) revisões e análises dos requisitos do *software* CNS/ATM satisfeitos pelo COTS;

b) testes baseados em requisitos dos requisitos do *software* CNS/ATM satisfeitos pelo COTS;

c) verificação de qualquer *software* suplementar (*glue-code*, partição, *wrappers*); e

d) verificação da integração do COTS no sistema CNS/ATM.

12.2.6.6 Uso de Experiência de Serviço como Crédito para Fins de Validação de *Software* COTS

O uso de experiência de serviço para fins de validação depende de dois fatores: suficiência e relevância. Suficiência da documentação da experiência de serviço que pode ser disponível através do uso de sistema paralelo com o sistema operacional no ambiente operacional, longa duração de simulação de novos sistemas CNS/ATM e múltiplas operações paralelas executadas em diferentes localidades. Documentação da relevância da experiência de serviço pode ser disponível para sistemas CNS/ATM pelo reuso de *software* COTS em uso nos sistemas CNS/ATM, ou verificação do sistema CNS/ATM e atividades pré-operação, como treinamento. Para *software* COTS sem precedente uso nos sistemas CNS/ATM, muitos

processos podem ser usados para coletar experiência de serviço; exemplos incluem processo de validação, processo de treinamento do operador, testes de qualificação, avaliação operacional do sistema e demonstrações em campo. As seguintes considerações se aplicam ao acúmulo de experiência de serviço:

a) o uso, condições de uso e resultados da experiência de serviço devem ser definidos, avaliados quanto à segurança operacional e submetidos à autoridade certificadora;

b) o ambiente operacional durante a experiência de serviço deve ser avaliado quanto à relevância para o uso almejado no CNS/ATM. Se há diferenças entre o ambiente operacional da aplicação, entre o existente e almejado, verificações adicionais devem ser realizadas para assegurar que a aplicação COTS irá operar no sistema CNS/ATM como desejado nesse ambiente. Deve ser assegurado que as capacidades COTS serão exercitadas em todos os modos operacionais. E análise deve ser feita para verificar que as permutações relevantes dos dados de entradas foram executadas;

c) quaisquer mudanças no COTS durante o tempo de experiência de serviço devem ser analisadas. Deve-se avaliar quando essas mudanças alteram a aplicabilidade da documentação de experiência de serviço para o período precedente às mudanças;

d) todos os problemas no uso devem ser avaliados pelo potencial de efeito adverso na operação do sistema CNS/ATM. Quaisquer problemas durante o período de experiência de serviço, em que a implicação tem resultado adverso não consistente com a avaliação de segurança, devem ser documentados. Quaisquer problemas devem ser considerados como falha. Uma falha invalida o uso da documentação de experiência de serviço para o período de experiência precedente à correção do problema;

e) capacidades COTS que não são necessárias para satisfazer aos requisitos CNS/ATM devem mostrar que não têm efeito adverso nas operações CNS/ATM; e

f) o período da experiência de serviço deve ser acumulado em horas de serviço. O número de cópias/exemplares em serviço deve ser levado em consideração para calcular o período de experiência, dado que cada cópia e o ambiente operacional associado mostram relevância, e que a uma cópia conta um percentual pré-negociado do total.

NOTA: Documentação disponível do COTS pode não conseguir demonstrar satisfação de todos os objetivos dos processos. Por exemplo, testes de requisitos de alto nível para operação robusta e normal podem ser demonstrados, mas os mesmos testes para requisitos de baixo nível podem não alcançar resultados satisfatórios. O uso de experiência de serviço pode ser proposto para demonstrar a conformidade aos objetivos dessas verificações. A extensão da experiência de serviço a ser usada é baseada em julgamento técnico e experiência com as operações dos sistemas CNS/ATM. Os resultados dos modelos de confiabilidade de *software* não podem ser usados para justificar tempo de experiência de serviço. Um método possível para diferentes níveis de garantia é dado abaixo:

- não pode ser aplicado para AL1;

- a duração da experiência de serviço sem falhas para AL2 com a duração negociada com a autoridade certificadora;
- mínimo de um ano (8760 horas) de experiência de serviço sem falhas para AL3;
- mínimo de seis meses (4380 horas) de experiência de serviço sem falhas para AL4; e
- os objetivos de AL5 são satisfeitos geralmente sem a necessidade de meios alternativos.

12.2.7 PROCESSO DE CONTROLE DE CONFIGURAÇÃO COTS

O gerenciamento do controle de configuração do fornecedor COTS não está sob o alcance do controle de configuração do sistema CNS/ATM. O controle de configuração deve incluir controle de versões de COTS.

12.2.7.1 Objetivos do Processo de Controle de Configuração COTS

Os objetivos do processo de controle de configuração são que:

a) a configuração específica e a documentação (ex.: *software*, documentação de adaptação) sejam identificadas no sistema de controle de configuração de *software* CNS/ATM;

b) o reporte de problemas do sistema CNS/ATM inclua gerenciamento dos problemas encontrados no COTS;

c) o processo de Controle de mudanças assegure que a incorporação de novas versões de COTS é controlada; e

d) a configuração específica de COTS e a documentação sejam incluídas no repositório CNS/ATM.

12.2.7.2 Atividades do Processo de Controle de Configuração COTS

As atividades associadas ao controle de configuração COTS incluem:

a) definir método de identificação para configuração COTS;

NOTA: O método de identificação pode ser baseado na identificação suprida pelo fornecedor COTS e qualquer informação adicional como versão ou data de compra.

b) reportar problemas CNS/ATM deve incluir gerenciamento dos problemas encontrados nos COTS, e um mecanismo bidirecional deve ser estabelecido com o fornecedor COTS;

c) estabelecer processo de controle de mudanças para novas versões de COTS incorporadas;

d) analisar o impacto de mudanças na base COTS antes da incorporação de novas versões de COTS; e

NOTA: Gerar e observar a lista de mudanças (correções e mudanças, ou funções extintas) implementadas para cada nova versão do COTS.

e) garantir que o repositório CNS/ATM inclua a configuração e a documentação COTS.

NOTA: Deve haver considerações sobre a obsolescência tecnológica para acesso de documentação e contratos.

12.2.8 GARANTIA DA QUALIDADE COTS

O processo de garantia da qualidade COTS deve avaliar os processos e evidências geradas para assegurar que os objetivos associados ao COTS são satisfeitos.

NOTA: É recomendado que o processo de garantia do fornecedor COTS esteja coordenado com o processo de qualidade do sistema ou equipamento CNS/ATM, quando possível.

