sid.inpe.br/mtc-m19/2012/03.26.23.04-TDI

UMA PROPOSTA DE APERFEIÇOAMENTO DE UM PROCESSO DE GERENCIAMENTO DE REQUISITOS DE SISTEMA E DE SOFTWARE E SUA APLICAÇÃO A SISTEMAS ESPACIAIS E AERONÁUTICOS EMBARCADOS

João Paulo Marques Reginato

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 13 de abril de 2012.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3BJTJGH>

INPE
São José dos Campos
2012

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

sid.inpe.br/mtc-m19/2012/03.26.23.04-TDI

UMA PROPOSTA DE APERFEIÇOAMENTO DE UM PROCESSO DE GERENCIAMENTO DE REQUISITOS DE SISTEMA E DE SOFTWARE E SUA APLICAÇÃO A SISTEMAS ESPACIAIS E AERONÁUTICOS EMBARCADOS

João Paulo Marques Reginato

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Marcelo Lopes de Oliveira e Souza, aprovada em 13 de abril de 2012.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3BJTJGH>

INPE
São José dos Campos
2012

Dados Internacionais de Catalogação na Publicação (CIP)

R263p

Reginato, João Paulo Marques.
Uma proposta de aperfeiçoamento de um processo de gerenciamento de requisitos de sistema e de software e sua aplicação a sistemas espaciais e aeronáuticos embarcados / João Paulo Marques Reginato. – São José dos Campos : INPE, 2012.
xxii + 149 p. ; (sid.inpe.br/mtc-m19/2012/03.26.23.04-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012.
Orientador : Dr. Marcelo Lopes de Oliveira e Souza.

1. gerenciamento de requisitos. 2. processo de desenvolvimento. 3. confiabilidade. 4. sistemas espaciais embarcados. 5. software embarcado. I.Título.

CDU 629.78

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Mestre em

Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais

Dr. Petrônio Noronha de Souza

Presidente / INPE / SJCampos - SP

Dr. Marcelo Lopes de Oliveira e Souza

Orientador(a) / INPE / SJCampos - SP

Dr. Mauricio Gonçalves Vieira Ferreira

Membro da Banca / INPE / SJCampos - SP

Dr. Henrique Mohallem Paiva

Convidado(a) / MECTRON / SJCampos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

Aluno (a): João Paulo Marques Reginato

São José dos Campos, 13 de abril de 2012

“A maior riqueza do homem
é a sua incompletude.
Nesse ponto sou abastado.
Palavras que me aceitam como sou - eu não aceito.”

Manoel de Barros (Retrato do artista quando coisa, 1998).

A minha mãe pela compreensão; a minha avó por todo o carinho; a meu irmão pelo companheirismo; e a minha futura esposa pela paciência.

AGRADECIMENTOS

Agradeço ao INPE e aos professores e colegas do Curso de Engenharia e Tecnologia Espaciais na Área de Engenharia e Gerenciamento de Sistemas Espaciais pelo tempo investido e suporte em todas as horas.

Agradeço em especial ao Professor Dr. Marcelo Lopes de Oliveira e Souza. Sua paciência e capacidade de entender as dificuldades dos alunos, aliadas a sua dedicação ao ensino de alto nível fazem acreditar que tudo é possível.

Agradeço à EMBRAER por disponibilizar horas de trabalho para que eu pudesse assistir às aulas e iniciar este trabalho.

RESUMO

Este trabalho apresenta uma proposta de aperfeiçoamento de um processo de gerenciamento de requisitos de sistema e de *software* e sua aplicação a sistemas espaciais e aeronáuticos embarcados. Para isto, o trabalho: 1) revisa e analisa as normas mais utilizadas pelas indústrias espacial (normas da ESA) e aeronáutica (normas da SAE, RTCA) brasileiras; 2) através da comparação entre as normas, propõe um processo aperfeiçoado, seus passos, *checklists* e *roadmap* para implementações futuras; 3) ilustra-o pela sua aplicação a um exemplo espacial (SCAO de um satélite) e a um exemplo aeronáutico (indicação no cockpit); 4) apresenta uma proposta adequada a sua utilização espacial, especialmente no Instituto Nacional de Pesquisas Espaciais - INPE, de forma a melhorar a confiabilidade dos sistemas espaciais e aeronáuticos embarcados. Ao fim deste trabalho, foi possível compreender as normas de desenvolvimento de requisitos de sistemas e de *software* e os benefícios de sua aplicação, bem como apresentar atividades detalhadas e exemplos de utilização.

A PROPOSAL FOR IMPROVEMENT OF A SYSTEM AND SOFTWARE REQUIREMENTS MANAGEMENT PROCESS AND ITS APPLICATION TO EMBEDDED SPACE AND AERONAUTICAL SYSTEMS

ABSTRACT

This work presents a proposal for Improvement of a system and software requirements management process and its application to embedded space and aeronautical systems. To accomplish this, the work: 1) revises and analyzes the standards most used by the brazilian space industry (ESA standards) and aeronautical industry (SAE, RTCA standards); 2) through comparison of the standards, proposes an improved process, its steps, checklists and roadmap for future implementations; 3) illustrates it through its application to one example from the space industry (satellite ACDH) and one example from the aeronautical industry (indication in the cockpit); 4) presents a proposal suitable for use in the space industry, specially at the National Institute for Space Research- INPE, to improve the reliability of embedded space and aeronautical systems. At the end of this work it was possible to understand the systems and software development requirements standards and the benefits of its application, and to present detailed activities and examples of its use.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

AC	Advisory Circular
ANAC	Agência Nacional de Aviação Civil
ARP	Aeronautical Recommended Practices
ASIC	Application-Specific Integrated Circuit
CBERS	China-Brazil Earth Resources Satellite
CCA	Common Cause Analysis
COTS	Commercial-Off-The-Shelf
DAL	Design Assurance Level
DD	Dependence Diagram
DO	Directive Order
EASA	European Aviation Safety Agency
ECSS	European Cooperation for Space Standardization
ESA	European Space Agency
FHA	Functional Hazard Assessment
FMEA	Failure Mode and Effect Analysis
FMES	Failure Mode and Effect Summary
FPGA	Field-Programmable Gate Array
FTA	Fault Tree Analysis
ICD	Interface Control Document
INPE	Instituto Nacional de Pesquisas Espaciais
IRD	Interface Requirements Document
MA	Markov Analysis
MCDC	Modified Condition and Decision Coverage
MTBF	Mean Time Between Failures
NASA	National Aeronautics and Space Administration
PLD	Programmable Logic Device
PMM	Plataforma MultiMissão
PSAC	Plan for Software Aspects of Certification
PSSA	Preliminary System Safety Assessment
RAMS	Reliability, Availability, Maintainability and Safety
RB	Requirements Baseline
RBD	Reliability Block Diagram
SAE	Society of Automotive Engineers
SCMP	Software Configuration Management Plan
SDP	Software Development Plan
SQAP	Software Quality Assurance Plan
SRS	Software Requirements Specification
SSA	System Safety Assessment
SSS	Software System Specification
SVP	Software Verification Plan
SW	Software
TS	Technical Specification
TT&C	Telemetria e Telecomando

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação e Justificativa

Sistemas espaciais, tais como satélites e foguetes, e sistemas aeronáuticos, tais como aviões e helicópteros, são exemplos de aplicações complexas e altamente integradas, onde o mau funcionamento do sistema pode provocar grandes perdas econômicas, causar danos ao meio ambiente, ou mesmo acarretar perdas de vidas humanas. Consequentemente, existe uma grande preocupação em relação aos requisitos dos sistemas envolvidos nestas aplicações, tanto no contexto de *hardware* quanto no de *software*.

A experiência tem mostrado que a preocupação com os requisitos do sistema no início do projeto melhora aspectos como prazo, custo, qualidade e confiabilidade. Um trabalho de gerenciamento de requisitos iniciado em fases preliminares do projeto e que se estenda pelas demais fases pode ser a diferença entre o sucesso e o fracasso.

Na indústria espacial, as normas para desenvolvimento de sistemas, *software* e circuitos integrados passam por constantes atualizações com o intuito de torná-las cada vez mais confiáveis, robustas e eficazes.

Na indústria aeronáutica existe a preocupação específica com os aspectos de segurança, que fomentaram a elaboração de diversas normas e padrões que auxiliam na busca pelo cumprimento de requisitos mínimos de segurança, tanto no escopo do sistema quanto em suas funções.

Essas diretrizes podem ser aplicadas a sistemas espaciais, de forma a melhorar a confiabilidade dos mesmos e aumentar a probabilidade de cumprimento de sua missão. Esse trabalho pretende mostrar a melhor forma de utilizar os conhecimentos e boas práticas da aeronáutica em aplicações espaciais.

Em relação ao INPE, Almeida (2011) mostra que:

> Introduzido em engenharia de sistemas no final da década de 1960, o INPE somente se envolveu decididamente no desenvolvimento de um satélite em meados da década de 1980 quando uma fase de intensa atividade, notada pela instalação de um centro de controle, estações de rastreio e laboratórios especializados, foi concluída, em 1993, com o lançamento de seu primeiro satélite. Apesar do ambiente de rápida evolução, a partir do início dos anos 1990, provocada pela introdução de microcomputadores em todas as atividades de engenharia, ocorreu na engenharia de sistemas do INPE uma fase de estabilidade, em que novos processos e ferramentas de automatização não foram introduzidas (ALMEIDA, 2011, p. 1).

1.2 Objetivos do Trabalho

O objetivo deste trabalho é apresentar uma proposta de aperfeiçoamento de um processo de gerenciamento de requisitos de sistema e de software e sua aplicação a sistemas espaciais e aeronáuticos embarcados. Para tanto, dividiu-se o trabalho em etapas, descritas a seguir:

a) Revisar a literatura com o objetivo de identificar e analisar as normas utilizadas pelas indústrias espacial e aeronáutica para desenvolvimento de sistemas e de *software*.

b) Comparar as normas aplicáveis para aperfeiçoar os processos de gerenciamento de requisitos de modo a serem adequados ao uso pelo INPE.

c) Especificar processos que devem ser aplicados em algumas das fases do desenvolvimento de sistemas e de *software*, incluindo detalhamento das atividades.

d) Apresentar exemplos de aplicação dos processos especificados.

1.3 Organização do Trabalho

Para atingir os objetivos propostos, é necessário conhecer e uniformizar a terminologia, os conceitos e as normas aplicáveis a este trabalho. O Capítulo 2

(Conceitos Básicos e Revisão Bibliográfica) visa apresentar a terminologia, os conceitos e as normas no contexto desta proposta.

O Capítulo 3 (Comparação entre as Normas de Desenvolvimento de Sistemas e *Software* da Indústria Espacial e da Indústria Aeronáutica) apresenta a comparação entre as normas da indústria espacial europeia e da indústria aeronáutica, no contexto de desenvolvimento de sistemas e de *software*, de forma a buscar processos aperfeiçoados.

O Capítulo 4 (Uma Proposta de Aperfeiçoamento de um Processo de Gerenciamento de Requisitos de Sistema e de *Software*) apresenta os processos e atividades propostos por este trabalho que são adequados à aplicação pelo INPE, tanto para desenvolvimento de sistemas quanto para desenvolvimento de *software*.

O Capítulo 5 (Exemplos de Aplicação a Sistemas Espaciais Embarcados e a Sistemas Aeronáuticos Embarcados) baseado: 1) no Subsistema de Controle de Atitude e Tratamento de Dados da Plataforma MultiMissão); e 2) numa indicação luminosa no *cockpit* de uma aeronave fictícia, mostra exemplos práticos de aplicação dos processos, através de considerações sobre a Plataforma MultiMissão e sobre tal *cockpit* fictício.

O Capítulo 6 (Conclusões, Recomendações e Sugestões para Trabalhos Futuros) se dedica às conclusões e considerações finais.

2 CONCEITOS BÁSICOS E REVISÃO BIBLIOGRÁFICA

Este Capítulo destina-se a uniformizar a terminologia, os conceitos e apresentar algumas das normas e padrões aplicáveis ao escopo deste trabalho, uma vez que o desenvolvimento de sistemas embarcados possui inúmeras referências que poderiam ser estudadas. Com o embasamento teórico apresentado neste Capítulo, é possível entender o detalhamento e a comparação entre as normas e a adequação do processo proposto ao INPE.

Especificamente, este Capítulo discute os conceitos de Requisitos (incluindo Validação e Verificação de Requisitos) e de Confiabilidade, além de descrever brevemente algumas normas das Indústrias Espacial e Aeronáutica.

2.1 Requisitos

Segundo Young (2004), “um requisito é um atributo necessário de um sistema, uma afirmação que identifica uma capacidade, característica ou fator de qualidade de um sistema para que ele tenha valor e utilidade para um consumidor ou usuário”.

Sua captura é chamada de elicitação (*elicitation*). Young (2004) indica que elicitação refere-se ao processo de “entender as necessidades dos consumidores ou usuários para o sistema planejado e suas expectativas”. Além de usuários, pode-se estender o termo para o conceito de interessados (*stakeholders*). Sua manutenção durante todo o ciclo devida do sistema é chamada de gerenciamento (*management*).

A experiência mostra que investir em elicitação de requisitos na fase inicial de um projeto e gerenciamento de requisitos durante o projeto ajuda a economizar recursos, minimizar esforços e melhorar prazos durante o projeto. Todavia, a pressão de chefes, clientes e fornecedores, entre outros, faz com que pouco tempo seja gasto com elicitação de requisitos na fase inicial de um projeto.

O Gerenciamento de Requisitos pode ser definido como:

> A ação de registrar, armazenar, alterar, manter histórico e recuperar informações atualizadas sobre requisitos, atributos de requisitos, as relações entre requisitos e entre outras informações sobre os requisitos de um sistema durante o seu ciclo de vida (ALMEIDA, 2011, p. 7).

Existe a necessidade de se ter um "produto" pronto, seja ele um *software*, um *hardware* ou um sistema. A grande questão é que um produto gerado sem o devido cuidado com a elicitação e o gerenciamento de requisitos tem alta probabilidade de não atender às expectativas de todos os interessados.

Para os defensores da Engenharia de Requisitos, o sucesso de um sistema está diretamente ligado à qualidade de seus requisitos, uma vez que sem bons requisitos um projeto pode atrasar, exceder custos ou produzir um sistema que nunca será utilizado em sua plenitude.

A Engenharia de Requisitos pode ser definida como:

> A subdisciplina da engenharia de sistemas e da engenharia de software que se preocupa em determinar, analisar, clarificar, documentar, validar e gerenciar as finalidades, os atributos, as capacidades, as características, as qualidades, as restrições e as funções de um sistema ou software durante seu ciclo de vida (ALMEIDA, 2011, p. 7).

Os requisitos devem ser a indicação clara, usualmente em linguagem natural, das necessidades dos usuários. Por outro lado, eles devem ser escritos de forma que o desenvolvedor possa entendê-los, traduzi-los em linguagem técnica e especificar o que o sistema deve fazer para produzir o resultado esperado no todo ou em parte.

Alexander e Stevens (2002) indicam que:

> Requisitos são a chave para o sucesso do projeto. Nós todos sabemos disso, mas frequentemente esquecemos – e pagamos o preço. Muitos projetos, tanto na indústria quanto no setor público, falham em fazer o que é desejado. Eles entregam com atraso, acima do orçamento e com baixa qualidade. Esquecer-se dos requisitos é desastroso (ALEXANDER E STEVENS, 2002, p. v).

Os autores mostram um estudo das principais causas que levam ao fracasso de projetos, conforme a Tabela 2.1 abaixo:

Tabela 2.1 – Razões para falhas em projetos.

Razão	Frequência
Requisitos incompletos	13,1%
Não envolveu usuários	12,4%
Recursos insuficientes	10,6%
Expectativas irreais	9,9%
Falta de suporte gerencial	9,3%
Mudança de requisitos	8,7%
Planejamento fraco	8,1%
Não é mais necessário	7,4%

Fonte: Standish Group (1995), citado por Alexander e Stevens (2002).

Almeida (2011) indica que

> Qualquer pessoa com experiência sabe avaliar a importância de se ter bons requisitos no início de um projeto, pois eles são a base para todo o trabalho que se realizará a seguir e, mal-entendidos em requisitos, geralmente, resultam em trabalho e esforço perdidos. Apesar disso, a abundância de citações e de casos judiciais de projetos que tiveram prazos e custos excedidos ou foram cancelados por motivos relacionados com requisitos são exemplos tanto de sua importância quanto da insuficiente atenção que lhes é dada (ALMEIDA, 2011, p. 13).

2.1.1 Validação de Requisitos

De acordo com o contexto, a validação pode ser vista de três formas diferentes: validação dos requisitos dos interessados (*stakeholders*), validação dos requisitos do sistema e validação da implementação.

Segundo Easterbruck e Nuseibeh (2000), a validação dos requisitos dos interessados é “o processo de estabelecer que os requisitos e modelos elicitados forneçam um relato preciso dos requisitos dos *stakeholders*”.

A indústria aeronáutica aplica o processo de validação de requisitos de sistema com o objetivo de garantir que os requisitos de sistema elicitados estejam corretos e completos, de forma que o produto final possa estar de acordo com o desejado.

A SAE ARP-4754 (ver Seção 2.4.2) define os termos “correto” (*correctness*) e “completo” (*completeness*) da seguinte maneira:

- Um requisito é correto quando não possui ambiguidades ou erros nos seus atributos.
- Um requisito é completo quando nenhum de seus atributos foi omitido e os apresentados são essenciais.

Alguns autores consideram a validação da implementação dos requisitos como a confirmação de que os requisitos foram implementados no sistema final. Essa interpretação acontece por uma mudança de perspectiva da avaliação dos requisitos de sua elicitação para sua implementação.

Young (2004) indica que a validação de requisitos é “o processo para confirmar que os requisitos estão implementados no sistema entregue. A ordem de validação dos requisitos deve ser priorizada, uma vez que existe um limite de recursos disponíveis”.

Para evitar confusão na utilização dos termos, este trabalho deixa claro qual a interpretação da validação nos diferentes contextos em que é utilizada.

2.1.2 Verificação de Requisitos

Engel (2010) mostra que, “alguns dizem que validação e verificação são uma e a mesma coisa, outros dizem que verificação lida com especificações, outros dizem que é a validação que lida com especificações e ainda há outros que dizem que ambas o fazem”.

Segundo o autor existem algumas tendências que não são universalmente aceitas:

- Verificação lida com satisfazer as especificações escritas dos sistemas;
- Verificação envolve a correção das estruturas internas dos sistemas;
- Verificação relaciona-se com a evolução dos processos de ciclo de vida dos sistemas.

O principal objetivo do processo de verificação de requisitos é assegurar que cada nível de implementação satisfaz aos requisitos especificados e validados. Segundo Young (2004), “requisitos verificados são requisitos satisfeitos na solução projetada”.

Normalmente, entende-se a verificação de requisitos como testes a serem feitos com protótipos ou no produto final. Todavia, o teste é apenas um dos meios de verificação de requisitos. Outros métodos encontrados na indústria são, por exemplo, análises, revisões, inspeções e simulações.

Enquanto o processo de validação de requisitos busca mostrar que o requisito está correto, o processo de verificação de requisitos usualmente busca mostrar que a implementação do requisito está correta.

Em relação à diferença entre validação e verificação de requisitos, Young (2004) indica que “engenheiros de sistema e desenvolvedores de *software* tendem a usar as duas palavras de forma inversa, e isso causa uma tremenda confusão”.

A definição sugerida pelo autor para verificação de requisitos é “um processo para assegurar que a solução de projeto satisfaz os requisitos”.

2.2 Confiabilidade

Uma miríade de definições para o termo Confiabilidade (*Reliability*) pode ser encontrada na literatura. Segundo Basovsky (2004), “dicionários e enciclopédias possuem várias interpretações para este palavra, ranqueando-a na categoria de conceitos abstratos, como bondade, beleza e honestidade”. Todavia, segundo o autor, “em Engenharia e em Matemática Estatística, confiabilidade tem um significado exato. Não apenas pode ser definida, mas também pode ser calculada, avaliada objetivamente, medida, testada e até projetada em um pedaço de equipamento de engenharia”.

Assim, uma definição simples de confiabilidade, apresentada por Basovsky (2004) é “a capacidade de um equipamento não falhar em operação”. De forma mais completa, Souza e Carvalho (2005), seguindo IEEE (1973), definem Confiabilidade (*Reliability)* como “a probabilidade de que um dispositivo ou sistema irá exercer uma função requerida, sob condições especificadas, por um período de tempo determinado”.

As definições apresentadas acima mostram um caráter quantitativo aplicável à confiabilidade. Todavia, existem aspectos qualitativos que devem ser observados para garantir que o “equipamento não falhe em operação”. Dessa forma, muitos autores utilizam o termo Dependabilidade (*Dependability)* para descrever as duas faces da análise, aplicando o termo confiabilidade apenas para os aspectos quantitativos.

A norma ECSS-P-001A (2000) define Dependabilidade como “um termo cumulativo usado para descrever a performance de disponibilidade e seus fatores de influência: performance de confiabilidade, performance de manutenabilidade e performance do suporte à manutenção”.

Como visto acima, outros conceitos associados à confiabilidade são a Disponibilidade (*Availability*) e a Manutenabilidade (*Maintainability*). De acordo com a mesma norma, a Disponibilidade é definida como “a capacidade de um item estar no estado de executar uma função requerida em condições determinadas, em um dado instante de tempo ou em um intervalo de tempo, supondo que todos os recursos externos necessários são fornecidos”. Além disso, a definição de Manutenabilidade é “a capacidade de um item, sob dadas condições de uso, ser conservado ou restaurado a um estado em que ele possa executar sua função requerida, quando a manutenção é executada sob dadas condições e usando procedimentos e recursos prescritos”.

Outro conceito importante relacionado ao tema é a Segurança (*Safety*). Segundo a SAE ARP-4754 (1996), Segurança é “o estado em que o risco é menor do o risco aceitável. O risco aceitável é definido por um processo ou é por declaração”. Já a ECSS-P-001A (2000) define segurança como “um estado

no qual o risco de ferir (pessoas) ou causar danos é limitado a um nível aceitável". Nota-se que ambas as definições são extremamente subjetivas e carecem de esclarecimentos adicionais.

Tais esclarecimentos podem ser encontrados no trabalho de Manelli Neto (2011), indicando que:

> No setor civil aeronáutico, Segurança é um termo muito bem definido pelos requisitos de certificação, em especial pelo requisito FAR 25.1309 e *Advisory Circular* 25.1309, que estabelecem o que as autoridades certificadoras chamam de *Safe Design Concept* ou Conceito de Projeto Seguro. O Conceito de Projeto Seguro (mesmo conceito adotado pelas agências de certificação européia – EASA – e a brasileira – ANAC) estabelece uma relação inversa entre severidade das falhas e sua probabilidade (Confiabilidade), i.e., quanto mais severa a falha, menos provável ela deve ser. (MANELLI NETO, 2011, p. 15).

O mesmo autor indica que, segundo a AC 25.1309, as falhas são divididas em 5 diferentes níveis de severidade:

a) Catastrófico (traduzindo do Inglês, Catastrophic).

b) Perigoso (traduzindo do Inglês, Hazardous);

c) Maior (traduzindo do Inglês, Major);

d) Menor (traduzindo do Inglês, Minor);

e) Sem impacto em Segurança (traduzindo do Inglês, No Safety Effect);

Manelli Neto (2011) comenta ainda que:

> A avaliação da severidade é um processo longo que envolve a identificação das falhas, a caracterização de seus efeitos nos passageiros, no aumento da carga de trabalho da tripulação e na aeronave e, finalmente, a classificação propriamente dita da severidade. Todo esse trabalho é normalmente capturado em um documento chamado FHA (MANELLI NETO, 2011, p. 15).

Segundo Serra (2008), um projeto seguro é a consequência lógica de decisões corretas tomadas ao longo de todo um programa. Essas decisões decorrem de compromissos equilibrados, tomados com competência, experiência e conhecimento dos diversos fatores que influenciam os requisitos do programa.

A segurança é o valor central que deve ser equilibrado com outros valores vitais ao programa como custo do produto, custo operacional e desempenho.

O requisito FAR 25.1309 indica que “a ocorrência de qualquer condição de falha que possa impedir a continuidade do voo e pouso seguro da aeronave deve ser extremamente improvável”. A definição quantitativa de extremamente improvável pode ser encontrada na *Advisory Circular* 25.1309: “Falhas extremamente improváveis são aquelas que têm uma ordem de grandeza menor ou igual à $1x10^{-9}$ por hora”.

A explicação sobre a origem da definição da probabilidade de $1x10^{-9}$ falha por hora pode ser encontrada em Serra (2008):

- O nível máximo aceitável para ocorrência de um acidente catastrófico era de 1 acidente por milhões de horas de voo, ou seja, a probabilidade de ocorrência era $1x10^{-6}$;
- A proporção de acidentes causados por sistemas era 10% do total, portanto o nível máximo de risco para acidentes causados pelo conjunto de todos os sistemas era de $1x10^{-7}$;
- Arbitrou-se na época que um avião teria 10 sistemas capazes de produzir falhas catastróficas, o que trazia o risco máximo aceitável para $1x10^{-8}$ por sistema;
- Arbitrou-se ainda que cada um desses 10 sistemas teria 10 condições de falhas catastróficas, levando o risco máximo para $1x10^{-9}$ por condição de falha.

2.3 Normas e Padrões da Indústria Espacial

A *European Cooperation for Space Standardization* – Cooperação Europeia para a Normatização do Espaço - ECSS é considerada um esforço conjunto entre a Agência Espacial Europeia - ESA, as agências espaciais nacionais e a indústria europeia com o objetivo de desenvolver normas e padrões comuns.

Algumas dessas normas são aplicáveis ao escopo deste trabalho e, portanto, farão parte dos estudos realizados. As seções subsequentes ilustram as principais normas utilizadas para este trabalho.

A *National Aeronautics and Space Administration* – Administração Nacional de Aeronáutica e Espaço - NASA também possui normas e padrões para o desenvolvimento de sistemas e de *software*, como o NASA-GB-8719.13 (2004), intitulado "*NASA Software Safety Guidebook*" (Guia de Segurança de *Software* da NASA) e o NASA-GB-A201 (1989), intitulado "*Software Assurance Guidebook*" (Guia de Garantia de *Software*). Todavia, historicamente, o INPE utiliza mais as referências oriundas da Europa. Portanto, para adequar os processos à utilização do INPE, este trabalho não utiliza as referências da agência espacial americana, buscando como fonte primária as normas da ECSS.

2.3.1 ECSS-P-001

A norma ECSS-P-001 (2000) é o glossário de termos que são utilizados nos demais padrões da ECSS. Um conjunto de importantes definições para este trabalho, segundo a referida norma, pode ser visto a seguir.

- Confiabilidade (*reliability*): A probabilidade de um item realizar a função requerida sob condições determinadas em um intervalo de tempo determinado;
- Requisito (*requirement*): Uma condição que deve ser cumprida;
- Validação (*validation*): Confirmação através de análise de evidência objetiva de que os requisitos para um uso específico foram atingidos;
- Verificação (*verification*): Confirmação através de análise de evidência objetiva de que o requisito especificado foi cumprido;

2.3.2 ECSS-E-ST-10C

A norma ECSS-E-ST-10C (2009) intitulada "*System Engineering General Requirements*" (Requisitos Gerais para Engenharia de Sistemas) é um padrão publicado pela ECSS em 06 de março de 2009 que especifica os requisitos de

implementação da Engenharia de Sistemas para o desenvolvimento de produtos e sistemas espaciais.

A Figura 2.1, extraída da norma ECSS-E-ST-10C (2009), mostra as funções e fronteiras da Engenharia de Sistemas.

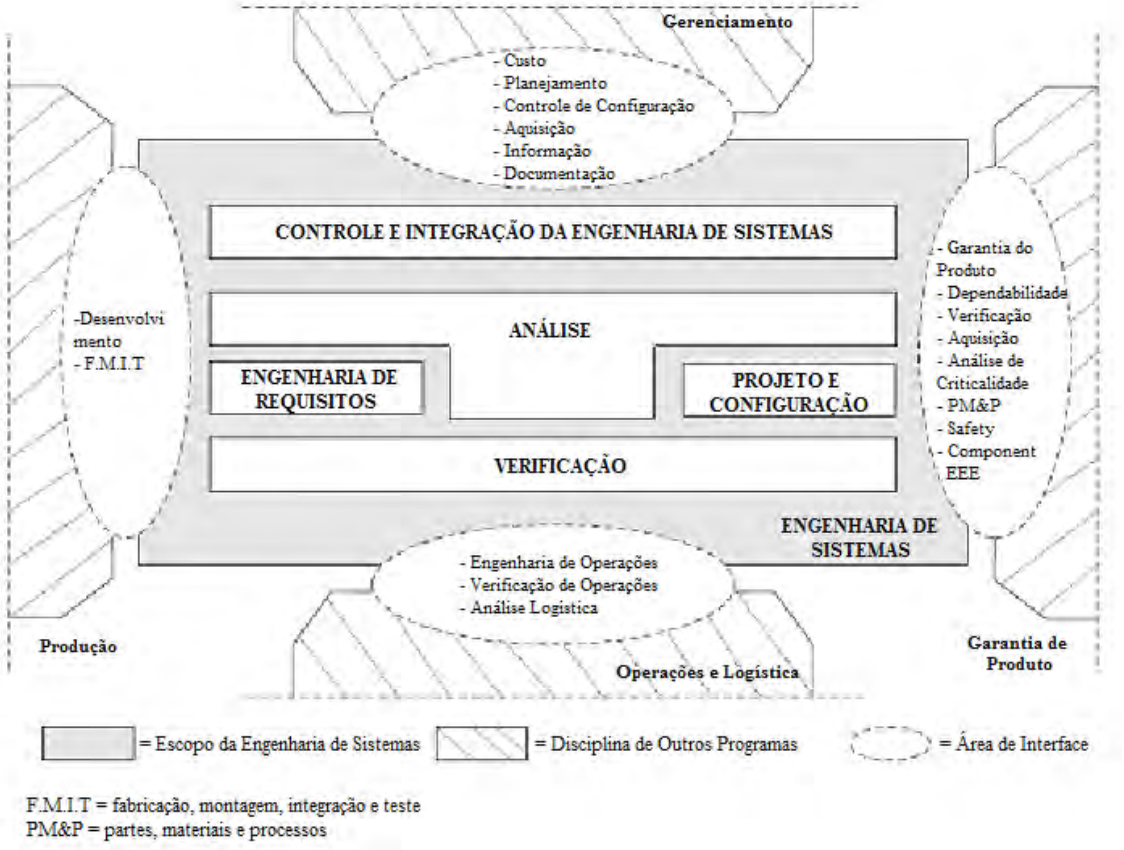

Figura 2.1 – Funções e fronteiras da Engenharia de Sistemas.

