UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

JACSON RODRIGUES BARBOSA

Estudo e Definição de uma Metodologia de Teste de Software no Contexto de Sistemas Embarcados Críticos

Goiânia
2011

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE DISSERTAÇÃO EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Estudo e Definição de uma Metodologia de Teste de Software no Contexto de Sistemas Embarcados Críticos

Autor(a): Jacson Rodrigues Barbosa

Goiânia, 28 de Julho de 2011.

Jacson Rodrigues Barbosa – Autor

Auri Marcelo Rizzo Vincenzi – Orientador

JACSON RODRIGUES BARBOSA

Estudo e Definição de uma Metodologia de Teste de Software no Contexto de Sistemas Embarcados Críticos

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

Área de concentração: Sistema de Informação.

Orientador: Prof. Auri Marcelo Rizzo Vincenzi

Goiânia
2011

JACSON RODRIGUES BARBOSA

Estudo e Definição de uma Metodologia de Teste de Software no Contexto de Sistemas Embarcados Críticos

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Computação, aprovada em 28 de Julho de 2011, pela Banca Examinadora constituída pelos professores:

Prof. Auri Marcelo Rizzo Vincenzi
Instituto de Informática – UFG
Presidente da Banca

Prof. Fabiano Cutigi Ferrari
Departamento de Computação – UFSCar

Prof. Plínio de Sá Leitão Júnior
Instituto de Informática – UFG

Jacson Rodrigues Barbosa

Graduou-se em Ciência da Computação na UFG - Universidade Federal de Goiás. Durante sua graduação, participou de um projeto de pesquisa no Departamento de Redes de Comunicação - IFG, sendo bolsista da CAE-IFG. Durante o Mestrado, também na UFG, foi bolsista da FAPEG, desenvolveu artigos à comunidade científica e participou de diversos eventos de Engenharia de Software.

À Deus pelo cuidado e sustento diário, mesmo não merecendo, um dia Ele me escolheu e me deu vida em abundância. Obrigado Pai, pelo amor incondicional.

“Sabemos que todas as cousas cooperam para o bem daqueles que amam a Deus, daqueles que são chamados segundo o seu propósito” (Romanos 8:28)

Agradecimentos

Esta conquista é resultado de apoio e ajuda de muitas pessoas que contribuíram para a realização deste trabalho.

Primeiramente agradeço à Deus pela capacitação e orientação, concedidas à mim.

Agradeço à minha querida e linda esposa, Cristhiane, que pacientemente compreendeu os momentos em que fiquei horas dedicando aos estudos.

Agradeço aos meus pais, Victor (Vitim) e Zelinda, que sempre buscaram proporcionar aos filhos uma boa formação acadêmica, mesmo que isso implicasse em sacrifícios pessoais.

Meus agradecimentos ao Auri, professor, orientador, colega e amigo.

Agradeço à colega Katia Romero Felizardo (ICMC-USP), ao Professor Torgeir Dingsøyr (SINTEF ICT) e ao Professor Fabiano Cutigi Ferrari (UFSCar), por terem ajudado muito durante a realização da revisão sistemática.

Agradeço aos colegas do mestrado: Ênio, Marcos Ivamoto, Marcelo Quinta, Marcelo Henrique, Jean Paulo e tantos outros que me ajudaram a entender melhor os conceitos apresentados em sala de aula.

Por fim, quero agradecer também aos amigos do INF: Berenice (Berê), Edir, Bosco (Boscão) e Ricardo.

Sou muito grato a todos vocês, Brigaduuuuuuuu!!!

À FAPEG e à UFG pelo auxílio financeiro.

Ora, àquele que é poderoso para fazer infinitamente mais do que tudo quanto pedimos ou pensamos, conforme o seu poder que opera em nós, a ele seja a glória, na Igreja e em Cristo Jesus, por todas as gerações, para todo o sempre. Amém.

Efésios 3:20-21,
Bíblia Sagrada, Revista e Atualizada no Brasil, 2a Edição.

Resumo

Rodrigues Barbosa, Jacson. **Estudo e Definição de uma Metodologia de Teste de Software no Contexto de Sistemas Embarcados Críticos**. Goiânia, 2011. 112p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

A computação ganha cada vez mais espaço em aplicações embarcadas críticas e, dependendo do software, seu mau funcionamento pode provocar desde um grave prejuízo financeiro até a perda de vidas humanas. Considerando este cenário, é apresentado uma revisão sistemática com o objetivo de investigar a evolução dos trabalhos relacionados a atividade de teste de software embarcado crítico visando avaliar o nível de aderência dos trabalhos encontrados em relação à norma DO-178B (*Software Considerations in Airborne Systems and Equipment Certification*). Esta pesquisa, além de realizar a revisão sistemática dos trabalhos relacionados ao tema, apresenta como resultado a composição de estudos primários para definição de um processo de teste de qualidade e que contemple as exigências da DO-178B em seus diferentes níveis de criticalidade.

Palavras–chave

Teste de software, Sistemas embarcados críticos e DO-178B.

Abstract

Rodrigues Barbosa, Jacson. **Study and Definition of a Methodology for Software Testing in the Context of Critical Embedded Systems**. Goiânia, 2011. 112p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Computing is becoming increasingly critical in the embedded applications space and depending on the software, its malfunction may result in a severe financial loss to the loss of human life. Considering this scenario, we presented a systematic literature review in order to investigate the evolution of work-related activity test critical embedded software in order to evaluate the level of compliance found in the work to the standard DO-178B (Software Considerations in Airborne Systems and Equipment Certification). This research, in addition to conducting a systematic review of publications about this issue, has resulted in the composition of primary studies to define a process of quality testing and including the requirements of DO-178B at their different levels of criticality.

Keywords

Software Testing, Critical embedded systems and DO-178B.

Conteúdo

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Devido à crescente evolução da tecnologia da informação, tem-se como resultado a redução do consumo de energia dos dispositivos computacionais, a melhoria do desempenho e a redução do tamanho físico do mesmos. Estes avanços tecnológicos têm beneficiado o aumento do uso de sistemas embarcados em diversas aplicações (TIAN et al., 2009).

Segundo Tian et al. (2009), um sistema embarcado é uma parte de um produto com o qual um usuário final não interage e nem controla diretamente e possui as seguintes características:

- Utiliza a mecânica, eletrônica, tecnologia de hardware e software e esses elementos estão intimamente relacionados entre si. Como existe uma elevada dependência entre o software e o hardware, a funcionalidade e o desempenho dos sistemas embarcados são implementadas por meio da cooperação entre ambos.
- Usualmente tem recursos de hardware limitados, como tamanho da memória e velocidade de processamento.
- Deve ser robusto, ou seja, o comportamento do mesmo deve ser controlado, mesmo durante falhas no sistema.

Durante a fase de projeto de sistemas embarcados, centra-se na implementação de um conjunto de funcionalidades que devem satisfazer um conjunto de restrições. A escolha do tipo de implementação define quais funcionalidades serão implementadas, como um componente de hardware ou de software. Em geral, a complexidade aliada à constante evolução da especificação acaba direcionando os projetistas a optarem por implementações mais flexíveis e que podem ser alteradas rapidamente, no caso implementações de software (TIAN et al., 2009).

Mas quando o sistema embarcado pode levar à morte ou a lesões graves às pessoas, a danos ambientais ou a perdas financeiras volumosas caso o sistema apresente uma falha ou mau funcionamento, é classificado como sistema embarcado crítico (MALDONADO, 2008), também conhecido como sistema de segurança crítica.

Segundo Tracey et al. (2002), sistema de segurança crítica refere-se a um sistema cuja falha pode resultar em ferimentos ou perda de vidas. Software de segurança crítica é qualquer software que possa direta ou indiretamente contribuir para a ocorrência de um estado perigoso em um sistema de segurança crítica. Um estado de risco é uma condição de um sistema que, juntamente com outras condições no ambiente, poderá resultar em um acidente.

Com o objetivo de aumentar o nível de conhecimento, competência e qualidade no país sobre o desenvolvimento de sistemas embarcados críticos, tendo em vista que se trata de uma tecnologia importante para o desenvolvimento do país em determinadas áreas, como a do meio ambiente, a de segurança e a de agricultura, recentemente foi criado o Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC), que visa pesquisar: diretrizes para a certificação da inclusão de Veículos Aéreos Não Tripulados (VANTs) no espaço aéreo controlado; metodologia para o desenvolvimento de sistemas de software embarcado críticos; comunicação em sistemas críticos; metodologia para o desenvolvimento de sistemas embarcados de hardware críticos; sistemas de computação de alto desempenho para processamento de sinais e imagens em tempo real; sistemas de controle inteligente para sistemas embarcados; desenvolvimento de sistemas robóticos embarcados; sistemas de aplicação orientados a dados dinâmicos (aquisição, tratamento e visualização de contextos em sistemas complexos de monitoramento); ambientes inovadores de treinamento para forças de segurança na preparação e resposta a emergências; avaliação de desempenho de sistemas embarcados críticos; e mecanismos de apoio ao ensino e treinamento (MALDONADO, 2008).

Em se tratando de sistemas com estas características, o principal objetivo do processo de desenvolvimento de software é garantir que o risco de acidentes se torne aceitável (conforme o impacto de possíveis falhas). Considerando este contexto, existem diversas normas e padrões que visam disponibilizar orientações e métodos para apoiar o desenvolvimento de software embarcado crítico, como, por exemplo, a DO-178B (*Software Considerations in Airborne Systems and Equipment Certification*) e a IEC61508 (*Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*).

A DO-178B é uma norma que disponibiliza um conjunto de diretrizes para o desenvolvimento de software embarcado em sistemas e equipamentos com um alto nível de segurança e que atende aos requisitos de aeronavegabilidade exigidos nas certificações das aeronaves (SOUZA; DIAS, 2009). Já o padrão IEC61508 é uma norma internacional destinada a ser um padrão de segurança funcional básica aplicável a todos os tipos de indústria, que abrange o ciclo de vida completo de segurança. A norma define segurança funcional como: “parte da segurança global relativa ao Equipamento em Controle (*Equipment Under Control*)”.

1.1 Objetivos

A partir deste trabalho, objetiva-se identificar o estado da arte do teste de software no contexto de sistemas embarcados críticos por meio de uma revisão sistemática. Tem-se como alvo também a atual demanda do INCT-SEC, definição de uma metodologia própria para o teste de software embarcado crítico. Para atender a esse objetivo, foram apresentadas respostas para as seguintes questões de pesquisa:

- Quais técnicas e critérios de teste de software tem sido investigados para o teste de software de sistemas embarcados críticos?
- Quais processos de software tem sido investigados para o teste de software de sistemas embarcados críticos?
- Que tipo de estudos experimentais tem sido empregados no contexto de teste de software embarcado crítico e qual é o respectivo grau de aderência com a norma DO-178B?

1.2 Metodologia

Para que fosse possível atender os objetivos do presente trabalho, inicialmente foi planejada (definido um protocolo) e executada uma revisão sistemática (RS). Após a sumarização dos resultados da revisão sistemática e realização do mapeamento dos estudos selecionados para as correspondentes atividades de verificação e validação (V&V), iniciou-se a definição da metodologia de teste de software embarcado crítico (SEC), considerando o contexto do INCT-SEC e os estudos primários selecionados na RS. A Figura 1.1 apresenta todas as etapas da metodologia deste trabalho.

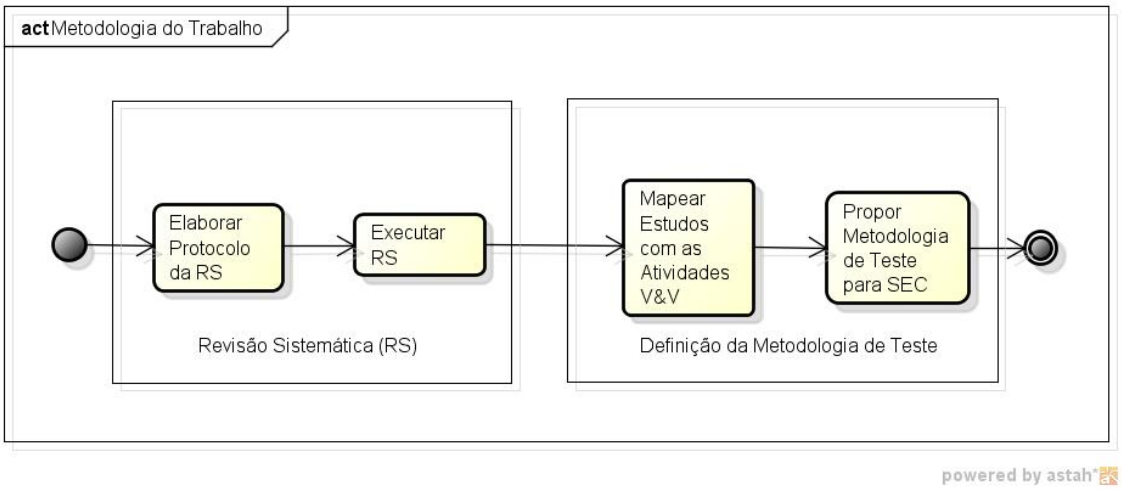

Figura 1.1: *Etapas da Metodologia do Trabalho*

Mais detalhes referente ao planejamento da revisão sistemática estão disponíveis no Capítulo 3 desta dissertação. Já os resultados da condução da revisão sistemática encontram-se no Capítulo 4 e no Apêndice B.

Durante a condução da revisão sistemática, vários artigos e relatórios descrevendo a metodologia e os resultados obtidos foram submetidos a conferências e periódicos especializados. Um dos artigos aceito foi publicado na *Trilha de Computação aplicada à Engenharia* no *Computer on The Beach 2011* (BARBOSA et al., 2011b). Outro artigo foi aceito para ser publicado no *The Third International Conference on Advances in System Testing and Validation Lifecycle (VALID 2011)* (BARBOSA et al., 2011a).

1.3 Organização da Dissertação

Esta dissertação está organizada em seis capítulos. Neste capítulo foram apresentados os principais conceitos relacionados a esta pesquisa, o objetivo e a metodologia visando delinear o tema explorado nesta dissertação.

O Capítulo 2 apresenta os principais aspectos relacionados com a norma DO-178B e uma visão geral dos processos definidos na mesma.

O Capítulo 3 apresenta o planejamento da revisão sistemática.

O Capítulo 4 apresenta uma síntese e análise dos estudos selecionados durante a condução da revisão sistemática.

O Capítulo 5 apresenta uma metodologia para o teste de software no contexto de sistemas embarcados críticos, proposta considerando as características do Instituto Nacional de Ciências e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC) e as exigências da DO-178B.

No Capítulo 6 são feitas considerações finais, apresentadas as limitações e os possíveis desdobramentos decorrente desta pesquisa.

O Apêndice A apresenta o Glossário com as respectivas definições dos principais termos utilizados no trabalho.

Finalmente, no Apêndice B são apresentados os resultados da condução da revisão sistemática.

CAPÍTULO 2

Aspectos Gerais da Norma DO-178B

A computação está ganhando cada vez mais espaço em aplicações embarcadas em aeronaves e dependendo da aplicação do software, seu mau funcionamento pode provocar desde um grave prejuízo financeiro até a perda de vidas (SOUZA; DIAS, 2009).

Considerando este cenário, a RTCA (*Radio Technical Commission for Aeronautics*), uma associação de organizações aeronáuticas dos Estados Unidos da América (tanto do governo como da indústria), iniciou em maio de 1980 o trabalho para criação e documentação de práticas de desenvolvimento de software embarcado crítico, dando origem assim ao documento DO-178, de título "Considerações sobre Certificação de Software de Sistemas e Equipamentos Aeronáuticos", que após revisões resultou na DO-178B em 1992.

A DO-178B é uma norma reconhecida pelo FAA (*Federal Aviation Administration*), cujo objetivo é disponibilizar as diretrizes necessárias para o desenvolvimento de software embarcado em sistemas e equipamentos com um alto nível de segurança e que satisfaça os requisitos de aeronavegabilidade exigidos nas certificações das aeronaves (SOUZA; DIAS, 2009).

Neste capítulo é apresentado na Seção 2.1 uma visão geral da norma DO-178B, na Seção 2.2 são discutidos alguns aspectos dos processos de ciclo de vida do sistema. O processo de verificação de software é discutido na Seção 2.3. Em seguida, na Seção 2.4 é apresentado o processo de certificação. Finalmente, na Seção 2.5 são feitas as considerações finais sobre este capítulo.