12.2.9 OBJETIVOS ESPECÍFICOS AOS COTS

Os objetivos contidos no Anexo B (Tabelas B-1 a B-3) devem ser satisfeitos adicionalmente aos objetivos contidos no Anexo A (Tabelas A-1 a A-10)

12.3 PROCESSO DE ADAPTAÇÃO

12.3.1 Adaptação é utilizada para customizar os elementos do sistema CNS/ATM para os propósitos designados numa localidade específica. Esses sistemas são geralmente configurados para acomodar as características específicas do local. Essas dependências de localidade são desenvolvidas em conjuntos de adaptação. Adaptação inclui:

a) os dados que configuram o *software* para um dado local geográfico; e

b) os dados que configuram uma *workstation* para as preferências e/ou funções de um operador.

12.3.2 Exemplos incluem, mas não se limitam à:

a) os dados geográficos – latitude e longitude da instalação do radar;

b) os dados de ambiente – dados selecionáveis pelo operador;

c) os dados de Espaço Aéreo – dados do setor específico; e

d) os procedimentos – customização operacional para prover o papel desejado do operador.

12.3.3 Adaptação pode ter a forma de uma mudança na base de dados dos parâmetros da aplicação ou pode ter a forma de opções pré-programadas. Em alguns casos, adaptação envolve a religação do código para incluir bibliotecas diferentes. Importante, isso não pode ser confundido com "recompilação", em que uma nova versão do código é gerada. Nesse último caso, o *software* representa uma nova base e as instruções do capítulo 8 se aplicam. Instruções referentes a *software* modificável pelo usuário podem ajudar a determinar como garantir a adaptação.

12.3.4 Adaptação deve ser desenvolvida para o mesmo nível de confiança do código no qual ela se aplica.

12.3.5 OBJETIVOS DO PROCESSO DE ADAPTAÇÃO

O propósito do processo de adaptação é identificar dados que precisam ser adaptados durante a configuração ou operação do sistema CNS/ATM, para especificar os mecanismos para realizar a adaptação e assegurar que os requisitos para estes dados são

capturados como parte dos requisitos de alto nível e/ou associados aos requisitos de baixo nível. Os objetivos para o processo são:

a) definir os dados que podem ser adaptados para uma localidade particular;

b) definir os mecanismos para gerar e modificar os dados de adaptação para cada localidade; e

c) satisfazer os objetivos para verificação, SQA, SCM e processo de validação (referências ao Anexo A desse documento) para cada localidade.

12.3.6 ATIVIDADES DO PROCESSO DE ADAPTAÇÃO

As atividades do processo de adaptação incluem, mas não são limitados a:

a) nos processos do ciclo de vida deve constar o uso da adaptação; e

b) os requisitos de alto nível devem identificar os dados a serem adaptados.

12.3.6.1 Dos documentos de planejamento devem constar os mecanismos para gerar e modificar os dados de adaptação.

12.3.7 CONSIDERAÇÕES DE GERENCIAMENTO DE CONFIGURAÇÃO

Se *software* anteriormente desenvolvido é usado, o processo de gerenciamento de configuração para a nova aplicação deverá ser incluído, adicionalmente às instruções do capítulo 8:

a) rastreabilidade do produto de *software* e produtos de *software* do ciclo de vida da aplicação anterior para a nova aplicação; e

b) controle de mudanças que habilitam o reporte de problemas, resolução de problemas e rastreabilidade de mudanças nos componentes de *software* usados em mais de uma aplicação.

12.3.8 CONSIDERAÇÕES DE GARANTIA DA QUALIDADE

Se *software* anteriormente desenvolvido é usado, o processo de garantia da qualidade para a nova aplicação deverá ser incluído, adicionalmente às instruções do capítulo 9:

a) garantia de que os componentes de *software* satisfazem ou excedem aos critérios do nível de garantia de *software* para a nova aplicação; e

b) garantia de que as mudanças para os processos do ciclo de vida são documentadas nos planos do *software*.

12.4 QUALIFICAÇÃO DE FERRAMENTAS

12.4.1 Qualificação de ferramentas é necessária quando os processos desse documento são eliminados, reduzidos ou automatizados pelo uso de uma ferramenta de *software* sem que seus resultados sejam verificados assim como está especificado no capítulo 7. O uso de ferramentas de *software* para automação de atividades do ciclo de vida de *software* pode ajudar a satisfazer aos objetivos de segurança do sistema, na medida em que eles podem forçar a conformidade aos padrões de desenvolvimento através do uso de verificadores automáticos.

12.4.2 O objetivo do processo de qualificação é assegurar que as ferramentas provejam confiabilidade no mínimo equivalente ao do processo que ele elimina, reduz ou automatiza. Se a separação das funções da ferramenta pode ser demonstrada, somente aquelas funções que são usadas para eliminar, reduzir ou automatizar as atividades dos processos de ciclo de vida, e cujas saídas não são verificadas, precisam ser qualificadas.

12.4.3 Somente ferramentas determinísticas podem ser qualificadas, que são ferramentas que podem produzir os mesmos resultados para as mesmas entradas quando operando num mesmo ambiente. O processo de qualificação de ferramentas pode ser aplicado tanto para uma única ferramenta como para um conjunto de ferramentas.

12.4.4 Ferramentas de *software* podem ser classificadas em dois tipos:

a) ferramentas de desenvolvimento: ferramentas cuja saída faz parte do *software* desenvolvido, podendo introduzir erros. Por exemplo, a ferramenta que gera código diretamente dos requisitos de baixo nível poderia ser qualificada se o código gerado não é verificado como especificado no capítulo 7; e

b) ferramentas de verificação: ferramentas que podem não introduzir erros, mas podem falhar na detecção dos mesmos. Por exemplo, um analisador estático, que automatiza as atividades do processo de verificação, deve ser qualificado como se a função que realiza não é verificada por outra atividade. Verificadores de tipo (*Type checkers*), ferramentas de análise e ferramentas de testes são outros exemplos.

12.4.5 As instruções para qualificação de ferramentas incluem:

a) as ferramentas devem ser verificadas de acordo como o tipo especificado acima;

b) a combinação de ferramentas de desenvolvimento e ferramentas de verificação deve ser qualificada para conformidade com as instruções do item 12.4.8, “Critérios de Qualificação de Ferramentas de Desenvolvimento”, a menos que a separação entre as funções possa ser demonstrada; e

c) os objetivos do processo de gerenciamento de configuração e do processo de garantia da qualidade do nível de garantia de *software* da aplicação desenvolvida se aplicam às ferramentas a serem qualificadas.

12.4.6 Os objetivos do processo de verificação para ferramentas de desenvolvimento são descritos no item 12.4.8, "Critérios de Qualificação de Ferramentas de Desenvolvimento", alínea "d".