Fonte: Adaptada de ECSS-E-ST-10C (2009).

O documento mostra quais são as atividades de Engenharia de Sistemas que devem ser desenvolvidas para cada uma das fases de projeto:

- Fase 0: Análise da missão – Identificação das necessidades
- Fase A: Viabilidade
- Fase B: Definição Preliminar
- Fase C: Definição Detalhada
- Fase D: Qualificação e Produção
- Fase E: Operação/Utilização
- Fase F: Descarte

Todavia, o documento apenas cita que uma das atividades de gerenciamento da Engenharia de Sistemas é a análise de confiabilidade, disponibilidade, manutenabilidade e segurança (RAMS – *Reliability, Availability, Maintainability and Safety*). Não existe um maior detalhamento de quais análises devem ser conduzidas e de que forma devem ser feitas. Também não é encontrada uma referência para outro padrão que inclua as análises de confiabilidade em seu escopo.

Em relação a requisitos de *Software*, a norma define de maneira inespecífica que devem ser feitas atividades para garantir que o projeto contemple as características de *software* e que a verificação do produto seja feita também nesse nível. Existem referências explícitas à norma ECSS-E-ST-40C – *Software General Requirements*, que é detalhada na Seção 2.3.3.

2.3.3 ECSS-E-ST-40C

A norma ECSS-E-ST-40C (2009) intitulada "*Software General Requirements*" (Requisitos Gerais de *Software*) é um padrão publicado pela ECSS em 06 de março de 2009 que cobre os aspectos de Engenharia de *Software* espacial, incluindo definição, *design*, produção, verificação e validação de requisitos, transferência, operação e manutenção.

Similarmente à norma ECSS-E-ST-10C (Seção 2.3.2), esta norma não estabelece de forma clara quais as atividades relativas às análises de confiabilidade do *software* em desenvolvimento. Ela apenas menciona que a Especificação de Requisitos de *Software* (SRS – *Software Requirements Specification*) deve listar os requisitos de confiabilidade aplicáveis ao item de *software*.

A Figura 2.2 mostra a estrutura da norma ECSS-E-ST-40C:

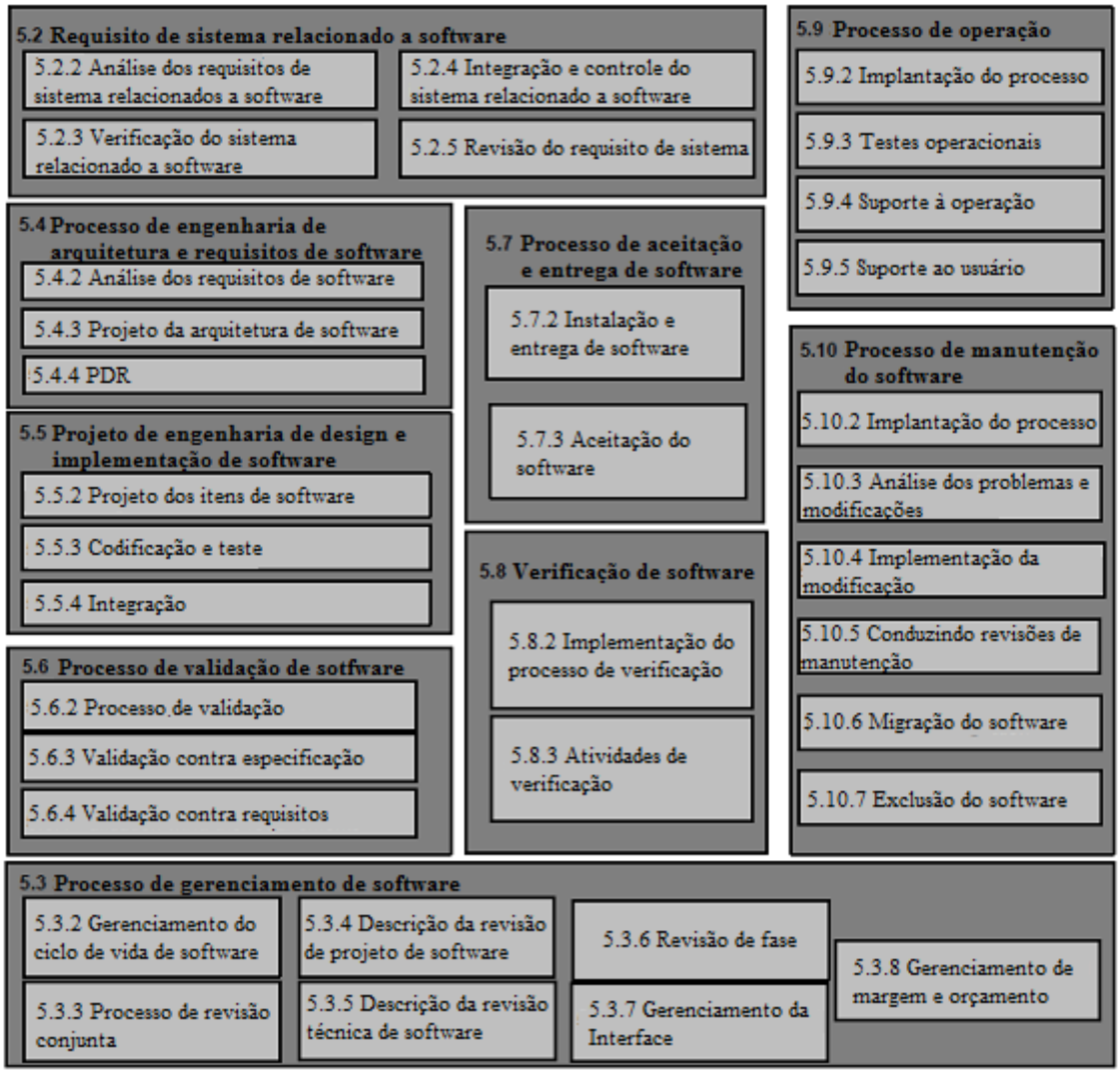

Figura 2.2 – Estrutura da norma ECSS-E-ST-40C.

Fonte: Adaptada de ECSS-E-ST-40C (2009).

O anexo R da norma identifica quais atividades de desenvolvimento de *software* devem ser aplicadas de acordo com a criticalidade do mesmo. A definição das categorias de criticalidade do *software*, baseada na severidade das consequências de falha do sistema, pode ser encontrada no Anexo D da norma ECSS-Q-ST-80C (*Software Product Assurance* – Garantia do Produto para *Software*) e está apresentada na Tabela 2-1.

Categoria	Definição
A	Software que, se não executado ou não corretamente executado, ou cujo comportamento anômalo pode causar ou contribuir para uma falha no sistema resultando em: → consequências catastróficas
B	Software que, se não executado ou não corretamente executado, ou cujo comportamento anômalo pode causar ou contribuir para uma falha no sistema resultando em: → consequências críticas
C	Software que, se não executado ou não corretamente executado, ou cujo comportamento anômalo pode causar ou contribuir para uma falha no sistema resultando em: → consequências graves (*major*)
D	Software que, se não executado ou não corretamente executado, ou cujo comportamento anômalo pode causar ou contribuir para uma falha no sistema resultando em: → consequências insignificantes (*minor*)

Tabela 2-1 – Categorias de criticalidade do sofware.

Fonte: Adaptada de ECSS-Q-ST-80C (2009).

Contudo, o documento não explicita as diferenças entre as definições das consequências citadas (e.g. catastrófica e crítica). Dessa forma, não fica claro de que forma é possível extrair requisitos de confiabilidade baseados na severidade da consequência de falhas do *software*.

2.3.4 ECSS-Q-ST-60-02C

A norma ECSS-Q-ST-60-02C (2008) intitulada "*ASIC and FPGA Development*" (Desenvolvimento de ASICs e FPGAs) é um padrão publicado pela ECSS em 31 de julho de 2008 que define um conjunto de requisitos para os desenvolvedores de circuitos integrados analógicos, digitais e analógico-digitais, tais como ASICs (*Application Specific Integrated Circuits* – Circuitos Integrados para Aplicações Específicas) e FPGAs (*Field Programmable Gate Arrays* – Arranjos de Portas Programáveis em Campo).

As fases de desenvolvimento de ASICs e FPGAs descritas na norma são as seguintes:

•Fase de definição;

•Definição de arquitetura;

•Projeto detalhado;

•Layout;

•Implementação

•Validação e entrega (*release)*

A Figura 2.3 identifica a documentação do projeto de circuitos integrados prevista na norma ECSS-Q-ST-60-02C:

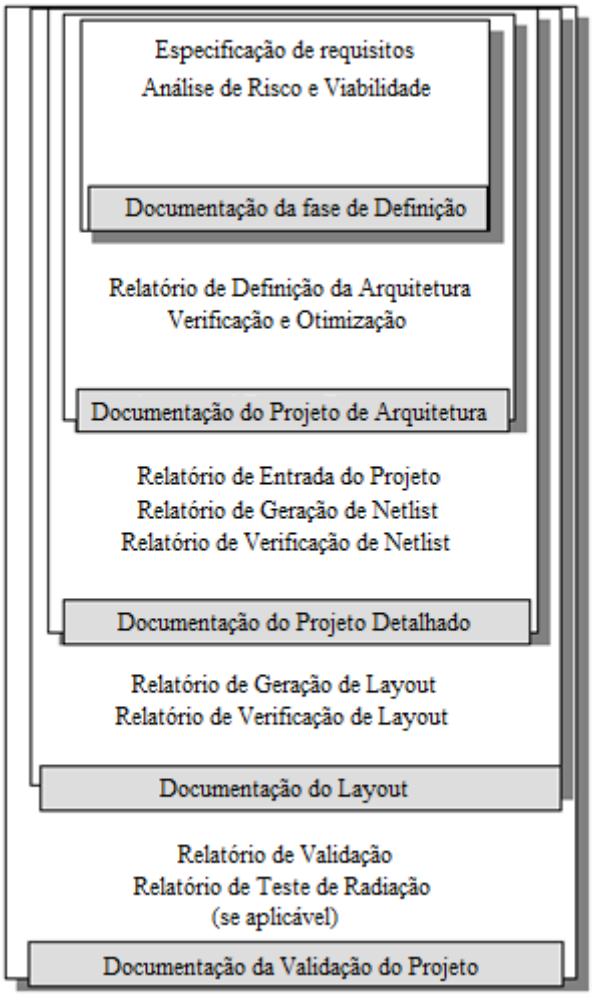

Figura 2.3 – Documentação de projeto para *hardware* eletrônico embarcado.

Fonte: Adaptada de ECSS-Q-ST-60-02C (2009).

O documento não diferencia o desenvolvimento dos circuitos integrados baseado na criticalidade das funções exercidas, diferentemente da norma ECSS-E-ST-40C que apresenta quatro categorias distintas (conforme Tabela 2-1).

2.3.5 ECSS-Q-ST-80C

A norma ECSS-Q-ST-80C (2009) intitulada "*Software Product Assurance*" (Garantia do Produto de *Software*) é um padrão publicado pela ECSS em 06 de março de 2009 que define um conjunto de requisitos para garantia do produto para os desenvolvedores de software para sistemas e aplicações espaciais.

De acordo com a norma, o principal objetivo da garantia do produto é prover confiança adequada ao cliente e ao fornecedor de que o *software* desenvolvido satisfaz seus requisitos durante todo o ciclo de vida do sistema.

A ECSS-Q-ST-80C divide-se em três partes principais:

- Implantação do programa de garantia do produto de *software*
- Garantia do processo de *software*
- Garantia da qualidade do produto de *software*

A Figura 2.4 mostra o relacionamento da ECSS-Q-ST-80C com outras normas ECSS, em especial com a norma ECSS-E-ST-40C.

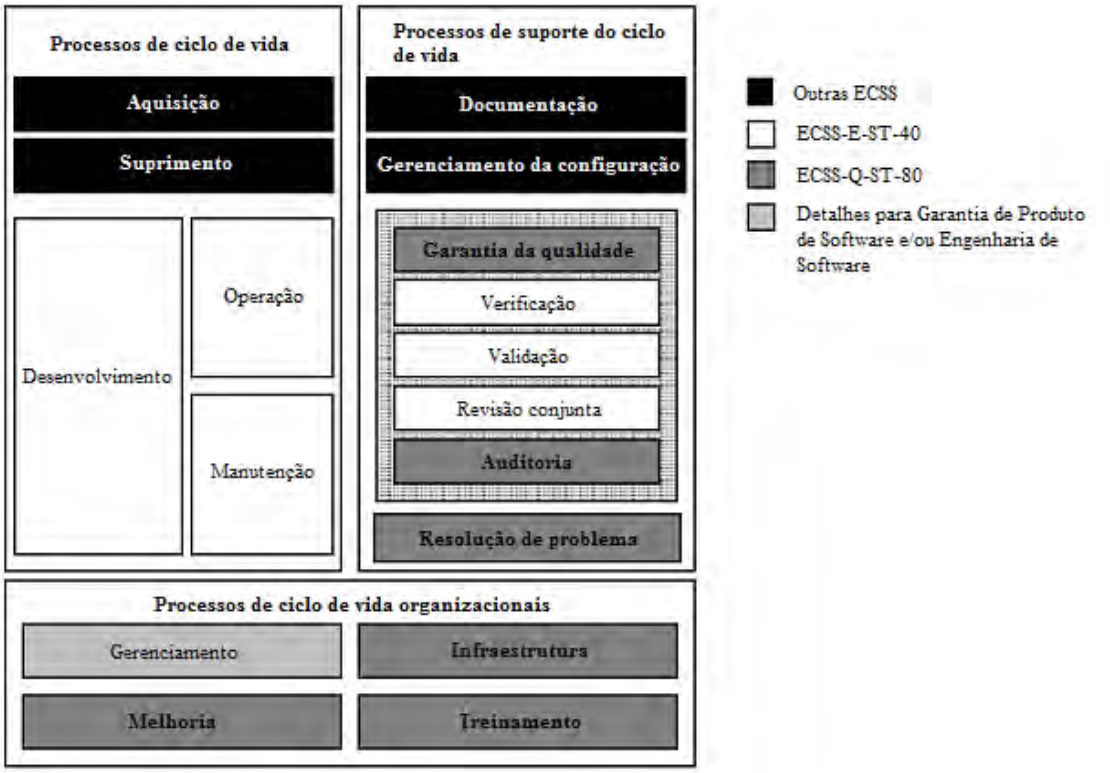

Figura 2.4 – Processos de *software* nas normas ECSS.
Fonte: Adaptada de ECSS-Q-ST-80C (2009).

2.4 Normas e Padrões da Indústria Aeronáutica

A indústria aeronáutica, por tratar com sistemas complexos ou altamente integrados, utiliza uma série de normas e padrões para desenvolvimento de sistemas, *software* e *hardware*.

Algumas dessas referências podem ser aplicadas na indústria espacial de forma a melhorar a qualidade e a confiabilidade dos sistemas desenvolvidos.

2.4.1 SAE ARP-4761

A SAE ARP-4761 (1996) é uma *Aerospace Recommended Practice* (Prática Aeroespacial Recomendada) intitulada *"Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment"*

(Diretrizes e Métodos para Conduzir o Processo de *Safety Assessment* em Sistemas e Equipamentos Embarcados Civis) publicada pela *Society of Automotive Engineers - SAE* em 1996 com o objetivo de ser um guia para os processos de *Functional Hazard Assessment* (FHA), *Preliminary System Safety Assessment* (PSSA) e *System Safety Assessment* (SSA).

A SAE ARP-4761 também apresenta informações sobre métodos de análise como *Fault Tree Analysis* (FTA), *Dependence Diagram* (DD), *Markov Analysis* (MA), *Failure Mode and Effect Analysis* (FMEA), *Failure Modes and Effect Summary* (FMES) e *Common Cause Analysis* (CCA).

De acordo com a norma, o FHA é definido como uma análise sistemática e abrangente das funções para identificar e classificar as condições de falha de acordo com sua severidade. A classificação dessas condições de falha estabelece os requisitos de segurança que a aeronave deve atingir.

O PSSA é utilizado para demonstrar de forma preliminar de que maneira o sistema irá atingir os requisitos quantitativos e qualitativos para os riscos (*hazards*) identificados. O PSSA é uma análise iterativa embutida no desenvolvimento do sistema que deve ocorrer desde as primeiras fases do projeto.

Já o SSA é uma avaliação sistemática e abrangente do sistema implementado para mostrar que os requisitos de segurança relevantes foram atingidos. O processo de análise é similar ao PSSA, mas com alteração de escopo.

Obviamente, o conceito de segurança (*safety*) é extremamente importante para a indústria aeronáutica, considerando o risco de vida de pessoas associado a um projeto. Todavia, as análises e processos propostos pela SAE ARP-4761 podem ser utilizados para melhoria da confiabilidade dos sistemas espaciais. O intento desse trabalho é identificar quais métodos podem auxiliar os engenheiros de sistemas espaciais a projetarem sistemas com níveis de segurança adequados.

2.4.2 SAE ARP-4754

A SAE ARP-4754 (1996) é uma *Aerospace Recommended Practice* (Prática Aeroespacial Recomendada) intitulada *“Certification Considerations for Highly Integrated or Complex Aircraft Systems”* (Considerações de Certificação para Sistemas Altamente Integrados ou Complexos) publicada pela *Society of Automotive Engineers* - SAE em 1996, com o objetivo de ser um guia para autoridades certificadoras e para as empresas candidatas (*applicants*) à certificação.

A grande preocupação dessa norma é relativa aos sistemas complexos ou altamente integrados. O termo “altamente integrado” refere-se a sistemas que contribuem para múltiplas funções na aeronave. O termo “complexo” refere-se a sistemas cuja segurança (“*safety*”) não pode ser demonstrada somente por testes e cuja lógica é difícil de compreender sem o auxílio de ferramentas analíticas.

A SAE ARP-4754 delineia o modelo de desenvolvimento de sistemas através de múltiplos processos de desenvolvimento de sistemas. A Figura 2.5 mostra um modelo de desenvolvimento de sistemas baseado no escopo do documento:

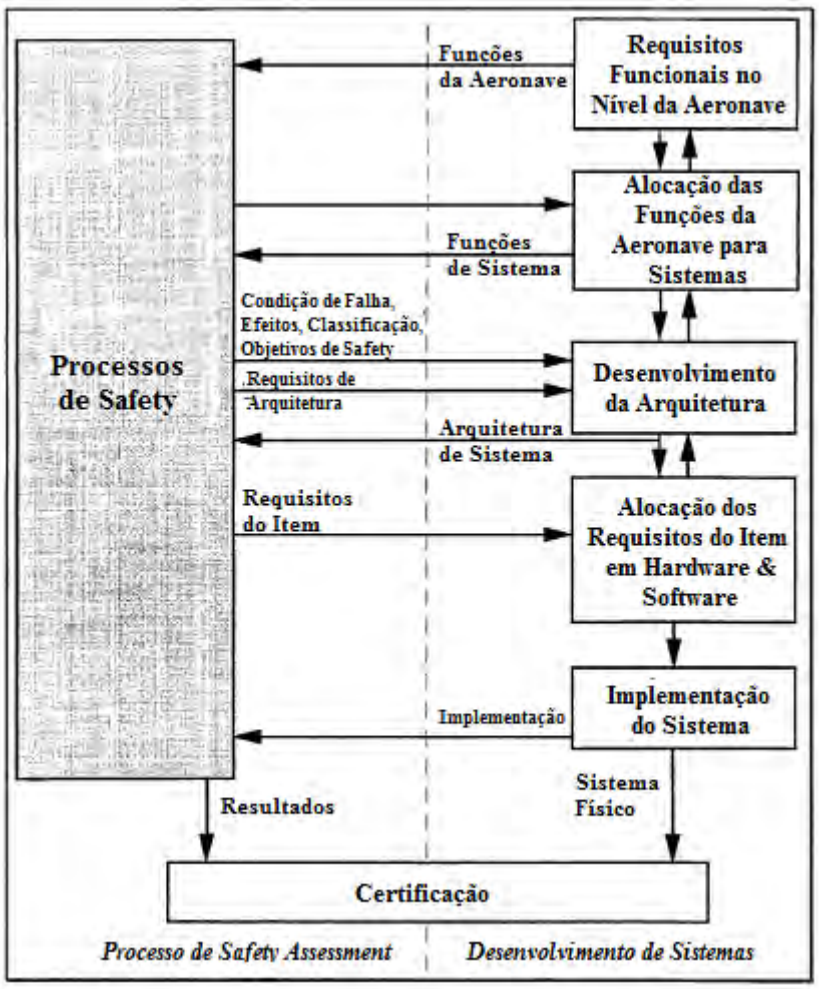

Figura 2.5 – Modelo do Processo de Desenvolvimento de Sistemas.

Fonte: Adaptada de SAE ARP-4754 (1996).

A SAE ARP-4754 introduz o conceito de garantia de desenvolvimento para os sistemas como uma atividade de suporte ao desenvolvimento de sistemas. De acordo com a norma, a garantia de desenvolvimento é um processo de ações sistemáticas e especificamente planejadas que, em conjunto fornecem a confiança de que erros e omissões de requisitos ou projeto foram identificados e corrigidos.

Os requisitos são divididos da seguinte forma:

• Requisitos de segurança (*safety)*;

• Requisitos funcionais (requisitos do cliente, requisitos operacionais, requisitos de performance, requisitos físicos e de instalação, requisitos de manutenabilidade e requisitos de interface);

• Requisitos de certificação.

A SAE ARP-4754 mantém relação estreita com a SAE ARP-4761 (ver Seção 2.4.1), uma vez que os processos de desenvolvimento do sistema são baseados na criticalidade do sistema verificada pelo *Safety Assessment*.

2.4.3 RTCA DO-178B

A RTCA DO-178B (1992), intitulada "*Software Considerations in Airborne Systems and Equipment Certification*" (Considerações de *Software* em Certificação de Equipamentos e Sistemas Embarcados) é um documento publicado pela *Radio Technical Comission for Aeronautics* - RTCA em 1992 que mostra as diretrizes de desenvolvimento e produção de *software* embarcado que exerce suas funções com um nível de confiança compatível com os requisitos de aeronavegabilidade.

Conforme a Seção 1.1 da RTCA DO-178B mostra, essas diretrizes se dão das seguintes maneiras:

•Objetivos para o ciclo de vida do *software*;

•Descrição das atividades e considerações de projeto para alcançar esses objetivos;

•Descrição da evidência que indica que os objetivos foram satisfeitos.

O documento mostra que os requisitos de sistema devem fluir para requisitos de *software* e deve existir o fluxo contrário na forma de iterações para se atingir o desenvolvimento de forma mais eficaz. A Figura 2.6 mostra esse relacionamento:

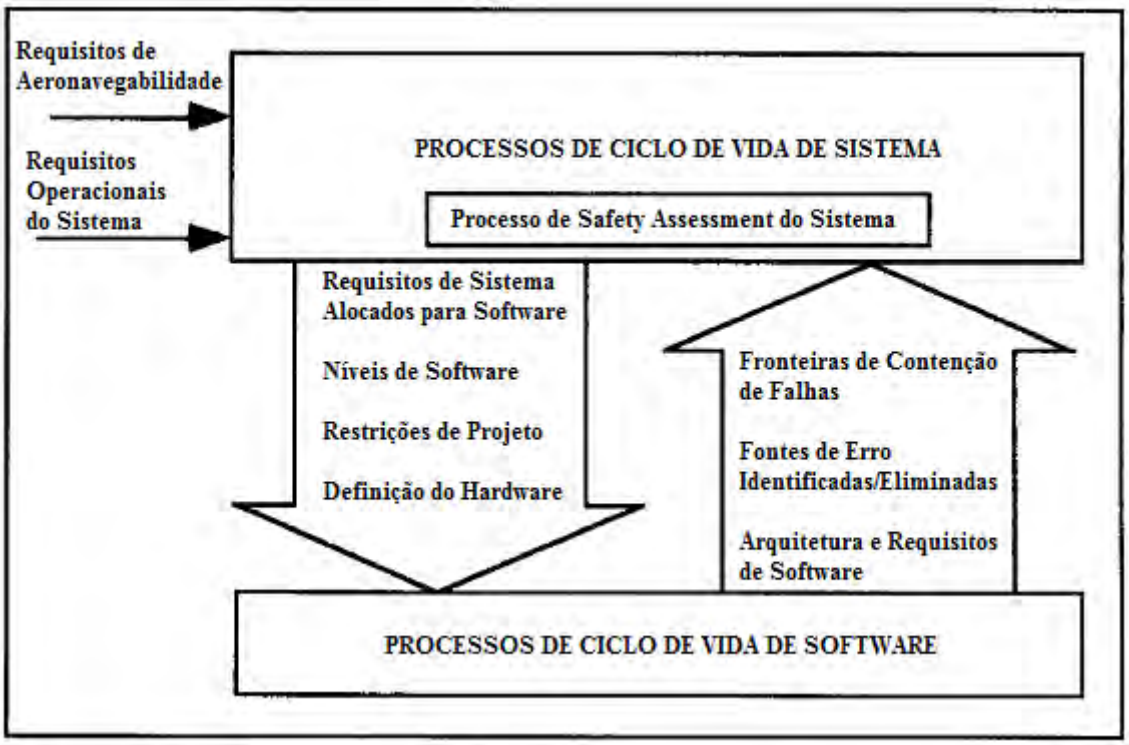

Figura 2.6 – Fluxo de informações entre os desenvolvimentos de sistemas e de *software*.
Fonte: Adaptada de RTCA DO-178B (1992).

A RTCA DO-178B explicita os processos de Planejamento, Desenvolvimento, Verificação, Controle de Configuração e Garantia da Qualidade (além de aspectos de certificação que não são escopo desse trabalho).

A fase de planejamento tem como objetivos principais garantir que as atividades de desenvolvimento de *software* sejam estabelecidas de acordo com os requisitos de sistema; determinar o ciclo de vida do *software* e selecionar o ambiente de desenvolvimento. Os planos de *software* compreendidos pela RTCA DO-178B são os seguintes:

• PSAC – *Plan for Software Aspects of Certification* (Plano para os aspectos de certificação de *software*)
• SDP – *Software Development Plan* (Plano de desenvolvimento de *software*)
• SVP – *Software Verification Plan* (Plano de verificação de *software*)
• SCMP – *Software Configuration Management Plan* (Plano de controle de configuração de *software*)
• SQAP – *Software Quality Assurance Plan* (Plano de garantia de qualidade *software*)

A fase de desenvolvimento é compreendida pelos seguintes processos:

• Processo de requisitos de *software*

• Processo de *design* de *software*

• Processo de codificação de *software*

• Processo de integração

A fase de verificação tem como principais objetivos:

• Garantir que os requisitos de sistema foram alocados em requisitos de *software* de alto nível que por sua vez foram alocados em requisitos de arquitetura e de baixo nível.

• Garantir que o Código Fonte e o Objeto Executável desenvolvidos satisfaçam os requisitos de *software* de baixo nível.

A verificação não é composta apenas de testes. Tipicamente, o processo de verificação é uma combinação de revisões, análises e testes.

O principal objetivo do Controle de Configuração é prover uma configuração definida e controlada do *software* em todas as fases do seu ciclo de vida. Dessa forma, é possível replicar de forma consistente o Código Executável no processo de manufatura.

Portanto, o Controle de Configuração deve ser exercitado durante todas as fases do ciclo de vida do *software*, inclusive no planejamento.

Da mesma forma, a Garantia da Qualidade deve fazer parte de todos os processos de *software*. Seu principal objetivo é garantir que estes processos sejam executados de acordo com o planejado.

Como é possível observar, excetuando-se os aspectos de certificação de *software* que são específicos da Indústria Aeronáutica, a RTCA DO-178B pode ser aplicada quase integralmente na Indústria Espacial. O intento desse trabalho é aproximar essas duas realidades de forma a utilizar os processos da RTCA DO-178B a fim de aumentar a confiabilidade dos *softwares* embarcados em Sistemas Espaciais.

2.4.4 RTCA DO-254

A RTCA DO-254 (2000), intitulada "*Design Assurance Guidance for Airborne Electronic Hardware*" (Diretrizes para Garantia de Projeto para *Hardware* Eletrônico Embarcado) é um documento publicado pela *Radio Technical Comission for Aeronautics* - RTCA em 2000 que, de forma similar à RTCA DO-178B, mostra as diretrizes de desenvolvimento e produção de *hardware* eletrônico embarcado para que o mesmo execute sua função de forma segura, em um ambiente especificado.

De acordo com a Seção 1.1 da RTCA DO-254, os objetivos do documento são:
• Definir os objetivos de garantia de projeto de *hardware*;
• Descrever a base para esses objetivos, de forma a garantir a correta interpretação das diretrizes;
• Fornecer a descrição dos objetivos para permitir o desenvolvimento dos métodos de cumprimento com estas e outras diretrizes;
• Prover diretrizes das atividades de garantia de projeto para cumprir os objetivos de garantia de projeto;
• Permitir flexibilidade na escolha dos processos necessários para cumprir os objetivos deste documento incluindo melhorias.