2.1 Visão geral da DO-178B

A Figura 2.1 localizada na Página 21 apresenta a organização da norma DO-178B, sendo que no presente capítulo são discutidos em detalhes as seções destacadas na figura.

A Seção 2 da DO-178B expõe os aspectos do sistema relacionado ao desenvolvimento do software, já na Seção 10 é apresentado uma visão geral a respeito de aeronave e certificação de máquina.

O restante das seções da DO-178B discorre a respeito dos processos de ciclo de vida do software, um conjunto de processos específicos definidos para que uma organização tenha condições de produzir um produto de software.

A Seção 3 da DO-178B apresenta o ciclo de vida do software, já na Seção 4 é apresentado o processo de planejamento de software, responsável por confeccionar planos e definir padrões de software que são utilizados para direcionar a execução dos processos de desenvolvimento de software e processos integrais (RTCA/EUROCAE, 1992).

A Seção 5 da DO-178B apresenta os processos de desenvolvimento de software, que são aplicados e definidos conforme o processo de planejamento de software. Os processos de desenvolvimento de software são: processo de requisito de software, processo de projeto arquitetural de software, processo de codificação de software e processo de integração.

A Seção 11 da DO-178B apresenta dados referente ao ciclo de vida do software, dados estes que são produzidos ao longo do ciclo de vida do software para planejar, definir, registrar ou fornecer evidências da execução de determinadas atividades. Esta seção discute características, forma, controle de gerenciamento de configuração e conteúdo dos dados referente ao ciclo de vida do software.

Já a Seção 12 da DO-178B discute algumas considerações adicionais referente à certificação de aspectos do software em relação ao uso de ferramentas de software qualificadas para o desenvolvimento de software crítico, métodos alternativos para atingir objetivos específicos e outras considerações.

Na DO-178B também são apresentados os processos integrais, que auxiliam os processos de desenvolvimento de software e permanecem ativos durante todo o ciclo de vida do software. Os processos integrais são o processo de verificação de software (Seção 6 da DO-178B), processo de gerenciamento da configuração de software (Seção 7 da DO-178B), processo de garantia da qualidade de software (Seção 8 da DO-178B) e processo de certificação (Seção 9 da DO-178B).

O processo de gerenciamento da configuração de software atua de forma cooperada com os outros processos de ciclo de vida de software visando assim disponibilizar e definir uma configuração controlada do software dentro do ciclo de vida do software inteiro.

Já o processo de garantia da qualidade de software visa proporcionar garantias de que os processos de ciclo de vida de software produzirão um produto de software em conformidade com os requisitos e assegurar que os processos executados também estejam em conformidade com o plano de software e padrões aprovados.

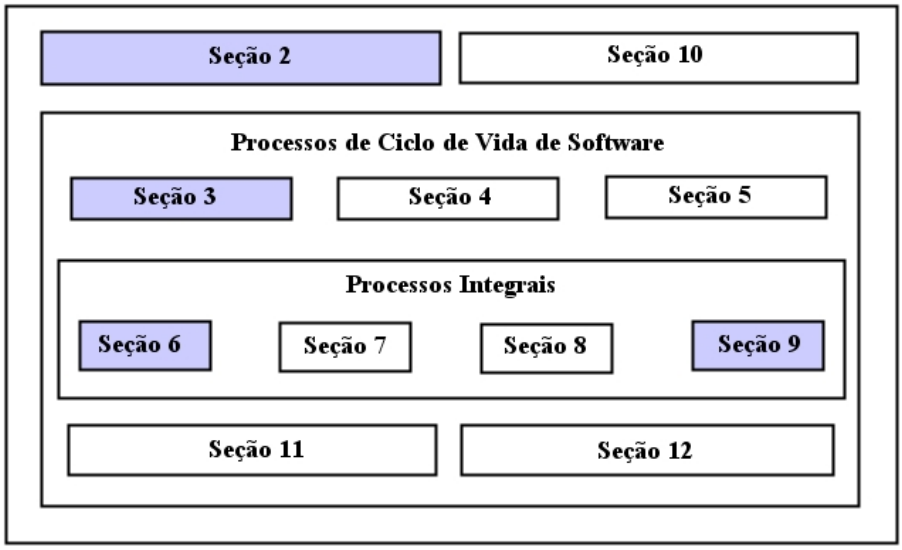

Figura 2.1: *Organização da DO-178B (adaptada de (RTCA/EUROCAE, 1992))*

Na Seção 2.2 desta dissertação, são apresentadas características gerais da DO-178B. A Seção 2.3 expõe os objetivos e atividades do processo de verificação de software. No que se refere a Seção 2.4, a mesma finaliza e apresenta o processo de certificação.

2.2 Aspectos do sistema relacionado ao desenvolvimento do software

Nesta seção são discutidos alguns aspectos dos processos de ciclo de vida do sistema necessários para o entendimento dos processos de ciclo de vida do software.

2.2.1 Fluxo de informações entre o ciclo de vida dos processos do sistema e do software

Fluxo de informações dos processos do sistema para os processos de software

O processo de avaliação de sistemas seguros determina e categoriza as condições de falhas do sistema (RTCA/EUROCAE, 1992). Dentro do processo de avaliação de sistemas de segurança, uma análise do projeto do sistema define a segurança dos requisitos relatados que especifica a imunidade pretendida de respostas do sistema para as respectivas condições de falha.

Estes requisitos são definidos para hardware e software visando eliminar ou limitar os efeitos das faltas e pode viabilizar a detecção de falha e tolerância à falha. Como as decisões são tomadas durante o processo de projeto de hardware e processo de desenvolvimento de software, o processo de avaliação de sistemas seguros analisa

o projeto arquitetural do sistema resultante para verificar se os requisitos de segurança relatados são satisfeitos.

Os requisitos de segurança relatados são parte dos requisitos do sistema que serão utilizados como insumo para o ciclo de vida dos processos de software. Os requisitos de sistema incluem ou referenciam os seguintes aspectos:

- Descrição do sistema e definição do hardware;
- Requisitos do sistema alocados para o software, incluindo requisitos funcionais, requisitos de desempenho e requisitos de segurança relatados;
- Nível do software, condições de falha e funções relatadas que são alocadas para o software;
- Estratégia de segurança e restrições de projeto, incluindo métodos de projeto, como o particionamento, múltiplas versões desiguais do software, redundância e monitoramento de segurança.

Fluxo de informações dos processos de software para os processos do sistema

O processo de avaliação de sistemas de segurança auxilia a identificação do impacto do projeto do software e implementação do sistema de segurança utilizando informações disponibilizadas pelo ciclo de vida do processo de software. Estas informações incluem limite de contenção de falhas, requisitos de software, arquitetura do software e fonte de defeitos que seriam detectados ou eliminados por meio da arquitetura de software ou com o uso de ferramentas/métodos aplicados no processo de projeto de software.

2.2.2 Condições de falha e nível do software

A categoria de condição de falha de um sistema é determinada a partir da severidade das condições de falha da aeronave e o respectivo impacto aos ocupantes da mesma. Um defeito em um software pode causar uma falha, e a DO-178B classifica falhas em cinco possíveis condições de falha apresentadas a seguir.

Categorização da condição de falha

As categorias de condição de falha são definidas levando em consideração os efeitos e consequências ocasionados à aeronave, tripulação e passageiros (VINCENZI, 2009):

- Catastrófica (*Catastrophic*): condição de falha que poderá causar a queda da aeronave.

- Perigosa (*Hazardous/Severe-Major*): condição de falha pode reduzir a habilidade da tripulação de operar a aeronave em decorrência da alta carga de trabalho ou pode causar danos fatais aos passageiros.
- Maior (*Major*): condição de falha pode provocar um impacto significativo, mas pode no máximo causar um desconforto ao passageiro.
- Menor (*Minor*): condição de falha ocasiona um impacto notável, mas pode no máximo causar inconvenientes ao passageiro como por exemplo mudança no plano de voo.
- Sem efeito (*No Effect*): condição de falha que não afeta a capacidade operacional da aeronave.

Definição do nível do software

A presente norma define níveis de exigência do software considerando os efeitos (condições de falha) caso o software apresente um comportamento anômalo. Sendo que para cada nível do software, também é exigido a cobertura de determinados critérios de teste, por exemplo, software cujo o nível seja igual a "C", é exigido que 100% dos requisitos do software sejam avaliados e que 100% dos comandos do código fonte sejam exercitados ao menos uma vez durante os testes (critério de cobertura SC). Na Tabela 2.1 é apresentada esta relação, considerando os cinco níveis de exigência da DO-178B.

Tabela 2.1: *Relação entre níveis e condição de falha (adaptada de (DOWNING, 2002)).*

Nível	*Condição de Falha*	*Critério de cobertura*	*Exigência de Cobertura*
A	Catastrófica	MC/DC	Nível B + 100% do critério MC/DC
B	Perigosa	DC	Nível C + 100% do critério DC
C	Maior	SC	Nível D + 100% do critério SC
D	Menor		100% de cobertura dos requisitos
E	Sem Efeito		Não apresenta exigência de cobertura

O Nível A se refere ao nível mais rigoroso e o correspondente critério de cobertura MC/DC (*Modified Condition/Decision Coverage*) exige que cada condição em uma decisão tenha que assumir todos os possíveis valores pelo menos uma vez e, cada decisão tenha que assumir todos os possíveis valores pelo menos uma vez e cada condição tenha que ser atribuída para afetar de forma independente o resultado da decisão. Uma condição é atribuída para afetar de forma independente o resultado de uma decisão, variando apenas uma delas e mantendo as outras condições fixas. Já o critério DC (*Decision Coverage*) exige que cada decisão assuma todos os possíveis valores pelo menos uma vez e, o critério SC (*Statement Coverage*) exige que todos os comandos do código sejam exercitados ao menos uma vez (YU; LAU, 2006).

Determinação do nível do software

Para se determinar o nível de um software adequadamente, deve-se executar o processo de avaliação do sistema de segurança adequadamente para os componentes de software sem levar em consideração o projeto do sistema.

Quando um determinado componente de software apresenta um comportamento anômalo que resulta em mais de uma condição de falha, então considera-se a mais severa condição de falha para categorizar o nível do software para o respectivo componente de software.

Uma função do sistema pode ser alocada para mais de um componente de software particionado (trata se de uma técnica que visa o isolamento entre componentes de software independentes funcionalmente). O uso do paralelismo é uma forma de se implementar uma função do sistema com múltiplos componentes de software de tal maneira que o comportamento anômalo de mais de um componente é necessário para produzir a condição de falha.

2.2.3 Ciclo de vida do software

Um projeto pode ter mais de um ciclo de vida, tendo em vista que, dado um conjunto de processos, definem-se as respectivas atividades para os mesmos, especifica-se a sequência de execução das atividades (considerando as característica específicas do projeto) e atribuem-se responsabilidades para as atividades (RTCA/EUROCAE, 1992).

Na Figura 2.2 é apresentado a ordem de execução dos processos de desenvolvimento de vários componentes de um produto de software com diferentes ciclos de vida. Na Figura 2.2 os processos de desenvolvimento de software apresentados são: processo de requisitos de software (R), processo de projeto de software (D), processo de codificação (C) e processo de integração (I). Por exemplo, o componente W implementa um conjunto de requisitos do software, utilizando os mesmos para definir o projeto do software que é insumo para implementação do código fonte, por fim, o componente construído é integrado com o hardware, empregando-se assim um ciclo de vida do software sequencial (Modelo Cascata). Já o componente Z foi construído empregando-se um ciclo de vida do software iterativo.

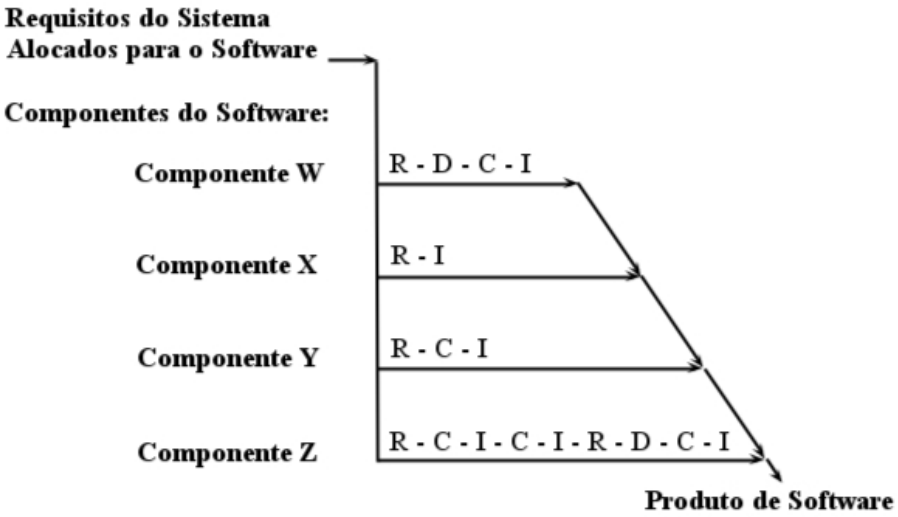

Figura 2.2: *Exemplo de um projeto de software utilizando os processos de desenvolvimento (adaptada de (RTCA/EUROCAE, 1992))*

2.3 Processo de verificação de software

Nesta seção discutem-se os objetivos e atividades do processo de verificação de software. Verificação é uma técnica de avaliação dos resultados dos processos de desenvolvimento de software, visando avaliar a consistência e equivalência dos artefatos produzidos durante o processo. O processo de verificação é aplicado conforme previsto no processo de planejamento de software e no plano de verificação de software (seção 11.3 da DO-178B) (RTCA/EUROCAE, 1992).

É importante ressaltar que verificação não se resume apenas à realização de testes, visto que teste de software não permite revelar a ausência de defeitos. Como resultado disso, na DO-178B emprega-se o termo "verificar" em vez de "testar" quando os objetivos do processo de verificação de software visam discutir a combinação de revisão, análise e teste (RTCA/EUROCAE, 1992).

As atividades de V&V são divididas em estáticas e dinâmicas. As estáticas são as que não exigem a execução do software, como a revisão e a inspeção, já as dinâmicas exigem que o software seja executado utilizando para isso entradas específicas e seja verificado se o comportamento do mesmo condiz com o esperado, como realizado pelos testes (DELAMARO et al., 2007).

2.3.1 Objetivos do processo de verificação de software

Segundo a DO-178B, o propósito principal do processo de verificação de software é detectar e relatar defeitos que foram introduzidos durante os processos de desen-

volvimento de software. A remoção dos defeitos é uma atividade de desenvolvimento de processos de software. Os objetivos gerais do processo de verificação visam verificar que:

- Requisitos do sistema alocados para o software deram origem aos requisitos de software de alto nível que satisfazem os requisitos do sistema;
- Os requisitos de alto nível deram origem à arquitetura de software e aos requisitos de baixo nível, que satisfazem os requisitos de alto nível. Caso o código fonte seja gerado diretamente a partir dos requisitos de alto nível, isso significa que esse objetivo não foi aplicado adequadamente;
- A arquitetura do software e os requisitos de baixo nível deram origem ao código fonte que satisfaz os requisitos de baixo nível e a arquitetura de software;
- O código objeto executável satisfaz os requisitos de software;
- Os meios utilizados para satisfazer estes objetivos são tecnicamente corretos e completos para o respectivo nível do software.

2.3.2 Atividades do processo de verificação de software

Os objetivos do processo de verificação de software são satisfeitos por meio da combinação de revisões, análises, desenvolvimento de casos de teste e procedimentos e subsequente execução dos mesmos. As revisões e análises viabilizam a avaliação da acurácia, completude e verificabilidade dos requisitos de software, da arquitetura de software e código fonte.

O desenvolvimento dos casos de teste viabiliza a avaliação futura da consistência interna e completude dos requisitos. Já com relação à execução dos procedimentos de teste, visa avaliar o grau de conformidade com os requisitos.

Os documentos que são utilizados como insumos para a execução do processo de verificação de software são: requisitos do sistema, requisitos e arquitetura de software, dados de rastreabilidade, código fonte, código objeto executável e plano de verificação de software.

Após a execução do processo de verificação, espera-se que sejam disponibilizados os seguintes documentos: registro de casos de teste, procedimentos de verificação de software e resultados da verificação de software.

O processo de verificação proporciona rastreabilidade entre a implementação do software e os respectivos requisitos de software:

- A rastreabilidade entre os requisitos de software e os casos de teste pode ser realizada pela análise de cobertura baseada em requisitos;
- A rastreabilidade entre a estrutura do código e os casos de teste pode ser realizada pela análise de cobertura estrutural.