12.4.7 A ferramenta pode ser qualificada para uso específico no sistema onde a intenção de seu uso esteja documentada no Plano para Validação de *Software*. Uso da ferramenta para outros sistemas pode necessitar outra qualificação.

12.4.8 CRITÉRIOS DE QUALIFICAÇÃO DE FERRAMENTAS DE DESENVOLVIMENTO

Os critérios para qualificação de ferramentas de desenvolvimento incluem:

a) se a ferramenta será qualificada, o processo de desenvolvimento para a ferramenta deve satisfazer aos mesmos objetivos dos processos de desenvolvimento do *software* desenvolvido;

b) o nível associado à ferramenta deverá ser o mesmo do *software* que ele produz, a menos que o fornecedor possa justificar a redução do nível de garantia de *software* para a autoridade certificadora;

NOTA: A redução do nível de garantia de *software* pode ser baseada na significância das atividades do processo de verificação a serem eliminadas, reduzidas ou automatizadas, com respeito ao conjunto de atividades de verificação. A significância é a função do tipo de atividade de verificação a ser eliminada, reduzida ou automatizada. Por exemplo, a atividade de verificação para conformidade do Código-Fonte com o padrão de endentação é menos significativo que a atividade de verificação para conformidade do Código-Executável com os requisitos de alto nível. Existe a probabilidade de que outras atividades de verificação detectem a mesmo erro.

c) o fornecedor deve demonstrar que as ferramentas estão em conformidade com os Requisitos Operacionais de Ferramentas (item 12.4.12). Essa demonstração pode envolver um período de avaliação durante o qual as saídas da ferramenta são analisadas, gravadas e corrigidas; e

d) ferramentas de desenvolvimento devem ser verificadas com respeito à correção, consistência e completeza dos Requisitos Operacionais da Ferramenta e verificação contra esses requisitos. A verificação das ferramentas pode ser realizada pela:
   - revisão dos Requisitos de Ferramentas Operacionais descritas no item 7.4.1, alíneas "a" e "b";
   - demonstração de que a ferramenta está conforme com os Requisitos Operacionais da Ferramenta sobre as condições normais de operação;
   - demonstração de que a ferramenta está conforme os Requisitos de Ferramentas Operacionais, enquanto executando sob condições anormais, incluindo distúrbios externos e falhas selecionadas aplicadas à ferramenta e ao ambiente;
   - análise de cobertura de requisitos e testes adicionais para completar a cobertura de requisitos;
   - análise de cobertura estrutural apropriada para o nível de garantia de *software* da ferramenta;

- testes de robustez para ferramentas com complexidade do fluxo de dados ou controle, especificado no item 7.5.5.2, apropriado ao nível de garantia de *software* da ferramenta; e
- análise de erros potenciais produzidos pela ferramenta, para confirmar a validade do Plano de Qualificação da Ferramenta.

12.4.9 CRITÉRIOS DE QUALIFICAÇÃO DE FERRAMENTAS DE VERIFICAÇÃO

Os critérios de qualificação de ferramentas de verificação devem ser atingidos pela demonstração de que a ferramenta está conforme com os Requisitos de Ferramentas Operacionais sob condições normais de operação.

12.4.10 EVIDÊNCIAS PARA QUALIFICAÇÃO DE FERRAMENTAS

As instruções para evidências para qualificação de ferramentas incluem:

a) quando qualificando uma ferramenta, o Plano para Validação de *Software* do *software* CNS/ATM deve especificar as ferramentas a serem qualificadas e as evidências para demonstração da qualificação;

b) as evidências da qualificação devem ser controladas como Categoria de Controle I (CCI) para ferramentas de desenvolvimento e CC2 para ferramentas de verificação; e

c) para ferramentas de desenvolvimento, as evidências da qualificação devem ser consistentes com as evidências do capítulo 11 e devem ter as mesmas características e conteúdo como um *software* CNS/ATM, considerando que:
   - o Plano de Qualificação da Ferramenta satisfaz aos mesmos objetivos do Plano para Validação de *Software* do *software* CNS/ATM; e
   - os Requisitos Operacionais da Ferramenta satisfazem aos mesmos objetivos dos Requisitos de *Software* do *software* CNS/ATM.

NOTA: O Sumário de Realizações da Ferramenta satisfaz aos mesmos objetivos do Sumário de Realizações de *Software* CNS/ATM

12.4.11 PLANO DE QUALIFICAÇÃO DE FERRAMENTAS

Para ferramentas de desenvolvimento a serem qualificadas, o Plano de Qualificação da Ferramenta descreve o processo de qualificação. Esse plano inclui:

a) identificação da ferramenta;

b) detalhes dos créditos da ferramenta, isto é, as atividades do processo de verificação que são eliminadas, reduzidas ou automatizadas;

c) o nível de garantia de *software* proposto para a ferramenta;

d) a descrição da arquitetura da ferramenta;

e) as atividades de qualificação a serem realizadas; e

f) as evidências da qualificação a serem produzidas.

12.4.12 REQUISITOS OPERACIONAIS DA FERRAMENTA

Os Requisitos Operacionais da Ferramenta descrevem a funcionalidade da ferramenta. Esses dados incluem:

a) a descrição das funções da ferramenta e as características técnicas. Para ferramentas de desenvolvimento, isso inclui as atividades de desenvolvimento de *software* realizadas pela ferramenta;

b) informação ao usuário, como guias de instalação e manuais de uso;

c) a descrição do ambiente operacional da ferramenta; e

d) para ferramentas de desenvolvimento, as respostas esperadas para ferramenta sob condições anormais de operação.

12.4.13 VALIDAÇÃO DA QUALIFICAÇÃO DA FERRAMENTA

12.4.13.1 A autoridade certificadora dá o aval para uso da ferramenta em dois passos:

a) em um primeiro momento, para ferramentas de desenvolvimento, com a aprovação do Plano de Qualificação da Ferramenta. E para ferramentas de verificação, com a aprovação do Plano para Validação do *Software* CNS/ATM; e

b) finalmente, para ferramentas de desenvolvimento, com a aprovação do Sumário de Realizações de Ferramenta. E para ferramentas de verificação, com a aprovação do Sumário de Realizações do *Software* CNS/ATM.

12.5 MÉTODOS ALTERNATIVOS DE CONFORMIDADE

12.5.1 Alguns métodos não foram discutidos nos capítulos anteriores por não estarem devidamente maduros quando esse documento foi escrito, ou pela limitada aplicabilidade ao *software* CNS/ATM. Não é intenção deste documento restringir a implementação de nenhum método corrente ou futuro. Qualquer método alternativo discutido neste item não é considerado como uma alternativa ao conjunto de métodos recomendados por este documento, mas pode ser usado para satisfazer a um ou mais objetivos.