As diretrizes da RTCA DO-254 são aplicáveis tanto a ASICs (*Application Specific Integrated Circuits* – Circuitos Integrados para Aplicações Específicas) quanto a PLDs (*Programmable Logic Devices* – Dispositivos Lógicos Programáveis), bem como a componentes COTS (*Commercial-Off-The-Shelf* – Componentes "de Prateleira").

Assim como na RTCA DO-178B, os requisitos de *hardware* devem se originar de requisitos de sistema e a definição dos níveis de garantia de desenvolvimento de sistema associados.

A Figura 2.7 mostra a relação entre os sistemas, *safety assessment*, desenvolvimento de *software* e desenvolvimento de *hardware*.

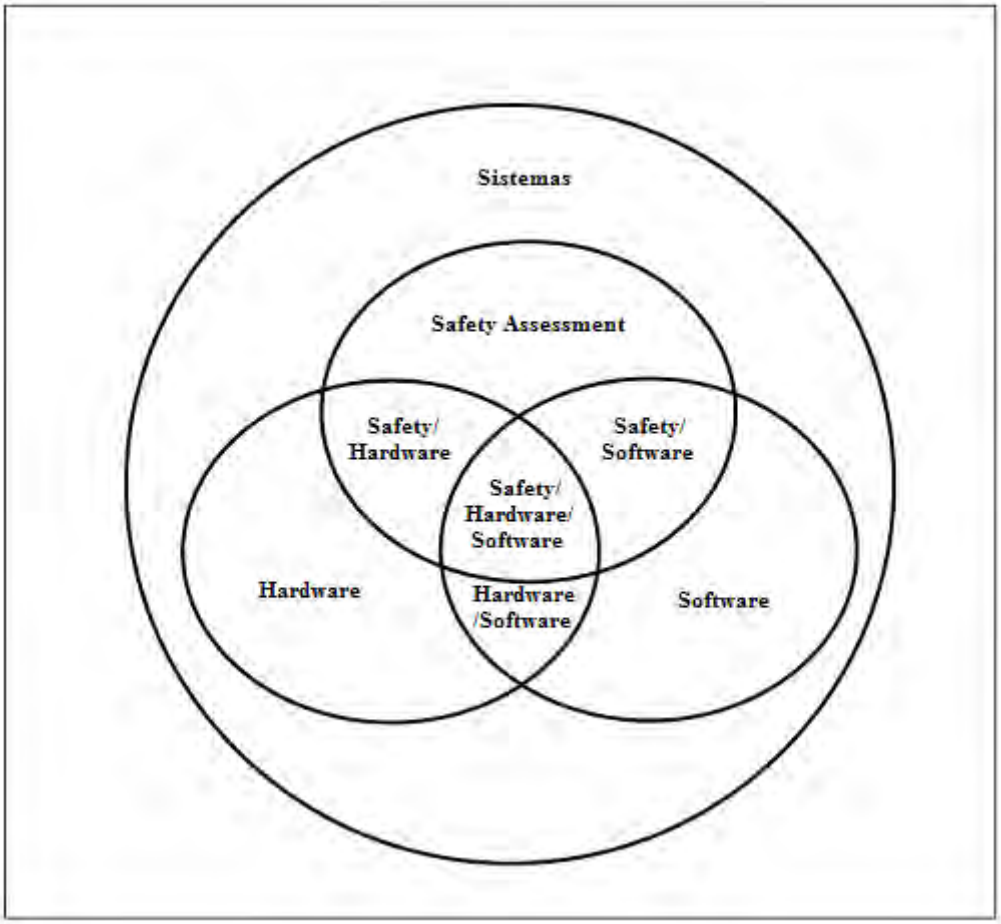

Figura 2.7 – Relacionamento entre sistemas, *software, hardware* e segurança (*safety*).
Fonte: RTCA DO-254 (2000).

A RTCA DO-254 engloba os seguintes processos: Planejamento, Projeto de *Hardware* (*Hardware Design*), Validação e Verificação, Controle de Configuração e Garantia de Processo.

Os objetivos da fase de planejamento incluem:

• Garantir que os processos do ciclo de vida de projeto de *hardware* sejam definidos;

• Garantir que os padrões sejam selecionados e definidos;

• Selecionar ou definir os ambientes de desenvolvimento e verificação.

Os processos de projeto de *hardware* devem produzir um item de *hardware* que cumpre os requisitos alocados para *hardware* oriundos dos requisitos de sistema. A RTCA DO-254 descreve cinco processos principais para o projeto

de *hardware*: Captura de Requisitos, Projeto Conceitual, Projeto Detalhado, Implementação e Transição para Produção.

Considerando as definições descritas nas Seções 2.1.1 e 2.1.2, a validação busca assegurar que os requisitos derivados do item de *hardware* estejam corretos e completos, em relação aos requisitos de sistemas alocados em *hardware*. Enquanto isso, a verificação busca assegurar que a implementação do *hardware* cumpra todos os requisitos de *hardware* (inclusive os derivados).

Os processos de Controle de Configuração e Garantia de Processo têm objetivos similares aos da RTCA DO-178B (Seção 2.4.3). São atividades suporte que devem ser aplicadas durante todo o ciclo de vida do *hardware*.

Assim, da mesma maneira que a RTCA DO-178B, os processos da RTCA DO-254 podem ser aplicados à Indústria Espacial. Novamente, este trabalho busca relacionar essas duas realidades a fim de melhorar a confiabilidade dos sistemas espaciais.

2.4.5 SAE ARP-4754A e RTCA DO-178C

Recentemente, foram emitidas revisões das normas SAE ARP-4754 e RTCA DO-178B. A SAE ARP-4754A (2010) teve seu título alterado para *Guidelines for Development of Civil Aircraft and Systems* (Diretrizes para o Desenvolvimento de Sistemas e Aeronaves Civis). A norma indica que: “Este documento discute o desenvolvimento de sistemas da aeronave levando em consideração o ambiente de operação geral e as funções da aeronave. Isto inclui validação de requisitos e verificação da implementação do projeto para certificação e garantia de produto. Ele fornece práticas para demonstrar cumprimento com os regulamentos e serve para auxiliar a companhia em desenvolver seus padrões internos considerando essas diretrizes”.

A principal diferença da nova revisão é a metodologia proposta para a alocação do nível de garantia de desenvolvimento. Basicamente, a norma define os conceitos de Nível de Garantia de Desenvolvimento da Função (FDAL –

Function Development Assurance Level) e Nível Garantia de Desenvolvimento do Item (IDAL – *Item Development Assurance Level*), bem como os métodos para alocação desses níveis.

Já a RTCA DO-178C (2011) indica que: “Este documento fornece recomendações para a produção de *software* para sistemas e equipamentos embarcados que executam suas funções pretendidas com um nível de confiança e segurança que cumpre com os requisitos de aeronavegabilidade. Cumprimento com os objetivos da DO-178C é o meio primário para obter aprovação do *software* utilizado em produtos de aviação civil”.

A revisão da norma resolve inconsistências de terminologias e erros da RTCA DO-178B. A principal diferença entre as normas é a inclusão de suplementos para acomodar novas metodologias de desenvolvimento de *software*, como Desenvolvimento Baseado em Modelos (MDB – *Model-Based Development*), Orientação a Objetos (OOT – *Object Oriented Technology*) e métodos formais (FM – *Formal Methods*).

Considerando a novidade das revisões, as mesmas não serão exploradas por este trabalho, uma vez que as aplicações práticas ainda são incipientes.

2.5 Plataforma MultiMissão (PMM)

A Plataforma MultiMissão foi concebida para ser uma plataforma modular capaz de servir como base para várias missões científicas, de comunicação e de observação da Terra em baixas órbitas terrestres. A PMM é constituída de subsistemas básicos que fornecem o essencial para o funcionamento do satélite e um suporte para a integração de uma carga útil que será escolhida de acordo com a missão que o satélite irá desempenhar.

Os subsistemas que compõem a PMM são:

- Subsistema de Estrutura, para prover suporte mecânico para todos os subsistemas da PMM, para equipamentos e acessórios e para o módulo

de carga útil, enquanto mantida no solo, durante o lançamento e no ambiente espacial.

- Subsistema de Suprimento de Energia, para converter a energia solar incidente no conjunto de painéis fotovoltaicos em energia elétrica, para armazenar essa energia em baterias e para prover controle e distribuição dessa energia para vários equipamentos da plataforma e da carga útil, usando uma arquitetura não regulada de distribuição de energia.
- Subsistema de Controle de Temperatura, para prover distribuição de calor de forma que todos os equipamentos a bordo operem dentro de suas faixas designadas de temperatura, em todas as possíveis atitudes do satélite experimentadas durante a vida nominal da missão.
- Subsistema de Controle de Atitude e Tratamento de Dados, para prover controle de atitude e órbita em um modo estabilizado em três eixos, possibilitando apontamento para a Terra, para uma posição inercial ou para o Sol, com as respectivas precisões especificadas. Este subsistema também deverá prover processamento de dados, capacidade de armazenamento, gerenciar a comunicação com a carga útil e para controlar os equipamentos da PMM através de um computador de bordo central.
- Subsistema de Propulsão, para prover aquisição de órbita e manutenção usando um tipo de propulsor monopropelente a hidrazina.
- Subsistema de Telemetria e Telecomando (TT&C), para prover comunicação entre a plataforma e as estações terrestres de TT&C garantindo a monitoração e o controle do veículo espacial durante todas as fases da missão.

A PMM deve prover os seguintes serviços para a carga útil:

- Telemetria, telecomando e transmissão de dados através de linhas seriais dedicadas.
- Suprimento de energia.
- Armazenamento de dados.

Uma visão da disposição interna dos equipamentos da PMM é apresentada na Figura 2.8.

Figura 2.8 – Plataforma MultiMissão (PMM).

Fonte: INPE (2001).

3 COMPARAÇÃO ENTRE AS NORMAS DE DESENVOLVIMENTO DE SISTEMAS E *SOFTWARE* DA INDÚSTRIA ESPACIAL E DA INDÚSTRIA AERONÁUTICA

Como apresentado no Capítulo 2, as indústrias automobilística e aeronáutica possuem normas para sistemas e *software* que possibilitam o desenvolvimento de produtos complexos ou altamente integrados através de processos e atividades que garantem adequadamente a segurança e a confiabilidade desses sistemas.

Da mesma forma, a indústria espacial tem buscado atualizar seus padrões no sentido de buscar a melhoria dos processos de desenvolvimento com objetivos semelhantes aos das indústrias supracitadas.

Considerando as características inerentes a essas tecnologias e as diferenças conceituais, esse trabalho pretende alinhar e comparar definições e abordagens de tais indústrias de forma a absorver as melhores práticas e propor aperfeiçoamentos aos processos de desenvolvimento de sistemas e *software* do INPE.

3.1 Desenvolvimento de Sistemas: ECSS-E-ST-10C e SAE ARP-4754

Como visto anteriormente, as normas que regem o desenvolvimento de sistemas nas indústrias espacial e aeronáutica são, respectivamente, a ECSS-E-ST-10C e a SAE ARP-4754. Nas seções subsequentes, é possível avaliar e comparar processos e atividades descritas nas normas supracitadas.

3.1.1 Determinação de Requisitos

Tanto a ECSS-E-ST-10C quanto a SAE ARP-4754 indicam que os processos de Engenharia de Requisitos devem passar, naturalmente, pela geração de um conjunto de requisitos que servirão como base para o desenvolvimento dos sistemas.

Uma das maneiras de iniciar a geração dos requisitos de sistemas é através do processo de elicitação de requisitos (*requirements elicitation*) que é a busca pelas necessidades dos interessados. Aurum e Wohlin (2005) mostram que: “o processo de elicitação de requisitos envolve um conjunto de atividades que devem permitir a comunicação, priorização, negociação e colaboração com todos os interessados (*stakeholders*) relevantes”.

De acordo com a ECSS-E-ST-10C, a organização deve derivar, gerar, controlar e manter um conjunto de requisitos para os elementos de nível mais baixo, definindo suas restrições operacionais e de projeto e os parâmetros de funcionalidade, atuação (*performance*) e verificação necessários para atender os requisitos do cliente.

De forma semelhante, a SAE ARP-4754 indica a importância da Determinação e Captura de Requisitos, primeiramente através da identificação das funções da aeronave e os requisitos associados a essas funções.

Uma primeira diferença entre as normas é a distinção apresentada na SAE ARP-4754 entre os tipos de requisitos, que não é encontrada na ECSS-E-ST-10C, como pode ser visto a seguir.

1- Requisitos de Segurança (*safety*): Requisitos determinados através das análises de modos de falha das funções e sua severidade, conforme descrito na SAE ARP-4761 (ver Seção 2.4.1). A identificação desses requisitos é de vital importância para o ciclo de vida de desenvolvimento do sistema, pois permite a avaliação correta do impacto de mudanças desses requisitos na segurança do sistema (preocupação primária da indústria aeronáutica).
2- Requisitos Funcionais: Requisitos necessários para atingir a performance desejada do sistema nas condições especificadas. Podem ser classificados de acordo com a sua natureza em Requisitos de Cliente, Requisitos Operacionais, Requisitos de Performance, Requisitos de

Instalação, Requisitos de Manutenção (*Maintainability*) e Requisitos de Interface.

3- Requisitos de Certificação: Por se tratar com autoridades de certificação de variadas nações, os sistemas aeronáuticos são sujeitos a funções ou atributos adicionais requeridos por essas autoridades que devem ser incorporados ao desenvolvimento dos sistemas.

Outra diferença entre as normas é a preocupação específica apresentada pela SAE ARP-4754 com os chamados requisitos derivados. De acordo com a SAE ARP-4754, em cada fase de desenvolvimento, decisões são tomadas sobre como um requisito, ou grupo de requisitos, devem ser atendidos. As consequências dessas escolhas de projeto tornam-se requisitos para a próxima fase de desenvolvimento. Como esses requisitos resultam do processo propriamente dito, eles podem não estar relacionados com requisitos de nível superior e são referenciados como requisitos derivados.

A maior vantagem de identificar claramente quais são os requisitos derivados do sistema é permitir a análise do impacto desses requisitos (e de eventuais modificações) nas funções de mais alto nível (inclusive o impacto na segurança da aeronave).

A ECSS-E-ST-10C apenas indica que se deve verificar a alocação dos requisitos derivados em componentes do sistema.

3.1.2 Verificação de Requisitos

Para comparar a verificação de requisitos entre as normas de desenvolvimento de sistemas SAE ARP-4754 e ECSS-E-ST-10C, primeiramente é necessário equalizar conceitos fundamentais dessas normas.

Em relação à verificação, a SAE ARP-4754 indica que: “O objetivo da verificação é assegurar que cada nível de implementação cumpre seus requisitos especificados”.

A SAE ARP-4754 descreve os objetivos do processo de verificação da seguinte forma:

a) Confirmar que as funções pretendidas foram corretamente implementadas.
b) Confirmar que os requisitos foram satisfeitos.
c) Assegurar que as análises de *safety* continuam válidas para o sistema implementado.

Ainda, de acordo com a norma: “O processo de verificação assegura que a implementação satisfaz os requisitos validados”.

A partir das definições apresentadas, pode-se notar claramente que para a SAE ARP-4754, a verificação acontece após a validação de requisitos.

De forma semelhante, a ECSS-E-ST-10C indica que o objetivo da verificação é: “demonstrar que a entrega (*deliverables*) está conforme com os requisitos especificados, incluindo qualificação e aceitação”.

A ECSS-E-ST-10C faz inúmeras referências à norma ECSS-E-10-02A (1998), intitulada *Space Engineering – Verification* (Engenharia Espacial – Verificação) que define os seguintes objetivos para a verificação:

a) qualificar o projeto;
b) assegurar que o produto está de acordo com o projeto qualificado, está livre de defeitos de fabricação e é aceitável para uso;
c) verificar que o sistema espacial (incluindo ferramentas, procedimentos e recursos) será capaz de cumprir os objetivos da missão;
d) confirmar a integridade do produto e a performance após fases específicas do ciclo de vida do projeto (e.g. pré-lançamento, em órbita, pós-aterrissagem).

Ambas as normas indicam quais são os métodos que devem ser utilizados durante a verificação. Existem algumas diferenças entre os métodos e as suas utilizações, como pode ser visto a seguir.

A norma ECSS-E-10-02A apresenta os seguintes métodos de verificação: Teste, Análise, Revisão de Projeto e Inspeção. Já a SAE ARP-4754 acrescenta aos métodos supracitados a Experiência em Serviço, que é considerada parte da Análise na ECSS-E-10-02A (nesse caso, denominada Similaridade).

Para ambas as normas, a Inspeção está ligada à verificação da implementação física e requisitos de manufatura, enquanto a Revisão de Projeto está relacionada com documentações técnicas e relatórios de projeto. A SAE ARP-4754 ainda explicita a preocupação com o funcionamento do sistema ou item em condições normais e não normais (anormais).

Em relação à Análise, como mencionado anteriormente, a ECSS-E-10-02 considera a Análise de Similaridade como parte desse método. De forma semelhante à Experiência em Serviço da SAE ARP-4754, a verificação por similaridade é aplicável quando o item sendo verificado é similar a outro item que foi verificado anteriormente.

Além da similaridade, ambas as normas indicam o Modelamento como método de verificação aceitável. Com a crescente complexidade dos sistemas e os altos custos envolvidos nos testes, esse método tem sido cada vez mais utilizados pelas indústrias aeronáutica e espacial.

Hull, Jackson e Dick (2011) indicam que:

> O modelamento de sistemas auxilia a análise e o processo de *design*, introduzindo um grau de formalidade na maneira como os sistemas são definidos. Durante o desenvolvimento de sistemas, é usual o caso de figuras serem usadas para ajudar a visualizar alguns aspectos do desenvolvimento. Modelamento fornece uma maneira de formalizar essas representações, através de diagramas (HULL, JACKSON E HILL, 2011).

Por fim, o método mais utilizado para verificação é o de Teste. De acordo com a SAE ARP-4754, os objetivos da verificação por testes são:

a) Demonstrar que a implementação do sistema ou item executa suas funções pretendidas (*intended functions*). Teste da função pretendida envolve avaliação contra critérios de passa/falha (*pass/fail criteria*) estabelecidos pelos requisitos de *safety*.

b) Prover confiança de que o sistema implementado não executa funções não pretendidas (i.e., não fazem parte do projeto conscientemente) que impactam na segurança (*safety*). Testes *ad hoc*, e de vigilância especial durante testes normais, podem ser utilizados para identificar operações não pretendidas do sistema ou item ou efeitos colaterais. Deve ser notado que a ausência completa de funções não pretendidas nunca pode ser estabelecida por teste.

Ainda de acordo com a SAE ARP-4754, os seguintes pontos devem ser especificados para cada teste:

a) Entradas necessárias – a variabilidade deve ser considerada ao estabelecer os critérios de teste
b) Ações requeridas
c) Resultados esperados e as tolerâncias associadas a esses resultados.

Uma grande diferença entre as normas de desenvolvimento de sistemas aeronáuticos e espaciais é a correlação entre essas normas e as discussões relativas à segurança (*safety*) e confiabilidade. Como visto acima, a SAE ARP-4754 explicita a preocupação com os impactos em segurança durante a verificação. De forma diferente, a ECSS-E-ST-10C (bem como a ECSS-E-10-02A) não apresenta a garantia da segurança como um dos objetivos da verificação.

Essa preocupação específica com a segurança se traduz no processo de Atribuição de Nível de Garantia de Desenvolvimento, conhecido com DAL (*Development Assurance Level*) *Assignment*.

O DAL de um sistema é determinado de acordo com a classificação da condição de falha mais severa associada às funções exercidas pelo sistema. Quando aplicável, o DAL pode ser associado a um subsistema ou até a um grupo de requisitos que se relacionam com determinada função que tem suas condições de falha avaliadas.

A Tabela 3-1 mostra a relação entre a classificação da condição de falha e o DAL do sistema, de acordo com a SAE ARP-4754.

Tabela 3-1 – Classificação da condição de falha e DAL do sistema.

Failure Condition Classification	System Development Assurance Level
Catastrophic	A
Hazardous / Severe Major	B
Major	C
Minor	D
No Safety Effect	E

Fonte: SAE ARP-4754 (1996).

Considerando a alocação de DAL apresentada, a verificação dos requisitos de sistema pode ser mais exigente ou mais simplificada, de acordo com a Tabela 3-2, onde R indica que o método é recomendado (*recommended*), A indica que pode ser negociado (*as negotiated*) e N indica que o método não é necessário (*not required*).

Tabela 3-2 – Métodos de verificação de requisitos baseados no DAL do sistema

Método	DAL A e B	DAL C	DAL D	DAL E
Inspeção, Revisão, Análise ou Teste	Teste e ao menos um outro	Um ou mais	A	N
Teste (função não pretendida)	R	A	A	N
Experiência em serviço	A	A	A	A

Fonte: Adaptada de SAE ARP-4754 (1996).

3.1.3 Validação de Requisitos

Assim como na verificação, é importante harmonizar os conceitos entre as duas normas, a fim de garantir uma comparação mais precisa.

As referências à validação de requisitos na norma ECSS-E-ST-10C são exíguas. O objetivo mais claro da validação encontrado é a indicação de que,

na fase 0 (Análise da Missão), os requisitos de sistema devem ser validados contra as necessidades expressas em conjunto com o cliente. Todavia, a norma não explicita métodos ou técnicas para que essa validação seja feita. Também não existe diferenciação entre as atividades de validação de requisitos baseada na severidade da função e/ou do sistema implementado por esses requisitos.

Existe ainda uma referência à verificação de consistência dos requisitos, onde a organização deve garantir que os requisitos são individual e globalmente consistentes e não redundantes, além de garantir que os requisitos de nível mais baixo são consistentes com os requisitos de sistema. Novamente, a norma não explicita os métodos ou técnicas para análise dessa consistência.

Já a SAE ARP-4754 apresenta atividades e objetivos claros para a validação de requisitos de sistema. Além disso, apresenta a diferenciação dessas atividades baseada no DAL dos requisitos, de forma similar à verificação de requisitos.

A SAE ARP-4754 mostra que a validação de requisitos é uma combinação de processos subjetivos e objetivos para assegurar que os requisitos especificados estão suficientemente corretos e completos, conforme definição apresentada na Seção 2.4.2. De acordo com a norma, os erros na definição de requisitos podem surgir de três causas primárias:

1- Ambiguidade
2- Descrição incorreta
3- Descrição incompleta

A correção, ou exatidão (*correctness*), de um requisito ou de um conjunto de requisitos pode ser feita através da avaliação de perguntas direcionadas, tais como:

a) Os requisitos são rastreáveis a suas fontes?
b) As limitações e premissas foram adequadamente definidas?
c) A implementação do sistema foi adequadamente especificada?

De forma semelhante, alguns exemplos de perguntas que direcionam a avaliação da completude (*completeness*) de um requisito ou de um conjunto de requisitos são:

a) Todos os requisitos estão corretamente descritos?

b) Todas as premissas estão corretas?

Pode-se observar que o caráter objetivo da avaliação é obtido pelo fato das perguntas serem do tipo SIM ou NÃO. Todavia, a subjetividade aparece quando existe um julgamento em relação à adequação das especificações e definições para poder responder a essas perguntas.

A forma mais usada pela indústria aeronáutica para a validação de requisitos é a utilização de *checklists* que contém perguntas que auxiliam a avaliação de correção e completude dos requisitos.

Assim como na verificação de requisitos de sistema, a SAE ARP-4754 apresenta métodos aceitos para a validação: Rastreabilidade (*Traceability*), Análise, Modelagem, Teste, Similaridade e Julgamento de Engenharia.

A Rastreabilidade é considerada parte essencial do processo de validação. O requisito deve ser rastreável a um requisito pai ou aos dados e decisões de projeto de onde o requisito foi derivado.

Em relação à Análise, a SAE ARP-4754 indica que uma grande variedade de técnicas e métodos de análise pode ser utilizada para a validação de requisitos. Especificamente, a norma cita a utilização das análises e métodos descritos na SAE ARP-4761 (ver Seção 2.4.1), tais como *Fault Tree Analysis* (FTA) e *Failure Mode and Effect Analysis* (FMEA).

O uso de Modelagem, já prevista na SAE ARP-4754, para validação de requisitos vem ganhando mais espaço na indústria, uma vez que os sistemas são cada vez mais complexos e os modelos auxiliam no entendimento desses sistemas sem exigir grandes custos de infraestrutura.

De forma semelhante, Testes podem ser usados para validação de requisitos, utilizando-se de maquetes, protótipos e simulações. Assim como no uso de modelos, devem-se tomar precauções para garantir que a maquete ou protótipo utilizado seja representativo do sistema final.

A validação por Similaridade (ou por Experiência em Serviço) é feita através da comparação com requisitos similares de sistemas já em uso. Portanto, quanto mais tempo de serviço o sistema comparado possui, mais forte é o argumento de validação por similaridade. É importante analisar o ambiente operacional dos dois sistemas para garantir o "reuso" de requisitos.

Por fim, o uso de Julgamento de Engenharia é feito através de revisões, inspeções e demonstrações que auxiliam na demonstração da correção e completude dos requisitos. As evidências dessas revisões devem ser claramente rastreadas aos requisitos.

Para guiar os esforços de validação de requisitos, a SAE ARP-4754 sugere os métodos e atividades baseados no DAL do requisito, conforme a Tabela 3-3, onde R indica que o método é recomendado (*recommended*), A indica que pode ser negociado (*as negotiated*) e N indica que o método não é necessário (*not required*).

Tabela 3-3 – Métodos de validação de requisitos baseados no DAL do sistema.

Método	DAL A e B	DAL C	DAL D	DAL E
Rastreabilidade	R	A	A	N
Análise, Modelagem ou Teste	R	R (Ao menos um)	A	N
Similaridade	A		A	N
Julgamento de Engenharia	A		A	N

Fonte: Adaptada de SAE ARP-4754 (1996).

3.1.4 Atividades de Controle

A ECSS-E-ST-10C inclui uma seção relacionada às atividades de controle e integração de Engenharia de Sistemas, que inclui os seguintes itens:

1- Gerenciamento das atividades de Engenharia de Sistemas

2- Planejamento
3- Dados de Engenharia
4- Controle de Interface
5- Coordenar sistemas e unidades
6- Orçamentos técnicos e Política de margens
7- Tecnologia
8- Controle de risco
9- Controle de mudanças e não-conformidades

Essa divisão é um indício claro de uma das principais diferenças entre as normas. Enquanto o foco da SAE ARP-4754 é majoritariamente técnico, a ECSS-E-ST-10C apresenta preocupações focadas nas demandas organizacionais.

Dentre as atividades de Gerenciamento indicadas pela ECSS-E-ST-10C, está a Garantia de Produto (*Product Assurance*). Apesar de a norma não apresentar uma referência explícita nessa seção, existe uma norma específica para a implantação da garantia de produto, a ECSS-Q-ST-10C (2008), intitulada "*Product assurance management*" (Gerenciamento da garantia de produto).

De acordo com a ECSS-Q-ST-10C, o objetivo primário da garantia de produto é assegurar que os produtos espaciais realizem seus objetivos de missão definidos de forma segura, disponível e confiável (*safe, available and reliable way*). Pode-se observar a estreita relação entre segurança e confiabilidade apresentada pelas normas de desenvolvimento. Esse processo atende explicitamente a essas duas preocupações de um projeto.

A norma prevê auditorias feitas por um time dedicado à Garantia de Produto. O objetivo dessas auditorias é verificar a implementação e a efetividade dos processos definidos. De acordo com a ECSS-Q-ST-10C, as auditorias podem ser feitas através de amostras e de forma periódica.

De forma semelhante, a SAE ARP-4754 indica que os objetivos da garantia de produto (chamada na norma de Garantia do Processo – *Process Assurance*) são:

a) Assegurar que todos os planos estão em prática ou sendo desenvolvidos para todos os aspectos do desenvolvimento de sistemas
b) Assegurar que os processos e atividades de desenvolvimento estão sendo conduzidos de acordo com esses planos.
c) Assegurar que as evidências que mostram a implementação dos planos estão disponíveis.

A norma, assim como na ECSS-Q-ST-10C, apresenta a necessidade de revisões periódicas nos planos e processos de desenvolvimento de sistemas. Além disso, cita como exemplo de evidência dessas auditorias o uso de *checklists*.

Além da Garantia de Produto, a ECSS-E-ST-10C indica que a organização deve prover uma avaliação técnica em toda proposta de mudança em uma *baseline* e em toda não-conformidade do produto. De acordo com a norma, o processo para controle de mudanças é definido na ECSS-M-ST-40C (2009), intitulada "*Configuration and Information Management*" (Gerenciamento de Configuração e Informação).

Essa norma define o gerenciamento de configuração como o processo para estabelecer e manter o registro das características físicas e funcionais de um produto comparado com seus requisitos de projeto e operacionais.

A SAE ARP-4754 apresenta três atividades principais relacionadas ao Controle de Configuração: Identificação, Controle de Modificações e Arquivamento. O objetivo da identificação é marcar de forma não ambígua cada item ou dado (e suas versões sucessivas) para estabelecer uma base de controle e referência.

O controle de modificações (que inclui a documentação dos problemas) tem como principal objetivo garantir que as modificações e problemas sejam identificados, avaliados e resolvidos, bem como garantir que todos os dados

gerados nessas atividades sejam armazenados para consulta futura. De acordo com a SAE ARP-4754, as seguintes diretrizes são aplicáveis:

a) Estabelecer meios para documentar as resoluções de problemas e as modificações dos dados de desenvolvimento (não apenas dos requisitos).
b) Preservar a integridade dos itens através de proteção contra modificações não autorizadas.
c) Assegurar que qualquer modificação em um item é apropriadamente identificada.
d) Assegurar que as modificações de um item requerem a modificação de todos os dados associados com esse item.

Por fim, o arquivamento tem como principal objetivo garantir que todos os dados relacionados ao sistema possam ser corretamente armazenados e recuperados. Para tal, é necessário garantir que modificações não autorizadas nos itens armazenados não possam ser feitas, bem como selecionar mídias para armazenamento que minimizem os erros de deterioração.