Seguem abaixo orientações para as atividades de verificação de software:

- Rastreabilidade entre os requisitos de alto nível deve ser verificada;
- Os resultados da análise baseada em requisitos e da cobertura estrutural deve mostrar que cada requisito de software é rastreável para o código;
- Caso o código testado não seja idêntico ao software da aeronave, essas diferenças devem ser especificadas e justificadas;
- Quando não é possível verificar um requisito de software específico utilizando um ambiente de teste, outros meios devem ser utilizados e justificados para satisfazer os objetivos do processo de verificação de software definidos no plano de verificação ou nos resultados da verificação de software;
- Discrepâncias e falhas identificadas durante o processo de verificação de software devem ser direcionadas aos processos de desenvolvimento de software para esclarecimento e correção.

2.3.3 Análises e revisões de software

Revisões e análises são aplicadas aos resultados dos processos de desenvolvimento de software e ao processo de verificação de software. Conceitualmente, a diferença entre revisão e análise é que o objetivo básico da análise é identificar várias evidências da corretude do software, enquanto a revisão tem como alvo avaliar qualitativamente a corretude do software.

A revisão consiste na inspeção dos documentos de saída de um processo apoiada por um *checklist* ou outro recurso similar. Já com relação à análise, pode examinar detalhes referente à funcionalidade, desempenho, rastreabilidade e implicações de segurança do componente de software relacionadas a outros componentes existentes no sistema ou em um equipamento embarcado crítico.

Análises e revisões dos requisitos de software de alto nível

O objetivo destas revisões e análises é identificar e relatar desvios nos requisitos os quais, em geral, foram introduzidos durante o processo de requisitos de software. Essas revisões e análises visam garantir que os requisitos de alto nível satisfazem os seguintes objetivos:

1. Conformidade com os requisitos do sistema: objetiva assegurar que as funções do sistema para serem desempenhadas por um software precisam ser definidas como funcionais, de desempenho ou requisitos relacionados a aspectos de segurança do sistema que são satisfeitos por requisitos de software de alto nível;

2. Exatidão e consistência: visam assegurar que cada requisito de alto nível é exato, não-ambíguo, suficientemente detalhado e nenhum requisito conflita com os outros requisitos;
3. Compatibilidade com o computador de implantação: objetiva assegurar que não existe conflito entre os requisitos de alto nível e os componentes de hardware/ software do computador alvo da implantação;
4. Verificabilidade: objetiva assegurar que cada requisito de alto nível pode ser verificado;
5. Conformidade com os padrões: visa assegurar que os padrões de requisitos de software foram atendidos durante o processo de requisitos e que desvios aos padrões foram justificados;
6. Rastreabilidade: visa assegurar que requisitos funcionais, de desempenho e relacionados a aspectos de segurança do sistema alocados ao software são mapeáveis aos requisitos de software de alto nível;
7. Aspectos algorítmicos: objetivam assegurar a exatidão e o comportamento proposto pelo algoritmo.

Análises e revisões dos requisitos de software de baixo nível

O objetivo destas revisões e análises é identificar e relatar desvios ou inconsistências nos requisitos que podem ter sido introduzidos durante o processo de projeto de software. Essas revisões e análises visam garantir que os requisitos de baixo nível satisfazem os seguintes objetivos:

1. Conformidade com os requisitos de software de alto nível: objetiva assegurar que os requisitos de software de baixo nível satisfazem os requisitos de software de alto nível e que requisitos derivados e projeto básico são corretamente definidos;
2. Exatidão e consistência: visa assegurar que cada requisito de baixo nível é exato, não-ambíguo e nenhum requisito conflita com os outros requisitos;
3. Compatibilidade com o computador de implantação: objetiva assegurar que não existe conflito entre os requisitos de software e os componentes de hardware/ software do computador alvo da implantação;
4. Verificabilidade: objetiva assegurar que cada requisito de baixo nível pode ser verificado;
5. Conformidade com os padrões: visa assegurar que os padrões de projeto de software foram atendidos durante o processo de projeto de software e que desvios aos padrões foram justificados;
6. Rastreabilidade: visa assegurar que requisitos de software de alto nível e requisitos derivados deram origem aos requisitos de software de baixo nível;

7. Aspectos algorítmicos: objetivam assegurar a exatidão e o comportamento proposto pelo algoritmo.

Análises e revisões da arquitetura de software

O objetivo destas revisões e análises é identificar e relatar defeitos que podem ter sido introduzidos durante o processo de desenvolvimento da arquitetura de software. Essas revisões e análises visam garantir que a arquitetura de software satisfaz os seguintes objetivos:

1. Conformidade com os requisitos de software de alto nível: objetiva assegurar que a arquitetura de software não conflita com os requisitos de alto nível, especialmente funções que asseguram a integridade do sistema;
2. Consistência: visa assegurar um relacionamento correto entre os componentes da arquitetura de software. Este relacionamento existe por meio do fluxo de dados e fluxo de controle;
3. Compatibilidade com o computador de implantação: objetiva assegurar que não existe conflito, especialmente na inicialização, operação assíncrona, sincronização e interrupção e entre a arquitetura de software e os componentes de hardware/ software do computador alvo da implantação;
4. Verificabilidade: objetiva assegurar que a arquitetura de software pode ser verificada;
5. Conformidade com os padrões: visa assegurar que os padrões de projeto de software foram atendidos durante o processo de projeto de software e que desvios aos padrões foram justificados;
6. Integridade de particionamento: o objetivo é assegurar que as violações do particionamento são impedidas ou isoladas.

Análises e revisões do código fonte

Visa identificar e relatar defeitos que foram introduzidos no processo de codificação de software. Essas revisões e análises confirmam que os documentos de saída do processo de codificação de software são corretos, completos e podem ser verificados. A principal preocupação desse tipo de análise e revisão se refere à corretude do código em relação ao documento de requisitos de software, à arquitetura de software e à conformidade com o padrão de codificação de software. Pode-se atender os objetivos dessa atividade da seguinte forma:

1. Conformidade com os requisitos de software de baixo nível: visa assegurar que o código fonte é correto e completo comparado com os requisitos de software de baixo nível;

2. Conformidade com a arquitetura de software: objetiva assegurar que o código fonte coincide com o fluxo de dados e com o fluxo de controle definidos na arquitetura de software;
3. Verificabilidade: visa assegurar que o código fonte não contém sentenças e estruturas que não podem ser verificadas e que o código não foi alterado para ser testado;
4. Conformidade com os padrões: objetiva assegurar que o padrão de codificação de software foi seguido durante o desenvolvimento do código, especialmente em sistemas críticos cujas restrições de complexidade e restrições do código devem ser consistentes com os objetivos desses sistemas;
5. Rastreabilidade: objetiva assegurar que os requisitos de software de baixo nível resultaram no respectivo código fonte;
6. Corretude e consistência: visa determinar o nível de corretude e consistência do código fonte, incluindo o uso da pilha de execução, *overflow* do ponto aritmético fixado e resolução, manipulação de exceção, uso de variável não inicializada ou constante, variável não usada ou constante, corrupção de dados devido à execução de uma tarefa ou interrupção de conflitos.

Análises e revisões dos documentos de saída do processo de integração

Visa assegurar que os resultados do processo de integração são completos e corretos. Isso pode ser feito a partir de um exame datalhado dos dados de *linking* e *loading* e mapeamento da memória. Neste tópico pode-se incluir os seguinte alvos:

1. Endereço de hardware incorreto;
2. Sobreposição da memória;
3. Ausência de componente de software.

Análises e revisões dos casos de teste, procedimentos e resultados

O objetivo destas revisões e análises é assegurar que o teste do código foi executado de forma correta e completa. Segue abaixo atividades que visam apoiar esses objetivos:

1. Casos de teste: a verificação dos casos de teste é exposta na Seção 2.3.4;
2. Procedimentos de teste: visa verificar que os casos de teste foram corretamente desenvolvidos a partir dos procedimentos de teste e dos resultados esperados;
3. Resultados de teste: visa assegurar que os resultados de teste estão corretos e que discrepâncias entre o resultado obtido e o esperado foram explicadas.

2.3.4 Processo de teste de software

O teste em sistemas embarcados críticos tem alguns objetivos complementares ao processo de verificação de software. Um dos objetivos visa demonstrar que o referido software satisfaz os correspondentes requisitos. O outro objetivo tem como alvo demonstrar com um alto grau de confiança, que os defeitos que poderiam levar a condições de falha inaceitáveis, tal como determinado pelo processo de avaliação do sistema de segurança, foram removidos.

Na Figura 2.3 é apresentado um diagrama do processo de teste de software, que ilustra o relacionamento entre os três tipos de teste e sugere uma sequência de execução por meio de caminhos diretos e/ou alternativos. Segue abaixo os três tipos de teste e os respectivos objetivos:

- Teste de baixo nível: verificar a implementação dos requisitos de software de baixo nível;
- Teste de integração de software: verificar o inter-relacionamento entre os requisitos de software e os componentes, também verificar a implementação dos requisitos de software e componentes de software com a arquitetura de software;
- Teste de integração de hardware/software: verificar a correta operação do software no computador de implantação.

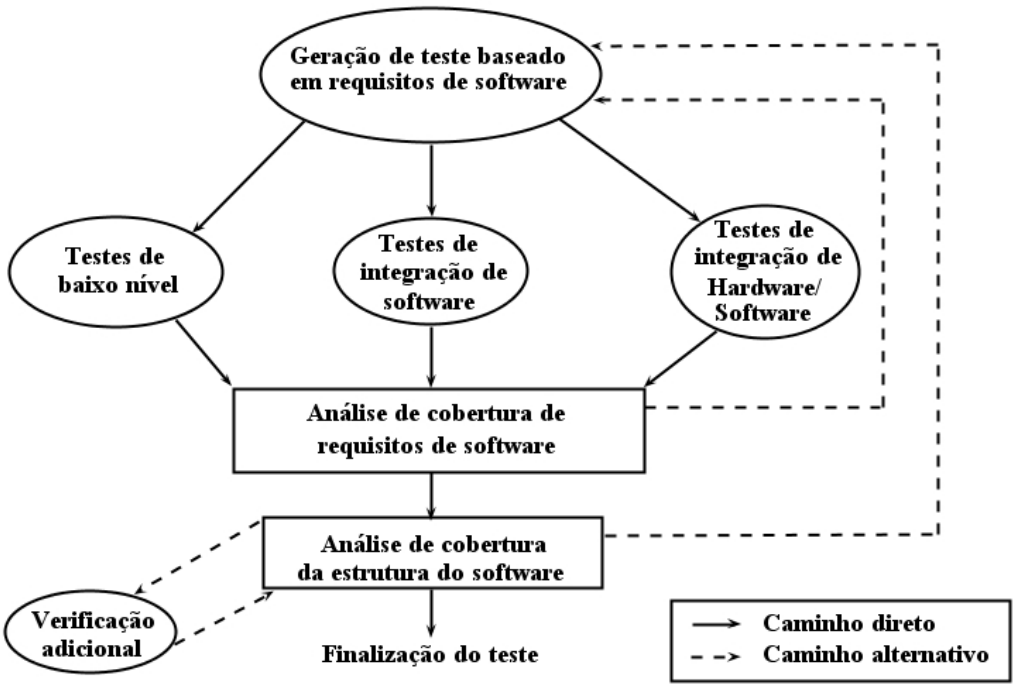

Figura 2.3: *Processo de teste de software (adaptada de (RTCA/EUROCAE, 1992))*

Para satisfazer os objetivos do teste de software:

1. Casos de teste devem basear-se essencialmente nos requisitos de software;
2. Casos de teste devem ser desenvolvidos para verificar funcionalidades corretas e para estabelecer condições que resultem em falhas potenciais;
3. A análise de cobertura de requisitos de software deve determinar quais requisitos de software não foram testados;
4. A análise de cobertura estrutural deve determinar quais estruturas do software não foram executadas.

Ambiente de teste

Para atender a demanda do teste de software, pode ser necessário mais de um ambiente de teste. Um ambiente ideal de teste de software inclui a configuração do software no computador de implantação e a execução do teste no ambiente de computador de implantação cuja simulação apresenta um alto grau de fidelidade.

A certificação do software pode ser concedida para testes realizados em um emulador de computador de implantação ou um simulador do computador. Segue uma orientação para o ambiente de teste:

1. Os testes selecionados devem ser realizados em um ambiente de computador de implantação, enquanto falhas forem identificadas durante a execução nesse ambiente.

Seleção de caso de teste baseado em requisitos

Teste baseado em requisitos é uma boa alternativa para apoiar a efetiva identificação de defeitos. Seguem orientações para seleção de casos de teste baseado em requisitos:

1. Para implementar os objetivos do teste de software, duas categorias de casos de teste podem ser incluídas: casos de teste de intervalo normal (testes positivos) e casos de teste de robustez (testes negativos);
2. Casos de teste específicos devem ser desenvolvidos a partir de requisitos de software tendo em vista que defeitos no código aparecem inerentemente aos processos de desenvolvimento de software.

Casos de teste de intervalo normal

O objetivo dos casos de teste de intervalo normal é demonstrar a habilidade do software em responder a entradas e a condições normais, que é atendido da seguinte forma:

1. Variável de entrada inteira e real deve ser exercitada usando classe de equivalência e valores limite;

2. Para funções em relação ao tempo, como filtros, integradores e atrasadores, múltiplas iterações do código devem ser desempenhadas para avaliar as características dentro do contexto;
3. Para transições de estado, casos de teste devem ser desenvolvidos para exercitar as possíveis transições durante a operação normal;
4. Para requisitos de software que foram definidos utilizando equações lógicas, casos de teste de intervalo normal devem verificar o uso das variáveis e os operadores booleanos.

Casos de teste de robustez

O objetivo dos casos de teste de robustez é demonstrar a habilidade do software em responder a entradas e condições anormais, que e atendido da seguinte forma:

1. Variável de entrada inteira e real deve ser exercitada usando a seleção de classe de equivalência de valores inválidos;
2. Sistema de inicialização deve ser executado durante condições anormais;
3. Os modos de falha possíveis dos dados de entrada devem ser determinados;
4. Para laços de iteração nos quais tem-se contadores de laço cujo os valores são calculados, casos de teste devem ser desenvolvidos para tentar calcular valores dos contadores de laço que estejam fora do intervalo previsto no respectivo laço;
5. Uma verificação deve ser feita para assegurar que os mecanismos de proteção para as estruturas de dados são respondidos corretamente quando se tentar exceder os limites das mesmas;
6. Para funções em relação ao tempo, como filtros, integradores e atrasadores, casos de teste devem ser desenvolvidos para provocar um *overflow* no mecanismo de proteção aritmética;
7. Para transições de estado, casos de teste devem ser desenvolvidos para conduzir a transições que não são permitidas pelos requisitos de software.

Métodos de teste baseado em requisitos

Métodos de teste baseados em requisitos são subdivididos em três métodos: integração de hardware/software baseado em requisitos, teste de integração de software baseado em requisitos e teste de baixo nível baseado em requisitos. Seguem abaixo detalhes dos três métodos:

1. Teste de integração de hardware/software baseado em requisitos: Este método de teste deve concentrar-se em possíveis defeitos cuja fonte esteja associada com a operação do software no ambiente de implantação e com as funcionalidades de alto nível. Os defeitos típicos revelados por este método são:

- Manipulação de interrupção incorreta;
- Falha para atender requisitos de tempo de execução;
- Resposta incorreta do software para transições do hardware ou falhas do hardware;
- Barramento de dados e outros recursos de contenção apresentam problemas, por exemplo, mapeamento de memória;
- Incapacidade para construção do teste visando a detecção de falhas;
- Falhas na interface hardware/software;
- Comportamento incorreto dos laços de iteração;
- Controle incorreto do gerenciamento da memória do hardware ou de outro dispositivo de hardware sob o controle do software;
- Estouro de pilha (*Stack overflow*);
- Mecanismo de operação usado incorretamente para confirmar a corretude e compatibilidade dos campos carregáveis do software;
- Violação do particionamento do software.