12.5.2 Métodos alternativos podem ser usados para dar suporte a outro. Por exemplo, métodos formais podem ajudar a qualificação de ferramentas, ou uma ferramenta qualificada pode assistir ao uso de métodos formais.

12.5.3 Um método alternativo não pode ser considerado isolado dos processos de desenvolvimento. O esforço para obter crédito para validação de um método alternativo é dependente do nível de garantia de *software* e do impacto do método alternativo nos processos do ciclo de vida de *software*.

12.5.4 Instruções para uso de métodos alternativos devem garantir que:

a) um método alternativo deve mostrar que satisfaz aos objetivos deste documento;

b) o fornecedor deve especificar no Plano para Validação de *Software*, sob a aprovação da autoridade certificadora, o seguinte:
   - relatório de impacto do método proposto nos processos de desenvolvimento de *software;*
   - relatório de impacto do método proposto nos produtos do ciclo de vida de *software;* e
   - exposição das razões para o uso do método alternativo que mostram que os objetivos de segurança são satisfeitos.

c) a exposição de razões deve ser consubstanciada por planos do *software*, processos, resultados esperados e evidências para uso do método.

12.5.5 MÉTODOS FORMAIS

12.5.5.1 Métodos formais envolvem o uso de lógica formal, matemática discreta e uso de linguagem computacional para aprimorar a especificação e verificação de *software*.

12.5.5.2 Esses métodos formais podem produzir uma implementação cujo comportamento operacional é conhecido com confiabilidade dentro de um domínio definido. Na aplicação mais intensa, métodos formais podem ser equivalentes à análise exaustiva dos requisitos do sistema. Essas análises podem prover:

a) evidências que o sistema é completo e correto com respeito aos requisitos; e

b) determinação de qual código, requisitos de *software*, ou arquitetura de *software* satisfazem ao próximo nível mais alto dos requisitos de *software*.

12.5.5.3 O objetivo da aplicação de métodos formais é prevenir e eliminar erros nos requisitos, do projeto e na codificação durante os processos de desenvolvimento. Assim, métodos formais são complementares às atividades de testes.

12.5.5.4 Testes podem mostrar que requisitos funcionais são satisfeitos e detectar erros, métodos formais podem aumentar a confiança de que comportamentos anômalos não ocorrerão ou que serão improváveis.

12.5.5.5 Métodos formais podem ser aplicados ao desenvolvimento de *software* com a consideração dos seguintes fatores:

a) níveis de refinamento: o uso de métodos formais começa pela especificação dos requisitos de alto nível na linguagem formal e pela verificação de provas formais de que a especificação satisfaz aos requisitos de sistemas, especialmente às restrições da operação aceitável. Depois, demonstra-se que os requisitos de baixo nível atendem aos requisitos de alto nível. Fazendo esse processo até o Código-Fonte, evidências de que o *software* satisfaz aos requisitos de sistemas são demonstradas. A aplicação de métodos formais pode começar e parar em níveis consecutivos de refinamentos de projeto, provendo evidências de que aqueles níveis de requisitos são especificados corretamente; e

b) cobertura de requisitos e arquitetura de *software*: métodos formais podem ser aplicados aos requisitos de *software,* tais que:
   - estejam relacionados à segurança;
   - possam ser definidos com matemática discreta; e
   - envolva comportamento complexo, como concorrência, processamento distribuído, gerenciamento de redundância e sincronização.

12.5.5.6 Esses critérios podem ser usados para determinar o conjunto de requisitos num nível de refinamento de projeto para o qual métodos formais são aplicados.

a) grau de rigor: métodos formais incluem os seguintes níveis de rigor em escala ascendente:
   - especificações formais sem provas;
   - especificações formais com provas manuais; e

– especificações formais com checagem ou geração de provas automática.

12.5.5.7 O uso de especificações formais conduz a requisitos não ambíguos. Prova manual é um processo bem entendido que pode ser usado quando há poucos detalhes. Checagem ou geração automática de provas pode ser usada para ajudar o processo de prova manual e oferecer um grau maior de confiança, especialmente para as provas mais complicadas.

12.5.6 TESTE EXAUSTIVO DAS ENTRADAS

Há situações nas quais o componente de *software* de um sistema ou equipamento CNS/ATM é simples e isolado, tal que o conjunto de entradas e saídas pode ser limitado. Nesse caso, é possível demonstrar que testes exaustivos desse espaço de entradas podem substituir as atividades do processo de verificação. Para esse método alternativo, o fornecedor deve incluir:

a) uma completa definição do conjunto de entradas válidas e suas saídas;

b) uma análise que confirme o isolamento das entradas de *software;*

c) exposição de razões para uso dos procedimentos e casos de testes com exaustão de entradas; e

d) os casos, procedimentos e resultados de testes.

12.5.7 CONSIDERAÇÕES PARA VERIFICAÇÃO DE *SOFTWARE* DISSIMILAR MÚLTIPLA-VERSÃO

12.5.7.1 As instruções que se seguem dizem respeito ao processo de verificação quando se aplica *software* dissimilar múltipla-versão. Se o processo de verificação é modificado, por causa do uso de *software* dissimilar múltipla-versão, evidências devem ser providas de que os objetivos do processo de verificação são satisfeitos e que equivalente detecção de erros para cada versão de *software* é obtida.

12.5.7.2 As técnicas utilizadas para produção de *software dissimilar* ou combinação das mesmas são as que se seguem:

a) o Código-Fonte é implementado em duas ou mais linguagens de programação;

b) o Código-Objeto é gerado usando dois ou mais diferentes compiladores;

c) cada versão de *software* do código é executada em um processador dissimilar separado, ou em um processador com meios de prover a separação entre as diferentes versões;

d) os requisitos de *software*, o projeto e o Código-Fonte são desenvolvidos por duas ou mais equipes de desenvolvimento ou diferentes ambientes de desenvolvimento, e/ou cada versão é verificada usando diferentes ambientes de desenvolvimento;

e) o código executável é ligado e carregado usando dois ou mais programas de edição diferentes e dois ou mais carregadores diferentes; e

f) os requisitos de *software*, o projeto e/ou o Código-Fonte são desenvolvidos em conformidade com dois ou mais diferentes padrões para Requisitos de *Software*, de Projeto e de Código respectivamente.