3.2 Desenvolvimento de *Software*: RTCA DO-178B e ECSS-E-ST-40C

A discussão sobre confiabilidade e segurança de sistemas inclui métodos qualitativos para demonstrar que o sistema sendo avaliado atende às necessidades relativas à disponibilidade e integridade, entre outros. Esses métodos incluem o Diagrama de Blocos de Confiabilidade (RBD – *Reliability Block Diagram)* e as Análises de Árvores de Falhas (FTA – *Fault Tree Analysis*).

Todavia, quando se trata de *software*, não é possível atribuir taxas de falha, probabilidades de falha, confiabilidade ou MTBF (*Mean Time Between Failures* – Tempo Médio Entre Falhas) para avaliar essas necessidades.

A estratégia utilizada, tanto na indústria aeronáutica quanto na indústria espacial, para garantir que o *software* utilizado em seus sistemas se comporte

de maneira confiável e segura é colocar foco no processo de desenvolvimento desse *software*. Como será visto nas seções subsequentes, as normas de ambas as indústrias seguem a filosofia de tratar com diferentes níveis de rigor o processo de desenvolvimento baseado na severidade da falha do *software*.

3.2.1 Estrutura do Documento

A primeira diferença clara entre as normas RTCA DO-178B e ECSS-E-ST-40C é que a primeira explicita em forma de tabelas, no final do documento, quais são os objetivos que devem ser atingidos, quais os tipos artefatos devem ser gerados e qual o rigor de controle de configuração que deve ser estabelecido para cada objetivo.

Essas tabelas, incluídas no anexo A da RTCA DO-178B, mostram ainda quais objetivos devem ser cumpridos baseados no DAL do *software*, bem como qual o grau de independência entre as atividades.

O contexto de cada uma das tabelas supracitadas pode ser visto a seguir:

- A-1: Processo de Planejamento do *Software*
- A-2: Processos de Desenvolvimento do *Software*
- A-3: Verificação das Saídas do Processo de Requisitos do *Software*
- A-4: Verificação das Saídas do Processo de *Design* do *Software*
- A-5: Verificação das Saídas dos Processos de Codificação e Integração do *Software*
- A-6: Teste das Saídas do Processo de Integração
- A-7: Verificação dos Resultados do Processo de Verificação
- A-8: Processo de Gerenciamento da Configuração do *Software*
- A-9: Processo de Garantia da Qualidade do *Software*
- A-10: Processo de Certificação

No total, a RTCA DO-178B apresenta 65 objetivos para o desenvolvimento de *software*, sendo que a quantidade de objetivos que devem ser cumpridos varia de acordo com o DAL do *software*, conforme a Tabela 3-4:

Tabela 3-4 – Quantidade de objetivos da RTCA DO-178B baseado no DAL do *software*.

DAL	Quantidade de Objetivos
A	65
B	64
C	57
D	28

Fonte: Adaptada de RTCA DO-178B (1992).

A norma ECSS-E-ST-40C também apresenta no final do documento um guia, em forma de tabela, baseado na criticalidade do *software*. Todavia, como o documento não apresenta os objetivos de forma clara, esse guia é considerado como “requisito” para o desenvolvimento do *software* para cada seção do documento.

Dessa forma, a norma ECSS-E-ST-40C apresenta um total de 233 requisitos para o desenvolvimento de *software*. No entanto, essa tabela não apresenta qual a independência requerida para cada um dos objetivos nem o controle de configuração considerado aceitável. A Tabela 3-5 mostra a quantidade de requisitos que devem ser cumpridos para cada nível de *software*.

Tabela 3-5 – Quantidade de requisitos da ECSS-E-ST-40C baseado no DAL do *software*.

Nível do *Software*	Quantidade de Requisitos
A	233
B	231
C	218
D	202

Fonte: Adaptada de ECSS-E-ST-40C (2009).

É importante ressaltar que pode não haver diferença entre os níveis A e B, considerando os requisitos que não foram incluídos na Tabela 3-5:

1- Cobertura de Decisão e Condição Modificada do Código Fonte (MCDC – *Modified Condition and Decision Coverage*): A norma exige 100% de análise de cobertura para nível A, mas deixa o valor do nível B em

aberto. Assim, pode ser estabelecido que para determinado *software* de nível B, esse requisito tem que ser atingido em 100%.

2- Cobertura do Código Objeto: Esse requisito só precisa ser cumprido quando a rastreabilidade entre o código fonte e o código objeto não puder ser verificada. Nesse caso, para *software* nível A é exigida a análise de cobertura de 100% do código objeto. Ou seja, caso o compilador permita a rastreabilidade entre código fonte e código objeto, não haverá diferença entre os níveis A e B.

3.2.2 Validação de Requisitos

O uso dos termos validação e verificação entre a RTCA DO-178B e a ECSS-E-ST-40C é bastante distinto. No início do documento, a ECSS-E-ST-40C apresenta a seguinte definição para validação: processo para confirmar que as funções e performances da *baseline* de requisitos estão corretamente e completamente implementadas no produto final.

Outra definição encontrada no documento é de que a validação de *software* refere-se a testar o produto contra a especificação técnica e a *baseline* de requisitos. Assim, de acordo com a norma ECSS-E-ST-40C, o processo de validação de *software* consiste de:

- Implementação do processo de validação;
- Atividades de validação com respeito à especificação técnica;
- Atividades de validação com respeito à *baseline* de requisitos.

Nesse momento, é importante estabelecer a diferença entre especificação técnica e *baseline* de requisitos, de acordo com a ECSS-E-ST-10C. A *baseline* de requisitos (RB – *Requirements Baseline*) é constituída por dois documentos principais: *Software System Specification* (SSS – Especificação de Sistema do *Software*) e *Software Interface Requirements Document* (IRD – Documento de Requisitos de Interface de *Software*). Já a Especificação Técnica (TS – *Technical Specification*) é constituída por: *Software Requirements Specification*

(SRS – Especificação dos Requisitos de *Software*) e *Software Interface Control Document* (ICD – Documento de Controle de Interface de *Software*).

O objetivo de cada um desses documentos pode ser visto a seguir:

- *Software System Specification (SSS)*: A especificação de sistema do *software* contém os requisitos do cliente. É a descrição de mais alto nível do *software* e, em conjunto com o Documento de Requisitos de Interface do *Software*, provê os critérios usados para validar e aceitar o *software*.
- *Software Interface Requirements Document (IRD):* O documento de requisitos de interface do *software* é um componente essencial da *baseline* de requisitos e é a entrada primária para revisão dos requisitos de sistema. O documento contém os requisitos do cliente. É a descrição de mais alto nível das interfaces do *software* e, em conjunto com a Especificação de Sistema do *Software*, provê os critérios usados para validar e aceitar o *software*.
- *Software Requirements Specification (SRS):* A especificação dos requisitos do *software* é um componente essencial da especificação técnica. Descreve os requisitos funcionais e não funcionais aplicáveis aos itens de *software*.
- *Software Interface Control Document (ICD):* O documento de controle de interface do *software* é um componente essencial da especificação técnica. Descreve todas as interfaces externas (preliminares e atualizadas).

Nota-se pelas definições que a norma ECSS-E-ST-40C não trata da validação de requisitos como uma fase anterior à implementação do produto, que auxilia a antecipação de problemas e reduz custos com modificações, tal qual visto na ECSS-E-ST-10C (ver Seção 3.1.3). Além disso, essa definição de validação do *software* é bastante semelhante ao que foi visto nas normas de desenvolvimento de sistemas como verificação de requisitos por teste.

No objetivo do *Software System Specification* surge outro termo que se relaciona com a validação: a aceitação (*acceptance*). Na ECSS-E-ST-40C, a

aceitação do *software* é definida como um conjunto de testes feitos em conjunto com o cliente utilizando o Ambiente Alvo (*Target Environment*).

Já a RTCA DO-178B define o termo validação como o processo para determinar que os requisitos sejam corretos e completos (semelhante à SAE ARP-4754, ver Seção 3.1.3). Todavia, a única referência encontrada sobre validação é no escopo do documento que indica que a RTCA DO-178B não pretende descrever os processos de Validação.

3.2.3 Verificação de Requisitos

A norma ECSS-E-ST-40C define a verificação como o processo para confirmar que entradas e especificações adequadas existem para todas as atividades, e que os resultados das atividades estão corretos e consistentes com as entradas e especificações.

Assim, nota-se que o processo de verificação relaciona-se primariamente com as transições entre as fases de desenvolvimento do *software*, pois mantém foco na consistência entre entradas (*inputs*), atividades e saídas (*outputs*).

As atividades de verificação, bem como sua descrição podem ser vistas a seguir.

a) Verificação da *Baseline* de Requisitos: o cliente deve verificar se a *baseline* de requisitos (incluindo o IRD) especifica características como ambiente de operação do *software*, sistemas externos que interagem com o *software*, modos/submodos e transição entre modos e margens de memória e alocação de processador.
b) Verificação da Especificação Técnica: o fornecedor deve verificar se a especificação técnica (incluindo o ICD) para garantir, entre outros, que os requisitos de *software* e de interface são consistentes, o rastreio entre requisitos de sistema e de *software* está completo, os requisitos de *software* são verificáveis e que as restrições de implementação foram identificadas.

c) Verificação da Arquitetura do *Software*: o fornecedor deve verificar a arquitetura do *software* e o projeto de interface para garantir, entre outros, que a arquitetura está consistente com os requisitos de *software*, o rastreio entre os requisitos e os componentes do *software* está completo, a operação e manutenção são factíveis e que o projeto implementa uma sequência apropriada de eventos, entradas, saídas, interfaces, fluxo lógico, alocação de tempo e espaço e tratamento de erros.

d) Verificação do Projeto Detalhado de *Software*: o fornecedor deve verificar o projeto detalhado de *software* para garantir, entre outros, que o projeto detalhado é consistente com a arquitetura, o rastreio entre arquitetura e projeto detalhado está completo; teste, operação e manutenção são factíveis, as características do projeto (definição de tarefas e prioridades, mecanismos de sincronização, gerenciamento dos recursos compartilhados) são fornecidas e as escolhas de tempo real são justificadas.

e) Verificação do Código: o fornecedor deve verificar o código para assegurar, entre outros, que este seja consistente com os requisitos e com o projeto do *software*; que é rastreável até o projeto e requisitos de *software*; é testável, correto e conforme com os padrões de codificação; os efeitos de erros em tempo de execução (*run-time errors*) estão controlados e mecanismos de proteção numérica estão implementados. Além disso, o fornecedor deve garantir que a cobertura de código foi atingida, de acordo com a Tabela 3-6:

Tabela 3-6 – Cobertura de código exigida pela ECSS-E-ST-40C baseada no DAL do *software*.

Cobertura do código / Criticalidade	**A**	**B**	**C**	**D**
Source code statement coverage	100%	100%	AM	AM
Source code decision coverage	100%	100%	AM	AM
Source code modified condition and decision coverage	100%	AM	AM	AM
Nota: AM significa que o valor deve ser discutido com o cliente.				

Fonte: ECSS-E-ST-40C (2009).

f) Verificação dos Testes de *Software*: o fornecedor deve verificar os resultados dos testes de *software* para garantir, entre outros, que os testes são consistentes com os requisitos e o projeto detalhado; que

eles são rastreáveis até os requisitos, projeto e código; que os resultados dos testes estão de acordo com os resultados esperados e que os resultados, dados, casos e procedimentos de teste estão mantidos sob controle de configuração.

g) Verificação da Integração do *Software*: o fornecedor deve verificar que a integração do *software* foi feita de acordo com o planejado, assegurando o rastreio até o projeto detalhado do *software*, consistência interna, objetivos de teste de interface e conformidade com os resultados esperados.

h) Verificação da Validação do *Software* com Respeito à *Baseline* de Requisitos e à Especificação Técnica: o fornecedor deve verificar os resultados da validação do *software* assegurando que os requisitos de teste, os casos de teste, as especificações de teste, análise, inspeção e revisão do projeto cobrem todos os requisitos de *software* da especificação técnica e da *baseline* de requisitos.

i) Avaliação da Validação (validação complementar no nível de sistema): o fornecedor deve identificar os requisitos da especificação técnica e da *baseline* de requisitos que não podem ser testados no seu ambiente e solicitar que o cliente faça a validação no nível de sistema.

j) Verificação da Documentação de *Software*: o fornecedor deve verificar a documentação de *software* para assegurar que a documentação está adequada, completa, consistente, foi preparada no tempo correto e o controle de configuração dos documentos segue os procedimentos especificados.

k) Análise de Escalonabilidade (*schedulability analysis*) de *Software* em Tempo Real: o fornecedor deve usar um modelo analítico (caso não exista modelo analítico, pode utilizar modelos e simulação) para realizar uma análise de escalonabilidade e provar que o projeto é factível.

l) Gestão de Orçamentos Técnicos: o fornecedor deve estimar o orçamento técnico (incluindo tamanho de memória e utilização do processador) durante a verificação dos requisitos de *software*. Essas estimativas devem ser atualizadas durante a verificação do projeto detalhado. Durante a codificação, testes e validação, os valores devem ser medidos e comparados com as margens estimadas.

m) Verificação de Comportamento por Modelos: o fornecedor deve verificar o *software* utilizando a visão comportamental do modelo lógico, como suporte para as atividades de verificação dos requisitos, da arquitetura e do projeto detalhado de *software*.

Já a norma RTCA DO-178B define a verificação como sendo uma avaliação técnica do resultado dos processos de desenvolvimento de *software* e do processo de verificação de *software*. Ainda de acordo com a norma, verificação não é simplesmente teste. Testes, em geral, não podem demonstrar a ausência de erros. Como resultado, o termo “verificar” é utilizado quando os objetivos sendo discutidos são basicamente uma combinação de revisões, análises e testes.

Dentre os objetivos apresentados pela norma no Apêndice A (ver Seção 3.2.1), as Tabelas de A-3 a A-7 possuem aqueles objetivos que fazem parte do processo de verificação do *software*. Em linhas gerais, os objetivos da verificação podem ser descritos como:

a) Verificar que os requisitos de sistema alocados para *software* foram desenvolvidos em requisitos de alto nível de *software*.
b) Verificar que os requisitos de alto nível de *software* foram desenvolvidos em arquitetura e requisitos de baixo nível.
c) Verificar que a arquitetura e os requisitos de baixo nível de *software* foram desenvolvidos em código fonte.
d) Verificar que o código objeto executável satisfaz os requisitos de *software*.
e) Verificar que os meios utilizados para satisfazer esses objetivos estão tecnicamente corretos e completos para o nível de *software*.

Em relação às atividades propriamente ditas, a RTCA DO-178B divide suas diretrizes entre Revisões e Análises do *software* e Processo de Teste do *software*. A seguir, pode-se observar um detalhamento das atividades de revisão e análise, bem como seus objetivos:

a) Revisões e Análises dos requisitos de alto nível: Identificar erros nos requisitos que podem ter sido introduzidos durante o processo de elicitação de requisitos de *software*. Os requisitos de alto nível devem:

- atender os requisitos de sistema,
- ser precisos e consistentes,
- ser compatíveis com o *hardware*,
- ser verificáveis,
- estar conforme com os padrões,
- ter rastreabilidade, e
- conter algoritmos precisos.

b) Revisões e Análises dos requisitos de baixo nível: Identificar erros nos requisitos que podem ter sido introduzidos durante o processo de projeto do *software*. Os requisitos de baixo nível devem:

- atender os requisitos de alto nível de *software*,
- ser precisos e consistentes,
- ser compatíveis com o *hardware*,
- ser verificáveis,
- estar conforme com os padrões,
- ter rastreabilidade, e
- conter algoritmos precisos.

c) Revisões e Análises da arquitetura do *software*: Identificar erros nos requisitos que podem ter sido introduzidos durante o desenvolvimento da arquitetura do *software*. A arquitetura do *software* deve:

- ser compatível com os requisitos de alto nível de *software*,
- ser consistente,
- ser compatível com o *hardware*,
- ser verificável,
- estar conforme com os padrões, e
- conter integridade de particionamento.

d) Revisões e Análises do código fonte: Identificar erros nos requisitos que podem ter sido introduzidos durante o processo de codificação do *software*. Essas revisões devem confirmar que as saídas do processo de

codificação são precisas, completas e podem ser verificadas. O código fonte deve:

- atender os requisitos de baixo nível de *software*,
- atender a arquitetura do *software*,
- ser verificável,
- estar conforme com os padrões,
- ter rastreabilidade, e
- ser preciso e consistente.

e) Revisões e Análises das saídas do processo de integração: Assegurar que os resultados do processo de integração estão corretos e completos. As análises devem conter:

- endereços incorretos de *hardware*,
- sobreposições de memória, e
- componentes de *software* faltantes.

f) Revisões e Análises dos casos, procedimentos e resultados dos testes: Garantir que os testes do código foram desenvolvidos e realizados de forma precisa e completa. Deve-se analisar:

- os casos de teste,
- se os casos de teste foram corretamente desenvolvidos em procedimentos de teste e resultados esperados, e
- se os resultados dos testes estão corretos e discrepâncias entre o resultado real e o esperado foram explicadas.

Já os testes de *software* possuem dois objetivos complementares. O primeiro é demonstrar que o *software* satisfaz os seus requisitos. O segundo objetivo é demonstrar com um alto grau de confiança que erros que podem levar a condições de falha inaceitáveis foram removidos. Esse segundo objetivo está diretamente ligado ao escopo deste trabalho, pois as falhas de *software* decorrentes de problemas no processo de desenvolvimento afetam diretamente a disponibilidade e a integridade do sistema.

A Figura 3.1, retirada da RTCA DO-178B, mostra o processo de teste do *software*. Os objetivos dos três tipos de teste são:

a) Teste de integração *hardware*/*software*: verificar a correta operação do *software* no ambiente alvo (*target computer environment*).

b) Teste de integração de *software*: verificar as inter-relações entre os requisitos de *software* e os componentes e verificar a implementação dos requisitos e componentes de *software* na arquitetura do *software*.

c) Teste de baixo nível: verificar a implementação dos requisitos de baixo nível do *software*.

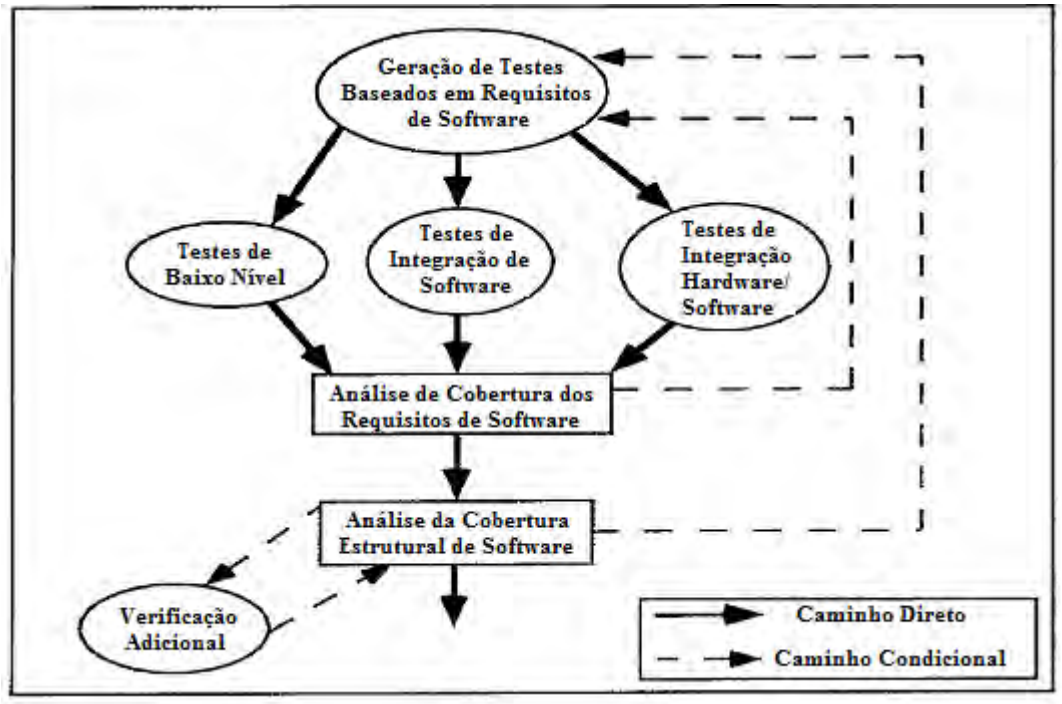

Figura 3.1 – Processo de teste de *software*.

Fonte: Adaptada de RTCA DO-178B (1992).

A RTCA DO-178B indica explicitamente que os casos de testes devem ser baseados nos requisitos (tanto de alto quanto de baixo nível). De acordo com a norma, os testes baseados em requisitos (*requirements-based testing*) são enfatizados porque essa estratégia se mostrou a mais efetiva em revelar erros. As diretrizes para os testes baseados em requisitos indicam que duas categorias de casos de teste devem ser incluídas: testes na faixa normal (*normal range*) e testes de robustez (*robustness*).

O objetivo dos testes de faixa normal é demonstrar a capacidade do *software* de operar com entradas e condições normais. Os testes de faixa normal incluem:

- Variáveis de entrada reais e inteiras devem ser exercitadas usando equivalência de classes válida e limite de valores.
- Para funções temporais (e.g. filtros, integradores), devem ser realizadas múltiplas iterações do código para verificar as características da função no contexto.
- Para transições de estado, os casos de teste devem exercitar as possíveis transições durante a operação normal.
- Para requisitos expressos por equações lógicas, os casos de teste devem verificar o uso de variáveis e os operadores booleanos.

Já os testes de robustez têm como objetivo demonstrar a capacidade do *software* de operar em condições e entradas anormais. Os testes de robustez incluem:

- Variáveis reais e inteiras devem ser exercitadas utilizando equivalência de classes de valores inválidos.
- A inicialização do sistema deve ser verificada em condições anormais.
- Os possíveis modos de falha dos dados de entrada devem ser determinados, especialmente sinais digitais de um sistema externo.
- Quando o contador de *loop* for um valor computado, os casos de teste devem tentar computar valores fora da faixa para o contador do *loop*.
- Verificação para assegurar que os mecanismos de proteção para tempos de quadro excedidos respondem corretamente.
- Para funções temporais (e.g. filtros e integradores), devem-se verificar os mecanismos de proteção contra *overflow* aritmético.
- Para transições de estado, as transições não permitidas pelos requisitos de *software* devem ser forçadas.

Além dos testes, a RTCA DO-178B indica a necessidade de se fazer análise de cobertura dos testes, sendo a atividade dividida em dois processos: Análise de

Cobertura dos Testes Baseados em Requisitos e Análise de Cobertura Estrutural.

O objetivo da Análise de Cobertura dos Testes Baseados em Requisitos é determinar se os testes realizados realmente verificaram a implementação dos requisitos de *software*. Dessa forma, a análise deve mostrar que existem casos de teste para todos os requisitos de *software* e que os casos de teste satisfazem os critérios de teste de faixa normal e de robustez apresentados acima.

Em relação aos testes de robustez, a ECSS-E-ST-40C indica entre os objetivos dos testes de *software* que o teste deve exercitar:

- o código utilizando valores de fronteira em n-1, n, n+1, incluindo instruções de *looping*;
- todas as mensagens e casos de erro definidos no documento de *design;*
- o acesso a todas as variáveis globais;
- valores fora da faixa, incluindo valores que podem causar resultados errados em funções matemáticas; e
- o *software* ao limite dos seus requisitos (teste de fadiga – *stress testing*).

Já o objetivo da análise de cobertura estrutural é determinar quais estruturas do código não foram exercitadas pelos procedimentos de teste baseados em requisitos. De acordo com a norma, a análise deve confirmar o grau apropriado de cobertura estrutural baseado no nível de *software*. De acordo com a Tabela A-7 da RTCA DO-178B, as seguintes análises de cobertura são apropriadas para cada nível de *software*:

Tabela 3-7 – Cobertura de código exigida pela RTCA DO-178B baseada no DAL do *software*.

Análise de Cobertura / Nível de SW	**A**	**B**	**C**	**D**
Statement coverage	C	C	S	N/A
Decision coverage	C	C	N/A	N/A
Modified condition and decision coverage	C	N/A	N/A	N/A
Nota: C indica com independência e S significa sem independência				

Fonte: Adaptada de RTCA DO-178B (1992).

Pode-se observar que a cobertura apropriada para a RTCA DO-178B é bastante semelhante à cobertura requerida pela ECSS-E-ST-40C.

Além disso, a análise de cobertura estrutural deve confirmar o acoplamento de dados e o acoplamento de controle entre os componentes do código. Novamente, a Tabela A-7 da RTCA DO-178B indica para quais níveis de *software* essa análise é necessária:

Tabela 3-8 – Análise de acoplamento de dados e de controle exigida pela RTCA DO-178B.

Análise de cobertura/ Nível de SW	**A**	**B**	**C**	**D**
Acoplamento de dados e acoplamento de controle	C	C	S	N/A
Nota: C indica com independência e S significa sem independência				

Fonte: Adaptada de RTCA DO-178B (1992).

A análise de cobertura estrutural pode revelar estruturas de código não exercitadas durante os testes, que requerem verificação adicional. Essa parte de código não executada pode ser resultado de:

- Deficiências nos casos ou procedimentos de teste;
- Inadequações nos requisitos de *software*;
- Código desativado (*deactivated code*): é uma porção de código utilizada em condições específicas em ambiente controlado, mas que não deve ser executada durante a operação do *software*. Nesse caso, devem ser feitas análises e testes para garantir que essa porção de código não é executada de forma inadvertida.
- Código morto (*dead code*): nesse caso, a porção de código deve ser removida e deve ser avaliada a necessidade de fazer uma nova verificação no código completo.

Cabe ressaltar que a ECSS-E-ST-40C indica explicitamente que o conceito e as preocupações com código desativado (*deactivated code*) foram retirados da RTCA DO-178B. Todavia, o conceito de código morto (*dead code*) não foi introduzido na norma ECSS-E-ST-40C.

3.2.4 Atividades de Controle

De forma semelhante à SAE ARP-4754, a RTCA DO-178B dedica um capítulo para o processo de controle de configuração do *software* e um capítulo para o processo de garantia da qualidade do *software* (na SAE ARP-4754 a atividade é chamada de garantia de processo).

Em relação ao processo de controle de configuração, a RTCA DO-178B apresenta os seguintes objetivos, entre outros:

- Prover configuração definida e controlada do *software* através de todo o ciclo de vida;
- Prover a capacidade de replicar de forma consistente o código objeto executável;
- Prover controle das entradas e saídas de todos os processos do ciclo de vida do *software*;
- Prover pontos conhecidos para revisões e controle de mudanças;
- Prover meios para garantir que os problemas são analisados e as modificações são guardadas, aprovadas e implementadas;
- Prover evidência da aprovação do *software*;

As atividades de controle de configuração previstas pela RTCA DO-178B, bem como seus objetivos podem ser vistos a seguir:

a) Identificação da Configuração: marcar de forma não ambígua cada item ou dado (e suas versões sucessivas) para estabelecer uma base de controle e referência. (Nota: a definição é igual à da SAE ARP-4754).

b) *Baseline* e Rastreabilidade: estabelecer uma base para as atividades do ciclo de vida subsequentes e permitir referência, controle e rastreio entre itens de configuração.

c) Relato de Problemas e Ações Corretivas: registrar falhas de cumprimento com os planos e padrões, registrar deficiências nas saídas dos processos do ciclo de vida do *software*, registrar comportamento anômalo dos produtos de *software* e assegurar resolução desses problemas.

d) Controle de Modificações: permitir o registro, avaliação, resolução e aprovação de mudanças através de todo o ciclo de vida do *software*.

e) Revisão de Modificações: assegurar que os problemas e modificações são analisadas, aprovadas ou rejeitadas, implementadas e que o parecer é fornecido para os processos afetados.

f) Relato do Status da Configuração: prover dados para o gerenciamento da configuração dos processos do ciclo de vida do *software*.

g) Arquivamento, Recuperação e Liberação: garantir que os dados do ciclo de vida do *software* associados com o produto possam ser recuperados em caso de necessidade, bem como garantir que apenas *software*s autorizados sejam utilizados.

h) Controle de Carregamento do *Software*: garantir que o código objeto executável foi carregado no sistema embarcado com as salvaguardas apropriadas, através de procedimentos de identificação e marcação.

i) Controle do Ambiente de Ciclo de Vida do *Software*: garantir que as ferramentas utilizadas para produzir o *software* sejam identificadas, controladas e recuperáveis.

Além disso, a RTCA DO-178B designa duas categorias de controle, CC1 e CC2, de acordo com a Tabela 3-9:

Tabela 3-9 – Categorias de controle da RTCA DO-178B.

Objetivo do Processo de Controle	CC1	CC2
Identificação da configuração	●	●
Baselines	●	
Rastreabilidade	●	●
Relato de problemas	●	
Controle de modificações - integridade e identificação	●	●
Controle de modificações - rastreamento	●	
Revisão das modificações	●	
Relato do status de configuração	●	
Recuperação	●	●
Proteção contra modificações não autorizadas	●	●
Seleção de mídia, *refreshing* e duplicação	●	
Liberação	●	
Retenção de dados	●	●

Fonte: RTCA DO-178B (1992).

Essa diferenciação de categorias de controle de configuração se reflete diretamente nos objetivos do Apêndice A da RTCA DO-178B, uma vez que cada um dos objetivos tem sua categoria de controle identificada nas tabelas.

Diferentemente da RTCA DO-178B, que apresenta no corpo do documento as informações referentes ao controle de configuração, a ECSS-E-ST-40C apresenta referências para outra norma que contém essas informações: a ECSS-Q-ST-80C, intitulada "*Software Product Assurance"* (Garantia do Produto de *Software*), que, por sua vez indica que a norma de gerenciamento da configuração aplicável ao *software* é a ECSS-M-ST-40C (2009), intitulada "*Configuration and Information Management*" (Gerenciamento de Configuração e Informação).