2. Teste de integração de software baseado em requisitos: O objetivo deste método é assegurar que os componentes de sofware interagem corretamente com os outros e satisfaz os requisitos de software e a arquitetura de software. Os defeitos típicos revelados por este método são:
   - Inicialização incorreta das variáveis e constantes;
   - Falhas durante a passagem de parâmetro;
   - Dados corrompidos (especialmente dados globais);
   - Resolução numérica inadequada;
   - Sequenciamento e operação de eventos incorreto.
3. Teste de baixo nível baseado em requisitos: Objetiva assegurar que cada componente de software satisfaz os seus respectivos requisitos de software de baixo nível. Os defeitos típicos revelados por este método são:
   - Falha no algoritmo para satisfazer um requisito de software;
   - Operação do laço de iteração incorreto;
   - Decisão lógica incorreta;
   - Falha para processar corretamente combinações de condições de entrada;
   - Resposta incorreta para ausência do dado de entrada ou fornecido de forma incorreto;
   - Tratamento incorreto das exceções;
   - Sequência de processamento incorreto;
   - Precisão do algoritmo inadequada.

Análise de cobertura de teste

A análise de cobertura de teste é composta de duas etapas: análise de cobertura baseado em requisitos e análise de cobertura estrutural. Na primeira, analisam-se casos de teste em relação aos requisitos de software para verificar se os casos de teste satisfazem o critério especificado. Já na segunda etapa, casos de teste são definidos com o objetivo de exercitar a estrutura do código. Orientações adicionais, por exemplo, referente ao código inativo, são discutidas na Seção 2.3.4.

Análise de cobertura de teste baseado em requisitos

O objetivo desta análise é determinar o quanto dos requisitos de software implementados foram verificados com o teste baseado em requisitos. A análise pode revelar a necessidade de utilizar casos de teste adicionais. Esta análise pode mostrar que:

1. Existem casos de teste para cada requisito de software;
2. Casos de teste satisfazem o critério de teste de intervalo normal e de robustez, conforme definido na Seção 2.3.4.

Análise de cobertura estrutural

O objetivo desta análise é determinar o quanto da estrutura do código não foi exercitada pelo teste baseado em requisitos. Esta análise pode mostrar que:

1. A análise deve confirmar o grau de cobertura estrutural apropriada para o nível do software;
2. A análise de cobertura estrutural pode ser realizada utilizando o código fonte. Entretanto, para software classificado no nível A e para os quais o compilador gera código objeto que não é mapeável diretamente para as sentenças do código fonte, verificação adicional deve ser realizada considerando também o código objeto. Compiladores que geram automaticamente código de verificação de limites de vetores é um exemplo de código objeto não rastreável no código fonte e, desse modo, teste de cobertura no código objeto é necessário para verificar se tais partes do código foram testadas;
3. A análise deve demonstrar o acoplamento dos dados e o acoplamento de controle entre os componentes do código.

Resolução de análise de cobertura estrutural

A análise de cobertura estrutural pode revelar parte do código fonte não exercitada durante a realização dos testes. Em virtude disso, a resolução pode requerer atividade-

des adicionais do processo de verificação de software. A parte do código fonte que não foi exercitada, pode ser resultado de:

1. Procedimentos ou casos de teste insuficientes: os casos de teste devem ser suplementados ou os procedimentos de teste devem ser alterados para aumentar a cobertura do código, podendo assim, ser revistos os métodos de análise de cobertura baseado em requisitos;
2. Inadequações nos requisitos de software: os requisitos de software devem ser modificados e adicionalmente desenvolver casos de teste e executar os referidos procedimentos de teste;
3. Código morto: o código morto deve ser removido e uma análise posterior do código deverá ser realizada;
4. Código desativado: para o código desativado que somente será executado em determinadas configurações no ambiente de implantação, a configuração operacional para execução normal deste código deve ser definida e casos de teste e procedimentos específicos devem ser desenvolvidos para satisfazer os objetivos de corbetura exigidos.

2.4 Processo de certificação

O objetivo do processo de certificação é estabelecer uma comunicação e entendimento entre o requente e a autoridade certificadora de todo o ciclo de vida do software para apoiar o processo de certificação (definido na Seção 9 da DO-178B) (RTCA/EUROCAE, 1992).

O processo de certificação é aplicado conforme previsto no processo de planejamento de software e no plano para certificação de aspectos do software (Seção 11.1 da DO-178B).

2.4.1 Planejamento e meios de cumprimento

O requerente propõe meios de cumprimento do que está definido para o desenvolvimento do sistema da aeronave ou equipamento que servirá de base para certificação. O plano para certificação dos aspectos de software (Seção 11.1 da DO-178B) define os aspectos do software do sistema da aeronave ou equipamento dentro do contexto de cumprimento. Este plano também apresenta os níveis do software determinados pelo processo de avaliação de segurança do sistema. O requerente deverá:

- Apresentar o plano para certificação dos aspectos do software e outros dados solicitados para a autoridade certificadora para realizar revisão quando os efeitos de mudanças são mínimos;

- Resolver questões identificadas pela autoridade certificadora relativo à certificação de aspectos do software;
- Obter acordo com a autoridade certificadora com relação ao plano para certificação de aspectos de software.

2.4.2 Comprovação de conformidade

O requerente disponibiliza evidências de que os processos de ciclo de vida satisfazem os planos de software. A autoridade certificadora realiza revisões dos documentos e/ou artefatos, podendo dessa forma envolver os requerentes ou fornecedores nas discussões. O requerente organiza estas revisões das atividades dos processos de ciclo de vida de software e torna disponível dados do ciclo de vida do software quando necessário. O requerente deverá:

- Resolver questões levantadas pela autoridade certificadora durante as revisões;
- Apresentar o resumo de realização de software (Seção 11.20 da DO-178B) e o índice de configuração de software (Seção 11.16 da DO-178B) para a autoridade certificadora;
- Apresentar ou tornar disponível outros dados ou evidências requisitads pela autoridade certificadora.

2.4.3 Mínimo de dados do ciclo de vida de software que é apresentado à autoridade certificadora

O mínimo de dados do ciclo de vida de software que é apresentado à autoridade certificadora é:

- Plano para certificação de aspectos de software;
- Índice de configuração de software;
- Resumo dos dados referente à implementação do software (características do software, histórico de mudanças, ciclo de vida do software e outros dados).

2.4.4 Dados do ciclo de vida de software relacionados com o tipo de projeto

Salvo decisão em contrário da autoridade de certificação, a regulamentação relativa à recuperação e aprovação dos dados do ciclo de vida de software relacionada com o tipo de projeto aplica-se a:

- Dados dos requisitos de software;

- Descrição do projeto;
- Código fonte;
- Código objeto executável;
- Índice de configuração de software;
- Resumo dos dados referente à implementação do software.

2.5 Considerações finais

Neste capítulo procurou-se apresentar alguns aspectos gerais de sistemas embarcados críticos no contexto da DO-178B. Embora tenha-se discutido a respeito do ciclo de vida de software, a DO-178B não define nenhum modelo de ciclo de vida específico, apenas apresenta condutas que o processo de ciclo de vida deve atender, tornando assim a DO-178B bastante flexível, tendo em vista que cada organização pode definir o seu próprio processo de ciclo de vida de acordo com o nível de exigência que a respectiva condição falha do produto desenvolvido exige (VINCENZI, 2009).

Na Seção 2.1 foram descritos os principais processos da DO-178B e seus respectivos objetivos. Foram apresentados, de forma geral, os objetivos de alguns dos processos da DO-178B, como não é definido na DO-178B a forma de se implementar os respectivos processos, a empresa deve definir como os processos serão implementados. Na literatura existem algumas propostas, uma delas propõe implementar os processos da DO-178B utilizando o OPENUP (*Open Unified Process*), que se trata de um projeto aberto derivado do *Unified Process* (UP) (BERTRAND; FUHRMAN, 2008). O OPENUP procura disponibilizar um modelo de processo que contém os elementos fundamentais do desenvolvimento de software.

Na Seção 2.2 são discutidos alguns aspectos dos processos de ciclo de vida do sistema necessário para o entendimento dos processos de ciclo de vida do software.

Na Seção 2.3 foi apresentado outro processo integral, processo de verificação de software. As atividades de Validação e Verificação (V&V) visam dar condições para que a maneira como o software está sendo construído como o produto de software em si estejam em conformidade com o especificado (DELAMARO et al., 2007).

Nessa seção discutiram-se tanto as atividades dinâmicas como as estáticas. Ficou claro que o teste baseado em requisitos se trata justamente do teste funcional, conhecido como teste caixa preta, cujo objetivo é utilizar entradas específicas (casos de teste) e testar o software para verificar se as saídas geradas estão em conformidade com o esperado no documento de requisitos de software. Já a análise de cobertura estrutural se trata do teste estrutural (teste caixa branca), cujo objetivo é definir requisitos de teste considerando a implementação do software para executar partes ou componentes elementares do software (DELAMARO et al., 2007).

Na Seção 2.4 foi apresentado o último processo integral, processo de certificação. Na Seção 12 da DO-178B apresentam-se considerações adicionais sobre os aspectos de certificação de software, qualificação de ferramentas de software e métodos para atingir os objetivos definidos nas seções anteriores da DO-178B.

CAPÍTULO 3

Protocolo de uma Revisão Sistemática na Área de V&V para Sistemas Embarcados Críticos

A revisão sistemática da literatura (muitas vezes mencionada como uma revisão sistemática) é o meio de identificar, avaliar e interpretar todas as pesquisas disponíveis relevantes para uma questão de pesquisa específica, ou área temática, ou fenômeno de interesse. Estudos individuais que contribuem para uma revisão sistemática são chamados de estudos primários; uma revisão sistemática é uma forma de estudo secundário (KITCHENHAM et al., 2007).

Há muitas razões para a realização de uma RS. Os motivos mais comuns são:

- Para resumir as evidências existentes em relação a um tratamento ou tecnologia, por exemplo, resumir evidência empírica dos benefícios e limitações de um método ágil específico;
- Para identificar as eventuais lacunas existentes na pesquisa atual, a fim de sugerir futuras áreas de investigação;
- Para fornecer uma visão geral/subsídios que permitem avançar o conhecimento na investigação de novas áreas.

No entanto, revisões sistemáticas também podem ser utilizadas para examinar a extensão em que a evidência empírica suporta/contradiz hipóteses teóricas, ou até mesmo para auxiliar a geração de novas hipóteses (KITCHENHAM et al., 2007).

Neste capítulo é apresentado na Seção 3.1 a metodologia da revisão sistemática com as respectivas etapas. Em seguida, na Seção 3.2 é apresentado o planejamento da RS. Finalmente, na Seção 3.3 são feitas as considerações finais sobre este capítulo.

3.1 Metodologia da Revisão Sistemática

O planejamento da revisão sistemática foi realizado de acordo com o modelo de protocolo apresentado por Biolchini et al. (2007), cujos passos são ilustrados na Figura 3.1. Como pode ser observado na Figura 3.1, a primeira etapa do processo de

desenvolvimento da revisão sistemática, Formulação da Questão de Pesquisa, defini-se questões que se espera que sejam esclarecidas, definindo assim o escopo da pesquisa.

Na sequência, deve ser definido na etapa Qualidade e Amplitude da Questão elementos básicos que visam definir uma questão de pesquisa bem formulada. Já com a definição de Estratégia de Busca para Seleção de Estudos Primários, procura-se estabelecer características para avaliar o local onde os estudos primários serão buscados.

A partir da consolidação das etapas anteriores, deve-se na etapa de Execução de Busca Piloto definir uma string de busca, para a(s) respectiva(s) fonte(s) escolhida(s), com o objetivo de executá-la para realização de uma avaliação inicial dos estudos primários retornados. Após isso, deve-se definir os critérios de inclusão e exclusão de estudos primários na etapa Critério e Procedimento para Seleção dos Estudos.

Já na etapa Processo de Seleção dos Estudos Primários são definidos os processos de seleção preliminar e final. Por fim, na etapa Estratégias de Extração e Sumarização dos Resultados, define-se os dados que serão coletados de cada estudo primário selecionado, e após a conclusão desta atividade define-se a forma como serão organizados/sumarizados os dados.

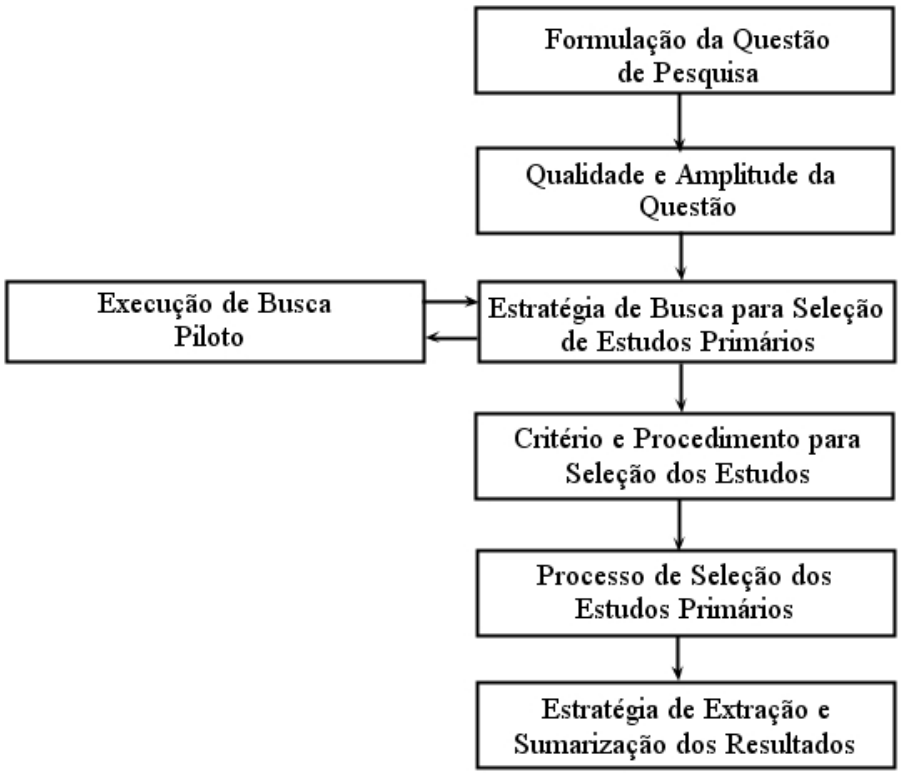

Figura 3.1: *Etapas da revisão sistemática*

3.2 Planejamento

O planejamento da revisão sistemática foi realizado de acordo com o modelo de protocolo apresentado por Biolchini et al. (2007). Nesta seção, são apresentados os principais pontos do plano elaborado.

3.2.1 Formulação da Questão de Pesquisa

Objetivou-se procurar respostas para as seguintes questões neste trabalho de revisão sistemática:

- **Questão de Pesquisa Primária 1:** Quais técnicas e critérios de teste de software tem sido investigados para o teste de software de sistemas embarcados críticos?
  - Questão de Pesquisa Secundária 1.1: Quais padrões e normas tem sido investigados para o teste de software de sistemas embarcados críticos?
- **Questão de Pesquisa Primária 2:** Quais são as evidências de que os estudos primários selecionados estão relacionados (aderentes) com os objetivos e atividades do processo de teste de software definido na DO-178B?
  - Questão de Pesquisa Secundária 2.1: Quais são as evidências de que os objetivos e atividades do processo de teste de software definidos na DO-178B proporcionam um alto nível de qualidade em sistemas embarcados críticos?
- **Questão de Pesquisa Primária 3:** Qual tem sido a força (*strength*) das evidências em apoio as conclusões apresentadas?

3.2.2 Qualidade e Amplitude da Questão

Uma questão de pesquisa bem formulada é composta pelos seguintes elementos:

Palavras-chaves e Sinônimos

Foram consideradas como palavras-chaves da língua inglesa as seguintes palavras:

1. critical embedded, safety-critical, mission-critical, embedded software
2. software test, system test

Intervenção

No contexto desta revisão, foram observados processos de teste de software, técnicas e critérios de teste de software aplicados no contexto de sistemas embarcados críticos.

Controle

Foram identificados artigos relevantes para o contexto deste trabalho e que serviram como artigos de controle da string de busca, conforme consta na Tabela 3.1. Caso a expressão de busca retornasse todos estes artigos, então haveria indícios de que estaria adequada.