12.5.7.3 Quando múltiplas versões de *software* são usadas, os métodos de verificação podem ser modificados em relação àqueles usados para verificar o *software* de versão única. Eles podem ser aplicados às atividades de desenvolvimento de *software multi-thread*, assim como equipes de desenvolvimento múltiplas e separadas. O processo de verificação é dependente da combinação do *hardware* e da arquitetura de *software* desde que afete a dissimilaridade das múltiplas versões de *software*. Os seguintes objetivos adicionais ao processo de verificação devem ser atingidos:

a) demonstrar que os requisitos de compatibilidade entre versões são satisfeitos, incluindo a compatibilidade durante operações normais e anômalas e estados de transição; e

b) demonstrar que a detecção de erros equivalente é alcançada.

12.5.7.4 Outras mudanças nas atividades do processo de verificação podem ser aprovadas pela autoridade certificadora, desde que essas mudanças sejam consubstanciadas por exposição de razões que confirmem a cobertura de verificação equivalente.

12.5.7.5 Independência de *Software* Dissimilar Múltipla-Versão

Quando *software* dissimilar múltipla-versão é desenvolvido, independentemente, usando um método gerencial, o processo de desenvolvimento tem o potencial de revelar certas classes de erros em que a verificação de cada versão é equivalente à verificação independente do processo de desenvolvimento de *software*. Para efetivar as vantagens desse potencial, as instruções devem incluir que:

a) o fornecedor deve demonstrar que diferentes times, com limitada interação, desenvolveram cada versão de *software,* dos requisitos de *software*, do projeto e do Código-Fonte; e

b) análises de cobertura dos testes independentes foram conduzidas como uma única versão.

12.5.7.6 Verificação referente a Múltiplos Processadores

12.5.7.6.1 Quando cada versão de *software* dissimilar operar em um tipo de processador diferente, a verificação de alguns aspectos de compatibilidade do código com o processador (item 7.5.6, alínea "a") pode ser trocada pela verificação separada de que os diferentes processadores produzem os resultados corretos.

12.5.7.6.2 Essa verificação consiste em testes de integração em que os resultados dos múltiplos processadores são comparados em casos de testes baseados em requisitos. O fornecedor deve mostrar que:

a) detecção de erros equivalente é alcançada;

b) cada processador foi projetado por um desenvolvedor diferente; e

c) os resultados das múltiplas versões são equivalentes.

12.5.7.7 Verificação de Código-Fonte Múltipla-Versão

12.5.7.7.1 As instruções para análise de cobertura estrutural (item 7.5.5.2) podem ser modificadas para sistemas ou equipamentos CNS/ATM usando *software* dissimilar múltipla-versão.

12.5.7.7.2 Análise pode ser realizada no nível de código, mesmo que o Código-Objeto não seja diretamente rastreado às linhas de Código-Fonte, conquanto o fornecedor demonstre que:

a) cada versão de *software* é codificada usando uma linguagem de programação diferente; e

b) cada compilador usado é de um desenvolvedor diferente.

12.5.7.8 Qualificação de Ferramentas para *Software* Dissimilar Múltipla-Versão

Se *software* dissimilar múltipla-versão foi usado, o processo de qualificação da ferramenta pode ser modificado se, também, evidências são disponíveis de que as ferramentas de desenvolvimento de *software* múltiplo são dissimilares. Isso depende da demonstração de equivalência da atividade do processo de verificação no desenvolvimento de múltiplas versões de *software* usando ferramentas de *software* dissimilar. O fornecedor deve demonstrar que:

a) cada ferramenta foi obtida de diferentes desenvolvedores; e

b) cada ferramenta tem um projeto diferente.

12.5.7.9 Simulador e Verificação de *Software* Dissimilar Múltipla-Versão

Se, separadamente, simuladores dissimilares são usados para verificar versões de *software* dissimilar múltipla-versão, então a qualificação desses simuladores pode ser modificada. Isso depende da demonstração da equivalência da atividade do processo de verificação de *software* a simulação das múltiplas versões de *software* usando múltiplos simuladores. A menos que isso possa ser justificado como não necessário, para múltiplos simuladores serem dissimilares, as seguintes evidências devem estar disponíveis:

a) cada simulador foi desenvolvido por uma equipe diferente;

b) cada simulador tem diferentes requisitos, um projeto diferente, assim como, uma linguagem de programação diferente; e

c) cada simulador executa em processadores diferentes.

NOTA: Quando um sistema multiprocessado, usando *software* dissimilar múltipla-versão, é executado em processadores idênticos, pode ser difícil demonstrar dissimilaridade dos simuladores porque é baseado na informação obtida de fontes comuns, o fabricante do processador.

12.5.8 MODELOS DE CONFIABILIDADE

Com o objetivo de desenvolver requisitos numéricos para as probabilidades em *software* CNS/ATM, modelos para estimação de probabilidades de erros de *software* foram desenvolvidos. A conclusão alcançada foi, no entanto, que atualmente não existem métodos com confiabilidade em nível adequado para esse propósito. Por essa razão, o documento não provê instruções sobre taxas de erros de *software*. Se o fornecedor propõe o uso de modelos

de confiabilidade para validação, explicações para o modelo devem ser incluídas no Plano para Validação de *Software*, com o aceite da autoridade certificadora.

12.5.9 HISTÓRICO DE SERVIÇO

12.5.9.1 Crédito para validação do *software* pode ser dado mediante a demonstração de uso com o histórico de serviço. A aceitação desse método depende de:

a) gerência de configuração de *software*;

b) efetividade da atividade de reporte de problemas;

c) estabilidade e maturidade do *software*;

d) relevância do ambiente do histórico de serviço;

e) taxas de erros reais e histórico de serviço; e

f) impacto de modificações.