Além disso, a norma ECSS-Q-ST-80C indica que outros requisitos são aplicáveis ao controle de configuração, tais como:

- O sistema de gerenciamento de configuração do *software* deve permitir que qualquer versão possa ser gerada através dos *backups*.
- O arquivo de configuração do *software* e o documento de liberação do *software* devem ser fornecidos em cada entrega de *software*.
- Todos os componentes da ferramenta de geração de códigos que são customizáveis pelo usuário deve estar sob controle de configuração.
- O fornecedor deve identificar métodos e ferramentas para proteger o *software* entregue de ser corrompido.

A norma ECSS-M-ST-40 indica que o gerenciamento de configuração deve ser aplicado através de todo o ciclo de vida do produto e tem como objetivo permitir:

- conhecer, a qualquer tempo, a descrição técnica aprovada do produto;
- registrar e controlar a evolução da descrição técnica do produto;
- prover rastreabilidade da evolução na descrição técnica do produto;
- assegurar a consistência das interfaces internas;
- verificar e demonstrar que a documentação é, e permanece sendo, a imagem exata do produto;

- identificar a *baseline* de configuração atual para registrar discrepâncias detectadas durante a produção, entrega e operação;
- possibilitar o conhecimento das limitações e possibilidades operacionais de cada item do produto.

As atividades previstas no processo de gerenciamento de configuração, conforme a ECSS-M-ST-40, são:

a) Identificação da Configuração: estabelece meios de identificar produtos, documentação e requisitos.

b) Controle de Configuração: estabelece um processo de controle de modificações, registra e controla a configuração do produto em qualquer tempo durante sua evolução.

c) Relatório do Status da Configuração: fornece a definição do produto através da referência a relatórios de configuração aprovados.

d) Auditoria e Verificação da Configuração: verifica e demonstra que o produto atende suas características funcionais, físicas e de performance, bem como verifica que o sistema de controle de configuração é efetivo.

Diferentemente do que foi visto para a RTCA DO-178B, a ECSS-M-ST-40 não estabelece duas categorias para todo o processo de gerenciamento da configuração. Todavia, a norma estabelece duas categorias especificamente para as atividades de identificação da configuração:

a) Item de Configuração Desenvolvido: Item de configuração sendo total ou parcialmente desenvolvido pelo projeto. Nesse caso, o gerenciamento da configuração é responsabilidade do fornecedor.

b) Item de Configuração não Desenvolvido: Item de configuração sendo padronizado ou um produto “de prateleira” (*off-the-shelf*) que não está sendo desenvolvido especificamente para o projeto. Nesse caso, gerenciamento da configuração deve ser adequado para ser integrado com o próximo item de configuração.

Em relação ao controle de modificações, a norma identifica duas classes de mudanças possíveis, conforme definição abaixo:

a) Classe 1: mudança que afeta especificações técnicas aprovadas, incluindo interfaces do mesmo nível e termos associados aos acordos comerciais entre cliente e fornecedor.
b) Classe 2: qualquer mudança diferente dos critérios de classe 1.

Em termos de atividades, as modificações classificadas como classe 1 devem ser necessariamente aprovadas pelo cliente antes de sua implementação. Já as modificações de classe 2 devem ser aprovadas pelo próprio fornecedor antes da implementação, mas devem ser informadas ao cliente.

Outra atividade de controle, além do processo de gerenciamento da configuração, é a chamada garantia da qualidade (ou garantia do produto). Conforme apresentado anteriormente, a norma ECSS-Q-ST-80C é intitulada Garantia do produto de *software* e está fortemente ligada à ECSS-E-ST-40C. A norma ECSS-Q-ST-80C divide as atividades de garantia de produto em três processos principais: implantação de um programa de garantia do produto de *software*, garantia do processo de *software* e garantia da qualidade do produto de *software*.

Assim como a ECSS-E-ST-40C, existe uma diferenciação dos requisitos da norma ECSS-Q-ST-80C em relação à criticalidade do *software*, de acordo com a Tabela 3-10:

Tabela 3-10 – Quantidade de requisitos da ECSS-Q-ST-80C baseado no DAL do *software*.

Nível do *Software*	Quantidade de Requisitos
A	250
B	250
C	246
D	183

Fonte: Adaptada de ECSS-Q-ST-80C (2009).

Nota-se que não há diferença de quantidade de requisitos entre os níveis A e B, o que indica que as atividades de garantia de produto são as mesmas para esses dois níveis.

A norma ECSS-Q-ST-80C indica que as auditorias de *software* devem ser feitas de acordo com o estabelecido pela ECSS-Q-ST-10 (ver Seção 3.1.4).

A atividade similar encontrada na RTCA DO-178B é identificada como processo de garantia da qualidade do *software*. Os objetivos desse processo, conforme explicitado na norma, são:

- garantir que os processos de desenvolvimento do *software* cumprem com os planos e padrões aprovados;
- garantir que os critérios de transição entre os processos do ciclo de vida do *software* foram satisfeitos;
- garantir que uma revisão de conformidade do *software* foi conduzida.

A Tabela A-9 da RTCA DO-178B indica quais são os objetivos de garantia da qualidade que devem ser atendidos por cada nível de *software*, conforme reproduzido na Tabela 3-11 abaixo:

Tabela 3-11 – Objetivos da garantia da qualidade da RTCA DO-178B baseados no DAL do *software*.

Objetivos / Nível de SW	**A**	**B**	**C**	**D**
Garantir que o desenvolvimento cumpre com os planos e padrões aprovados	C	C	C	C
Garantir que os critérios de transição foram satisfeitos	C	C	N/A	N/A
Revisão da conformidade do *software* foi conduzida	C	C	C	C
Nota: C indica com independência e N/A significa não aplicável				

Fonte: Adaptada de RTCA DO-178B (1992).

3.2.5 Documentação Aplicável

Conforme observado nas seções anteriores, existem processos similares entre as normas da indústria aeronáutica (RTCA DO-178B) e da indústria espacial (ECSS-E-ST-40C), mas também existem algumas diferenças de escopo e atividades.

Da mesma forma, a documentação aplicável a cada um dos processos de desenvolvimento de *software* possui similaridades e diferenças. A Tabela 3-12 é um resumo comparativo de alguns dos documentos esperados por cada uma das normas:

Tabela 3-12 – Comparação da documentação aplicável ao desenvolvimento de *software*.

Tipo	RTCA DO-178B	ECSS	
		Documento	**Arquivo**
Certificação	*Plan for Software Aspects of Certification* (PSAC)	N/A	N/A
Plano de desenvolvimento	*Software Development Plan* (SDP)	*Software Development Plan* (SDP)	*Management File* (MGT)
Plano de verificação	*Software Verification Plan* (SVP)	*Software Verification Plan* (SVerP)	*Design Justification* File (DJF)
		Software Integration Test Plan (SUITP)	*Design Justification File* (DJF)
		Software Unit Test Plan (SUITP)	*Design Justification File* (DJF)
		Software Review Plan (SRevP)	*Management File* (MGT)
Plano de validação	N/A	*Software Validation Plan* (SValP)	*Design Justification File* (DJF)
Plano de controle de configuração	*Software Configuration Management Plan* (SCMP)	*Software Configuration Management Plan*	*Management File* (MGT)
Plano de garantia da qualidade	*Software Quality Assurance Plan* (SQAP)	*Software Product Assurance Plan* (SPAP)	*Product Assurance File* (PAF)
Padrão de requisitos	*Software Requirements Standards*	N/A	N/A
Padrão de design	*Software Design Standards*	*Modelling and Design Standards*	*Product Assurance File* (PAF)
Padrão de código	*Software Code Standards*	*Coding Standards*	*Product Assurance File* (PAF)
Requisitos de alto nível	*Software Requirements Data*	*Software Requirements Specification* (SRS)	*Technical Specification* (TS)
Requisitos de baixo nível	*Software Design Description*	*Software Design Document* (SDD)	*Design Definition File* (DDF)
Requisitos de arquitetura	*Software Design Description*	*Software Design Document* (SDD)	*Design Definition File* (DDF)
Requisitos do cliente	N/A	*Software System Specification* (SSS)	*Requirements Baseline* (RB)
		Interface Requirements Document (IRD)	*Requirements Baseline* (RB)

(continua)

Tabela 3-12 – Conclusão.

Tipo	RTCA DO-178B	ECSS	
		Documento	**Arquivo**
Requisitos de interface	N/A	*Interface Control Document* (ICD)	*Technical Specification* (TS)
Procedimentos de verificação	*Software Verification Cases and Procedures*	*Software Validation Specification* (SVS)	*Design Justification File* (DJF)
Resultados de verificação	*Software Verification Results*	*Software Verification Results* (SVR)	*Design Justification File* (DJF)
Índice de configuração do ambiente de desenvolvimento	*Software Life Cycle Environment Configuration Index* (SECI)	N/A	N/A
Índice de configuração do *software*	*Software Configuration Index* (SCI)	*Software Configuration File* (SCF)	*Design Definition File* (DDF)
Relatório de problemas	*Problem Reports*	*Software Problem Reports and Nonconformance Reports*	*Design Justification File* (DJF)
Relatório de controle de configuração	*Software Configuration Management Records*	N/A	N/A
Relatório de garantia da qualidade	*Software Quality Assurance Records*	*Software Product Assurance Milestone Report* (SPAMR)	*Product Assurance File* (PAF)
Resumo da realização do *software*	*Software Accomplishment Summary* (SAS)	N/A	N/A
Documento de liberação	N/A	*Software Release Document* (SRelD)	*Design Definition File* (DDF)
Manual do *software*	N/A	*Software User Manual* (SUM)	*Design Definition File* (DDF)
Nota: N/A significa que não existe explicitamente na norma nenhum documento que contenha a informação.			

Do exposto acima, observa-se que é possível absorver conceitos e práticas de ambas as indústrias para gerar processos e atividades que podem ser utilizadas pelo INPE para o desenvolvimento de sistemas e *softwares*, levando-se em conta variáveis como custo e prazo de execução.

3.3 Exemplos de Documentação Utilizada pelo INPE

De forma complementar à comparação entre as normas, é importante analisar de que forma a documentação e os processos são utilizados pelo INPE. Para tal objetivo foram analisados dois documentos relativos à Garantia de Produto do programa CBERS (*China-Brazil Earth Resources Satellite* – Satélite de Recursos Terrestres Sino-Brasileiro) que foram avaliados na perspectiva de processo de captura e rastreamento de requisitos por Almeida (2011).

O documento C-PAD-003 (1989), intitulado *Basic Requirements for Contractors' Product Assurance Plan for CBERS Spacecraft and Associated Equipment* (Requisitos Básicos para o Plano de Garantia de Produto dos Contratados para o Veículo Espacial CBERS e Equipamentos Associados) indica em seu início que a especificação "relaciona-se com confiabilidade, manutenabilidade, *safety* e garantia da qualidade de sistemas, subsistemas e equipamentos espaciais".

A Seção 8 do documento, intitulada *Software Quality Assurance* (Garantia da Qualidade de *Software*) indica que a documentação mínima para garantir que a implementação do *software* satisfaz os requisitos é a seguinte:

a) *Software Requirements Specification* (Especificação de Requisitos de *Software*);
b) *Software Design Description* (Descrição do Projeto de *Software*);
c) *Software Verification and Validation Plan* (Plano de Verificação e Validação de *Software*)
d) *Software Verification and Validation Report* (Relatório de Verificação e Validação de Software);
e) *Software User's Manual* (Manual do Usuário de Software)

O documento inclui na mesma seção informações relativas a Revisões e Auditorias, Gerenciamento de Configuração e Relatórios de Problemas. Todavia, essas informações não são indicam processos completos para as atividades, que é o escopo deste trabalho.

O segundo documento analisado é o RB-PAD-0002 (2005) intitulado *CBERS 3&4 Product Assurance Requirements* (Requisitos de Garantia de Produto dos CBERS 3&4).

O documento também indica que a especificação “relaciona-se com confiabilidade, manutenabilidade, *safety* e garantia da qualidade de sistemas, subsistemas e equipamentos espaciais”. Além disso, o documento também apresenta uma seção denominada *Software Quality Assurance* (Garantia da Qualidade de *Software*).

Nesta seção, o documento indica que de acordo com o efeito da falha do software, quatro graus podem ser associados ao *software*:

- Grau A: Relativo ao *software* mais crítico. *Software* de controle do satélite é usualmente Grau A.
- Grau B: Relativo a *software* crítico. *Software* de controle do satélite que não é Grau A é usualmente Grau B.
- Grau C: Relativo a *software* importante. *Software* auxiliar e de teste é usualmente Grau C.
- Grau D. Relativo a *software* normal. *Software* de teste em solo, análise e gerenciamento é usualmente Grau D.

Pode-se observar que a definição acima não permite uma avaliação muito clara de como classificar o *software*. Além disso, o documento não apresenta referências de qual a diferença de rigor na Garantia da Qualidade baseado no Grau do *software*.

Do exposto acima, nota-se a importância da proposta de aperfeiçoamento dos processos de desenvolvimento de sistemas e *software* (incluindo Garantia da Qualidade) para aplicação no INPE.

4 UMA PROPOSTA DE APERFEIÇOAMENTO DE UM PROCESSO DE GERENCIAMENTO DE REQUISITOS DE SISTEMA E DE *SOFTWARE*

Como exposto anteriormente, este trabalho propõe-se a estudar as normas de desenvolvimento de sistemas e *software* utilizadas pelas indústrias espacial e aeronáutica de forma a definir processos e atividades que possam ser aplicados ao INPE.

Em relação à utilização da Engenharia de Requisitos no INPE, Almeida (2011) mostra que:

> O INPE tem os pré-requisitos necessários e se adapta facilmente ao uso de uma ferramenta de gerenciamento de requisitos, pois já usa metodologias de engenharia de sistemas e tem uma cultura acostumada com requisitos, documentos, revisões de projeto, controle de configuração, rastreabilidade (ALMEIDA, 2011, p. 181).

Além disso, o autor indica que “é preciso que se criem processos adequados de preparação de requisitos e que o INPE ainda não tem”.

Este trabalho pretende propor atividades relacionadas ao gerenciamento dos requisitos de sistema e de *software* e mostrar um exemplo da aplicação dessas atividades considerando requisitos como os da Plataforma MultiMissão (PMM), desenvolvida pelo INPE.

4.1 Desenvolvimento de Sistemas

Primeiramente, é mister estabelecer qual a natureza das atividades que serão desenvolvidas neste trabalho. Em relação aos requisitos de sistema, a proposta é definir um processo que auxilie na determinação de requisitos, um processo para validação e verificação de requisitos e processos de controle.

Para o processo Validação dos Requisitos de Sistema, a definição de validação será a mesma apresentada pela SAE ARP-4754, uma vez que a norma da indústria espacial (ECSS-E-ST-10C) não apresenta uma preocupação explícita com o tema. Assim, considera-se para o escopo deste trabalho que a validação de requisitos de sistema é uma combinação de processos subjetivos e

objetivos para assegurar que os requisitos especificados estão suficientemente corretos e completos.

Já a Verificação dos Requisitos de Sistema é aplicada de forma semelhante nas duas normas de desenvolvimento de sistemas estudadas, Assim, para efeito didático, a Verificação de Requisitos de Sistema é entendida como o processo para garantir que a implementação satisfaz os requisitos validados.

Em relação às atividades de controle, serão consideradas as atividades com escopo técnico: garantia do produto e controle de configuração. Dessa forma, os objetivos dessas atividades são aqueles apresentados na norma ECSS-E-ST-10C. Assim, para fins deste trabalho, o objetivo da garantia de produto é assegurar que os produtos espaciais realizem seus objetivos de missão definidos de forma segura, disponível e confiável.

Da mesma forma, neste trabalho, o controle de configuração é o processo para estabelecer e manter o registro das características físicas e funcionais de um produto comparado com seus requisitos de projeto e operacionais.

4.1.1 Elicitação e Validação de Requisitos de Sistema

Considerando as definições apresentadas neste trabalho, os processos de elicitação e de validação de requisitos de sistema devem ser feitos de forma conjunta. As informações pertinentes à elicitação dos requisitos, tais como rastreio para requisitos de cliente, justificativas (*rationales*) e premissas (*assumptions*) servem como base para a atividade de validação de requisitos.

Primeiramente, os requisitos de sistema devem ser inseridos em um ambiente que permita o controle adequado de *baseline* e modificações (ver Seção 4.1.3). Para permitir o processo de validação, a base de requisitos deve conter atributos para cada um dos requisitos relacionados com essa atividade.

Dessa forma, os atributos de requisitos de sistema propostos por este trabalho são:

1- Identificador: de forma a identificar de maneira não ambígua cada um dos requisitos de sistema, o requisito deverá possuir um identificador único.
2- Texto do Requisito: é a descrição do requisito, que é passada para os próximos níveis de desenvolvimento.
3- Autor: indica de forma clara quem são os autores do requisito (e, posteriormente, quem fez modificações nesse requisito).
4- DAL: indica qual o nível de criticalidade que a função demandada pelo requisito tem no sistema. Baseado no DAL do sistema, algumas atividades de validação e independências poderão ser minimizadas.
5- Premissas: caso alguma premissa (*assumption*) tenha sido utilizada na elaboração do requisito ela deverá ser claramente identificada.
6- Rastreio: todo requisito que foi elaborado baseado em um requisito de nível superior (e.g. requisito de cliente) deverá possuir a identificação clara do rastreio.
7- Justificativa: todo requisito que não possuir um requisito de nível superior (considerado requisito derivado) deverá apresentar a justificativa (*rationale*) de porque esse requisito é necessário.
8- Método de Validação: conforme apresentado na Seção 3.1.3, existem diferentes métodos que auxiliam no processo de validação, pois demonstram a correção (*correctness*) do requisito. Todos os métodos que forem necessários para permitir a validação deverão ser apresentados.
9- Artefato de Suporte ao Método de Validação: para cada um dos métodos de validação apresentados, deverá se evidenciar o artefato que auxilia na avaliação da correção do requisito.
10- Resultado da Validação: após avaliar todos os atributos preenchidos pelo autor do requisito, o validador irá considerar o requisito como válido ou não válido.
11- *Checklist* de Validação: para avaliar a correção e completude dos requisitos e preencher o atributo de resultado da validação, o validador deverá preencher um *checklist* que serve de guia para a avaliação. Caso alguma das perguntas não seja aprovada, a justificativa de reprovação deve ser explicitada, para que o autor possa fazer as correções.

12- Validador: indica de forma clara quem foi responsável pela atividade de validação do requisito.

Como pode ser visto acima, existe uma forte dependência da utilização de *checklists* para o processo de validação. Um dos objetivos deste trabalho é definir um *checklist* para essa atividade que permita avaliar de forma clara a correção e completude dos requisitos, mas que não seja muito oneroso para o desenvolvimento de sistemas.

Em relação à diferenciação das atividades baseadas no DAL do requisito, este trabalho traz a seguinte proposta:

Tabela 4-1 – Proposta de métodos de validação baseados no DAL dos requisitos.

Método	DAL A e B	DAL C	DAL D	DAL E
Rastreabilidade	Ao menos um método com independência	Ao menos um método	Ao menos um método	Não aplicável
Análise, Modelos ou Teste				
Similaridade				
Julgamento de Engenharia				

Assim, a proposta para a validação de requisitos é garantir que a correção possa ser avaliada para todos os requisitos que afetam a disponibilidade do sistema, independente do DAL associado. Dessa forma, não existe a obrigatoriedade de um ou mais métodos para os requisitos mais críticos, enquanto que os requisitos menos críticos não são negligenciados.

Todavia, se apenas um método não for suficiente para que o validador possa avaliar a correção do requisito, outros métodos podem ser utilizados. Por exemplo, quando um requisito de sistema possui rastreio para um requisito de cliente, a justificativa (*rationale*) não é necessária. Em contrapartida, o requisito de sistema pode incluir informações adicionais ao requisito do cliente, que deverão ser avaliadas durante o processo de validação. Nesse caso, o autor do requisito poderá utilizar a rastreabilidade como um dos métodos (cujo artefato é o próprio rastreio para o requisito do cliente), mas também deverá incluir mais evidências para mostrar a correção do requisito (por exemplo, pode expor um julgamento de engenharia ou fazer uma análise).

A diferença que existe em relação aos requisitos de mais alta criticalidade (que possuem DAL A ou B) é a necessidade de independência no processo de validação. Nesse caso, busca-se uma avaliação mais criteriosa dos requisitos que tenham maior impacto na confiabilidade e na segurança do sistema. Assim, os campos “Autor” e “Validador” não poderão conter os mesmos nomes para esses requisitos.

Com esta proposta, entende-se que os objetivos da validação de requisitos são plenamente atendidos, pois este processo permite identificar erros na concepção dos requisitos do sistema antes de passar a *baseline* para o desenvolvimento do próximo nível, seja ele subsistema, *software* ou *hardware* de uma forma clara e possui carga de trabalho moderada.

Levando em conta essas premissas, foi elaborado um checklist (Tabela 4-2) no qual todas as perguntas devem ser respondidas. Quando a resposta for “Sim” não é necessário incluir nenhum comentário no campo “Justificativa”. Para respostas “N/A” (Não Aplicável), o preenchimento do campo “Justificativa” é desejável, pois possibilita o entendimento do porque a pergunta é não aplicável para determinado requisito. Por fim, para respostas “Não”, o preenchimento do campo “Justificativa” é obrigatório, pois servirá como base para que o autor do requisito faça as correções necessárias e o validador possa reavaliar os requisitos, até que o *checklist* seja preenchido sem nenhuma resposta “Não”.

O validador deve ser identificado no *checklist* para garantir consistência entre a evidência do processo de validação e o campo “Validador”. Além disso, o requisito deve estar claramente identificado no início, utilizando a informação do campo “Identificador” do requisito.

Tabela 4-2 – Proposta de checklist para validação de requisitos de sistema.

Checklist para validação de requisitos de sistema			**Revisão A**
Identificador do Requisito:			
Validador:		**Data: ___/___/____**	
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para uma fonte?	☐ Sim ☐ Não ☐ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	☐ Sim ☐ Não ☐ N/A	
3	A descrição do requisito indica apenas uma necessidade?	☐ Sim ☐ Não ☐ N/A	
4	Se o requisito não rastreia para um requisito de nível superior (i.e. requisito derivado), existe um *rationale* para justificar o requisito?	☐ Sim ☐ Não ☐ N/A	
5	O requisito é verificável?	☐ Sim ☐ Não ☐ N/A	
6	O requisito descreve as capacidades desejadas de forma positiva?	☐ Sim ☐ Não ☐ N/A	
7	O requisito apresenta as tolerâncias e faixas de operação, quando aplicável?	☐ Sim ☐ Não ☐ N/A	
8	Os artefatos de suporte são consistentes com os métodos de validação identificados?	☐ Sim ☐ Não ☐ N/A	
9	Considerando os artefatos de suporte, o requisito está correto?	☐ Sim ☐ Não ☐ N/A	
10	O requisito está consistente com os demais requisitos relacionados à mesma função?	☐ Sim ☐ Não ☐ N/A	

Cabe ressaltar que cabe ao validador decidir se o *checklist* será utilizado para apenas um requisito ou se será utilizado para validar um conjunto de requisitos relacionados. No caso da atividade ser realizada para mais de um requisito, todos devem estar claramente identificados no *checklist*, que deverá ser anexado apenas uma vez como evidência de validação, sendo referenciado pelos demais requisitos.

Assim, para todo requisito será gerada uma evidência do processo de validação, de forma a garantir a possibilidade de reuso de requisitos, justificativas (*rationales*), premissas (*assumptions*) e evidências de validação para futuros projetos de mesma natureza.

Por se tratar de uma atividade complexa, o processo de validação de requisitos está sujeito a erros durante a sua execução, uma vez que ele será executado por diferentes membros de uma equipe técnica, com diferentes níveis de experiência e instrução. Para garantir que esse processo aconteça da forma mais correta e completa, é necessário o processo de Garantia do Produto (ver Seção 4.1.4).

Preferencialmente, a transição da *baseline* de requisitos de sistema para os próximos níveis de desenvolvimento deve acontecer após a validação de requisitos, para evitar desperdícios e retrabalho, uma vez que durante as atividades de validação pode ocorrer a adição, remoção ou alteração de requisitos.

4.1.2 Verificação de Requisitos de Sistema

Como visto anteriormente, o processo de verificação de requisitos para as normas aeronáutica (SAE ARP-4754) e espacial (ECSS-E-ST-10C) apresentam muitas semelhanças. Assim, o processo proposto por este trabalho é uma tentativa de adequar as duas realidades para a prática do INPE.

Assim como o processo de validação, as atividades de verificação devem acontecer em ambientes com controle de configuração e as evidências dessas atividades devem ficar armazenadas. Para garantir essas condições, devem ser acrescentados aos atributos de elicitação e validação dos requisitos, apresentados na seção anterior, as informações referentes à verificação desses requisitos. Dessa forma, os seguintes campos devem ser adicionados:

1- Método de Verificação: deve-se indicar qual será o método de verificação utilizado.
2- Evidência de Verificação: a evidência irá depender do método utilizado. Por exemplo, no caso de verificação por teste o procedimento de testes e o relatório de resultados devem ser indicados.
3- Resultado da Verificação: após a atividade de verificação ser feita e as evidências geradas, o verificador irá considerar o requisito como verificado ou não verificado.
4- Verificador: indica de forma clara quem foi responsável pela atividade de verificação do requisito.

Em relação aos métodos apropriados para verificação de requisitos e a diferenciação baseada no DAL dos requisitos, este trabalho traz a seguinte proposta mostrada na Tabela 4-3:

Tabela 4-3 – Proposta de métodos de verificação baseados no DAL dos requisitos.

Método	DAL A e B	DAL C	DAL D	DAL E
Teste	M	M (ao menos um)	M (ao menos um)	Não aplicável
Inspeção, Revisão, Análise ou Experiência em serviço	R			
Nota: M indica mandatório e R indica recomendável				

Assim, para requisitos com DAL A ou B, a verificação deve acontecer obrigatoriamente através de testes. Além disso, recomenda-se que mais de um método seja utilizado. Caso um determinado requisito com DAL A ou B não possa ser verificado por teste, é permitido utilizar outro método.

Para os requisitos de DAL C ou D a verificação é obrigatória, mas o método pode ser escolhido de forma a minimizar o custo com testes. Assim, para esses requisitos é aceitável que a verificação seja feita apenas por Inspeção, Revisão, Análise ou Experiência em Serviço.

Conforme apresentado anteriormente, as evidências para a verificação por testes são os procedimentos utilizados e os resultados obtidos. Toda essa informação deve ser mantida em um ambiente controlado e as revisões devem ser claramente identificadas. Em relação aos demais métodos é esperado que a inspeção e a análise sejam feitas através de relatórios técnicos; a revisão deva ser contra documentação técnica dos níveis mais baixos de implementação (e.g. *software*) e para a experiência em serviço utilize documentação aprovada em outros projetos.

Considerando que a organização ou equipe responsável pelos requisitos de sistema é usualmente diferente da organização ou equipe responsável pela implementação desses requisitos em componentes de mais baixo nível (*software* e *hardware*), a independência no processo de verificação tende a ser atingida de forma bastante clara. Todavia, caso a mesma equipe seja responsável pela engenharia de requisitos de sistema e pela implementação propriamente dita, deve-se estabelecer independência entre quem fez a implementação e quem fez a verificação.

Assim como na validação, o processo de verificação é complexo e está sujeito a falhas de execução. O processo de Garantia do Produto (Seção 4.1.4) busca garantir a correta aplicação das atividades descritas, incluindo independência quando necessário.

4.1.3 Controle de Configuração

Como visto nas seções anteriores, os processos de Elicitação, Validação e Verificação de Requisitos de Sistema devem ocorrer em um ambiente controlado tal que, entre outros atributos, dificulte alterações não autorizadas e

permita o arquivamento e a recuperação de todos os dados relativos a esses processos.

Dessa forma, é importante que o ambiente escolhido para o desenvolvimento dos requisitos de sistema permita o correto gerenciamento das atividades. Por exemplo, para garantir a identificação única para cada requisito, esse ambiente deve ser capaz de gerar numeração automática para os requisitos. Além disso, deve permitir a geração de *baselines* que, uma vez geradas, não podem ser modificadas.

Outra atividade de grande importância para o desenvolvimento de sistemas é o controle de modificações, que podem ser originadas, entre outros motivos, por problemas encontrados durante a validação, verificação, produção ou operação; por mudança nos requisitos dos clientes e surgimento de novas tecnologias.

Usualmente, o ambiente de controle de modificações é diferente do ambiente de desenvolvimento de requisitos, por serem atividades de natureza distinta. Todavia, é primordial manter o rastreio entre os dois ambientes para garantir que as modificações aprovadas sejam efetivamente implementadas.

A proposta deste trabalho é apresentar quais são as informações obrigatórias que devem constar em cada proposta de modificação, independente da ferramenta escolhida para esse processo. Assim, a ferramenta de controle de modificações deve conter:

1- Identificador: de forma a identificar de maneira não ambígua cada uma das análises de modificação; a proposta deve possuir um identificador único.
2- Descrição da Modificação: a descrição deve conter o motivo da modificação (e.g. problema na verificação, novo requisito do cliente, falha durante operação) e, preferencialmente, apresentar o texto final do requisito a ser modificado.