Tabela 3.1: *Artigos de Controle*

#	**Título**	**Autor(es)**	**Base de Dados**
AC1	An industrial case study of structural testing applied to safety-critical embedded software	(GUAN et al., 2006)	ACM
AC2	A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions	(YU; LAU, 2006)	ACM
AC3	Combining unit-level symbolic execution and system-level concrete execution for testing nasa software	(PăSăREANU et al., 2008)	ACM
AC4	Testing to certify an embedded software system	(DAVIDSON; AMOUSSOU, 2010)	ACM
AC5	“Fly Me to the Moon”: Verification of Aerospace Systems	(GIANNAKOPOULOU, 2010)	IEEE
AC6	A practical approach to modified condition/decision coverage	(HAYHURST; VEERHUSEN, 2001)	IEEE
AC7	An empirical evaluation of the MC/DC coverage criterion on the HETE-2 satellite software	(DUPUY; LEVESON, 2000)	IEEE
AC8	Development of a Framework for Automated Systematic Testing of Safety-Critical Embedded Systems	(KANDL et al., 2006)	IEEE

População

O grupo observado foi o de pesquisadores e desenvolvedores de software que trabalham no projeto e construção de sistemas embarcados críticos.

Resultados

Atividades de Verificação e Validação (V&V) de software, metodologia de teste de software, técnicas e critérios de teste de software aplicados no contexto de sistemas embarcados críticos.

Aplicação

Projetos de desenvolvimento de software aplicados no contexto de sistemas embarcados críticos.

3.2.3 Estratégia de Busca para Seleção de Estudos Primários

A estratégia de busca e seleção dos estudos primários foi definida de acordo com as fontes de estudos, palavras-chave, idioma e os tipos de estudos primários selecionados para a revisão:

Critério de seleção das fontes

Bases de dados eletrônicas indexadas e máquinas de busca eletrônica.

Métodos de busca de fontes

Manual e máquina de busca na web.

Listagem de fontes

Bases de dados eletrônicas indexadas IEEE Xplore (IEEE) e ACM Digital Library (ACM), visto que referem-se a duas das principais bases de dados na área de computação.

Tipo dos estudos primários

Listas de referência de estudos primários, periódicos, relatórios técnicos, trabalhos em andamento e *proceedings* de conferências.

Idioma dos estudos primários

Inglês por essa ser a língua internacionalmente aceita para a redação de trabalhos científicos.

3.2.4 Execução de Busca Piloto

A partir das questões de pesquisa, dos seus respectivos atributos de qualidade e amplitude e estratégia de busca para seleção de estudos primários, definiu-se uma string de busca para uma base eletrônica indexada, visando executá-la na respectiva base para realização de uma avaliação inicial dos estudos primários retornados e caso seja necessário fazer adequações nas etapas anteriores.

3.2.5 Critérios e Procedimento para Seleção dos Estudos

Critérios de inclusão

Os seguintes critérios de inclusão de trabalhos foram definidos:

1. Aplicação de atividades de V&V (estáticas e/ou dinâmicas) de software no contexto de sistemas embarcados críticos (CI1);
2. Aplicação de técnicas e critérios de teste (dinâmicas) de software em sistemas embarcados críticos (CI2).

Critérios de exclusão

Os seguintes critérios de exclusão de trabalhos foram definidos:

1. Aplicação de atividades de V&V de software no contexto de sistemas embarcados não-críticos, por exemplo, aplicações para celulares (CE1);
2. Aplicação de técnicas e critérios de teste de software em sistemas embarcados não-críticos (CE2);
3. Não aborda atividades de V&V de software ou técnicas e critérios de teste de software no contexto de sistemas embarcados críticos (CE3);
4. Documento completo não encontrado (CE4);
5. Não caracteriza-se como um documento, por exemplo, se trata apenas de uma descrição de um evento científico (CE5);
6. Artigo repetido, mas com o código DOI (*Digital Object Identifier*) diferente ou pertencente a uma base de dados eletrônica indexada (ou fonte) distinta (CE6).

3.2.6 Processo de Seleção dos Estudos Primários

Processo de seleção preliminar

Nesta etapa, foram construídas strings de busca formadas pela combinação dos sinônimos das palavras-chaves identificadas. Essas strings foram utilizadas para se realizar as respectivas consultas nas máquinas de busca mencionadas (IEEE e ACM).

Os estudos primários obtidos por meio das respectivas consultadas, foram analisados pelos revisores (autor e orientador deste trabalho de dissertação), os quais foram responsáveis pela leitura dos títulos e dos resumos dos trabalhos, identificado a relevância de um trabalho (de acordo com pelo menos um dos critérios de inclusão), e existindo consenso entre os revisores, o referido trabalho foi selecionado para ser lido na íntegra. Não existindo consenso, o trabalho foi colocado em uma lista de espera, para definição futura pelos revisores.

Processo de seleção Final

Foi realizada a leitura completa dos trabalhos selecionados na etapa de seleção preliminar por pelo menos um dos revisores, que redigiu um documento com o resumo, metodologia e técnicas de teste mencionadas no trabalho e outros conceitos relacionados.

Avaliação de qualidade dos estudos primários

Os estudos selecionados resultante da execução do processo de seleção dos estudos primários foram avaliados pelos pesquisadores envolvidos de acordo com os critérios de qualidade definidos por Ali et al. (2010):

1. Existe uma razão específica que motivou a realização do estudo?
2. Existe uma descrição adequada do contexto (por exemplo indústria, laboratório, produtos utilizados e etc) em que a pesquisa foi realizada?
3. Existe uma justificativa e uma descrição para o projeto de pesquisa?
4. O pesquisador explicou como a amostra do estudo (participantes ou casos) foi identificada e selecionada e qual foi a justificativa para essa seleção?
5. Está claro como os dados foram coletados (por exemplo, por meio de entrevistas, formulários, observação, ferramentas e etc)?
6. O estudo fornece uma descrição e justificativa dos métodos de análise de dados utilizados?
7. Existem dados suficientes que foram apresentados com o objetivo de sustentar as conclusões?
8. Há uma declaração clara dos resultados?
9. O pesquisador analisou criticamente o seu próprio papel, o viés potencial e influência na formulação de questões de investigação, o recrutamento da amostra, coleta de dados, análise e seleção de dados para apresentação?
10. Os autores discutem a credibilidade dos seus resultados?
11. As limitações do estudo foram discutidas explicitamente?

Kitchenham et al. (2007) afirmam que os pesquisadores responsáveis pela condução da RS podem incluir uma terceira etapa no processo de seleção dos estudos primários, que é baseada no detalhamento dos critérios de qualidade dos respectivos estudos. Além dos critérios de inclusão/exclusão definidos na Seção 3.2.5, considera-se fundamental avaliar a qualidade dos estudos primários como uma maneira de se detalhar mais os critérios de inclusão/exclusão.

Considerando isso, a avaliação da qualidade de cada estudo definiu sua exclusão ou inclusão na lista de estudos que foi utilizada para extrair os dados. Ao avaliar cada estudo segundo os critérios acima, foi obtido uma correspondente nota final numa escala

de 0 a 11 pontos, sendo que cada questão foi pontuada da seguinte forma: se a resposta foi "Sim"(1 ponto), se foi "Não" (0 ponto) e se foi "Parcialmente" (0.5 ponto). Ao final da avaliação, foram excluídos estudos que eram "Muito fraco" (0 a 2.4 pontos) ou "Fraco" (2.5 a 4.4 pontos), restando assim apenas estudos "Regular" (4.5 a 5.9 pontos), "Bom" (6 a 8.4 pontos) e "Muito Bom" (8.5 a 11 pontos).

3.2.7 Estratégias de Extração e Sumarização dos Resultados

Para cada estudo primário selecionado, foi utilizado a ferramenta JabRef para armazenar os dados (ALVER et al., 2008). Segue abaixo os dados que foram coletados:

- Fonte de dados;
- Técnica de teste de software enfatizada;
- Critério de teste de software;
- Objetivos da DO-178B;
- Tipo de estudo experimental;
- Resumo.

Sumarização dos resultados

Os resultados coletados foram organizados de maneira tabular.

3.2.8 Força das evidências

A força geral de um corpo de evidência é normalmente referido como a força da evidência. Uma análise da força da evidência é muito importante para que os leitores de uma RS tenham condições de identificar o grau de confiança que se pode colocar nas conclusões e recomendações resultantes dessas revisões (ALI et al., 2010) (DYBÅ; DINGSØYR, 2008b).

Existem diversos sistemas para avaliar a força das evidências, porém neste trabalho foi escolhido o GRADE (*Grading of Recommendations Assessment, Development and Evaluation*), devido as definições prevista no GRADE abordar a maioria das fragilidades dos sistemas de classificação de evidências baseado em hierarquia e também por ser utilizado por outros pesquisadores em engenharia de software (ALI et al., 2010).

O GRADE define quatro graus de força das evidências: alta, moderada, baixa e muito baixa (conforme Tabela 3.2). A força das evidências é determinada por meio da combinação de quatro elementos: características do estudo, qualidade do estudo, consistência e objetividade (*directness*).

Tabela 3.2: *Definições utilizadas para classificar a força das evidências*

Grau	**Definição**
Alta	Pesquisas futuras são muito improváveis que mude a confiança na estimativa do efeito
Moderada	Pesquisas futuras são susceptíveis que provoque um impacto importante sobre a confiança na estimativa do efeito, podendo assim alterar a estimativa
Baixa	Pesquisas futuras são muito susceptíveis que provoque um impacto importante sobre a confiança na estimativa do efeito e é susceptível que altere a estimativa
Muito baixa	Qualquer estimativa do efeito é muito incerto

3.3 Considerações finais

Neste capítulo foi apresentado o protocolo da revisão sistemática na área de V&V para sistemas embarcados críticos. No qual cada uma das etapas foram apresentadas em detalhes. Diferentemente do que acontece em uma revisão bibliográfica tradicional, coleta e interpretação dos estudos selecionados de maneira informal e subjetiva, na RS tem-se como objetivo apresentar uma avaliação justa a respeito de um tópico de pesquisa, fazendo uso de uma metodologia de revisão que seja confiável, rigorosa e que permita auditagem (replicabilidade) (KITCHENHAM, 2004).

CAPÍTULO 4

Evolução do Teste de Software em Sistemas Embarcados Críticos por meio da Revisão Sistemática

Neste capítulo são apresentadas algumas evidências empíricas coletadas durante a condução da RS. Inicialmente, na Seção 4.1 é apresentado uma síntese dos trabalhos selecionados na RS. Em seguida, na Seção 4.2 é apresentado a análise dos trabalhos selecionados em relação às questões de pesquisa. Finalmente, na Seção 4.3 são feitas as considerações finais sobre este capítulo.

4.1 Síntese dos trabalhos selecionados

Os trabalhos selecionados na revisão sistemática são agrupados conforme as técnicas de teste de software exploradas e requisitos do processo de teste da DO-178B apresentados na Seção 2.3.4.

Conforme a Tabela 4.1, pode-se observar que alguns trabalhos exploram mais de uma técnica de teste em conjunto. Já outros, por sua vez, não exploram uma técnica específica, concentrando-se em propor estratégias de teste. Com relação aos requisitos do processo de teste da DO-178B explorados (Requisito DO-178B), foram abreviados da seguinte maneira: Casos de teste de intervalo normal (CTN), Casos de teste de robustez (CTR), Métodos de teste baseado em requisitos (MTR), Análise de cobertura de teste baseado em requisitos (ACR) e Análise de cobertura estrutural (ACE).

Tabela 4.1: *Súmario dos estudos primários selecionados, adaptado de (FERRARI; MALDONADO, 2007)*

Estudo Primário	Fonte	Técnica Explorada	Estudo Experimental	Requisito DO-178B
(ABDUL-BAKI et al., 2000)	IEEE	Teste funcional	não se aplica	ACR
(ALAGAR et al., 2003)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
Continua na próxima página...				

Tabela 4.1: *Súmario dos estudos primários selecionados, adaptado de (FERRARI; MALDONADO, 2007)*

Estudo Primário	Fonte	Técnica Explorada	Estudo Experimental	Requisito DO-178B
(ANDREWS; ZHANG, 2000)	IEEE	Teste baseado em modelos	*Survey*	não enfatizam
(ANGELETTI et al., 2009)	ACM	Teste estrutural	Experimento	ACE
(AWEDIKIAN et al., 2009)	ACM	Teste estrutural	Experimento	ACE
(AUGUSTON et al., 2005a)	ACM	Teste baseado em modelos	Estudos de caso	não enfatizam
(AUGUSTON et al., 2005b)	ACM	Teste baseado em modelos	Estudo de caso	não enfatizam
(BELL; BRAT, 2008)	IEEE	Teste estrutural	Estudos de caso	ACE
(BARBIER; BELLOIR, 2003)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(BHATEJA, 2009)	ACM	Teste baseado em modelos	não se aplica	não enfatizam
(BHOGARAJU et al., 2000)	IEEE	Teste funcional	Estudo de caso	ACR
(BLACKBURN; BUSSER, 1996)	IEEE	Teste funcional	não se aplica	não enfatizam
(BRITT, 1994)	IEEE	Teste funcional	Estudo de caso	CTN e ACR
(CARPENTER, 1999)	ACM	Teste funcional	Estudo de caso	CTN, CTR e MTR
(CHANG et al., 1997)	IEEE	Teste funcional	Estudo de caso	CTN, CTR e MTR
(CHEON et al., 2004)	IEEE	Teste funcional	Estudo de caso	MTR e ACR
(CHEON et al., 2005)	IEEE	não enfatizam	Estudo de caso	não enfatizam
(CLANCY et al., 2006)	ACM	Teste funcional	Estudos de caso	MTR
(COULTER, 1999)	IEEE	Teste funcional, teste estrutural e teste de mutação	não se aplica	CTN, CTR, MTR, ACR e ACE
(CULLYER; STOREY, 1994)	IEEE	não enfatizam	*Survey*	não enfatizam
(DAVIDSON; AMOUSSOU, 2010)	ACM	Teste funcional	Estudo de caso	não enfatizam
(DENG; SANG, 2008)	ACM	Teste funcional	Experimento	MTR
(DO et al., 2006)	ACM	Teste de mutação	Estudo de caso	não enfatizam
(DONG et al., 2009)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(DRUSINSKY et al., 2007)	ACM	Teste baseado em modelos	Experimento	não enfatizam
(DRUSINSKY et al., 2006)	ACM	Teste baseado em modelos	Experimento	não enfatizam

Continua na próxima página...

Tabela 4.1: *Súmario dos estudos primários selecionados, adaptado de (FERRARI; MALDONADO, 2007)*

Estudo Primário	Fonte	Técnica Explorada	Estudo Experimental	Requisito DO-178B
(DUPUY; LEVESON, 2000)	IEEE	Teste funcional e teste estrutural	Estudo de caso	CTN, CTR, ACR e ACE
(KUMAR et al., 2010)	IEEE	Teste funcional e teste estrutural	Experimento	CTN, CTR, ACR e ACE
(LOUBACH et al., 2006)	IEEE	Teste funcional e teste estrutural	Estudos de caso	CTN, CTR, MTR, ACR e ACE
(MORTON, 1999)	IEEE	Teste funcional e teste estrutural	não se aplica	ACR e ACE
(ESSER; STRUSS, 2007)	ACM	Teste baseado em modelos	Estudo de caso	não enfatizam
(GOTLIEB, 2009)	ACM	Teste estrutural	Estudo de caso	ACE
(GROSS et al., 2009)	ACM	Teste estrutural	Estudos de caso	ACE
(GUAN et al., 2006)	ACM	Teste estrutural	Estudo de caso	ACE
(HAUSE et al., 2010)	IEEE	Teste baseado em modelos	Estudos de caso	não enfatizam
(GIANNAKOPOULOU, 2010)	IEEE	Teste estrutural	Estudos de caso	ACE
(HEIMDAHL, 2007)	IEEE	não enfatizam	*Survey*	não enfatizam
(HU, 2004)	ACM	Teste baseado em modelos	Experimento	ACR
(KLOOS; ESCHBACH, 2010)	ACM	Teste baseado em modelos	não se aplica	não enfatizam
(KLOOS; ESCHBACH, 2009)	ACM	Teste baseado em modelos	Estudo de caso	não enfatizam
(KANDL et al., 2006)	IEEE	Teste baseado em modelos e teste estrutural	Estudo de caso	ACE
(JAW et al., 2008)	IEEE	Teste baseado em modelos	Experimento	ACE
(JASKE et al., 1996)	IEEE	Teste funcional	Estudo de caso	CTN, CTR e MTR
(KROPP et al., 1998)	IEEE	Teste funcional	Experimento	CTN e CTR
(LI et al., 1999)	IEEE	Teste funcional	não se aplica	CTN e CTR
(JOUNG et al., 2009)	IEEE	não enfatizam	não se aplica	não enfatizam
(QIAN; ZHENG, 2009)	IEEE	não enfatizam	Estudo de caso	não enfatizam
(LAKEHAL; PARISSIS, 2005a)	ACM	Teste estrutural	Experimento	ACE
(LAKEHAL; PARISSIS, 2005b)	ACM	Teste estrutural	Experimento	ACE
(LAKEHAL; PARISSIS, 2007)	ACM	Teste estrutural e teste de mutação	Estudo de caso	ACE

Continua na próxima página...