12.5.9.2 Instruções para uso de histórico de serviço incluem:

a) o fornecedor deve mostrar que o *software* e as evidências associadas usadas para conformidade aos objetivos de segurança de sistema estavam com gerenciamento de configuração por todo o histórico de serviço;

b) o fornecedor deve mostrar que o relatório de problemas, durante o histórico de serviço, assegura a representatividade dos dados disponíveis e que os problemas em serviço foram reportados e armazenados, e que são recuperáveis;

c) mudanças de configuração durante o histórico de serviço devem ser identificadas e seu efeito analisado para confirmar a estabilidade e maturidade do *software*. Mudanças não controladas no código executável durante o histórico de serviço podem invalidar o uso do histórico de serviço;

d) o uso almejado do *software* deve ser analisado para mostrar a relevância do histórico de serviço;

e) se os ambientes operacionais das aplicações existentes e propostas forem diferentes, verificações adicionais devem confirmar a conformidade com os objetivos de segurança do sistema;

f) a análise das mudanças de configuração e do ambiente do histórico de serviço pode requerer o uso de requisitos de *software* e do projeto para confirmar a aplicabilidade do ambiente do histórico de serviço;

g) se o *software* é um subconjunto de um *software* que é ativo durante o período de serviço, então a análise deve confirmar a equivalência do novo ambiente com o ambiente anterior e determinar os componentes de *software* que não foram executados durante a operação normal;

NOTA: Verificações adicionais podem ser necessárias para confirmar a conformidade com os objetivos de segurança do sistema para esses componentes.

h) o histórico de relatório de problemas deve ser analisado para determinar como e quais os problemas relacionados à segurança foram corrigidos;

i) os problemas que são indicativos de processos inadequados, como projeto e erros na codificação, devem ser indicados separadamente daqueles cuja causa esteja fora do escopo desse documento, como problemas de requisitos de *hardware* ou requisitos de sistemas; e

j) os dados descritos acima e os itens a seguir devem ser especificados no Plano para Validação de *Software*:
   - análise da relevância do ambiente do histórico de serviço do produto;
   - a extensão do período e as explicações sobre o cálculo de horas de serviço, incluindo os fatores como modos operacionais, número de cópias independentes em operação na instalação e em serviço, e a definição de "operação normal" e "tempo de operação normal";
   - definição do que foi contado como erro e as explicações para essa definição; e
   - taxas de erros propostas como aceitáveis e explicações, dentro do período do histórico de serviço, em relação à segurança de sistemas e taxas de erros propostas. Se a taxa de erros é maior que o identificado no plano, esses erros devem ser analisados e suas análises deverão ser revisadas pela autoridade certificadora.

13 VISÃO GERAL DA CERTIFICAÇÃO

Este capítulo, que é a visão geral do processo de certificação com respeito à validação do *software* de sistemas e equipamentos CNS/ATM, é provido para fins de informação apenas. A autoridade certificadora considera o *software* como parte integrante de um sistema ou equipamento CNS/ATM, ou seja, a autoridade certificadora não certifica de forma isolada o *software* como produto único.

13.1 BASE DE CERTIFICAÇÃO

13.1.1 A autoridade certificadora estabelece uma base de certificação para o sistema ou equipamento CNS/ATM consultando o fornecedor.

13.1.2 Para sistemas ou equipamentos modificados, a autoridade certificadora considera o impacto que a modificação tem sobre a base de certificação originalmente estabelecida. Em alguns casos, a base de certificação para uma modificação pode não mudar a base de certificação original. No entanto, os meios para mostrar conformidade do produto original podem não se aplicar para demonstração de conformidade do produto modificado.

13.2 VALIDAÇÃO DO *SOFTWARE*

A autoridade certificadora avalia se o "Plano para Validação do *Software*" é completo e consistente com a demonstração de conformidade acordada para satisfazer à base de certificação. Além disso, verifica se o nível de *software* proposto pelo fornecedor é consistente com a avaliação de segurança do sistema e com os outros documentos do ciclo de vida. A autoridade certificadora informa ao fornecedor os problemas levantados com os planos do *software*, que devem ser sanados antes do acordo para validação.

13.3 DETERMINAÇÃO DE CONFORMIDADE

13.3.1 Antes da certificação, a autoridade determina se o sistema ou equipamento CNS/ATM está em conformidade com a base de certificação. Para o *software*, essa conformidade é cumprida através do documento "Sumário de Realizações" e evidências de conformidade.

13.3.2 A autoridade certificadora pode rever a sua discrição os processos e resultados do ciclo de vida do *software* durante o ciclo de vida, como discutido no item 10.3.

14 DISPOSIÇÕES FINAIS

14.1 As instruções estabelecidas neste documento são de caráter geral e devem ser periodicamente revisadas.

14.2 Casos omissos deverão ser levados à apreciação do Exmo. Sr. Chefe do SDTE.

REFERÊNCIAS

ALMEIDA, A. T. de et al. Gestão da manutenção na direção da competitividade. Recife: Editora Universitária, 2001.

BRASIL. Comando da Aeronáutica. Departamento de Controle do Espaço Aéreo. *Gerenciamento do Ciclo de Vida de Sistemas e Materiais do SISCEAB,* **ICA 400-31**. [Rio de Janeiro], 2010.

BRASIL. Comando da Aeronáutica. Departamento de Controle do Espaço Aéreo. *Gerenciamento do Risco à Segurança Operacional (GRSO) no SISCEAB,* **ICA 63-26**. [Rio de Janeiro], 2010.

BRASIL. Comando da Aeronáutica. Departamento de Controle do Espaço Aéreo. *Procedimentos Administrativos para Homologação, Ativação e Desativação de Auxílios, Equipamentos, Sistemas e Órgãos Operacionais no Âmbito do SISCEAB,* **MCA 63-4**. [Rio de Janeiro], 2010.

PAN, J. *Software reliability: dependable embedded systems*. Carnegie Mellon University, 1999. Disponível em: <http://www.ece.cmu.edu/~koopman/des_s99/sw_reliability/>. Acesso em: 26 jun. 2008.

RADIO TECHNICAL COMMISION FOR AERONAUTICS. *Software Considerations in Airborne Systems and Equipment Certification,* **DO-178B.** [EUA], 1992.

RADIO TECHNICAL COMMISION FOR AERONAUTICS. *Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance,* **DO-278.** [EUA], 2002.

Anexo A – Objetivos dos Processos e Saídas por Nível de *Software*

Tabela A-1 – Processo de Planejamento de *Software*

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	As atividades dos processos de desenvolvimento e de integração estão definidas	5.1.a 5.3	s	s	s	s	s	Plano para Validação de *Software* Plano de Desenvolvimento do *Software* Plano de Verificação do *Software* Plano SCM	11.2 11.3 11.4 11.5	1 1 1 1	1 1 2 1	1 2 2 2	1 2 2 2	1 2 2 2
2	Os critérios de transição, as relações e a sequência entre processos estão definidos	5.1.b 5.3	s	s	s	s	n							
3	O ambiente do ciclo de vida está definido	5.1.c	s	s	s	s	n							
4	As condições adicionais foram desenvolvidas	5.1.d	s	s	s	s	s	Plano SQA	11.6	1	1	2	2	2
5	Os padrões de desenvolvimento foram definidos	5.1.e	s	s	s	s	n	Padrões de Requisitos de *Software* Padrões de Projeto de *Software* Padrões de Codificação	11.7 11.8 11.9	1 1 1	1 1 1	2 2 2	2 2 2	n n n
6	Os planos do *software* estão em conformidade a este documento	5.1.f 5.6	s	s	s	s	n	Registros SQA Resultados da Verificação de *Software*	11.20 11.15	2 2	2 2	2 2	2 2	n n
7	Os planos do *software* são coordenados	5.1.g 5.6	s	s	s	s	n	Registros SQA Resultados da Verificação de *Software*	11.20 11.15	2 2	2 2	2 2	2 2	n n