3- *Baseline* Afetada: para garantir rastreio quando existe mais de uma modificação no mesmo requisito durante o ciclo de vida de desenvolvimento.
4- Requisitos Afetados: todos os requisitos afetados pela proposta de modificação devem ser listados, a fim de garantir a análise completa do impacto da modificação proposta.
5- Disposição da Análise: toda proposta de modificação deve ser discutida por um corpo técnico habilitado para decidir se a mudança deve ou não ser implementada (e, eventualmente, quando deve ser implementada). Todos os pontos discutidos por esse grupo devem ser incluídos nesse campo, para consulta posterior.
6- Áreas Afetadas: a modificação proposta pode ter impacto em diferentes áreas do desenvolvimento, como subsistemas, áreas de interface, *software* e *hardware*. Esse campo deve conter todas as áreas afetadas e representantes dessas áreas devem fazer parte da análise.
7- Próxima Ação: após a abertura da proposta de modificação, diferente ações podem ser tomadas como análise, investigação e implementação. Esse campo deve conter o estado atual da proposta.
8- Responsável pela Próxima Ação: assim como diversas ações podem ser tomadas em cada proposta, diferentes responsáveis podem ser designados para essas ações.
9- Conclusão: ao final do processo de análise da modificação, a proposta pode ter sido rejeitada, aprovada e implementada, aprovada e programada para implementação, entre outros vereditos.

Da mesma forma que os requisitos afetados pela modificação devem constar na ferramenta de controle de modificações, o identificador da proposta de mudança que efetivamente alterou um requisito deve ser incluído como atributo desse requisito. Assim, além dos atributos referentes à elicitação, validação e verificação incluídos no ambiente de gerenciamento dos requisitos de sistemas, é necessário incluir um atributo com esse identificador, denominado “Modificado por:“.

Por fim, é preciso garantir o correto arquivamento de todos os artefatos de desenvolvimento de sistema, incluindo as *baselines* de requisitos, os procedimentos e resultados de teste, relatórios de análise e as análises de modificações.

A organização deve garantir que a mídia utilizada minimize erros de corrupção de arquivos e que seja protegida contra alterações indesejadas. Idealmente, devem ser mantidas cópias de todos esses arquivos em um espaço segregado, para evitar que um problema não contido impossibilite a recuperação completa dos dados.

4.1.4 Garantia do Produto

As seções anteriores explicitam a grande quantidade de atividades que permeiam o desenvolvimento de sistemas, principalmente quando se trata de sistemas complexo ou altamente integrados, como a Plataforma MultiMissão (PMM). Como o objetivo deste trabalho é propor processos que auxiliem na melhora da disponibilidade e confiabilidade dos sistemas, a garantia do bom andamento das atividades de determinação, validação e verificação de requisitos é condição *sine qua non* para atingir esse objetivo. Da mesma forma, o processo de controle de modificações deve passar por revisões que garantam o correto uso das atividades planejadas.

Para as atividades de garantia de produto, a proposta do trabalho é a realização de auditorias para avaliação do andamento das atividades durante o processo de desenvolvimento. Essas auditorias devem ser feitas por um time com conhecimento dos processos sendo utilizados e que não executa nenhuma das atividades sendo auditadas. Considerando que todos os processos são complexos e que cada um deles possui características distintas, um *checklist* é proposto para cada processo, a fim de servir como guia para a auditoria e permitir o registro de problemas e suas resoluções.

Primeiramente, deve-se discutir qual a periodicidade indicada para cada uma das auditorias. Para decidir essa periodicidade deve ser levado em conta que

um problema de execução dos processos encontrados prematuramente é facilmente gerenciado, mas também que até o último momento erros podem ser cometidos.

Assim, a proposta para a periodicidade das auditorias proposta por este trabalho é:

a) Elicitação e Validação de Requisitos de Sistema: a primeira auditoria deve acontecer quando o total de requisitos validados atingir 30%, pois garante que problemas de experiência e treinamento do time sejam observados no início do processo. A segunda auditoria deve acontecer com 70% dos requisitos validados, pois já existe quantidade de requisitos validados suficiente para uma análise abrangente e permite que problemas encontrados possam ser resolvidos antes do *baseline* ser enviado para os próximos níveis de desenvolvimento. A partir do primeiro envio de *baseline* de requisitos validados, as auditorias devem acontecer sempre que houver modificações significativas nos requisitos (por exemplo, inclusão de uma nova função).

b) Verificação de Requisitos de Sistema: Os requisitos de sistema dificilmente conseguem ser verificados antes de haver uma implementação no próximo nível, como *hardware* ou *software*. Após iniciado o processo de verificação, a primeira auditoria deve acontecer quando o total de requisitos verificados atingir 30%, para identificar problemas no início das atividades. A segunda auditoria deve acontecer com 70% dos requisitos verificados. A partir de então, as auditorias devem acontecer sempre que houver modificações significativas nos requisitos (por exemplo, inclusão de uma nova função).

c) Controle de Modificações: Considerando que o controle de modificações se iniciará em algum momento entre o início da determinação de requisitos e a validação completa dos requisitos, a primeira auditoria deve acontecer quando 10% dos requisitos forem modificados formalmente. A partir de então, cada vez que o total de requisitos modificados sofrer um acréscimo de 15% uma nova auditoria deve ser conduzida.

Além da periodicidade, é necessário definir o escopo de cada auditoria. Por questões de custo e prazo, não é esperado que as auditorias avaliem todos os requisitos do *baseline*, nem todas as propostas de modificação. Usualmente, são escolhidas amostras aleatórias representativas para serem auditadas de forma que, se for encontrado um problema sistêmico na execução do processo, ações de correção sejam efetuadas.

Assim, o escopo das auditorias proposto por este trabalho, para cada um dos processos é:

a) Elicitação e Validação de Requisitos de Sistema: Na primeira auditoria, devem ser selecionados 20% dos requisitos já validados. Na segunda auditoria, 10% dos requisitos validados posteriormente à primeira auditoria devem ser avaliados. A partir da geração do *baseline* validado, 10% dos requisitos modificados deverão ser avaliados.

b) Verificação de Requisitos de Sistema: Na primeira auditoria, devem ser selecionados 20% dos requisitos já verificados. Na segunda auditoria, 10% dos requisitos verificados posteriormente à primeira auditoria devem ser avaliados. As auditorias seguintes devem avaliar 10% dos requisitos modificados após cada auditoria anterior.

c) Controle de Modificações: Por se tratar de um processo que envolve diversas áreas de uma organização, a primeira auditoria deve analisar 30% das propostas de modificação que já foram concluídas. A partir de então, 20% das propostas de modificações posteriores à auditoria anterior devem ser avaliadas.

Considerando o exposto acima, existe a necessidade de três *checklists* para o processo de garantia de produto no desenvolvimento de sistemas. Para manter o histórico e possibilitar o rastreio das ações, esses *checklists* devem conter explicitamente quais foram os requisitos (ou propostas de modificação) avaliados e quais foram os problemas encontrados. Além disso, devem ser armazenados em ambiente controlado com acesso restrito aos times envolvidos nas auditorias.

A proposta deste trabalho para os três *checklists* supracitados pode ser vista a seguir, onde o campo “Justificativa” deve conter a identificação do requisito ou da proposta de modificação onde foi encontrado problema e qual foi esse problema:

Tabela 4-4 – Proposta de checklist de auditoria para processos de elicitação e validação de requisitos de sistema.

Checklist de Auditoria – Determinação e validação de requisitos			**Rev A**
Auditoria nº:		**Baseline:**	
Requisitos avaliados:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os requisitos possuem identificação do autor?	☐ Sim ☐ Não ☐ N/A	
2	Os requisitos possuem DAL associado?	☐ Sim ☐ Não ☐ N/A	
3	Os requisitos possuem rastreio para requisito de nível superior ou justificativa associada?	☐ Sim ☐ Não ☐ N/A	
4	Os requisitos possuem identificação do validador?	☐ Sim ☐ Não ☐ N/A	
5	Os requisitos de DAL A ou B possuem validador diferente do autor?	☐ Sim ☐ Não ☐ N/A	
6	Os requisitos possuem método de validação associado?	☐ Sim ☐ Não ☐ N/A	
7	Para todos os métodos de validação indicados existe um artefato de suporte associado?	☐ Sim ☐ Não ☐ N/A	
8	Os requisitos possuem checklist de validação preenchido?	☐ Sim ☐ Não ☐ N/A	
9	Todas as perguntas dos checklists estão corretamente preenchidas?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-5 – Proposta de checklist para auditoria do processo de controle de modificações.

Checklist de Auditoria – Controle de modificações			**Revisão A**
Auditoria nº:			
Propostas de modificação avaliadas:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	As propostas possuem descrição clara da modificação?	☐ Sim ☐ Não ☐ N/A	
2	As propostas indicam qual a baseline afetada?	☐ Sim ☐ Não ☐ N/A	
3	As propostas indicam quais os requisitos afetados?	☐ Sim ☐ Não ☐ N/A	
4	As propostas indicam quais as áreas afetadas?	☐ Sim ☐ Não ☐ N/A	
5	As disposições evidenciam que as áreas afetadas participaram das análises?	☐ Sim ☐ Não ☐ N/A	
6	Os requisitos com modificações aceitas foram alterados?	☐ Sim ☐ Não ☐ N/A	
7	Os requisitos alterados apresentam o identificador da proposta no campo “Modificado por:“	☐ Sim ☐ Não ☐ N/A	
8	A proposta apresenta a conclusão da modificação?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-6 – Proposta de checklist para auditoria do processo de verificação de requisitos de sistema.

Checklist de Auditoria – Verificação de requisitos			**Revisão A**
Auditoria nº:		**Baseline:**	
Requisitos avaliados:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os requisitos possuem ao menos um método de verificação associado?	☐ Sim ☐ Não ☐ N/A	
2	Para todos os métodos de validação indicados existe uma evidência de verificação associada?	☐ Sim ☐ Não ☐ N/A	
3	Os requisitos possuem identificação do verificador?	☐ Sim ☐ Não ☐ N/A	
4	Os requisitos de DAL A ou B incluem testes como método de verificação?	☐ Sim ☐ Não ☐ N/A	

4.2 Desenvolvimento de *Software*

Assim como apresentado para o desenvolvimento de sistemas, é essencial estabelecer quais atividades de desenvolvimento de *software* são escopo deste trabalho.

Como visto nas seções anteriores, a norma ECSS-E-ST-40C e outras normas de desenvolvimento de *software* indicam a necessidade de um processo de verificação, um processo de validação e processos de controle. Todavia, a definição de validação usada pela ECSS-E-ST-40C é bastante diferente da definição utilizada neste trabalho para a validação de requisitos de sistema, pois considera a validação de todo o sistema versus as necessidades.

As atividades previstas na norma que se assemelham ao processo de validação de requisitos apresentado no desenvolvimento de sistemas são a verificação da baseline de requisitos, a verificação da especificação técnica, a verificação da arquitetura de *software* e a verificação do projeto detalhado de *software*.

Assim, no contexto de desenvolvimento de *software*, o processo de validação deste trabalho está ligado à validação do conjunto de requisitos e não à avaliação de correção e completude dos requisitos de *software*. Neste trabalho, essa importante atividade que possibilita a observação de erros nos requisitos antes da implementação será incluída como parte dos processos de verificação.

Do exposto acima, o objetivo das atividades de validação é confirmar que as funções e performances da baseline de requisitos estão corretamente e completamente implementadas no produto final. Já as atividades de verificação têm objetivos específicos dependendo de qual nível está sendo verificado, mas de forma geral pode-se dizer que é uma avaliação técnica dos processos de desenvolvimento de *software*.

Por fim, existem as atividades de controle que são semelhantes às atividades de controle para desenvolvimento de sistemas: controle de configuração e garantia do produto. Para o desenvolvimento de *software* este trabalho considera que o objetivo do controle de configuração é prover configuração definida e controlada do *software* através de todo o ciclo de vida, enquanto que o objetivo da garantia de produto é garantir que os processos de desenvolvimento do *software* cumprem com os planos e padrões aprovados.

4.2.1 Elicitação de Requisitos

A Seção 4.2.2, referente às atividades de verificação, identifica quais os níveis de requisito são considerados no desenvolvimento de *software*. Independente do nível do requisito, o processo de elicitação e captura deve ocorrer em um ambiente que permita o controle adequado de *baseline* e modificações.

4.2.2 Atividades de Verificação

Como visto na seção anterior, as atividades de verificação possuem características distintas para cada nível do desenvolvimento de software. Assim, este trabalho traz a seguinte divisão de atividades baseada nesta diferenciação:

- Verificação dos requisitos de alto nível;
- Verificação dos requisitos de baixo nível e da arquitetura do *software*;
- Verificação do código;
- Verificação do processo de integração; e
- Verificação dos testes de *software*.

4.2.2.1 Verificação dos Requisitos de Alto Nível

Considerando a documentação prevista na norma ECSS-E-ST-40C, a proposta deste trabalho é considerar que a *baseline* de requisitos são os requisitos de sistema que servem como entrada para o desenvolvimento de *software*. O processo para garantir a correção desses requisitos é discutido na seção de

desenvolvimento de sistemas. Dessa forma, a documentação que inclui os requisitos de alto nível de *software* é a denominada especificação técnica (*Technical Specification*).

Para garantir que a gestão dos requisitos seja adequada, deve-se utilizar um ambiente de desenvolvimento semelhante ao apresentado no desenvolvimento de sistemas (idealmente, deve-se utilizar o mesmo ambiente para facilitar a integração e o rastreio das bases de requisitos).

A fim de manter a coerência entre as atividades de desenvolvimento de sistemas e de *software*, a proposta deste trabalho é considerar atributos para os requisitos de alto nível de *software* de forma semelhante ao apresentado na Seção 4.1. Dessa forma, os requisitos de alto nível de *software* devem conter os seguintes atributos:

1- Identificador: de forma a identificar de maneira não ambígua cada um dos requisitos de alto nível de *software*; o requisito deve possuir um identificador único.
2- Texto do Requisito: é a descrição do requisito, que é passada para os próximos níveis de desenvolvimento.
3- Autor: indica de forma clara quem são os autores do requisito (e, posteriormente, quem fez modificações nesse requisito).
4- DAL: indica qual o nível de criticalidade que a função exercida pelo requisito tem no *software*. Baseado no DAL do sistema, algumas atividades de verificação e independências podem ser minimizadas.
5- Premissas: caso alguma premissa (*assumption*) tenha sido utilizada na elaboração do requisito ela deve ser claramente identificada.
6- Rastreio: todo requisito que foi elaborado baseado em um requisito de nível superior (e.g. requisito de sistemas) deve possuir a identificação clara do rastreio.
7- Justificativa: todo requisito que não possuir um requisito de nível superior (considerado requisito derivado) deve apresentar a justificativa (*rationale*) de porque esse requisito é necessário.

8- Resultado da Verificação: após avaliar todos os atributos preenchidos pelo autor do requisito, o verificador irá considerar o requisito como verificado ou não verificado.

9- *Checklist* de Verificação: para fazer a verificação, o verificador deverá preencher um *checklist* que serve de guia para a avaliação. Caso alguma das perguntas não seja aprovada, a justificativa de reprovação deve ser explicitada, para que o autor possa fazer as correções.

10- Verificador: indica de forma clara quem foi responsável pela atividade de verificação do requisito de alto nível de *software*.

Considerando os objetivos apresentados na ECSS-E-ST-40C e na RTCA DO-178B, o verificador deve analisar os seguintes aspectos:

- consistência entre os requisitos de alto nível de *software*;
- rastreio entre os requisitos de sistema e de alto nível de *software*;
- possibilidade de testes dos requisitos;
- compatibilidade com o *hardware*; e
- conformidade com os padrões de escrita de requisitos de alto nível de *software*.

As atividades de verificação podem ser diferenciadas de acordo com o DAL do *software*, de forma semelhante ao apresentado no desenvolvimento de sistemas, bem como a independência requerida. Em relação à independência das atividades de verificação, a norma ECSS-E-ST-40C indica que o nível de independência deve ser incluído nos planos, mas não apresenta um guia claro do que é esperado. Assim, a independência para a verificação dos requisitos de alto nível de *software* proposta neste trabalho é uma simplificação da RTCA DO-178B, baseada no DAL do *software*.

A RTCA DO-178B permite que algumas atividades relativas ao processo de verificação de requisitos de alto nível não sejam feitas para *softwares* menos críticos. Já a ECSS-E-ST-40C indica apenas quais documentos podem deixar de ser submetidos de acordo com a criticalidade do *software*. Levando em

conta o exposto acima, a proposta deste trabalho para o processo de verificação de requisitos de alto nível de *software* é a da Tabela 4-7 seguinte:

Tabela 4-7 – Proposta para objetivos de verificação dos requisitos de alto nível de *software*.

Verificação (requisitos de alto nível)	DAL A e B	DAL C	DAL D
Consistência entre os requisitos	C	S	S
Rastreio para os requisitos de sistema	C	S	S
Conformidade com os padrões de requisitos	C	S	S
Compatibilidade com o *hardware*	C	S	S
Possibilidade de testes dos requisitos	C	S	N/A
Nota: C indica com independência e S indica sem independência			

Da tabela acima, observa-se que a verificação dos requisitos para *software* com DAL A ou B deve ser feita com independência. Além disso, os requisitos de alto nível de *software* com DAL D não precisam ser testados.

Por consistência, este trabalho propõe a utilização de *checklists* para as atividades de desenvolvimento de *software*, como feito com o processo de desenvolvimento de sistemas.

Assim como no *checklist* de validação de requisitos de sistema, quando a resposta for “Sim” não é necessário incluir comentário algum no campo “Justificativa”. Para respostas “N/A” (Não Aplicável), o preenchimento do campo “Justificativa” é desejável, pois possibilita o entendimento do porque a pergunta é não aplicável para determinado requisito. Por fim, para respostas “Não”, o preenchimento do campo “Justificativa” é obrigatório, pois servirá como base para que o autor do requisito faça as correções necessárias e o verificador possa reavaliar os requisitos, até que o resultado seja o *checklist* preenchido sem nenhuma resposta “Não”.

Especificamente para o processo de verificação de requisitos de alto nível de *software*, o *checklist* proposto é o da Tabela 4-8 seguinte:

Tabela 4-8 – Proposta para checklist de verificação dos requisitos de alto nível de *software*.

Checklist para verificação de requisitos de alto nível de *software*			**Rev A**
Identificador do Requisito:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para um requisito de sistema ou possui uma justificativa?	☐ Sim ☐ Não ☐ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	☐ Sim ☐ Não ☐ N/A	
3	O requisito é verificável?	☐ Sim ☐ Não ☐ N/A	
4	A descrição do requisito é compatível com o *hardware*?	☐ Sim ☐ Não ☐ N/A	
5	O requisito segue o padrão estabelecido para escrita de requisitos?	☐ Sim ☐ Não ☐ N/A	
6	O algoritmo utilizado é preciso?	☐ Sim ☐ Não ☐ N/A	

4.2.2.2 Verificação dos Requisitos de Baixo Nível e da Arquitetura de *Software*

Considerando a terminologia da ECSS-E-ST-40C e da RTCA DO-178B, os requisitos de baixo nível de *software* são usualmente denominados projeto de software ou projeto de *software* detalhado (*software detailed design*). A natureza dos requisitos de baixo nível é semelhante aos requisitos de alto nível, mas apresentam um maior detalhamento e possibilitam, com o auxílio da arquitetura de *software*, a codificação do *software*.

Assim, é possível inserir os requisitos de baixo nível no mesmo ambiente de controle utilizado pelos requisitos de alto nível, garantindo consistência e

rastreio entre as duas bases. De forma semelhante, os atributos associados aos requisitos de baixo nível são:

1- Identificador: de forma a identificar de maneira não ambígua cada um dos requisitos de baixo nível de *software;* o requisito deve possuir um identificador único.
2- Texto do Requisito: é a descrição do requisito, que deve permitir a codificação do *software*.
3- Autor: indica de forma clara quem são os autores do requisito (e, posteriormente, quem fez modificações nesse requisito).
4- DAL: indica qual o nível de criticalidade que a função exercida pelo requisito tem no *software*. Baseado no DAL do sistema, algumas atividades de verificação e independências podem ser minimizadas.
5- Premissas: caso alguma premissa (*assumption*) tenha sido utilizada na elaboração do requisito ela deve ser claramente identificada.
6- Rastreio: todo requisito que foi elaborado baseado em um requisito de nível alto nível de *software* deve possuir a identificação clara do rastreio.
7- Justificativa: todo requisito que não rastreia para um requisito de alto nível deve apresentar a justificativa (*rationale*) de porque esse requisito é necessário.
8- Resultado da Verificação: após avaliar todos os atributos preenchidos pelo autor do requisito, o verificador irá considerar o requisito como verificado ou não verificado.
9- *Checklist* de Verificação: para fazer a verificação, o verificador deverá preencher um *checklist* que serve de guia para a avaliação. Caso alguma das perguntas não seja aprovada, a justificativa de reprovação deve ser explicitada, para que o autor possa fazer as correções.
10- Verificador: indica de forma clara quem foi responsável pela atividade de verificação do requisito de baixo nível de *software*.

Além dos requisitos de baixo nível, a camada entre os requisitos de alto nível e a codificação do *software* contém a arquitetura do software (a denominação na ECSS-E-ST-40C é projeto arquitetural de *software* – *software architectural design*). Conforme apresentado anteriormente, a arquitetura do *software* deve rastrear para os requisitos de alto nível de *software* e deve indicar as entradas,

saídas, interfaces, fluxo lógico, alocação de tempo e espaço e tratamento de erros.

Por sua natureza, não é esperado que a arquitetura de *software* receba a mesma tratativa dos requisitos. Todavia, a atividade de verificação da arquitetura também deve acontecer.

Assim como os requisitos de alto nível, as atividades de verificação da arquitetura de *software* e dos requisitos de baixo nível e a independência dessas atividades são baseadas no DAL do *software* sendo desenvolvido, conforme as Tabelas 4.9 e 4.10 abaixo:

Tabela 4-9 – Proposta para objetivos de verificação dos requisitos de baixo nível de *software.*

Verificação (requisitos de baixo nível)	DAL A e B	DAL C	DAL D
Consistência entre os requisitos	C	S	S
Rastreio para os requisitos de alto nível	C	S	N/A
Conformidade com os padrões de requisitos	C	S	N/A
Compatibilidade com o *hardware*	C	S	N/A
Possibilidade de testes dos requisitos	C	S	N/A
Nota: C indica com independência e S indica sem independência			

Tabela 4-10 – Proposta para objetivos de verificação da arquitetura do *software.*

Verificação (arquitetura de *software*)	DAL A e B	DAL C	DAL D
Consistência da arquitetura	C	S	S
Conformidade com os padrões de arquitetura	C	S	S
Rastreio para os requisitos de alto nível	C	S	S
Compatibilidade com o *hardware*	C	S	S
Possibilidade de testes da arquitetura	C	S	N/A
Nota: C indica com independência e S indica sem independência			

Seguindo as diretrizes da ECSS-E-ST-40C, os requisitos de baixo nível para *software* com DAL D precisam ser verificados contra consistência, mas uma análise mais profunda não é necessária.

Para guiar as atividades de verificação de requisitos de baixo nível e de arquitetura de *software*, os seguintes *checklists* são propostos nas Tabelas 4.11 e 4.12:

Tabela 4-11 - Proposta de checklist para verificação dos requisitos de baixo nível de *software*.

Checklist para verificação de requisitos de baixo nível de *software*			**Rev A**
Identificador do Requisito:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para um requisito de alto nível de *software* ou possui uma justificativa?	☐ Sim ☐ Não ☐ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	☐ Sim ☐ Não ☐ N/A	
3	O requisito é verificável?	☐ Sim ☐ Não ☐ N/A	
4	A descrição do requisito é compatível com o *hardware*?	☐ Sim ☐ Não ☐ N/A	
5	O requisito segue o padrão estabelecido para escrita de requisitos?	☐ Sim ☐ Não ☐ N/A	
6	O algoritmo utilizado é preciso?	☐ Sim ☐ Não ☐ N/A	
7	O requisito é compatível com a arquitetura do *software*?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-12 – Proposta de checklist para verificação da arquitetura de *software*.

Checklist para verificação de arquitetura de *software*			**Rev A**
Documento avaliado:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	A arquitetura rastreia para os requisitos de alto nível de *software*?	☐ Sim ☐ Não ☐ N/A	
2	A arquitetura apresenta entradas e saídas consistentes com as interfaces?	☐ Sim ☐ Não ☐ N/A	
3	A arquitetura é verificável?	☐ Sim ☐ Não ☐ N/A	
4	A arquitetura é compatível com o *hardware*?	☐ Sim ☐ Não ☐ N/A	
5	A arquitetura segue o padrão estabelecido para definição de arquitetura?	☐ Sim ☐ Não ☐ N/A	
6	A arquitetura indica fluxo lógico com alocação de tempo e espaço adequados?	☐ Sim ☐ Não ☐ N/A	

É importante observar que no caso da arquitetura de *software*, o *checklist* deve ser preenchido para a arquitetura completa de determinada função (ou para subconjuntos completos de arquitetura), enquanto os *checklists* de requisitos são preenchidos para cada requisito.

4.2.2.3 Verificação do Código

Existe uma discreta diferença entre a RTCA DO-178B e a ECSS-E-ST-40C no processo de verificação de código. A ECSS-E-ST-40C considera que as análises de cobertura fazem parte desse processo, conforme apresentado na Seção 3.2.3, enquanto a RTCA DO-178B inclui um processo dedicado para verificação dos resultados do processo de verificação, onde deve ocorrer a análise de cobertura. A proposta deste trabalho é manter os processos

separados, pois a análise de cobertura deve ser feita após os testes do *software*, que acontecem depois de outro processo descrito nas normas como processo de integração (ver Seção 4.2.2.4).

Assim, este trabalho considera que o objetivo principal da verificação do código fonte é garantir a consistência do código em relação aos requisitos e à arquitetura (incluindo rastreabilidade) e garantir que o código implemente as sequências de eventos apropriadas, interfaces consistentes e fluxo de controle e de dados corretos.

Dessa forma, os objetivos da verificação do código fonte, diferenciados novamente pelo DAL do *software*, e a independência das atividades propostos por este trabalho são:

Tabela 4-13 – Proposta para objetivos de verificação do código fonte.

Verificação (código fonte)	DAL A e B	DAL C	DAL D
Cumprimento com requisitos de baixo nível	C	S	N/A
Cumprimento com a arquitetura	C	S	N/A
Rastreio para os requisitos de baixo nível	C	S	N/A
Conformidade com os padrões de codificação	C	S	N/A
Consistência do código fonte	C	S	N/A
Possibilidade de testes do código fonte	C	S	N/A
Nota: C indica com independência e S indica sem independência			

É importante observar que o fato de não haver verificação desses objetivos para o *software* com DAL D não indica que o desenvolvimento do código fonte para esse *software* possa ser negligenciado. A equipe responsável pela codificação faz a atividade baseada nos requisitos de baixo nível e na arquitetura e possui um padrão de codificação estabelecido pela organização que deve ser seguido. A diferença é que para *software* com DAL D, não é exigida uma atividade extra de conferência se cada linha do código fonte estiver consistente com os requisitos e com a arquitetura e se a rastreabilidade do código para os requisitos estiver completa.

Assim como os demais processos, um *checklist* que serve de guia para a verificação do código fonte é proposto, conforme a Tabela 4-14 abaixo:

Tabela 4-14 – Proposta de checklist para verificação do código fonte.

Checklist para verificação do código fonte			**Rev A**
Código avaliado:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	O código fonte rastreia para os requisitos de baixo nível de *software*?	☐ Sim ☐ Não ☐ N/A	
2	O código fonte é consistente com a arquitetura?	☐ Sim ☐ Não ☐ N/A	
3	O código fonte é verificável?	☐ Sim ☐ Não ☐ N/A	
4	O código fonte é consistente e preciso?	☐ Sim ☐ Não ☐ N/A	
5	O código fonte segue o padrão de codificação estabelecido?	☐ Sim ☐ Não ☐ N/A	

Usualmente, os *softwares* complexos são implementados utilizando uma série de códigos fonte que são integrados em um objeto executável durante o processo de integração. O *checklist* apresentado acima deve ser preenchido para cada um dos códigos fonte.

Por exemplo, se a linguagem C é utilizada, serão gerados vários arquivos do tipo “file.c” que implementam diversas funções do *software* de forma segregada. Nesse caso, deve-se estabelecer de forma clara o rastreio de cada requisito de baixo nível para os arquivos e, caso os arquivos sejam grandes, para partes do código dentro do arquivo.

4.2.2.4 Verificação do Processo de Integração

Como visto na seção anterior, o processo de integração ocorre após a codificação do *software* e está relacionado com a geração do código objeto executável.

Novamente, existe uma diferença entre a ECSS-E-ST-40C, que considera apenas a geração do código objeto executável (integração de *software*), e a RTCA DO-178B, que inclui o carregamento desse código no *hardware* alvo (integração *hardware/software*).

Este trabalho utiliza a definição da ECSS-E-ST-40C, uma vez que a compatibilidade com o *hardware* já é avaliada em todas as outras fases do desenvolvimento de *software*. De qualquer forma, a mudança não implicaria em grandes alterações, uma vez que o *checklist* da verificação do processo de integração seria acrescido de uma questão relacionada com a integração *hardware/software*.

Assim, os objetivos da verificação do processo de integração, diferenciados novamente pelo DAL do *software*, e a independência das atividades propostos por este trabalho são mostrados na Tabela 4-15:

Tabela 4-15 – Proposta para objetivos de verificação do processo de integração.

Verificação (processo de integração)	DAL A e B	DAL C	DAL D
Cumprimento com requisitos de alto nível	C	S	S
Cumprimento com requisitos de baixo nível	C	S	S
Nota: C indica com independência e S indica sem independência			

Observa-se, a partir dos objetivos apresentados acima (baseados na RTCA DO-178B), que a verificação do processo de integração está relacionada com os testes baseados nos requisitos de alto nível e de baixo nível. Como visto anteriormente, a ECSS-E-ST-40C indica que a atividade de validação da *baseline* de requisitos (considerada requisitos de cliente) e da especificação técnica (considerada requisitos de alto nível de *software*) deve ser feita por teste, mas não indica que testes devem ser feitos para os requisitos de baixo

nível (contidos no projeto detalhado do *software*). Para manter a consistência, este trabalho propõe incluir os testes de alto nível e baixo nível de *software* como parte da verificação do processo de integração, enquanto que os testes da baseline de *software* são escopo da validação do *software* (ver Seção 4.2.3).