Tabela 4.1: *Súmario dos estudos primários selecionados, adaptado de (FERRARI; MALDONADO, 2007)*

Estudo Primário	Fonte	Técnica Explorada	Estudo Experimental	Requisito DO-178B
(LöFFLER et al., 2010)	ACM	Teste baseado em modelos	não se aplica	ACE
(MORSCHHAUSER; LINDVALL, 2007)	IEEE	Teste baseado em modelos	Estudo de caso	ACE
(NAKAO; ESCHBACH, 2008)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(MATHAIKUTTY et al., 2007)	ACM	Teste estrutural	Estudo de caso	CTN, ACR e ACE
(MCDONALD et al., 2001)	ACM	Teste funcional e teste estrutural	Estudo de caso	ACE
(MALEKZADEH; AINON, 2010)	IEEE	Teste funcional	Estudo de caso	ACR
(NADEEM et al., 2006)	IEEE	Teste funcional	não se aplica	CTN e CTR
(NEBUT et al., 2006)	ACM	Teste funcional e teste estrutural	Estudos de caso	CTN, CTR, ACR e ACE
(PăSăREANU et al., 2008)	ACM	Teste estrutural	Estudo de caso	ACE
(PAPAILIOPOULOU, 2008)	ACM	Teste estrutural	não se aplica	ACE
(PEREZ; KAISER, 2009)	IEEE	Teste funcional	Estudo de caso	MTR
(RYU et al., 2008)	ACM	não enfatizam	não se aplica	não enfatizam
(REZA et al., 2008)	ACM	Teste funcional	Estudo de caso	MTR
(SANTHANAM, 2001)	ACM	Teste estrutural	Estudo de caso	ACE
(SANTIAGO et al., 2008a)	ACM	Teste funcional	Experimento	não enfatizam
(SANTIAGO et al., 2008b)	ACM	Teste baseado em modelos	Estudos de caso	não enfatizam
(SWARUP; RAMAIAH, 2008)	IEEE	não enfatizam	Experimento	não enfatizam
(TSAI et al., 2003)	IEEE	Teste funcional	Estudo de caso	não enfatizam
(TOMMASO et al., 2005)	ACM	Teste funcional	não se aplica	CTN e CTR
(TSAI et al., 2000)	ACM	Teste funcional	Experimento	CTN e CTR
(TSAI et al., 2002)	ACM	Teste funcional	Estudo de caso	ACR e MTR
(TSAI et al., 2005)	ACM	Teste funcional	Estudo de caso	ACR
(TRACEY et al., 2002)	ACM	Teste funcional, teste estrutural e teste de mutação	Estudos de caso	ACE
(TU et al., 1998)	ACM	Teste funcional	não se aplica	CTN e CTR
(TUDOR et al., 2004)	IEEE	Teste funcional	Experimento	não enfatizam

Continua na próxima página...

Tabela 4.1: *Súmario dos estudos primários selecionados, adaptado de (FERRARI; MALDONADO, 2007)*

Estudo Primário	Fonte	Técnica Explorada	Estudo Experimental	Requisito DO-178B
(VIEIRA et al., 2008)	ACM	Teste baseado em modelos	Estudo de caso	ACR
(ZHENG et al., 2008)	ACM	Teste baseado em modelos	Estudo de caso	não enfatizam
(XU; WU, 1999)	ACM	Teste funcional	Estudo de caso	CTN, CTR, MTR e ACR
(YOON et al., 2005)	IEEE	não enfatizam	Estudo de caso	não enfatizam
(YIN; LIU, 2009)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(YU; LAU, 2006)	ACM	Teste estrutural	Experimento	ACE
(YU; LAU, 2004)	ACM	Teste estrutural	Experimento	ACE
(YU et al., 2009)	ACM	Teste funcional	Estudo de caso	CTN, CTR, MTR e ACR
(YIN et al., 2010)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(TRAKHTENBROT, 2005)	IEEE	Teste baseado em modelos	Estudo de caso	ACE
(TU; WU, 1999)	IEEE	Teste baseado em modelos	Estudo de caso	ACR
(TUMMALA et al., 2006)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(YU et al., 2010)	IEEE	Teste baseado em modelos	Estudo de caso	não enfatizam
(HAYHURST; VEERHUSEN, 2001)	IEEE	Teste estrutural	não se aplica	ACE
(RAYADURGAM; HEIMDAHL, 2001)	IEEE	Teste estrutural	Estudo de caso	ACE
(SMITH; URI, 2004)	IEEE	Teste estrutural	não se aplica	ACE
(SANTO, 1988)	IEEE	Teste funcional	Estudo de caso	ACR
(WU; HUANG, 2003)	IEEE	Teste funcional	Estudo de caso	ACR
(WU; LI, 1999)	IEEE	Teste funcional	Estudo de caso	ACR
(XIE, 2004)	IEEE	Teste funcional	não se aplica	não enfatizam
(WANG, 2004)	IEEE	Teste funcional e teste estrutural	não se aplica	CTN, CTR, MTR e ACE
(WANG; TAN, 2005)	IEEE	Teste funcional e teste estrutural	não se aplica	CTN, CTR, MTR e ACE

Os trabalhos apresentados na próxima seção estão agrupados de acordo com os requisitos do processo de teste da DO-178B. Observa-se que alguns trabalhos abordam mais de um requisito em conjunto. Cada trabalho ou conjunto de trabalhos relacionados é apresentado em um parágrafo.

4.1.1 Casos de teste de intervalo normal

Chang et al. (1997) apresentam um estudo de caso que enfatiza o processo de verificação por meio de métodos formais e inspeção em todas as fases do processo de desenvolvimento. É definida uma estratégia de geração de casos de teste, na qual um espaço de teste de um determinado componente a ser testado é multidimensional, sendo que cada dimensão representa a variação de elemento de entrada ou uma condição de uso, no qual cada ponto no espaço representa um possível caso de teste. Mas no estudo, um subdomínio de teste é especificado a partir de um subconjunto do espaço multidimensional.

Jaske et al. (1996) descrevem um estudo de caso que aplica o critério de particionamento em classes de equivalência válidas, que descreve experiências no processo de teste de um complexo sistema de segurança crítica médica, um sistema de planejamento de tratamento braquiterapia.

Kropp et al. (1998) apresentam um estudo que explora a aplicação da metodologia *Ballista* para o teste de componentes COTS (*Commercial Off-The-Shelf*). Nesta metodologia é estabelecido um estado inicial do sistema, após isto é realizado uma chamada ao MuT (*Module under Test*) e avaliado a resposta do sistema a partir de um conjunto de entradas válidas.

4.1.2 Casos de teste de robustez

Jaske et al. (1996) apresentam um estudo de caso que aplica o critério de particionamento em classes de equivalência inválidas.

Kropp et al. (1998), ao utilizar a metodologia *Ballista* é estabelecido um estado inicial do sistema, após isto é realizado uma chamada ao MuT (*Module under Test*) e avaliado a resposta do sistema a partir de um conjunto de entrada inválida.

4.1.3 Métodos de teste baseado em requisitos

Chang et al. (1997) mencionam a realização do teste de integração, mas não é especificado se foi realizado o teste de integração de hardware/software ou o teste de integração de componentes de software.

Cheon et al. (2004) apresentam um ciclo de vida para as atividades de verificação e validação de software, no qual são definidos o teste de integração de hardware/software e o teste de integração de componentes de software.

Perez e Kaiser (2009) apresentam uma metodologia de teste de nível de integração que utiliza como referência o Modelo V de ciclo de vida de software. Esta metodolo-

gia de teste foi baseada na reutilização de casos de teste entre os diferentes níveis de teste, incluindo o comportamento de teste e adaptadores de teste.

4.1.4 Análise de cobertura de teste baseado em requisitos

Tu e Wu (1999) apresentam um *survey* que discute a injeção de falhas no teste funcional no contexto de um ambiente de simulação.

Tanto Britt (1994) como Abdul-Baki et al. (2000) apresentam trabalhos que relatam os processos e métodos empregados para definir uma especificação formal utilizando a linguagem de especificação de requisitos RSML (*Requirements State Machine Language*) para definir uma versão de um sistema de tráfego aéreo.

Malekzadeh e Ainon (2010) apresentam um gerador de casos de teste automático que recebe uma especificação em uma linguagem e obtém as causas e os efeitos de forma automática, permitindo assim gerar os respectivos casos de teste a partir do grafo causa-efeito.

4.1.5 Análise de cobertura estrutural

Angeletti et al. (2009) apresentam uma proposta que objetiva avaliar a cobertura do código de um sistema de Gestão de Tráfego Ferroviário. Os autores propõem a utilização de um núcleo principal conhecido como CBMC (*Bounded Model Checker for ANSI-C*) para apoiar a realização do processo de teste. É apresentada uma metodologia para geração automática de casos de teste, visando atingir uma cobertura estrutural do software de 100%, utilizando o critério Todas-Arestas. Esta metodologia inicialmente instrumenta o código (inserindo sentenças), depois o módulo CBMC é executado iterativamente até atingir uma cobertura de 100%, sendo que para cada iteração é ativado um conjunto de sentenças.

Awedikian et al. (2009) apresentam um experimento no qual empregam-se algoritmos genéticos. Uma função *fitness* inspirada no encadeamento de geração de casos de teste é definida para gerar de forma eficiente casos de teste que satisfaçam o critério MC/DC (*modified condition/decision coverage*).

Gotlieb (2009) apresenta uma ferramenta de teste baseada em restrição para verificação de software de segurança crítica desenvolvido na linguagem C, Euclide. A ferramente oferece os seguintes recursos: geração de casos de teste estrutural, geração de contra-exemplo e prova parcial da corretude de procedimentos do software. Objetiva satisfazer os critérios Todos-Nós e Todas-Arestas utilizando técnicas simbólicas e numéricas, chamadas de SSA (*Single Static Assignment form*).

Gross et al. (2009) apresentam uma avaliação de um protótipo de uma ferramenta de teste de software evolucionário aplicado na indústria automobilística. Cada um dos

quatro estudos de caso é composto de um módulo do software da indústria implementado na linguagem C, para cada estudo utilizou-se o protótipo ETF (*EvoTest Framework*) no qual avaliou-se o atendimento aos requisitos de teste do critério Todas-Arestas.

Guan et al. (2006) apresentam uma análise comparativa entre os casos de teste elaborados pela empresa (teste funcional manual) e os casos de teste que atendem os requisitos de teste do critério CACC (*Correlated Active Clause Coverage*), critério baseado em lógica. Este critério é equivalente à versão mascarada (*masking*) do critério MC/DC (AMMANN et al., 2003). Observou-se que ao empregar o critério CACC, identificaram-se grandes falhas críticas de segurança que não foram detectadas pelo teste funcional e, também com relação ao custo (esforço) necessário para satisfazer o critério CACC, observou-se que não é necessariamente superior aos critérios de teste funcional.

Lakehal e Parissis (2005a) apresentam a ferramenta Lustructu, que realiza a análise da cobertura estrutural dos programas implementados em Lustre (linguagem declarativa de fluxo de dados síncrono). Os critérios de cobertura, que também são apresentados por Lakehal e Parissis (2005b), são definidos a partir do conceito de ativação de condição, que se refere à condição sob a qual um fluxo de dados presente na aresta de entrada de um operador é transferido para a aresta de saída do operador. São definidos três critérios de cobertura: básica de n-caminho (assegura a ativação de todos i-caminhos com $i \leq n$); condição elementar de n-caminho (a variação da avaliação de um caminho de entrada torna-se obrigatória) e múltiplas condições de n-caminho (o resultado do caminho depende de todas as combinações das arestas, inclusive das aresta internas). Já Papailiopoulou (2008) apresenta um ambiente gráfico baseado na linguagem Lustre, SCADE (*Safety Critical Application Development Environment*), que permite a definição hierárquica dos componentes do sistema e a geração automática de código na linguagem C; os critérios de cobertura definidos para a linguagem Lustre devem ser extendidos para o ambiente SCADE, sendo então apresentado no trabalho esta extensão. Lakehal e Parissis (2007) apresentam um estudo de caso que ilustra a aplicação dos critérios de cobertura, sendo avaliado a respectiva aplicabilidade e utilidade dos mesmos.

Mathaikutty et al. (2007) apresentam um *framework* integrado para o desenvolvimento guiado por modelo; validação de projetos de sistemas que utilizam ESTEREL (linguagem de programação síncrona) e SystemC (conjunto de classes e macros na linguagem C++ que disponibiliza um Kernel de simulação guiado por evento em C++) e para a geração de casos de teste objetivando o atendimento de critérios de cobertura tradicionais (Todos-Nós, Todas-Arestas e o MC/DC).

Păsăreanu et al. (2008) apresentam uma abordagem para o teste de software no contexto de segurança crítica que emprega de forma combinada a execução simbólica no nível de unidade e a execução concreta no nível de sistema para apoiar a geração de casos de teste que satisfazem critérios de teste específicos do usuário (Todas-Arestas, regras de

vôo e combinação de regras de vôo/abortar).

Santhanam (2001) apresenta um módulo de automação do processo de teste de software desenvolvido na linguagem ADA em conjunto com a ferramenta TSE (*Test Set Editor*), que consiste em um conjunto de software proprietários que trabalham de forma integrada visando o atendimento dos seguintes critérios de cobertura estrutural: Todos-Nós e Todas-Arestas.

Yu e Lau (2004) apresentam uma discussão teórica e empírica a respeito dos critérios de teste. No trabalho, afirma-se que o critério de cobertura de condição (CC) e cobertura de decisão (DC), não são adequados para softwares de segurança crítica. Critérios mais rigorosos devem ser utilizados, sendo o critério MC/DC exigido pela indústria de aviação comercial para realização de certificação. Também identificou-se que alguns conjuntos de falhas não são detectadas por este critério, sendo necessário critérios mais rigorosos. Observou-se que o critério MUMCUT (no qual um conjunto de casos de teste satisfaz os requisitos de teste de todos os critério MUTP - *multiple unique true point*, MNFP - *multiple near false point* e CUTPNFP - *corresponding unique true point and near false point pair*) se mostram mais rigorosos do que o MC/DC. Já Yu e Lau (2006) propõe o critério M-CC (*multiple-condition coverage*) que se mostra mais rigoroso do que o MC/DC e o MUMCUT. A Figura 4.1 apresenta a relação de inclusão entre esses critérios de teste (vide Apêndice A para visualizar as definições dos critérios). Uma vez que a relação de inclusão não reflete necessariamente a habilidade dos critérios de teste em detectar falhas (FRANKL; WEYUKER, 1993; ZHU, 1996), Yu e Lau (2006) avaliaram os referidos critérios por meio de experimentação e obtiveram a seguinte relação entre os critérios em termos da capacidade dos mesmos em detectar falhas CC < DC < C/DC $\ll$ MC/DC < CUTPNFP < MUMCUT $\lessapprox$ M-CC, onde o símbolo $\ll$ significa que a detecção de falhas não é total, ou seja, alguns defeitos detectados pelos critérios CC, DC e C/DC não foram detectados pelo MC/DC; e o símbolo $\lessapprox$ indica que, apesar do critério M-CC detectar todos os defeitos detectados pelos critérios abaixo dele, seu custo de aplicação é extremamente elevado, sendo o critério MUMCUT mais recomendado tendo em vista que este é capaz de detectar quase todos os defeitos a um custo bem menor.

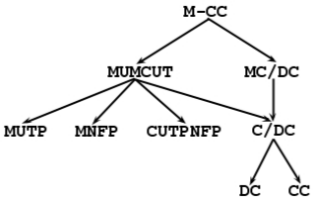

Figura 4.1: *Hierarquia de inclusão (adaptada de (YU; LAU, 2006)).*

McDonald et al. (2001) exploram questões referente ao isolamento de módulos a partir de um ambiente de tempo real, melhoria da integração do teste no ambiente de desenvolvimento, automação e planejamento de testes. Sendo realizado também uma

análise da cobertura estrutural segundo os critérios Todos-Nós e B/DC (*branch/decision coverage*), sendo este portanto um critério menos rigoroso do que o C/DC.