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-2 – Processo de Desenvolvimento de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os requisitos de alto nível são desenvolvidos	6.2.1.a	s	s	s	s	s	Requisitos de *Software*	11.10	1	1	1	1	1
2	Os requisitos derivados de alto nível são definidos	6.2.1.b	s	s	s	s	s	Requisitos de *Software*	11.10	1	1	1	1	1
3	A arquitetura é desenvolvida	6.3.1.a	s	s	s	s	s	Descrição de Projeto	11.11	1	1	2	2	n
4	Os requisitos de baixo nível são desenvolvidos	6.3.1.a	s	s	s	n	n	Descrição de Projeto	11.11	1	1	2	n	n
5	Os requisitos derivados de baixo nível são desenvolvidos	6.3.1.b	s	s	s	n	n	Descrição de Projeto	11.11	1	1	2	n	n
6	O Código-Fonte é desenvolvido	6.4.1.a	s	s	s	n	n	Código-Fonte	11.12	1	1	1	n	n
7	O código executável é produzido e integrado ao computador	6.5.1.a	s	s	s	s	s	Código-Executável	11.13	1	1	1	1	1
8	O Processo de Adaptação e os processos relacionados são definidos (quando aplicável)	12.3	s	s	s	s	s	Documentação da Adaptação	11.22	1	1	1	1	2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-3 – Verificação das Saídas do Processo de Requisitos de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os requisitos de alto nível são compatíveis com os requisitos de sistemas	7.4.1.a	i	i	s	s	s	Resultados da Verificação de *Software*	11.15	2	2	2	2	2
2	Os requisitos de alto nível são acurados e consistentes	7.4.1.b	i	i	s	s	s	Resultados da Verificação de *Software*	11.15	2	2	2	2	2
3	Os requisitos de alto nível são compatíveis com o computador	7.4.1.c	s	s	n	n	n	Resultados da Verificação de *Software*	11.15	2	2	N	n	n
4	Os requisitos de alto nível são verificáveis	7.4.1.d	s	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
5	Os requisitos de alto nível estão em conformidade aos padrões	7.4.1.e	s	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
6	Os requisitos de alto nível são rastreáveis aos requisitos de sistemas	7.4.1.f	s	s	s	s	s	Resultados da Verificação de *Software*	11.15	2	2	2	2	2
7	Os algoritmos são acurados	7.4.1.g	i	i	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-4 – Verificação das Saídas do Processo de Projeto de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os requisitos de baixo nível são compatíveis com os requisitos de alto nível	7.4.2.a	i	i	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
2	Os requisitos de baixo nível são acurados e consistentes	7.4.2.b	i	i	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
3	Os requisitos de baixo nível são compatíveis com o computador	7.4.2.c	s	s	n	n	n	Resultados da Verificação de *Software*	11.15	2	2	n	n	n
4	Os requisitos de baixo nível são verificáveis	7.4.2.d	s	s	n	n	n	Resultados da Verificação de *Software*	11.15	2	2	n	n	n
5	Os requisitos de baixo nível estão em conformidade com os padrões	7.4.2.e	s	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
6	Os requisitos de baixo nível são rastreáveis aos requisitos de alto nível	7.4.2.f	s	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
7	Os algoritmos são acurados	7.4.2.g	i	i	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
8	A arquitetura é compatível com os requisitos de alto nível	7.4.3.a	i	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
9	A arquitetura de *software* é consistente	7.4.3.b	i	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
10	A arquitetura de *software* é compatível com o computador	7.4.3.c	i	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
11	A arquitetura de *software* é verificável	7.4.3.d	s	s	n	n	n	Resultados da Verificação de *Software*	11.15	2	2	n	n	n
12	A arquitetura de *software* está em conformidade aos padrões	7.4.3.e	s	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
13	A integridade da partição é confirmada	7.4.3.f	i	s	s	s	s	Resultados da Verificação de *Software*	11.15	2	2	2	2	2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-5 – Verificação das Saídas dos Processos de Codificação e de Integração de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	O Código-Fonte é compatível com os requisitos de baixo nível	7.4.4.a	i	i	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
2	O Código-Fonte é compatível com a arquitetura de *software*	7.4.4.b	i	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
3	O Código-Fonte é verificável	7.4.4.c	s	s	n	n	n	Resultados da Verificação de *Software*	11.15	2	2	n	n	n
4	O Código-Fonte está em conformidade aos padrões	7.4.4.d	s	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
5	O Código-Fonte tem rastreabilidade aos requisitos de baixo nível	7.4.4.e	s	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
6	O Código-Fonte é acurado e consistente	7.4.4.f	i	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
7	Os resultados do processo de integração são completos e corretos	7.4.5	s	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-6 – Testes das Saídas do Processo de Integração

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	O código executável é compatível com os requisitos de alto nível	7.5.5.1 7.5.6	i	i	s	s	s	Procedimentos e Casos de Verificação de *Software* Resultados da Verificação de *Software*	11.14 11.15	1 2	1 2	2 2	2 2	2 2
2	O código executável é robusto com respeito ao requisitos de alto nível	7.5.5.2 7.5.6	i	s	s	s	s	Procedimentos e Casos de Verificação de *Software* Resultados da Verificação de *Software*	11.14 11.15	1 2	1 2	2 2	2 2	2 2
3	O código executável é compatível com os requisitos de baixo nível	7.5.5.1 7.5.6	i	i	s	n	n	Procedimentos e Casos de Verificação de *Software* Resultados da Verificação de *Software*	11.14 11.15	1 2	1 2	2 2	n n	n n
4	O código executável é robusto com respeito aos requisitos de baixo nível	7.5.5.2 7.5.6	i	s	s	n	n	Procedimentos e Casos de Verificação de *Software* Resultados da Verificação de *Software*	11.14 11.15	1 2	1 2	2 2	n n	n n
5	O código executável é compatível com o computador	7.5.6.a	s	s	s	s	s	Procedimentos e Casos de Verificação de *Software* Resultados da Verificação de *Software*	11.14 11.15	1 2	1 2	2 2	2 2	2 2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-7 – Verificação dos Resultados do Processo de Verificação