Em relação aos testes propriamente dito, foi discutido na Seção 3.2.3 que a RTCA DO-178B explicita a necessidade de realização dos chamados testes de robustez (*robustness tests*), que complementam os testes de faixa normal (*normal range tests*), enquanto a ECSS-E-ST-40C indica que o *software* deve ser testado para valores de fronteira e fora da faixa (incluindo valores que podem causar resultados errados em funções matemáticas).

Do exposto acima, nota-se a necessidade da inclusão de mais quatro atributos aos requisitos de alto e baixo nível, além dos apresentados nas Seções 4.2.2.1 e 4.2.2.2, referentes aos testes dos requisitos. A proposta deste trabalho é a seguinte:

1- Procedimento de Teste: o identificador do procedimento de teste que verifica cada requisito deve ser indicado nesse campo. A granularidade do procedimento de teste deve ser avaliada e, caso necessário, deve-se indicar qual passo do teste verifica o requisito.
2- Resultado do Teste: deve-se indicar qual foi o resultado do teste e o relatório com o resultado dos testes para verificação de cada requisito.
3- Responsável pela Verificação: indica de forma clara quem foi responsável pela atividade de verificação do processo de integração para o requisito específico.
4- *Checklist* de Verificação: para avaliar se os testes realizados foram completos, o responsável pela verificação deverá preencher um *checklist* que serve de guia para a avaliação.

Assim, a verificação do processo de integração é feita através da análise dos procedimentos e resultados de teste contra cada um dos requisitos de alto nível e de baixo nível pois, dessa forma, pode-se observar que o código executável cumpre com ambos os níveis de requisitos, conforme objetivo explicitado tanto pela RTCA DO-178B quanto pela ECSS-E-ST-40C.

Por fim, para manter a coerência com os demais processos propostos por este trabalho, a verificação do processo de integração também deve ser feita com o auxílio de um *checklist* que deve seguir a independência mostrada na Tabela 4-15. O *checklist* proposto por este trabalho pode ser visto na Tabela 4-16 a seguir:

Tabela 4-16 – Proposta de checklist para verificação do processo de integração.

Checklist para verificação do processo de integração			**Rev A**
Identificador do requisito:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	A referência para o procedimento de teste permite identificar os passos de teste do requisito?	☐ Sim ☐ Não ☐ N/A	
2	O procedimento de teste indicado verifica a correta implementação do requisito?	☐ Sim ☐ Não ☐ N/A	
3	O procedimento de teste indicado exercita a faixa normal de operação do requisito?	☐ Sim ☐ Não ☐ N/A	
4	O procedimento de teste indicado exercita o requisito fora da faixa normal de operação?	☐ Sim ☐ Não ☐ N/A	
5	Os resultados de teste indicam claramente que todos os testes do requisito tiveram resultados satisfatórios?	☐ Sim ☐ Não ☐ N/A	

4.2.2.5 Verificação dos Testes de *Software*

Como visto na seção anterior, a verificação do processo de integração envolve os testes do código baseados nos requisitos de alto nível e de baixo nível. Tanto a ECSS-E-ST-40C quanto a RTCA DO-178B indicam que é necessário fazer a verificação tanto dos procedimentos de teste quanto dos resultados. Além disso, existe a necessidade de analisar a cobertura estrutural dos testes, conforme Tabela 3-7.

A ECSS-E-ST-40C também indica explicitamente a necessidade de rastreio entre os testes e os requisitos de alto nível e de baixo nível. Dessa forma, os objetivos da verificação dos testes de *software* propostos por este trabalho bem como as independências, baseado nas diretrizes das duas normas, podem ser vistos na Tabela 4-17 a seguir:

Tabela 4-17 – Proposta para objetivos de verificação dos testes de *software*.

Verificação (testes de *software*)	**DAL A e B**	**DAL C**	**DAL D**
Consistência e correção dos procedimentos de teste	C	S	N/A
Consistência e correção dos resultados de teste	C	S	N/A
Cobertura dos requisitos de alto nível	C	S	S
Cobertura dos requisitos de baixo nível	C	S	S
Rastreio entre testes e requisitos	C	S	S
Cobertura estrutural do código	C	S	N/A
Nota: C indica com independência e S indica sem independência			

É importante ressaltar que os procedimentos de teste e resultados para *software* com DAL D devem ser gerados durante o processo de verificação do processo de integração. A proposta do trabalho é que não existe necessidade de se verificar a correção dessa documentação.

Em relação à análise de cobertura estrutural, a proposta deste trabalho é utilizar a tabela da RTCA DO-178B, repetida na Tabela 4-18 a seguir:

Tabela 4-18 – Proposta para análise de cobertura estrutural.

Análise de Cobertura / Nível de SW	**A**	**B**	**C**	**D**
Statement coverage	C	C	S	N/A
Decision coverage	C	C	N/A	N/A
Modified condition and decision coverage	C	N/A	N/A	N/A
Nota: C indica com independência e S significa sem independência				

Dessa forma, para software com DAL A, é necessário fazer a análise de cobertura de decisão e condição modificada (MCDC - *Modified Condition and Decision Coverage*). Para satisfazer o critério de MCDC, todos os itens abaixo devem ser observados durante os testes:

- Toda decisão tenta todas as possíveis saídas;
- Toda condição de uma decisão leva a cada resultado possível;
- Todo ponto de entrada e saída é exercitado;
- Toda condição em uma decisão afeta de forma independente a saída da decisão

Nesse caso, para se provar a independência de uma condição, deve-se mostrar que apenas uma condição é alterada por vez. Como se pode notar, a análise de MCDC é extremamente complexa e é, usualmente, feita com a utilização de ferramentas de análise.

Considerando o exposto acima, este trabalho propõe dois checklists distintos para guiar o trabalho de verificação dos testes de *software*. Um para os testes baseados em requisitos de alto nível e outro para os testes baseados em requisitos de baixo nível, conforme mostrado nas Tabelas 4.19 e 4.20 a seguir:

Tabela 4-19 – Proposta de checklist para verificação dos testes de requisitos de alto nível de *software.*

Checklist para verificação dos testes de requisitos de alto nível			**Rev A**
Procedimentos e resultados avaliados:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os procedimentos de teste são consistentes?	☐ Sim ☐ Não ☐ N/A	
2	É possível determinar que todos os requisitos de alto nível de *software* foram testados?	☐ Sim ☐ Não ☐ N/A	
3	Os procedimentos rastreiam para os requisitos de alto nível sendo testados?	☐ Sim ☐ Não ☐ N/A	
4	Os resultados dos testes são consistentes com os procedimentos?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-20 – Proposta de checklist para verificação dos testes de requisitos de baixo nível de *software.*

Checklist para verificação dos testes de requisitos de baixo nível			**Rev A**
Procedimentos e resultados avaliados:			
Verificador:			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os procedimentos de teste são consistentes?	☐ Sim ☐ Não ☐ N/A	
2	É possível determinar que todos os requisitos de baixo nível de *software* foram testados?	☐ Sim ☐ Não ☐ N/A	
3	Os procedimentos rastreiam para os requisitos de baixo nível sendo testados?	☐ Sim ☐ Não ☐ N/A	
4	Os resultados dos testes são consistentes com os procedimentos?	☐ Sim ☐ Não ☐ N/A	

4.2.3 Validação do *Software*

Conforme visto nas seções anteriores, as normas aeronáuticas e espaciais para o desenvolvimento de *software* estudadas não apresentam uma atividade para garantir correção e completude (*correctness* e *completeness*) dos requisitos antes de iniciar a implementação, que é considerada a fase de validação no desenvolvimento de sistemas.

As atividades de validação previstas na ECSS-E-ST-40C referem-se a testes da *baseline* de requisitos que geram os requisitos de *software*. Todavia, no escopo deste trabalho os requisitos de *software* são capturados a partir de requisitos de sistema formalmente validados e verificados (conforme processo descrito na Seção 4.1).

Dessa forma, não é necessário estabelecer um processo próprio para a validação de requisitos no desenvolvimento de *software*. Assim, nota-se que as atividades de teste do *software* são a principal maneira de garantir que o

software que será integrado no sistema irá funcionar de maneira segura e confiável.

4.2.4 Controle de Configuração

Assim como no âmbito de sistemas, o desenvolvimento de *software* deve acontecer em um ambiente controlado que, entre outros, dificulte alterações não autorizadas e permita o arquivamento e a recuperação de todos os dados relativos a esses processos.

Como visto na Seção 3.2.4, a RTCA DO-178B diferencia as atividades de controle de configuração em dois níveis que são aplicáveis a cada um dos objetivos apresentados no Apêndice A da referida norma.

Para simplificar o entendimento do processo e torna-lo compatível com o estabelecido pelas normas espaciais (em especial a norma ECSS-M-ST-40C) a proposta deste trabalho é não dividir as atividades de controle em duas categorias. Dessa forma, independentemente da atividade e do DAL do *software*, deve existir identificação da configuração, *baselines*, relatório de problemas e controle de modificações para requisitos, arquitetura e código.

De forma semelhante ao que foi visto no desenvolvimento de sistemas, deve existir uma ferramenta para controle de modificações que usualmente é diferente do ambiente de desenvolvimento do *software*. Não obstante, é necessário garantir o rastreio entre os dois ambientes para que as modificações aprovadas sejam efetivamente implementadas.

A proposta deste trabalho é apresentar quais são as informações obrigatórias que devem constar em cada proposta de modificação, independente da ferramenta escolhida para esse processo. Assim, a ferramenta de controle de modificações deve conter:

1- Identificador: de forma a identificar de maneira não ambígua cada uma das análises de modificação; a proposta deve possuir um identificador único.
2- Descrição da Modificação: a descrição deve conter o motivo da modificação (e.g. problema na verificação, novo requisito do cliente, falha durante operação) e, preferencialmente, apresentar o texto final do item a ser modificado (requisito ou código).
3- Item Afetado: deve-se explicitar quais itens são afetados pela modificação (requisitos de alto nível, requisitos de baixo nível, arquitetura, código fonte ou código executável)
4- *Baseline* Afetada: quando requisitos são afetados, a *baseline* deve ser identificada, para garantir rastreio quando existe mais de uma modificação no mesmo requisito durante o ciclo de vida de desenvolvimento.
5- Requisitos Afetados: todos os requisitos afetados pela proposta de modificação devem ser listados, a fim de garantir a análise completa da modificação proposta.
6- Código Afetado: usualmente, o código é subdivido em várias partes. A porção do código afetada deve ser identificada.
7- Disposição da Análise: toda proposta de modificação deve ser discutida por um corpo técnico habilitado para decidir se a mudança deve ou não ser implementada (e, eventualmente, quando deve ser implementada). Todos os pontos discutidos por esse grupo devem ser incluídos nesse campo, para consulta posterior.
8- Próxima Ação: após a abertura da proposta de modificação, diferentes ações podem ser feitas como análise, investigação e implementação. Esse campo deve conter o estado atual da proposta.
9- Responsável pela Próxima Ação: assim como diversas ações podem ser feitas em cada proposta, diferentes responsáveis podem ser designados para essas ações.
10- Conclusão: ao final do processo de análise da modificação, a proposta pode ter sido: rejeitada, aprovada e implementada, aprovada e programada para implementação, entre outros vereditos.

Da mesma forma que os requisitos afetados pela modificação devem constar na ferramenta de controle de modificações, o identificador da proposta de mudança que efetivamente alterou um requisito deve ser incluído como atributo desse requisito. Assim, além dos atributos referentes à determinação e verificação incluídos no ambiente de gerenciamento dos requisitos de sistemas, é necessário incluir um atributo com esse identificador, denominado “Modificado por:“.

Por fim, é preciso garantir o correto arquivamento de todos os artefatos de desenvolvimento de sistema, incluindo as *baselines* de requisitos, as porções de código fonte, os procedimentos e resultados de teste, relatórios de análise e as análises de modificações.

4.2.5 Garantia do Produto

As seções anteriores explicitam a grande quantidade de atividades que permeiam o desenvolvimento de *software*, com grande foco nas atividades de verificação. De forma semelhante ao que foi visto no âmbito de sistemas, o bom andamento dos processos é necessário para melhorar a disponibilidade e confiabilidade dos sistemas, incluindo o *software* embarcado.

Para as atividades de garantia de produto de *software*, a proposta do trabalho é a realização de auditorias para avaliação do andamento das atividades durante o processo de desenvolvimento. Assim como no desenvolvimento de sistemas, essas auditorias devem ser feitas por um time com conhecimento dos processos sendo utilizados e que não executa nenhuma das atividades sendo auditadas. Além disso, um *checklist* é proposto para cada processo, a fim de servir como guia para auditoria e permitir o registro de problemas e resoluções.

Em relação à periodicidade das auditorias, este trabalho propõe o seguinte:

a) Verificação de Requisitos de Alto Nível de *Software*: a primeira auditoria deve acontecer quando o total de requisitos de alto nível verificados atingir 30%, para identificar problemas no início das atividades. A segunda auditoria deve acontecer com 70% dos requisitos verificados. A

partir de então, as auditorias devem acontecer sempre que houver modificações significativas nos requisitos (por exemplo, inclusão de uma nova função).

b) Verificação de Requisitos de Baixo Nível e Arquitetura de *Software*: a primeira auditoria deve acontecer quando o total de requisitos de baixo nível e arquitetura de *software* verificados atingir 30%, para identificar problemas no início das atividades. A segunda auditoria deve acontecer com 70% dos requisitos verificados. A partir de então, as auditorias devem acontecer sempre que houver modificações significativas nos requisitos (por exemplo, inclusão de uma nova função).

c) Verificação do Código: a primeira auditoria deve acontecer quando o percentual de código fonte verificado atingir 30%, para identificar problemas no início das atividades. A segunda auditoria deve acontecer com 70% do código verificado. A partir de então, as auditorias devem acontecer sempre que houver modificações significativas na codificação.

d) Verificação dos Testes de *Software*: a primeira auditoria deve acontecer quando 30% dos procedimentos de teste tiverem sido executados. A segunda auditoria deve acontecer com 70% dos procedimentos de teste executados. A partir de então, as auditorias devem acontecer quando houver mudança significativa nos procedimentos de teste.

e) Controle de Modificações: considerando que o controle de modificações iniciar-se-á em algum momento entre o início da determinação de requisitos e a validação completa dos requisitos, a primeira auditoria deve acontecer quando 10% dos requisitos forem modificados formalmente. A partir de então, cada vez que o total de requisitos modificados sofrer um acréscimo de 15% deve ser conduzida uma nova auditoria.

Além da periodicidade, é necessário definir o escopo de cada auditoria. Por questões de custo e prazo, não é esperado que as auditorias avaliem todos os requisitos do *baseline*, nem todas as linhas de código. Usualmente, são escolhidas amostras aleatórias representativas para serem auditadas de forma que, se for encontrado um problema sistêmico na execução do processo, ações de correção sejam efetuadas.

Assim, o escopo das auditorias proposto por este trabalho, para cada um dos processos é:

a) Verificação de Requisitos de Alto Nível de *Software*: Na primeira auditoria, devem ser selecionados 20% dos requisitos já verificados. Na segunda auditoria, 10% dos requisitos verificados posteriormente à primeira auditoria devem ser avaliados. A partir de então, 10% dos requisitos modificados deverão ser avaliados.

b)Verificação de Requisitos de Baixo Nível e da Arquitetura de *Software*: Na primeira auditoria, devem ser selecionados 20% dos requisitos já verificados. Na segunda auditoria, 10% dos requisitos verificados posteriormente à primeira auditoria devem ser avaliados. A partir de então, 10% dos requisitos modificados deverão ser avaliados.

c) Verificação do Código: Na primeira auditoria, devem ser selecionadas 20% das partes do código já verificadas. Na segunda auditoria, 10% das partes de código verificadas posteriormente à primeira auditoria devem ser avaliadas. A partir de então, 10% das partes do código modificadas deverão ser avaliadas.

d) Verificação dos Testes de *Software*: Na primeira auditoria, devem ser selecionados 20% dos procedimentos de teste já executados. Na segunda auditoria, 10% dos procedimentos de teste executados posteriormente à primeira auditoria devem ser avaliados. A partir de então, 10% dos procedimentos de teste modificados deverão ser avaliados.

e) Controle de Modificações: Por se tratar de um processo que envolve diversas áreas de uma organização, a primeira auditoria deve analisar 30% das propostas de modificação que já foram concluídas. A partir de então, 20% das propostas de modificações posteriores à auditoria anterior devem ser avaliadas.

Nota-se que não foi estabelecida uma atividade de garantia de produto para o processo de Verificação do Processo de Integração. A razão desta ausência é o entendimento de que as demais atividades atingem o objetivo para a garantia de produto. Além disso, os testes de *software* (bem como sua verificação)

atuam de forma a garantir que o processo de integração ocorra da forma correta.

Considerando o exposto acima, existe a necessidade de cinco *checklists* para o processo de garantia de produto no desenvolvimento de *software*. Esses *checklist*s devem ser armazenados em ambiente controlado com acesso restrito aos times envolvidos nas auditorias.

A proposta deste trabalho para os cinco *checklists* supracitados pode ser vista nas Tabelas 4.21, 4.22, 4.23, 4.24 e 4.25 a seguir, onde o campo “Justificativa” deve conter a identificação do requisito, da parte do código, do procedimento de teste ou da proposta de modificação onde foi encontrado problema e qual foi esse problema:

Tabela 4-21 – Proposta de checklist para auditoria do processo de verificação de requisitos de alto nível de *software.*

Checklist de Auditoria – Verificação de Requisitos de Alto Nível de *Software*			**Rev A**
Auditoria nº:		**Baseline:**	
Requisitos avaliados:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os requisitos possuem identificação do autor?	☐ Sim ☐ Não ☐ N/A	
2	Os requisitos possuem DAL associado?	☐ Sim ☐ Não ☐ N/A	
3	Os requisitos possuem rastreio para requisito de nível superior ou justificativa associada?	☐ Sim ☐ Não ☐ N/A	
4	Os requisitos possuem identificação do verificador?	☐ Sim ☐ Não ☐ N/A	
5	Os requisitos de DAL A ou B possuem verificador diferente do autor?	☐ Sim ☐ Não ☐ N/A	
6	Os requisitos possuem checklist de verificação preenchido?	☐ Sim ☐ Não ☐ N/A	
7	Todas as perguntas dos checklists estão corretamente preenchidas?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-22 – Proposta de checklist para auditoria dos processos de verificação de requisitos de baixo nível e da arquitetura do *software.*

Checklist de Auditoria – Verificação de Requisitos de Baixo Nível e da Arquitetura de *Software*			**Rev A**
Auditoria nº:		**Baseline:**	
Requisitos avaliados:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os requisitos possuem identificação do autor?	☐ Sim ☐ Não ☐ N/A	
2	Os requisitos possuem DAL associado?	☐ Sim ☐ Não ☐ N/A	
3	Os requisitos possuem rastreio para requisito de nível superior ou justificativa associada?	☐ Sim ☐ Não ☐ N/A	
4	Os requisitos possuem identificação do verificador?	☐ Sim ☐ Não ☐ N/A	
5	Os requisitos de DAL A ou B possuem verificador diferente do autor?	☐ Sim ☐ Não ☐ N/A	
6	Os requisitos possuem checklist de verificação preenchido?	☐ Sim ☐ Não ☐ N/A	
7	Todas as perguntas dos checklists de verificação de requisitos estão corretamente preenchidas?	☐ Sim ☐ Não ☐ N/A	
8	A arquitetura possui checklist de verificação preenchido?	☐ Sim ☐ Não ☐ N/A	
9	Todas as perguntas do checklist de verificação da arquitetura estão corretamente preenchidas?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-23 – Proposta de checklist para auditoria do processo de verificação do código.

Checklist de Auditoria – Verificação do Código			**Revisão A**
Auditoria nº:			
Partes de código avaliadas:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	O código fonte apresenta rastreio para os requisitos de baixo nível de *software*?	☐ Sim ☐ Não ☐ N/A	
2	O padrão de codificação foi seguido?	☐ Sim ☐ Não ☐ N/A	
3	O código apresenta checklist de verificação?	☐ Sim ☐ Não ☐ N/A	
4	Todas as perguntas do checklist de verificação de código foram preenchidas corretamente?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-24 – Proposta de checklist para auditoria do processo de verificação dos testes de *software*.

Checklist de Auditoria – Verificação dos Testes de *Software*			**Revisão A**
Auditoria nº:			
Procedimentos de teste avaliados:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os requisitos de alto nível possuem procedimento de teste aplicável?	☐ Sim ☐ Não ☐ N/A	
2	Os requisitos de baixo nível possuem procedimento de teste aplicável?	☐ Sim ☐ Não ☐ N/A	
3	Os procedimentos de teste possuem checklist de verificação?	☐ Sim ☐ Não ☐ N/A	
4	Todas as perguntas do checklist de verificação de teste foram preenchidas corretamente?	☐ Sim ☐ Não ☐ N/A	

Tabela 4-25 – Proposta de checklist para processo de controle de modificações (*software*).

Checklist de Auditoria – Controle de modificações			**Revisão A**
Auditoria nº:			
Propostas de modificação avaliadas:			
Auditor(es):			**Data: ___/___/___**
Item	**Questão**	**Avaliação**	**Justificativa**
1	As propostas possuem descrição clara da modificação?	☐ Sim ☐ Não ☐ N/A	
2	As propostas indicam qual o item afetado?	☐ Sim ☐ Não ☐ N/A	
3	As propostas indicam quais os requisitos afetados?	☐ Sim ☐ Não ☐ N/A	
4	As propostas indicam quais as partes do código afetadas?	☐ Sim ☐ Não ☐ N/A	
5	Os requisitos com modificações aceitas foram alterados?	☐ Sim ☐ Não ☐ N/A	
6	Os requisitos alterados apresentam o identificador da proposta no campo “Modificado por:“	☐ Sim ☐ Não ☐ N/A	
7	As partes de código com modificações aceitas foram alteradas?	☐ Sim ☐ Não ☐ N/A	
8	A proposta apresenta a conclusão da modificação?	☐ Sim ☐ Não ☐ N/A	

4.3 *Roadmap* para Implementação do Processo Proposto

Considerando a grande quantidade de atividades propostas nas seções anteriores, é importante estabelecer um *roadmap* para a implementação deste processo.

Inicialmente, tomando-se um documento de requisitos ou de especificação técnica de um sistema ou subsistema, devem-se estabelecer quais são os requisitos de sistema, de subsistema e de *software* (alto nível e baixo nível).

Após essa definição, é necessário identificar quais são os *gaps* no nível de sistema e escrever os requisitos para construir uma *baseline* completa. Utilizando o processo de validação de requisitos de sistema proposto por este trabalho, a primeira *baseline* gerada deve ser avaliada quanto à correção e completude, gerando dessa forma uma *baseline* de requisitos de sistema validada.

Quando aplicável, o conjunto de requisitos de sistema deve ser alocado a requisitos de subsistema. Caso já existam requisitos definidos neste nível, eles devem ser rastreados para a *baseline* de requisitos de sistema validada. Se houverem requisitos de subsistema que não possuam rastreio para o nível de sistema, deve-se avaliar se é necessário escrever novos requisitos de sistema para fazer o rastreio ou tratar esses requisitos como derivados. Para os requisitos de sistema que não tenham sido alocados, devem-se escrever requisitos de subsistema e fazer o rastreio para o nível de sistema.

Utilizando o processo de validação de requisitos de sistema proposto por este trabalho, a primeira *baseline* gerada deve ser avaliada quanto à correção e completude, gerando dessa forma uma *baseline* de requisitos de subsistema validada.

Caso já existam requisitos definidos como de alto nível de *software*, eles devem ser rastreados para a *baseline* de requisitos de sistema (ou de subsistema, quando aplicável) validada. Se houverem requisitos de alto nível

de *software* que não possuam rastreio para o nível de sistema ou subsistema, deve-se avaliar se é necessário escrever novos requisitos de sistema ou subsistema para fazer o rastreio ou tratar esses requisitos como derivados. Para os requisitos de sistema ou subsistema que não tenham sido alocados, devem-se escrever requisitos de alto nível de *software* e fazer o rastreio para o nível de sistema ou subsistema.

Utilizando o processo de verificação de requisitos de alto nível de *software* proposto por este trabalho, a primeira *baseline* gerada deve ser avaliada quanto à consistência, gerando dessa forma uma *baseline* de requisitos de alto nível de *software* revisada.

De forma semelhante, deve-se estabelecer uma *baseline* de requisitos de baixo nível de *software* e da arquitetura do *software* e aplicar o processo de verificação de requisitos de baixo nível e de arquitetura de *software*.

Caso já exista código fonte gerado para o sistema/*software*, deve-se garantir que existe rastreio entre todos os requisitos de baixo nível de *software* e o código. Caso algum requisito não tenha sido implementado no código, deve-se completar o código para garantir cobertura completa da *baseline*.

Por fim, devem-se gerar os casos e procedimentos de teste para verificação do processo de integração baseados nos requisitos de alto nível e de baixo nível de software, aplicando os processos propostos de verificação do processo de integração e de verificação dos testes de *software*. Além disso, deve ser feita a verificação formal dos requisitos de sistema e subsistema, através do processo de verificação dos requisitos de sistema proposto.

Em paralelo a todas essas atividades, deve-se garantir que as mudanças sejam feitas utilizando os processos de controle de configuração. Além disso, devem ser feitas auditorias nos processo utilizando os *checklists* propostos, de forma a garantir o bom andamento dessas atividades.

Para garantir a implementação deste *roadmap* de maneira adequada, é importante selecionar um ambiente adequado para o gerenciamento de requisitos que apresente as seguintes características mínimas:

- Escrita de requisitos;
- Identificação automática;
- *Links* e Rastreabilidade;
- Controle de Versão e de Modificação;
- Possibilidade de implementação de *checklists*;
- Capacidade de importação e exportação para ambiente *Microsoft Office;*
- Capacidade de fazer *backup* e restauração de todos os dados.

Existem inúmeras ferramentas que permitem a implementação dos processos descritos neste trabalho, com as características apresentadas acima. O sítio do INCOSE (*International Council on Systems Engineering* – Conselho Internacional em Engenharia de Sistemas) apresenta uma pesquisa entre as principais ferramentas encontradas no mercado. Ver <http://www.incose.org/ProductsPubs/products/rmsurvey.aspx>

5 EXEMPLOS DE APLICAÇÃO A SISTEMAS ESPACIAIS EMBARCADOS E A SISTEMAS AERONÁUTICOS EMBARCADOS

5.1 Exemplos de Aplicação a Sistemas Espaciais Embarcados

Os exemplos de aplicação a sistemas espaciais embarcados apresentados a seguir são baseados nos relatórios (INPE, 2001) *Multi-Mission Platform Attitude Control and Data Handling (ACDH) Subsystem Equipment Specification* (Especificação de Equipamentos do Subsistema de Controle de Atitude e Tratamento de Dados da Plataforma MultiMissão) e (INPE, 2001) *Multi-Mission Platform Attitude Control and Data Handling (ACDH) Subsystem Specification* (Especificação do Subsistema de Controle de Atitude e Tratamento de Dados da Plataforma MultiMissão).

Estes exemplos não tem o objetivo de indicar falhas ou problemas nas especificações apresentadas nos documentos supracitados, mas apenas mostrar que os processos definidos por este trabalho tem, de fato, aplicação prática.

Esses documentos não possuem preocupação específica com a diferenciação entre requisitos de sistema e de *software*. Assim, para este exemplo essa distinção foi feita baseada nas diretrizes da RTCA DO-248B (2001), intitulada "*Final Report for Clarification of DO-178B Software Considerations in Airborne Systems and Equipment Certification*" (Relatório Final para Esclarecimento da DO-178B).

A DO-248B indica o seguinte exemplo de decomposição de requisitos de sistema para requisitos de alto nível de *software* e requisitos de baixo nível de *software*, onde é considerado um sistema de motor com um FADEC – *Full Authority Digital Engine Control* (Controle Digital de Motor com Autoridade Completa) que recebe um sinal de velocidade baixa de rotor (NL) em porcentagem de máximas revoluções por minuto (%):

Requisito de sistema: O FADEC deve detectar e acomodar o sinal NL para falhas em aberto e em curto circuito, e falhas de alcance de acordo com as limitações da especificação de requisitos de *software* dependentes de *hardware*.

Requisito de alto nível de *software*: Se a velocidade de rotor NL está fora da faixa entre 5% e 95% ou se a taxa de mudança de NL é maior que 10% por segundo, então uma falha deve ser declarada.

Requisito de baixo nível de *software*: A precisão de NL deve ser ao menos 0,01%.

Além dos exemplos baseados em requisitos da PMM, nesta seção são apresentados exemplos de problemas na execução dos processos estabelecidos que podem ser identificados através dos checklists de auditoria propostos.

5.1.1 Requisito de Sistema Validado

Identificador: PMM-ACDH-SYS-134

Texto do Requisito: O ACDH deve distribuir 16 comandos diretos para a PMM.

Autor: I. Newton

DAL: B

Premissas: N/A

Rastreio: N/A (requisito derivado)

Justificativa: O controle de atitude deve fornecer os comandos necessários para o correto posicionamento da PMM.

Método de Validação: Análise

Artefato de Suporte ao Método de Validação: PMM-ACDH-AN-01 (relatório fictício)

Resultado da Validação: Válido

Checklist de Validação: *Ver Seção 5.1.2*

Validador: G. Leibniz

5.1.2 Checklist de Validação de Requisitos de Sistema sem Problemas Identificados

Tabela 5-1 – Checklist de validação de requisitos de sistema sem problemas identificados.