Nebut et al. (2006) apresentam três estudos de caso que são avaliados com relação aos seguintes critérios de teste: Todos-Nós e Todas-Arestas.

Tracey et al. (2002) apresentam um *framework* que utiliza algoritmos genéticos para gerar dados de teste, estudos de caso são avaliados com relação ao critério de teste Todas-Arestas e MC/DC.

Löffler et al. (2010) apresentam uma metodologia que emprega especificação formal para descrever cenários de casos de uso e derivar modelos de teste a partir de transformações automáticas de modelos, tendo assim um modelo de teste para cada caso de uso. Os modelos são avaliados segundo os seguintes critérios de teste: Todas-Arestas, Todos-Nós, Todos Caminhos Básicos e Todos os Parametros.

Jaw et al. (2008) apresentam um processo de desenvolvimento de software embarcado crítico, cujo os processos de V&V devem ser apoiados por um conjunto de técnicas, *technique spectrum*, que são classificadas em: informal, estática, dinâmica e formal. Com relação às técnicas dinâmicas, é recomendado a realização do teste de cobertura utilizando o critério MC/DC.

Morschhauser e Lindvall (2007) apresentam uma ferramenta Reactis (uma ferramenta de software que automatiza o teste de modelos Simulink), sendo portanto empregado como critério de teste o MC/DC.

Trakhtenbrot (2005) ressalta um estudo de caso que utiliza a ferramenta *Statemate ModelChecker*, cuja a principal ideia é utilizar conhecimento sobre como executar o irreproduzível que viola uma propriedade *P* do sistema, de forma que o sistema seja configurado para acessar apenas as propriedades diferentes de P. O trabalho menciona a realização do teste estrutural, mas não menciona qual critério é aplicado.

Kandl et al. (2006) salientam um *framework* para a automação do teste de software embarcado crítico, que é apoiado pelo teste baseado em modelo, modelo este que é obtido automaticamente a partir do código fonte, sendo portanto os casos de teste obtidos a partir do modelo. Os casos de teste gerados devem atender os seguintes critérios de teste: Todas-Arestas, Todos-Nós e o critério MC/DC.

Bell e Brat (2008) apresentam estudos de caso que utilizam dois métodos formais para verificação automática do software, análise estática e verificação de modelos (*model checking*). No trabalho é utilizado o critério de teste Todas-Arestas.

Giannakopoulou (2010) evidencia estudos de caso que empregam duas abordagens: técnica de verificação exaustiva e o uso de verificação modelos para gerar casos de teste automaticamente. No trabalho é empregado o critério de teste MC/DC.

Hayhurst e Veerhusen (2001) discutem-se alguns passos práticos para avaliar produtos de software no contexto de aviação segundo o critério de teste MC/DC, também é

apresentado alguns enganos típicos que são comuns quando se emprega o critério MC/DC.

Rayadurgam e Heimdahl (2001) apresentam como um verificador de modelos pode ser utilizado para geração automática de teste que pode ser representado como um modelo de estado finito. É discutido como o verificador de modelos pode ser utilizado para gerar sequência de casos de teste para atender o critério MC/DC.

Smith e Uri (2004) apresentam um estudo relacionado à testabilidade de produtos COTS (*commercial-off-the-shelf*), sendo também avaliado o software segundo o critério de teste MC/DC.

Dupuy e Leveson (2000) apresentam um estudo de caso no qual os casos de teste gerados para satisfazer o critério de teste MC/DC detectaram importantes defeitos não detectados pelo teste funcional. Também observou-se que com o uso do critério de teste MC/DC detectou-se defeitos não identificados por critérios de teste estrutural de nível inferior na hierarquia de inclusão.

Kumar et al. (2010) apresentam um experimento no qual é avaliado a cobertura do atendimento do critério de teste de intensidade de segurança (*safety depth of testing*), também chamado de critério Todos-Possíveis-Caminhos/Condições (*all possible paths/-conditions*).

Loubach et al. (2006) apresentam estudos de caso, os quais são avaliados segundo o critério de complexidade de McCabe, é mencionado também a realização da análise de cobertura estrutural do software, mas não menciona qual(is) critério(s) de teste é(são) empregado(s) para a realização desta análise.

Morton (1999) discute a importância do uso dos critérios de teste MC/DC, Todos-Nós e Todas-Arestas.

Wang (2004) e Wang e Tan (2005) discutem a relevância do emprego dos seguintes critérios de teste: DC, CC, C/DC, M-CC e Todos-Nós.

Coulter (1999), com a utilização da metodologia *Graybox* (que combina teste funcional e teste estrutural com prova de corretude de programas e teste de mutação), propõe o uso de atividades de verificação em conjunto com a análise de cobertura de caminhos para avaliar o escore da conclusão do teste.

4.2 Análise dos trabalhos selecionados em relação às questões de pesquisa

As Tabelas 4.2, 4.3, 4.5, 4.6, 4.7 e 4.8 sintetizam os dados de 97 estudos primários, referentes às quantidades parciais em cada uma das bases de dados eletrônicas indexadas (IEEE e ACM), total de estudos (n) e o percentual total de estudos (%). Para visualizar mais detalhes sobre a condução da revisão sistemática, vide o Apêndice B.

4.2.1 Características dos estudos

Tipo de estudo experimental

Na Tabela 4.2 são apresentadas informações quantitativas sobre o tipo de estudo experimental empregado nos trabalhos selecionados, classificação esta realizada segundo a terminologia definida por (DYBÅ; DINGSØYR, 2008a). Sendo que, Estudos de Caso tratam-se de trabalhos que apresentaram mais de um caso (projeto). Ao examinar a tabela, constata-se que 59.79% dos estudos são de natureza observacional (Estudo de caso e Estudos de caso), indicando portanto que a maioria dos estudos são resultado da realização de monitoramento de um ou mais projetos em profundidade.

Tabela 4.2: *Tipo de estudo experimental*

Estudo Experimental	**IEEE**	**ACM**	**Quantidade**	%
Estudo de caso	28	20	48	49.48
Estudos de caso	4	6	10	10.31
Experimento	5	12	17	17.53
Survey	3	0	3	3.09
não se aplica	12	7	19	19.59
Total	52	45	97	100
Média	10.4	9	19.4	-
Desvio padrão	9.35	6.69	15.37	-

Escopo de atuação dos estudos

Na Tabela 4.3 são apresentadas informações quantitativas sobre o escopo de atuação dos estudos selecionados. Ao examinar a tabela, constata-se uma predominância de estudos realizados pela academia/laboratório (70.1%).

Tabela 4.3: *Escopo de atuação dos estudos*

Escopo	**IEEE**	**ACM**	**Quantidade**	%
Indústria	14	15	29	29.9
Academia/Laboratório	38	30	68	70.1
Total	52	45	97	100
Média	26	22.5	48.5	-
Desvio padrão	16.97	10.61	27.58	-

Dígrafo de citação interna

Para os estudos primários que referencia um ou mais estudos pertencentes ao conjunto selecionado pela RS, é possível construir um grafo direcionado (dígrafo) e identificar seus respectivos grau de entrada e saída. Na Tabela 4.4 são apresentadas

informações referente ao respectivo digrafo, já na Figura 4.2 é apresentada a representação gráfica do grafo direcionado, construído com o auxílio da ferramenta de desenho de grafos, *Hints for Graph Drawing* (NASCIMENTO, 2003).

O grau de entrada neste dígrafo corresponde ao total de vezes em que o referido estudo foi citado. Esta métrica sugere estudos que são mais relevantes no contexto da respectiva RS, ao observar a Figura 4.2 constatou-se que o estudo EP11 é o mais relevante (mais citado pelos estudos primários), observa-se também que o mesmo corresponde a um dos artigos de controle definido no planejamento da revisão sistemática (vide Seção 3.2.2 localizada na Página 43), sugerindo assim também que o conjunto de artigos de controle é de boa qualidade.

Como pode ser observado na Figura 4.2, o dígrafo é disjunto, anti-simétrico (se não existe um par (u, v) de vértices tal que $u \rightarrow v$ e $v \rightarrow u$) e tem oito grupos de estudos primários separados. O Grupo 4 possui o maior número de estudos interrelacionados, sendo que o estudo EP21 é uma evolução do EP22 e ambos foram escritos pelos mesmos autores. Já no Grupo 1, o estudo EP23 é uma evolução dos estudos EP16 e EP17, sendo que os três foram escritos pelos mesmos autores e discutem sobre a cobertura estrutural de programas escritos em Lustre, enquanto o EP20 é um estudo mais recente escrito por um autor diferente que discute uma outra linha de pesquisa: geração automática de casos de teste para programas Lustre/SCADE. Com relação ao restante dos grupos, para cada um deles, tem-se um estudo (grau de saída igual a um) que é resultado da evolução de um outro estudo (grau de entrada igual a um).

Tabela 4.4: *Dígrafo dos estudos primários selecionados*

Identificador	Estudo	Grau de saída	Grau de entrada
EP1	(HEIMDAHL, 2007)	1	0
EP2	(MORSCHHAUSER; LINDVALL, 2007)	1	0
EP3	(TU; WU, 1999)	1	0
EP4	(TUMMALA et al., 2006)	1	0
EP5	(GIANNAKOPOULOU, 2010)	1	0
EP6	(HAYHURST; VEERHUSEN, 2001)	1	0
EP7	(RAYADURGAM; HEIMDAHL, 2001)	0	1
EP8	(SMITH; URI, 2004)	1	0
EP9	(WU; HUANG, 2003)	1	0
EP10	(WU; LI, 1999)	0	1
EP11	(DUPUY; LEVESON, 2000)	0	7
EP12	(AUGUSTON et al., 2005a)	0	1
EP13	(DRUSINSKY et al., 2007)	1	0
EP14	(DRUSINSKY et al., 2006)	0	1

Continua na próxima página...

Tabela 4.4: *Dígrafo dos estudos primários selecionados*

Identificador	Estudo	Grau de saída	Grau de entrada
EP15	(GUAN et al., 2006)	1	1
EP16	(LAKEHAL; PARISSIS, 2005a)	0	2
EP17	(LAKEHAL; PARISSIS, 2005b)	0	2
EP18	(MATHAIKUTTY et al., 2007)	1	0
EP19	(PăSăREANU et al., 2008)	0	1
EP20	(PAPAILIOPOULOU, 2008)	2	0
EP21	(YU; LAU, 2006)	2	0
EP22	(YU; LAU, 2004)	1	1
EP23	(LAKEHAL; PARISSIS, 2007)	2	0
EP24	(DAVIDSON; AMOUSSOU, 2010)	1	0
EP25	(TU et al., 1998)	0	1

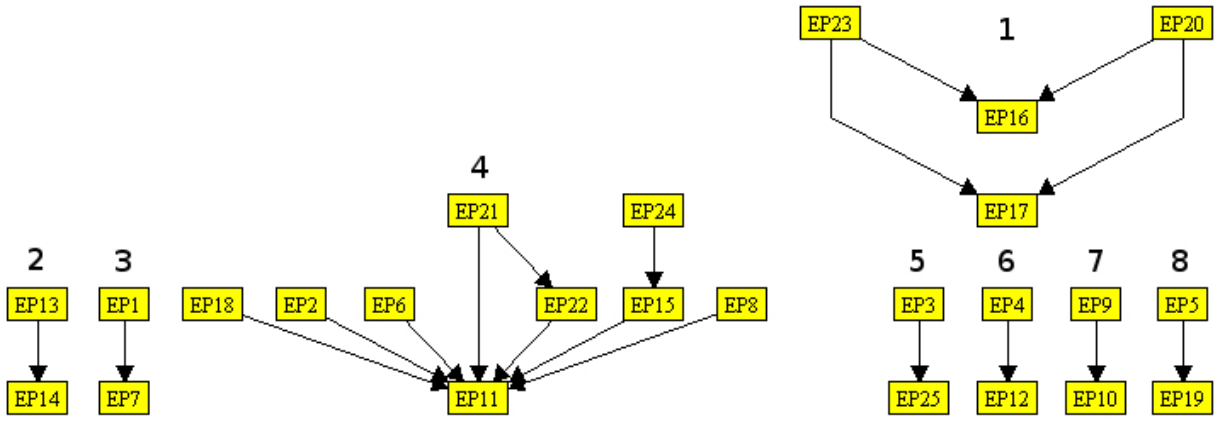

Figura 4.2: *Grafo direcionado das citações entre os estudos*

4.2.2 Técnicas de teste de software exploradas

A Tabela 4.5 apresenta as técnicas de teste de software exploradas pelos estudos. Caso um deles abordasse mais de uma técnica, por exemplo quantidade (q) igual a 3, para cada técnica seria atribuído um valor igual a 0.33 (q^{-1}). Ao examiná-la, constata-se que a técnica de teste funcional é a mais utilizada, (36.77%), em seguida tem-se o teste baseado em modelos (28.36%), que possibilita eliminar ambiguidades e viabiliza a derivação de casos de teste a partir do modelo. Yu e Lau (2006) apresentam o critério de cobertura de transição baseado em requisitos de segurança, como uma nova alternativa para a técnica de teste baseado em modelos, tendo em vista que tradicionalmente tem-se utilizado esta técnica com enfoque em requisitos funcionais.

Tabela 4.5: *Técnicas de teste de software exploradas*

Técnica Explorada	**IEEE**	**ACM**	**Quantidade**	%
Teste baseado em modelos	14.5	13	27.5	28.36
Teste de mutação	0.33	1.83	2.16	2.23
Teste estrutural	8.83	14.83	23.66	24.4
Teste funcional	21.33	14.33	35.66	36.77
Não enfatizam	7	1	8	8.25
Total	51.99	44.99	96.98	100
Média	10.4	9	19.4	-
Desvio padrão	7.1	6.23	12.45	-

Assim, o primeiro quesito da Questão de Pesquisa Primária 1 (definida na Seção 3.2.1 localizada na Página 42) foi respondido, deixando evidente que todas as técnicas de teste de software foram exploradas pelos estudos primários.

4.2.3 Critérios de teste de software explorados

A Tabela 4.6 apresenta os critérios de teste de software explorados pelo estudos, ao examiná-la constata-se que o critério funcional particionamento em classes de equivalência é o mais utilizado, (12.37%). Já com relação aos critérios de teste estrutural, o MC/DC é o mais utilizado, (10.23%). O restante dos critérios estruturais exigidos pela DO-178B foram explorados da seguinte forma: DC (0.68%) e SC (5.82%), contemplando assim o segundo quesito da Questão de Pesquisa Primária 1.

Tabela 4.6: *Critérios de teste de software explorados*

Critério Explorado	**IEEE**	**ACM**	**Quantidade**	%
C/DC	0.33	0.34	0.67	0.69
CACC	0	1	1	1.03
CC	0.32	0.34	0.66	0.68
CCC	0.16	0	0.16	0.17
Cobertura de caminhos lógicos	0.33	0	0.33	0.34
Cobertura de decisões/arestas	0	0.5	0.5	0.52
Cobertura de estados	0.5	0.5	1	1.03
Cobertura de parâmetros	0	0.25	0.25	0.26
Cobertura de transições	1.5	0.5	2	2.06
Cobertura do potenciais caminhos/condições	0.5	0	0.5	0.52
Critério de McCabe	1.5	0.25	1.75	1.81
CUTPNFP	0	0.14	0.14	0.14
D/CC	0.32	0	0.32	0.33
DC	0.32	0.34	0.66	0.68
M-CC	0	0.14	0.14	0.14
MC/DC	8.41	1.5	9.91	10.23

Continua na próxima página...