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os procedimentos de testes são corretos	7.4.6.b	i	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
2	Os resultados dos testes são corretos e as discrepâncias com o planejamento são explicadas	7.4.6.c	i	s	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n
3	A cobertura dos testes dos requisitos de alto nível é atingida	7.5.7.1	i	s	s	s	s	Resultados da Verificação de *Software*	11.15	2	2	2	2	2
4	A cobertura dos testes dos requisitos de baixo nível é atingida	7.5.7.1	i	s	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
5	A cobertura das estruturas de *software* (cobertura de decisão/condição modificada) é atingida	7.5.7.2.a 7.5.7.2.b	i	n	n	n	n	Resultados da Verificação de *Software*	11.15	2	n	n	n	n
6	A cobertura das estruturas de *software* (cobertura de decisões) é atingida	7.5.7.2.a 7.5.7.2.b	i	i	n	n	n	Resultados da Verificação de *Software*	11.15	2	2	n	n	n
7	A cobertura da estrutura de código (cobertura de linhas executáveis) é atingida	7.5.7.2.a 7.5.7.2.b	i	i	s	n	n	Resultados da Verificação de *Software*	11.15	2	2	2	n	n
8	A cobertura da estrutura de código (acoplamento de dados e de controle) é atingida	7.5.7.2.c	i	i	s	s	n	Resultados da Verificação de *Software*	11.15	2	2	2	2	n

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-8 – Processo de Gerenciamento de Configuração de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os itens de configuração são identificados	8.2.1	s	s	s	s	s	Registros SCM	11.19	2	2	2	2	2
2	As linhas de bases e a rastreabilidade são estabelecidas	8.2.2	s	s	s	s	s	Índice de Configuração de *Software* Registros SCM	11.17 11.19	1 2	1 2	1 2	1 2	1 2
3	O reporte de problemas, o controle de mudanças, a revisão de mudanças e o reporte de *status* de configuração são estabelecidos	8.2.3 8.2.4 8.2.5 8.2.6	s	s	s	s	s	Relatório de Problemas Registros SCM	11.18 11.19	2 2	2 2	2 2	2 2	2 2
4	O arquivamento, a recuperação e a liberação são estabelecidos	8.2.7	s	s	s	s	s	Registros SCM	11.19	2	2	2	2	2
5	O controle de carregamento é estabelecido	8.2.8	s	s	s	s	s	Registros SCM	11.19	2	2	2	2	2
6	O controle do ambiente do ciclo de vida de *software* é estabelecido	8.2.9	s	s	s	s	s	Índice de Configuração de Ambiente do Ciclo de Vida de *Software* Registros SCM	11.16 11.19	1 2	1 2	1 2	2 2	2 2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-9 – Processo de Garantia da Qualidade de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	A qualidade é obtida para os processos de desenvolvimento e de integração em conformidade com os planos e padrões adotados	9.1.a	i	i	i	i	i	Registros SQA	11.20	2	2	2	2	2
2	A qualidade é obtida com o critério de transição para o ciclo de vida sendo satisfeito	9.1.b	i	i	i	i	n	Registros SQA	11.20	2	2	2	2	n
3	A revisão de conformidade é conduzida	9.1.c	i	i	i	i	i	Registros SQA	11.20	2	2	2	2	2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo A – Objetivos dos Processos e Saídas por Nível de Software

Tabela A-10 – Processo de Validação de Software

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Existe comunicação e entendimento entre fornecedor e autoridade certificadora	10	s	s	s	s	s	Plano para Validação de *Software*	11.2	1	1	1	1	1
2	Os meios de conformidade são propostos e acordados com o Plano para Validação de *Software*	10.2	s	s	s	s	s	Plano para Validação de *Software*	11.2	1	1	1	1	1
3	Aceitação da conformidade é promovida	10.3	s	s	s	s	s	Sumário de Realizações Índice de Configuração	11.21 11.17	1 1	1 1	1 1	1 1	1 1

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Anexo B – Objetivos Específicos aos COTS

Tabela B-1 – Processo de Planejamento COTS

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os planos de aquisição e de integração são definidos	12.2.4.3	s	s	s	s	s	Plano de Desenvolvimento do *Software*	11.19	1	1	2	2	2
2	Os critérios de transição estão definidos	12.2.4.3	s	s	s	s	n	Plano de Desenvolvimento do *Software*	11.18	1	1	2	2	n
3	As evidências do planejamento COTS são consistentes com os planos do *software* CNS/ATM	12.2.4.3	s	s	s	s	n	Registros SQA	11.19	2	2	2	2	n

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo B –Objetivos Específicos aos COTS

Tabela B-2 – Processo de Aquisição COTS

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os requisitos CNS/ATM satisfeitos pelo *software* COTS são determinados	12.2.5.1	i	i	s	s	s	Plano de Desenvolvimento do *Software* Descrição do Projeto	11.3 11.11	2 2	2 2	2 2	2 2	2 2
2	A adequação das evidências do ciclo de vida são determinadas	12.2.5.1	s	s	s	s	s	Plano de Desenvolvimento do *Software*	11.2	1	1	2	2	2
3	Os requisitos derivados são definidos	12.2.5.1	s	s	s	s	s	Requisitos de *Software*	11.10	1	1	1	1	1
4	A compatibilidade do COTS com o *hardware* e com o outro *software* CNS/ATM é assegurada	12.2.5.1	s	s	s	s	s	Resultados da Verificação de *Software*	11.15.	2	2	2	2	2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)

Continuação do Anexo B –Objetivos Específicos aos COTS

Tabela B-3 – Processo de Gerenciamento de Configuração COTS

	Objetivos		Aplicabilidade					Resultados / Evidências		Categoria de Controle				
	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5	Descrição	Ref.	AL1	AL2	AL3	AL4	AL5
1	Os itens de dados e configuração do COTS são identificados	12.2.7.1	s	s	s	s	s	Registros SCM	11.19	2	2	2	2	2
2	O mecanismo de relatório de problemas é estabelecido	12.2.7.1	s	s	s	s	s	Relatórios de Problemas	11.18	2	2	2	2	2
3	A incorporação de versões para o COTS é controlada	12.2.7.1	s	s	s	s	s	Registros SCM	11.19	2	2	2	2	2
4	Os itens de dados e configuração do COTS são arquivados	12.2.7.1	s	s	s	s	s	Registros SCM	11.19	2	2	2	2	2

s - objetivo deve ser satisfeito

i - objetivo deve ser satisfeito com independência

n - não aplicável

1 - Resultados /evidências devem satisfazer aos objetivos da categoria de controle 1 (CC1)

2 - Resultados ou evidências devem satisfazer aos objetivos da categoria de controle 2 (CC2)