Checklist para validação de requisitos de sistema			**Revisão A**
Identificador do Requisito: PMM-ACDH-SYS-134			
Validador: G. Leibniz		**Data:** 24/11/2011	
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para uma fonte?	☐ Sim ☐ Não ■ N/A	O requisito é derivado
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim ☐ Não ☐ N/A	
3	A descrição do requisito indica apenas uma necessidade?	■ Sim ☐ Não ☐ N/A	
4	Se o requisito não rastreia para um requisito de nível superior (i.e. requisito derivado), existe um *rationale* para justificar o requisito?	■ Sim ☐ Não ☐ N/A	
5	O requisito é verificável?	■ Sim ☐ Não ☐ N/A	
6	O requisito descreve as capacidades desejadas de forma positiva?	■ Sim ☐ Não ☐ N/A	
7	O requisito apresenta as tolerâncias e faixas de operação, quando aplicável?	☐ Sim ☐ Não ■ N/A	Não apresenta tolerâncias
8	Os artefatos de suporte são consistentes com os métodos de validação identificados?	■ Sim ☐ Não ☐ N/A	
9	Considerando os artefatos de suporte, o requisito está correto?	■ Sim ☐ Não ☐ N/A	
10	O requisito está consistente com os demais requisitos relacionados à mesma função?	■ Sim ☐ Não ☐ N/A	

5.1.3 Requisitos de Sistema com Problemas Identificados na Validação

Identificador: PMM-ACDH-SYS-278

Texto do Requisito: As provisões de alinhamento (óticas e mecânicas) do ACDH devem ser localizadas em uma posição conveniente de forma que possam ser avaliadas do lado externo da espaçonave integrada.

Autor: I. Newton

DAL: C

Premissas: N/A

Rastreio: N/A (requisito derivado)

Justificativa: A localização das provisões de alinhamento deve ser definida.

Método de Validação: Julgamento de Engenharia

Artefato de Suporte ao Método de Validação: Para evitar manuseio indevido de equipamentos apenas para avaliação das provisões de alinhamento, elas devem estar visíveis do lado externo.

Resultado da Validação: Não Válido

Checklist de Validação: *Ver Seção 5.1.4*

Validador: G. Leibniz

Identificador: PMM-ACDH-SYS-519

Texto do Requisito: O receptor de GPS deve fornecer tempo e posição do satélite e velocidade a bordo e de forma autônoma.

Autor: I. Newton

DAL: B

Premissas: N/A

Rastreio: PMM-SYS-401 (requisito fictício)

Justificativa: N/A

Método de Validação: Julgamento de Engenharia

Artefato de Suporte ao Método de Validação: N/A.

Resultado da Validação: Não Válido

Checklist de Validação: *Ver Seção 5.1.4*

Validador: G. Leibniz

5.1.4 Checklist de Validação de Requisitos de Sistema com Problemas

Tabela 5-2 – Checklist de validação de requisitos de sistema com problemas identificados.

Checklist para validação de requisitos de sistema			**Revisão A**
Identificador do Requisito: PMM-ACDH-SYS-278			
Validador: G. Leibniz		**Data:** 24/11/2011	
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para uma fonte?	☐ Sim ☐ Não ■ N/A	O requisito é derivado
2	A descrição do requisito está clara, concisa e não ambígua?	☐ Sim ■ Não ☐ N/A	O uso do termo "conveniente" não define de forma clara a localização do equipamento. Mais informações devem ser fornecidas.
3	A descrição do requisito indica apenas uma necessidade?	■ Sim ☐ Não ☐ N/A	
4	Se o requisito não rastreia para um requisito de nível superior (i.e. requisito derivado), existe um *rationale* para justificar o requisito?	■ Sim ☐ Não ☐ N/A	
5	O requisito é verificável?	■ Sim ☐ Não ☐ N/A	
6	O requisito descreve as capacidades desejadas de forma positiva?	■ Sim ☐ Não ☐ N/A	
7	O requisito apresenta as tolerâncias e faixas de operação, quando aplicável?	☐ Sim ☐ Não ■ N/A	Não apresenta tolerâncias
8	Os artefatos de suporte são consistentes com os métodos de validação identificados?	■ Sim ☐ Não ☐ N/A	
9	Considerando os artefatos de suporte, o requisito está correto?	■ Sim ☐ Não ☐ N/A	
10	O requisito está consistente com os demais requisitos relacionados à mesma função?	■ Sim ☐ Não ☐ N/A	

Tabela 5-3 – Checklist de validação de requisitos de sistema com problemas identificados.

Checklist para validação de requisitos de sistema			**Revisão A**
Identificador do Requisito: PMM-ACDH-SYS-519			
Validador: G. Leibniz		**Data:** 24/11/2011	
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para uma fonte?	■ Sim □ Não □ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim □ Não □ N/A	
3	A descrição do requisito indica apenas uma necessidade?	□ Sim ■ Não □ N/A	O requisito apresenta várias necessidades no mesmo texto.
4	Se o requisito não rastreia para um requisito de nível superior (i.e. requisito derivado), existe um *rationale* para justificar o requisito?	□ Sim □ Não ■ N/A	
5	O requisito é verificável?	■ Sim □ Não □ N/A	
6	O requisito descreve as capacidades desejadas de forma positiva?	■ Sim □ Não □ N/A	
7	O requisito apresenta as tolerâncias e faixas de operação, quando aplicável?	□ Sim □ Não ■ N/A	Não apresenta tolerâncias
8	Os artefatos de suporte são consistentes com os métodos de validação identificados?	□ Sim ■ Não □ N/A	Não apresenta os artefatos de suporte para validação
9	Considerando os artefatos de suporte, o requisito está correto?	□ Sim ■ Não □ N/A	Não foi possível avaliar sem os artefatos de suporte para validação
10	O requisito está consistente com os demais requisitos relacionados à mesma função?	■ Sim □ Não □ N/A	

5.1.5 Requisito de Sistema Verificado

Identificador: PMM-ACDH-SYS-134

Texto do Requisito: O ACDH deve distribuir 16 comandos diretos para a PMM.

Autor: I. Newton

DAL: B

Premissas: N/A

Rastreio: N/A (requisito derivado)

Justificativa: O controle de atitude deve fornecer os comandos necessários para o correto posicionamento da PMM.

Método de Validação: Análise

Artefato de Suporte ao Método de Validação: PMM-ACDH-AN-01 (relatório fictício)

Resultado da Validação: Válido

Checklist de Validação: *Ver Seção 5.1.2*

Validador: G. Leibniz

Método de Verificação: Teste

Evidência de Verificação: PMM-ADCH-TP-01, Seção 2 (documento fictício).

Resultado da Verificação: Verificado

Verificador: M. Agnesi

5.1.6 Requisito de Alto Nível de *Software*

Identificador: PMM-ACDH-SWHLR-16

Texto do Requisito: A faixa do sinal de saída do magnetômetro deve ser ao menos de -600 mG a +600 mG e desvio menor ou igual a 10 mG.

Autor: J. Fourier

DAL: A

Premissas: N/A

Rastreio: N/A (requisito derivado)

Justificativa:

Resultado da Verificação: Verificado

Checklist de Verificação: *Ver Seção 5.1.7*

Verificador: P. Laplace

5.1.7 Checklist de Verificação de Requisito de Alto Nível de *Software*

Tabela 5-4 – Checklist de verificação de requisitos de alto nível de *software*.

Checklist para verificação de requisitos de alto nível de *software*			**Rev A**
Identificador do Requisito: PMM-ACDH-SWHLR-16			
Verificador: P. Laplace			**Data:** 28**/**11**/**2011
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para um requisito de sistema ou possui uma justificativa?	■ Sim □ Não □ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim □ Não □ N/A	
3	O requisito é verificável?	■ Sim □ Não □ N/A	
4	A descrição do requisito é compatível com o *hardware*?	■ Sim □ Não □ N/A	
5	O requisito segue o padrão estabelecido para escrita de requisitos?	■ Sim □ Não □ N/A	
6	O algoritmo utilizado é preciso?	□ Sim □ Não ■ N/A	Não foi utilizado algoritmo na descrição do requisito

5.1.8 Requisito de Baixo Nível de *Software*

Identificador: PMM-ACDH-SWLLR-78

Texto do Requisito: O sinal de saída do magnetômetro deve ter precisão igual, ou melhor, a 10 mG.

Autor: I. Kant

DAL: A

Premissas: N/A

Rastreio: PMM-ACDH-SWHLR-16

Justificativa: N/A

Resultado da Verificação: Verificado

Checklist de Verificação: *Ver Seção 5.1.9*

Verificador: G. Hegel

5.1.9 Checklist de Verificação de Requisitos de Baixo Nível de *Software*

Tabela 5-5 – Checklist de verificação de requisitos de baixo nível de *software*.

Checklist para verificação de requisitos de baixo nível de *software*			**Rev A**
Identificador do Requisito: PMM-ACDH-SWLLR-78			
Verificador: G. Hegel			**Data:** 02/12/2011
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para um requisito de alto nível de *software* ou possui uma justificativa?	■ Sim □ Não □ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim □ Não □ N/A	
3	O requisito é verificável?	■ Sim □ Não □ N/A	
4	A descrição do requisito é compatível com o *hardware*?	■ Sim □ Não □ N/A	
5	O requisito segue o padrão estabelecido para escrita de requisitos?	■ Sim □ Não □ N/A	
6	O algoritmo utilizado é preciso?	□ Sim □ Não ■ N/A	Não foi utilizado algoritmo na descrição do requisito
7	O requisito é compatível com a arquitetura do *software*?	■ Sim □ Não □ N/A	

5.1.10 Checklist para Auditoria de Validação de Requisitos de Sistema

Utilizando o exemplo do requisito PMM-ACDH-SYS-519, mas com um checklist incorretamente preenchido, conforme a seguir:

Identificador: PMM-ACDH-SYS-519
Texto do Requisito: O receptor de GPS deve fornecer tempo e posição do satélite e velocidade a bordo e de forma autônoma.
Autor: I. Newton
DAL: B
Premissas: N/A
Rastreio: PMM-SYS-401 (requisito fictício)
Justificativa: N/A
Método de Validação: Julgamento de Engenharia
Artefato de Suporte ao Método de Validação: N/A.
Resultado da Validação: Válido
Checklist de Validação: *Ver Tabela 5-6*
Validador: G. Leibniz

Tabela 5-6 – Checklist de validação de requisitos de sistema incorretamente preenchido.

Checklist para validação de requisitos de sistema			**Revisão A**
Identificador do Requisito: PMM-ACDH-SYS-519			
Validador: G. Leibniz		**Data:** 24/11/2011	
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para uma fonte?	■ Sim ☐ Não ☐ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim ☐ Não ☐ N/A	
3	A descrição do requisito indica apenas uma necessidade?	■ Sim ☐ Não ☐ N/A	
4	Se o requisito não rastreia para um requisito de nível superior (i.e. requisito derivado), existe um *rationale* para justificar o requisito?	☐ Sim ☐ Não ■ N/A	
5	O requisito é verificável?	■ Sim ☐ Não ☐ N/A	
6	O requisito descreve as capacidades desejadas de forma positiva?	■ Sim ☐ Não ☐ N/A	
7	O requisito apresenta as tolerâncias e faixas de operação, quando aplicável?	☐ Sim ☐ Não ■ N/A	Não apresenta tolerâncias
8	Os artefatos de suporte são consistentes com os métodos de validação identificados?	■ Sim ☐ Não ☐ N/A	
9	Considerando os artefatos de suporte, o requisito está correto?	■ Sim ☐ Não ☐ N/A	
10	O requisito está consistente com os demais requisitos relacionados à mesma função?	■ Sim ☐ Não ☐ N/A	

Tabela 5-7 – Checklist de auditoria para processos de determinação e validação de requisitos de sistema.

Checklist de auditoria – determinação e validação de requisitos			**Rev A**
Auditoria nº: 001		**Baseline:** 2.0	
Requisitos avaliados: PMM-ACDH-SYS-278/519			
Auditor(es):			**Data:** 10/01/2012
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os requisitos possuem identificação do autor?	■ Sim □ Não □ N/A	
2	Os requisitos possuem DAL associado?	■ Sim □ Não □ N/A	
3	Os requisitos possuem rastreio para requisito de nível superior ou justificativa associada?	■ Sim □ Não □ N/A	
4	Os requisitos possuem identificação do validador?	■ Sim □ Não □ N/A	
5	Os requisitos de DAL A ou B possuem validador diferente do autor?	■ Sim □ Não □ N/A	
6	Os requisitos possuem método de validação associado?	■ Sim □ Não □ N/A	
7	Para todos os métodos de validação indicados existe um artefato de suporte associado?	□ Sim ■ Não □ N/A	O requisito PMM-ACDH-SYS-519 não apresenta artefato de suporte associado ao método de “Julgamento de Engenharia”
8	Os requisitos possuem checklist de validação preenchido?	■ Sim □ Não □ N/A	
9	Todas as perguntas dos checklists estão corretamente preenchidas?	□ Sim ■ Não □ N/A	As perguntas 8 e 9 no checklist do requisito PMM-ACDH-SYS-519 estão incorretamente respondidas.

5.2 Exemplos de Aplicação a Sistemas Aeronáuticos Embarcados

Além dos exemplos relativos à indústria espacial, nesta Seção são apresentados exemplos de requisitos de sistema e *software* da indústria aeronáutica, considerando uma indicação luminosa no cockpit de uma aeronave fictícia; bem como um exemplo de verificação do processo de integração de software.

5.2.1 Requisito de Sistema Validado

Identificador: AVC-SYS-180
Texto do Requisito: A <<lâmpada do cockpit>> deve ser iluminada quando uma falha no <<sistema>> é detectada.
Autor: M. Proust
DAL: C
Premissas: O design do cockpit é compatível com o uso de lâmpadas.
Rastreio: N/A (requisito derivado)
Justificativa: A tripulação deve ser informada quando há uma falha no sistema.
Método de Validação: Julgamento de Engenharia
Artefato de Suporte ao Método de Validação: Iluminar uma lâmpada no cockpit é suficiente para chamar a atenção da tripulação quando ocorre uma falha.
Resultado da Validação: Válido
Checklist de Validação: *Ver Tabela 5-8*
Validador: F. Nietzche

Tabela 5-8 – Checklist de validação de requisitos de sistema.

Checklist para validação de requisitos de sistema			**Revisão A**
Identificador do Requisito: AVC-SYS-180			
Validador: F. Nietzche		**Data:** 11/03/2012	
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para uma fonte?	☐ Sim ☐ Não ■ N/A	O requisito é derivado
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim ☐ Não ☐ N/A	
3	A descrição do requisito indica apenas uma necessidade?	■ Sim ☐ Não ☐ N/A	
4	Se o requisito não rastreia para um requisito de nível superior (i.e. requisito derivado), existe um *rationale* para justificar o requisito?	■ Sim ☐ Não ☐ N/A	
5	O requisito é verificável?	■ Sim ☐ Não ☐ N/A	
6	O requisito descreve as capacidades desejadas de forma positiva?	■ Sim ☐ Não ☐ N/A	
7	O requisito apresenta as tolerâncias e faixas de operação, quando aplicável?	☐ Sim ☐ Não ■ N/A	Não apresenta tolerâncias
8	Os artefatos de suporte são consistentes com os métodos de validação identificados?	■ Sim ☐ Não ☐ N/A	
9	Considerando os artefatos de suporte, o requisito está correto?	■ Sim ☐ Não ☐ N/A	
10	O requisito está consistente com os demais requisitos relacionados à mesma função?	■ Sim ☐ Não ☐ N/A	

5.2.2 Requisito de Alto Nível de Software Validado

Identificador: AVC-SWHLR-47
Texto do Requisito: O <<software do subsistema>> deve enviar um output para a <<lâmpada do cockpit>> quando <<falha do subsistema 1>> ou <<falha do subsistema 2>> é detectada.
Autor: M. Bandeira
DAL: C
Premissas: N/A
Rastreio: AVC-SYS-180
Justificativa: N/A
Resultado da Verificação: Verificado
Checklist de Verificação: *Ver Tabela 5-9*
Verificador: M. Barros

Tabela 5-9 – Checklist de verificação de requisitos de alto nível de *software*.

Checklist para verificação de requisitos de alto nível de *software*			**Rev A**
Identificador do Requisito: AVC-SWHLR-47			
Verificador: M. Barros			**Data:** 12/03/2012
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para um requisito de sistema ou possui uma justificativa?	■ Sim □ Não □ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim □ Não □ N/A	
3	O requisito é verificável?	■ Sim □ Não □ N/A	
4	A descrição do requisito é compatível com o *hardware*?	■ Sim □ Não □ N/A	
5	O requisito segue o padrão estabelecido para escrita de requisitos?	■ Sim □ Não □ N/A	
6	O algoritmo utilizado é preciso?	□ Sim □ Não ■ N/A	Não foi utilizado algoritmo na descrição do requisito

5.2.3 Requisito de Baixo Nível de Software Validado

Identificador: AVC-SWLLR-113

Texto do Requisito: O <<subsys_output>> deve ser definido como TRUE quando <<sys_1_fail>> ou <<sys_2_fail>> são definidos como TRUE.

Autor: P. Laplace

DAL: C

Premissas: N/A

Rastreio: AVC-SWHLR-47

Justificativa: N/A

Resultado da Verificação: Verificado

Checklist de Verificação: *Ver Tabela 5-10*

Verificador: J. Fourier

Tabela 5-10 – Checklist de verificação de requisitos de baixo nível de *software*.

Checklist para verificação de requisitos de baixo nível de *software*			**Rev A**
Identificador do Requisito: AVC-SWLLR-113			
Verificador: J. Fourier			**Data:** 12/03/2012
Item	**Questão**	**Avaliação**	**Justificativa**
1	O requisito rastreia para um requisito de alto nível de *software* ou possui uma justificativa?	■ Sim □ Não □ N/A	
2	A descrição do requisito está clara, concisa e não ambígua?	■ Sim □ Não □ N/A	
3	O requisito é verificável?	■ Sim □ Não □ N/A	
4	A descrição do requisito é compatível com o *hardware*?	■ Sim □ Não □ N/A	
5	O requisito segue o padrão estabelecido para escrita de requisitos?	■ Sim □ Não □ N/A	
6	O algoritmo utilizado é preciso?	□ Sim □ Não ■ N/A	Não foi utilizado algoritmo na descrição do requisito
7	O requisito é compatível com a arquitetura do *software*?	■ Sim □ Não □ N/A	

5.2.4 Verificação do Processo de Integração

Tabela 5-11 – Resultado de teste de requisitos de alto nível de *software.*

AVC-TR-HLR-12			
Requisitos testados: AVC-SWHLR-47			
Passo	**Ação**	**Resultado Esperado**	**Resultado do Teste**
1	<<subsistema 1>> e <<subsistema 2>> operacionais	output para <<lâmpada do cockpit>> = 0	output para <<lâmpada do cockpit>> = 0
2	<<subsistema 1>> falhado e <<subsistema 2>> operacional	output para <<lâmpada do cockpit>> = 1	output para <<lâmpada do cockpit>> = 1
3	<<subsistema 1>> operacional e <<subsistema 2>> falhado	output para <<lâmpada do cockpit>> = 1	output para <<lâmpada do cockpit>> = 1
4	<<subsistema 1>> e <<subsistema 2>> falhados	output para <<lâmpada do cockpit>> = 1	output para <<lâmpada do cockpit>> = 1

Nota: O procedimento/resultado apresentado não faz parte da proposta do trabalho.

Tabela 5-12 – Checklist para verificação do processo de integração.

Checklist para verificação do processo de integração			**Rev A**
Identificador do requisito: AVC-SWHLR-47			
Verificador: A. Azevedo			**Data:** 28/03/2012
Item	**Questão**	**Avaliação**	**Justificativa**
1	A referência para o procedimento de teste permite identificar os passos de teste do requisito?	■ Sim □ Não □ N/A	
2	O procedimento de teste indicado verifica a correta implementação do requisito?	■ Sim □ Não □ N/A	
3	O procedimento de teste indicado exercita a faixa normal de operação do requisito?	■ Sim □ Não □ N/A	
4	O procedimento de teste indicado exercita o requisito fora da faixa normal de operação?	□ Sim □ Não ■ N/A	
5	Os resultados de teste indicam claramente que todos os testes do requisito tiveram resultados satisfatórios?	■ Sim □ Não □ N/A	

Tabela 5-13 – Checklist para verificação do processo de integração.

Checklist para verificação dos testes de requisitos de alto nível			**Rev A**
Procedimentos e resultados avaliados: AVC-TP-HLR-12			
Verificador: C. Lispector			**Data:** 01/04/12
Item	**Questão**	**Avaliação**	**Justificativa**
1	Os procedimentos de teste são consistentes?	■ Sim □ Não □ N/A	
2	É possível determinar que todos os requisitos de alto nível de *software* foram testados?	■ Sim □ Não □ N/A	
3	Os procedimentos rastreiam para os requisitos de alto nível sendo testados?	■ Sim □ Não □ N/A	
4	Os resultados dos testes são consistentes com os procedimentos?	■ Sim □ Não □ N/A	

6 CONCLUSÕES, RECOMENDAÇÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este trabalho se propôs a estudar as normas e padrões das indústrias espacial e aeronáutica relativos ao desenvolvimento de sistemas e de *software*, a fim de permitir a comparação entre essas normas e, em particular, elaborar uma proposta de aperfeiçoamento dos processos para se adequarem à realidade do INPE. O trabalho explicitou a importância do gerenciamento de requisitos durante todo o desenvolvimento.

A comparação entre as normas da ECSS e as normas utilizadas na indústria aeronáutica mostrou que existem semelhanças entre as mesmas, mas principalmente apontou a oportunidade para o aperfeiçoamento das normas da ECSS à luz da experiência aeronáutica, de forma a elaborar processos ao mesmo tempo compatíveis com a indústria espacial e adequados à aplicação no INPE.

Em relação aos processos propostos, o foco inicial foi o gerenciamento de requisitos para o desenvolvimento de sistemas. As atividades propostas para o desenvolvimento de sistemas foram Determinação e Validação de Requisitos, Verificação de Requisitos, Controle de Configuração e Garantia do Produto. O trabalho mostrou quais atividades devem ser executadas e qual o nível de independência requerido, baseado no DAL (*Design Assurance Level* – Nível de Garantia de Projeto) dos requisitos. Além disso, o trabalho propôs a utilização de *checklists* para servir como guia em grande parte das atividades.

Em relação ao desenvolvimento de *software*, as atividades propostas por este trabalho foram Verificação dos Requisitos de Alto Nível, Verificação dos Requisitos de Baixo Nível e da Arquitetura de *Software*, Verificação do Código, Verificação do Processo de Integração, Verificação dos Testes de *Software*, Validação do *Software*, Controle de Configuração e Garantia do Produto. De forma semelhante ao desenvolvimento de sistemas, o trabalho mostrou quais atividades de desenvolvimento de *software* devem ser executadas e qual o

nível de independência requerido, baseado no DAL do *software*, além de propor a utilização de *checklists*.

Por fim, utilizando a Plataforma MultiMissão desenvolvida pelo INPE, o trabalho apresentou exemplos da aplicação dos processos propostos, incluindo informação relativa a todos os campos necessários para o gerenciamento de requisitos e exemplos de *checklists* preenchidos, com o objetivo de esclarecer eventuais dúvidas existentes em relação à utilização destes processos. O mesmo foi feito utilizando uma indicação luminosa no cockpit de uma aeronave fictícia,

Ao longo do estudo dos processos e atividades de desenvolvimento por várias vezes foi necessário abandonar a discussão de um determinado tema por conta de sua grande extensão e em favor dos objetivos do trabalho. Dentre as várias possibilidades de temas a serem desenvolvidos por trabalhos futuros destacam-se os seguintes:

a) Detalhamento de outras atividades do desenvolvimento de sistemas e de *software*, como os procedimentos de teste.

b) Aperfeiçoamento dos processos de desenvolvimento de *hardware* eletrônico embarcado, baseado nas atividades da RTCA DO-254 e da ECSS-Q-ST-60-02C.

c) Discussão sobre o desenvolvimento de sistemas utilizando o conceito de IMA (*Integrated Modular Avionics* – Aviônica Modular Integrada) para sistemas embarcados operando em tempo-real, à luz da RTCA DO-297, intitulada “*Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations*” (Guia de Desenvolvimento e Considerações de Certificação para IMA).

d) Validação dos processos em relação às novas revisões das normas aeronáuticas (SAE ARP-4754A e DO-178C).

Existe ainda a possibilidade de implementar as atividades descritas neste trabalho em uma ferramenta de gestão de requisitos que permita a inclusão dos *checklist* propostos e o controle de configuração durante o ciclo de vida do sistema ou *software*, possibilitando a aplicação deste trabalho de forma mais clara, de acordo com o *roadmap* apresentado na Seção 4.3.

REFERÊNCIAS BIBLIOGRÁFICAS

ALEXANDER, I. F.; STEVENS, R. **Writing Better Requirements**, Edinburg: Pearson Education, 2002.

ALMEIDA, M. C. P. **Proposta de adoção de um processo de captura e rastreamento de requisitos baseada num estudo de caso e histórico das fases da engenharia de sistemas no INPE**. 219 p. Dissertação (Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011. Disponível em: <http://urlib.net/8JMKD3MGP7W/3AQPAJ8>. Acesso em: 13 abr. 2012.

AURUM, A.; WOHLIN, C. **Engineering and managing software requirements**, Berlin, Alemanha: Springer, 2005.

BAZOVSKY, I. **Reliability theory and practice,** Mineola, New York, EUA: Dover Publications Inc., 2004.

C-PAD-003 **Basic requirements for contractor's product assurance plan of CBERS spacecraft and associated documents,** October 28, 1989.

EASTERBROOK, S.; NUSEIBEH. B. **Requirements engineering:** a roadmap, Limerick, Ireland: Future of Software Engineering, 2000.

ENGEL. A. **Verification, validation and testing of engineered system**, Hoboken, New Jersey, EUA: John Wiley & Sons, 2010.

European Cooperation for Space Standardization - ECSS. **ECSS-E-ST-10C** space engineering – system engineering general requirements**,** Noordwijk, Holanda, 2009.

European Cooperation for Space Standardization- ECSS. **ECSS-E-ST-40C** space engineering – software**,** Noordwijk, Holanda, 2009.

European Cooperation for Space Standardization - ECSS. **ECSS-E-10-02A** space engineering - verification**,** Noordwijk, Holanda, 1998.

European Cooperation for Space Standardization - ECSS. **ECSS-M-ST-40C** space project management – configuration and information management**,** Noordwijk, Holanda, 2009.

European Cooperation for Space Standardization - ECSS. **ECSS-P-001** standardization policy**,** Noordwijk , Holanda, 2000.

European Cooperation for Space Standardization - ECSS. **ECSS-Q-ST-10C** space product assurance – product assurance management**,** Noordwijk, Holanda, 2008.

European Cooperation for Space Standardization - ECSS. **ECSS-Q-ST-60-02C** space product assurance – ASIC and FPGA development**,** Noordwijk, Holanda, 2008.

European Cooperation for Space Standardization - ECSS. **ECSS-Q-ST-80C** space product assurance – software product assurance**,** Noordwijk, Holanda, 2009.

Federal Aviation Administration – FAA. **FAA Advisory Circular AC 25.1309.** EUA. Disponível em www.faa.gov. Website consultado no dia 28 de Dezembro de 2011.

Federal Aviation Administration – FAA. **Federal Admistration Requirements FAR 25.1309.** EUA. Disponível em www.faa.gov. Website consultado no dia 28 de Dezembro de 2011.

HULL, E.; JACKSON, K.; DICK, J. **Requirements engineering**, 3. ed. London: Springer, 2011.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS – INPE. **Multi-Mission Platform Attitude Control and Data Handling (ACDH) Subsystem Equipment Specification**. São José dos Campos-SP, Brasil, 2001. (Relatório número A822700-SPC-02/03).

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS – INPE. **Multi-Mission Platform Attitude Control and Data Handling (ACDH) Subsystem Specification.** São José dos Campos-SP, Brasil, 2001. (Relatório número A822700-SPC-01/04).

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS – INPE. **Multi-Mission Platform data package for system requirements review.** São José dos Campos-SP, Brasil, 2001. 383p. (Relatório número A822000-DPK-01/D5a)

MANELLI NETO, H. **Estudo de requisitos e especificações para a tolerância a falha simples do sistema de controle de atitude da plataforma multimissão**. 2011. 207p. Dissertação (Mestrado em Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011. Disponível em: <http://urlib.net/8JMKD3MGP7W/3996485>. Acesso em: 13 abr. 2012.

National Aeronautics and Space Administration – NASA. **NASA software safety guidebook.** EUA, 2004.

National Aeronautics and Space Administration – NASA. **Software assurance guidebook.** EUA, 1989.

RTCA Inc. **DO-178B software considerations in airborne systems and equipment certification,** Washington, D.C., EUA, 1992.

RTCA Inc. **DO-178C software considerations in airborne systems and equipment certification,** Washington, D.C., EUA, 2011.

RTCA Inc. **DO-248B final report for clarification of DO-178B "Software Considerations in Airborne Systems and Equipment Certification",** Washington, D.C., EUA, 2001.

RTCA Inc. **DO-254 design assurance guidance for airborne electronic hardware,** Washington, D.C., EUA, 1992.

SAE International. **ARP-4754 certification considerations for highly-integrated or complex aircraft systems,** Warrendale, PA, EUA, 1996.

SAE International. **ARP-4754A guidelines for development of civil aircraft and systems,** Warrendale, PA, EUA,2010.

SAE International. **ARP-4761 guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment,** Warrendale, PA, EUA, 1996.

SERRA, P. R. **Análise de confiabilidade e segurança de sistemas aeronáuticos** (safety assessment of aircraft systems), São José dos Campos, SP, Brasil, 2008.

SOUZA, M. L. O.; CARVALHO, T. R. **The fault avoidance and the fault tolerance approaches for increasing the reliability of aerospace and automotive systems**, INPE, S.J. Campos, SP, 2005.

YOUNG, R. R. **The requirements engineering handbook**. Norwood, MA, EUA: Artech House, 2004.

Bibliografia Complementar

European Cooperation for Space Standardization - ECSS. **ECSS-E-10-03A** space engineering - testing**,** Noordwijk, Holanda, 2002.

LAPRIE, J. C. **Dependability:** basic concepts and terminology. Viena, Áustria: Springer, 1992.