Tabela 4.6: *Critérios de teste de software explorados*

Critério Explorado	**IEEE**	**ACM**	**Quantidade**	%
MCC	0.16	0	0.16	0.17
MUMCUT	0	0.34	0.34	0.35
N-caminhos básicos	0	1.32	1.32	1.36
N-caminhos com condições múltiplas	0	1.32	1.32	1.36
N-caminhos de condições elementares	0	1.32	1.32	1.36
Nível 1	0	0.5	0.5	0.52
Nível 2	0	0.5	0.5	0.52
Nível 3	0	0.5	0.5	0.52
Nível 4	0	0.5	0.5	0.52
Particionamento em classes de equivalência	7.99	4	11.99	12.37
Todas as arestas (AE)	0	0.2	0.2	0.21
Todas as precondições (APT)	0	0.2	0.2	0.21
Todas-Arestas em código fonte	0.91	5.91	6.82	7.04
Todas-Transições	0	1	1	1.03
Todos os casos de usos instaciados (AIUC)	0	0.2	0.2	0.21
Todos os nós (AV)	0	0.2	0.2	0.21
Todos os vértices e todos os casos de usos instanciados (AV-AIUC)	0	0.2	0.2	0.21
Todos-Nós	3.73	1.91	5.64	5.82
Não enfatizam	25	19	44	45.41
Total	51.98	44.92	96.9	100
Média	1.52	1.31	2.83	-
Desvio padrão	4.61	3.34	7.79	-

4.2.4 Normas e padrões relacionados com o desenvolvimento de software embarcado crítico explorados

A Tabela 4.7 apresenta normas e padrões relacionados com o desenvolvimento de software embarcado crítico (quesito da Questão de Pesquisa Secundária 1.1). Ao examiná-la constata-se que a norma DO-178B é a mais utilizada, (20.92%), indicando assim que os objetivos e atividades definidos na DO-178B proporcionam condições para construir um software embarcado crítico com qualidade (Questão de Pesquisa Secundária 2.1). Observa-se também que são utilizadas normas específicas para alguns contexto da indústria, por exemplo, as normas do CENELEC (*European Committee for Electrotechnical Standardization*) que são recomendações para o desenvolvimento e teste de sistemas de transporte ferroviário (KLOOS; ESCHBACH, 2009).

Tabela 4.7: *Normas e padrões explorados*

Normas e padrões	**IEEE**	**ACM**	**Quantidade**	%
BS 5750	1	0	1	1.03
CENELEC-EN 50126	0.16	2.2	2.36	2.43
CENELEC-EN 50128	0.16	1.33	1.49	1.54
CENELEC-EN 50129	0.16	0	0.16	0.17
Defence Standards 00-55	0.25	0	0.25	0.26
Defence Standards 00-56	0.25	0	0.26	0.26
DO-178B	9.25	11.03	20.28	20.92
DO-254	0	0.33	0.33	0.34
FAA-STD-O60b	0.5	0	0.5	0.52
IEC 61508	1.91	0.7	2.61	2.69
IEC 61511	0	0.2	0.2	0.21
IEC 62278	0.16	0	0.16	0.17
IEC 62279	0.16	0	0.16	0.17
IEEE Std 7-4.3.2	0.33	0	0.33	0.34
IEEE Std 829	0.58	0	0.58	0.6
IEEE Std 1008	0.58	0	0.58	0.6
IEEE Std 1012	0.91	0	0.91	0.94
IEEE Std 1028	0.25	0	0.25	0.26
IEEE Std 1228	0.33	0	0.33	0.34
ISO CD 26262	0	0.2	0.2	0.21
NASA-STD-8719.13A	0	2	2	2.06
MIL-STD-882	1	0	1	1.03
MND-TMM	0	1	1	1.03
Não enfatizam	34	26	60	61.9
Total	51.94	44.99	96.93	100
Média	2.16	1.87	4.04	-
Desvio padrão	7.03	5.62	12.59	-

4.2.5 Requisitos do processo de teste da DO-178B

Como pode ser observado na Tabela 4.8, tem-se investigado muito a respeito do requisito *Análise de cobertura estrutural* da DO-178B, possivelmente isso se deve ao grau de complexidade associado ao mesmo. Para satisfazer o nível de software A, o critério de teste estrutural *Modified Condition/Decision Coverage* (MC/DC) deve ser satisfeito, devido ser esse o critério estrutural mais rigoroso definido pela norma DO-178B. Dupuy e Leveson (2000) apresentam um estudo de caso que ao procurar satisfazer o critério MC/DC, defeitos importantes não identificados pela técnica funcional foram encontrados, demonstrando assim a efetividade do mesmo para identificar defeitos críticos e complementar a técnica funcional.

Tabela 4.8: *Aderência aos requisitos do processo de teste*

Requisito do Processo de Teste	**IEEE**	**ACM**	**Quantidade**	%
Casos de teste de intervalo normal	4.61	2.91	7.52	7.75
Casos de teste de robustez	4.48	2.58	7.06	7.28
Métodos de teste baseado em requisitos	3.11	4.33	7.44	7.67
Análise de cobertura de teste baseado em requisitos	9.65	3.58	13.23	13.64
Análise de cobertura estrutural	10.15	16.58	26.73	27.56
não enfatizam	20	15	35	36.09
Total	52	44.98	96.98	100
Média	8.67	7.5	16.16	-
Desvio padrão	6.27	6.47	11.91	-

Desta forma, ao finalizar a avaliação da aderência dos estudos primário aos objetivos e atividades da DO-178B, conclui-se que 63.91% dos estudos estão relacionados com pelo menos um dos requisitos do processo de teste da DO-178B, respondendo assim a Questão de Pesquisa Primária 2.

4.2.6 Força das evidências

Com relação as características dos estudos, 59.79% dos estudos são de natureza observacional (como pode ser observado na Tabela 4.2 localizada na Página 60), somente 17.53% corresponde a experimentos. Conforme as definições da GRADE, as evidências desta RS segundo o critério características do estudo são consideradas baixas (ALI et al., 2010).

Com relação à qualidade dos estudos, as abordagens de análise de dados nos respectivos estudos foram explicadas de forma moderada; questões como viés potencial, credibilidade e limitações dos estudos (questões nove, dez e onze definidas na Seção 3.2.6, respectivamente). Somente em um estudo o pesquisador analisou criticamente o seu próprio papel. Em 98.45% dos estudos a credibilidade foi discutida, já com relação às limitações dos estudos, 72.68% dos estudos discutiram. Baseado nos resultados apresentados, os estudos segundo o critério qualidade apresentam evidências consideradas moderadas.

Com relação ao critério consistência, foram identificadas similaridades entre 63.91% dos estudos no que se refere aos requisitos da DO-178B (conforme apresentado na Tabela 4.8), pois pelo menos um dos requisitos da norma foi possível ser associado em mais de um estudo primário, sendo que os restantes não enfatizam os requisitos (36.09%). Em virtude disso, entende-se que a força das evidências no que se refere à consistência pode ser classificada como moderada.

Por fim, com relação ao critério objetividade (*directness*), que consiste em avaliar se as pessoas envolvidas (estudantes ou profissionais de software), as intervenções e os resultados dos estudos estão de acordo com a área de interesse. No contexto desta

RS, conforme apresentado na Tabela 4.3 constatou-se que a maioria dos estudos foi no contexto da academia/laboratório, mas apenas um estudo mencionou o nível de conhecimento e experiência dos estudantes envolvidos, graduandos e pós-graduandos experientes (LOUBACH et al., 2006). Com relação à intervenção, foi possível realizar comparações objetivas com 63.91% dos estudos no que se refere aos requisitos da DO-178B, mesmo que apenas 20.92% dos estudos tenha mencionado explicitamente o uso da DO-178B (conforme Tabela 4.7). Por fim, com relação aos resultados obtidos, mesmo que a maioria dos estudos sejam de natureza observacional, tais estudos não são triviais, sendo assim comparáveis com os softwares/sistemas desenvolvidos pela indústria. A partir dessas informações, a força das evidências no que se refere à objetividade pode ser considerada entre baixa e moderada.

Combinando os quatros elementos para se determinar a força das evidências, pode-se afirmar que a força das evidências para esta RS pode ser classificada como moderada, respondendo assim a Questão de Pesquisa Primária 3. Portanto, pesquisas futuras são susceptíveis que provoque um impacto importante sobre a confiança na estimativa do efeito, podendo assim alterar a estimativa.

4.3 Considerações finais

Dentre as cinco questões de pesquisa (primárias e secundárias) investigadas neste trabalho, apenas uma delas apresenta um resultado tendencioso (subjetivo), Questão de Pesquisa Primária 3, devido a maioria dos elementos analisados para responder esta questão serem resultados da realização da avaliação subjetiva da qualidade dos estudos primário. Já o restante das questões se tratam de questões mais objetivas, minimizando significativamente a tendenciosidade.

Comparando as Tabelas 4.5 e 4.8, constata-se que a técnica de teste funcional foi a mais utilizada, porém os dois primeiros requisitos do processo de teste da DO-178B apresentados na Tabela 4.8 que são aderentes aos critérios de teste funcionais (Particionamento de equivalência e Análise do valor limite), apresentam poucos trabalhos relacionados. Isso se deve ao fato de não ter sido especificado nesse subconjunto de estudos primários selecionados os correspondentes critérios de teste funcionais empregados.

A partir da análise dos estudos selecionados, constatou-se que todos os requisitos do processo de teste da DO-178B foram explorados, mas poucos trabalhos discutiram a respeito da resolução de problemas da análise de cobertura estrutural (n=3), tais como: código morto e código desativado.

CAPÍTULO 5

Proposta de Metodologia de Teste de Software para Sistemas Embarcados Críticos

Sistemas embarcados críticos são em geral módulos computacionais de monitoração e controle utilizados em conjunto com dispositivos físicos, como robôs, veículos autônomos, aeronaves não tripuladas e outros. Sistemas esses que impõem restrições no que diz respeito à segurança, desempenho, confiabilidade e outras. Portanto, falhas nesses sistemas podem resultar em danos a vidas humanas, danos ambientais ou altas perdas financeiras.

Com o objetivo de garantir níveis de qualidade que reduzam as chances dessas ocorrências trágicas, a RTCA (*Radio Technical Commission for Aeronautics*) em conjunto com a EUROCAE (*European Organization for Civil Aviation Equipment*) criaram a norma DO-178B, que disponibiliza um conjunto de diretrizes necessárias para o desenvolvimento e certificação de software embarcado em sistemas e equipamentos aeronáuticos, pois estes equipamentos não podem ser comercializados pela indústria sem a sua homologação por essa norma (RTCA/EUROCAE, 1992).

Nesse contexto, recentemente foi criado o Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC), com o objetivo de criar uma rede de colaboração e pesquisa em sistemas críticos (MALDONADO, 2008). Dentre os objetivos do INCT-SEC, um deles objetiva pesquisar diretrizes para a certificação da inclusão de Veículos Aéreos Não Tripulados (VANTs) no espaço aéreo controlado.

A partir disso, o presente capítulo visa apoiar parte dos objetivos do INCT-SEC, avaliando os resultados da RS e o contexto do INCT-SEC visando a propor uma metodologia de teste de software aderente à norma DO-178B. Inicialmente, na Seção 5.1 são apresentados modelos de ciclo de vida de software no contexto de sistema embarcado crítico. Em seguida, na Seção 5.2 é apresentada a metodologia de teste proposta considerando os estudos primários selecionados. Finalmente, na Seção 5.3 são feitas as considerações finais sobre este capítulo.

5.1 Modelos de ciclo de vida de software embarcado crítico

Sabe-se que quanto antes as atividades de teste forem executadas, menores serão os custos das correções de defeitos (DELAMARO et al., 2007). A Figura 5.1 apresenta uma opção de modelo de desenvolvimento de sistema que agrega atividades de verificação e validação com as atividades de desenvolvimento que é conhecido como Modelo-V, que se trata de um modelo de desenvolvimento inspirado no tradicional modelo cascata. O referido modelo recomenda que para cada etapa do desenvolvimento do software (lado esquerdo do modelo) seja possível empregar uma correspondente fase de verificação e validação (lado direito do modelo) visando, assim, antecipar a detecção e remoção de defeitos o quanto antes (VINCENZI, 2009).

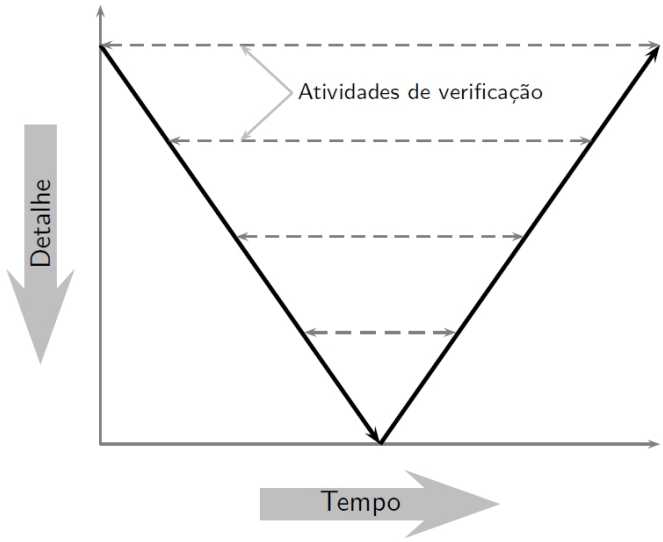

Figura 5.1: *Modelo-V inspirado no modelo em cascata (adaptada de Amey e Dion (2006)).*

Conforme ressaltado por Nicholson (2003), a geração atual dos sistemas embarcados críticos foi desenvolvida utilizando, tipicamente, o Modelo-V. Entretanto, Nicholson (2003) comenta que, no Modelo-V, a dependência estabelecida entre o projeto e o código torna o processo de alto custo, principalmente quando existe a necessidade de reprojeto.

Devido à necessidade de evolução rápida do sistema a partir de modificações realizadas no projeto, um novo modelo, denominado Modelo-Y foi definido (NICHOLSON, 2003). Esse modelo, ilustrado na Figura 5.2, parte do princípio que existe um mecanismo de geração automática de código confiável, tendo como base uma especificação formal. Tal suposição reduz de forma significativa o tempo de codificação e, consequentemente,

remove também algumas atividades de verificação no nível de unidade que resultam na entrega do produto final de forma mais rápida.

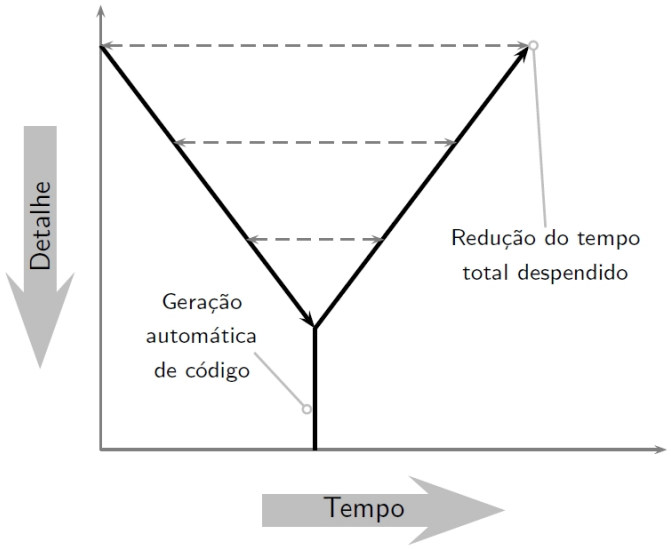

Figura 5.2: *Modelo-Y (adaptada de Amey e Dion (2006)).*

No entanto, sabe-se que a geração de 100% do código de maneira automática é difícil de ser alcançada na prática, devido à necessidade de se escrever código para acesso a interfaces de baixo nível, *drivers* para dispositivo e outros códigos para a integração de componentes. Considerando este contexto, um cenário mais próximo da realidade é uma solução híbrida (vide Figura 5.3). Para o código escrito manualmente, tem-se a necessidade de aplicar uma diversidade de atividades de verificação, dentre elas a de teste de software (unitário, integração e sistemas).

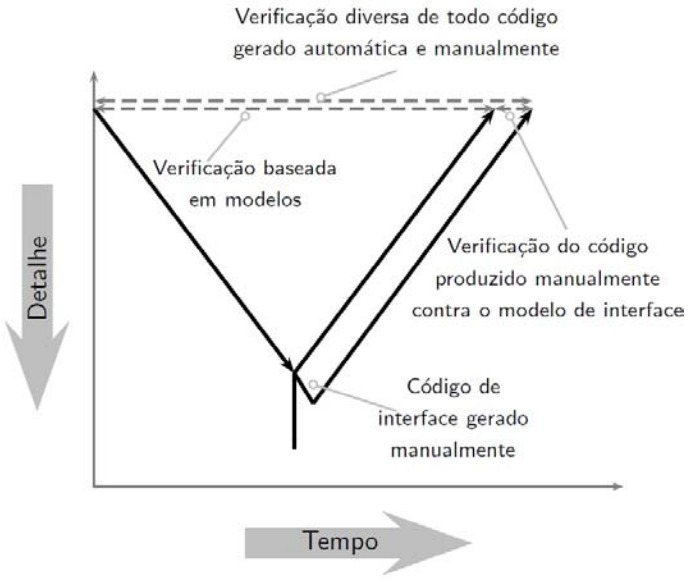

Figura 5.3: *Modelo-YV (adaptada de Amey e Dion (2006)).*

Dessa forma, na seção seguinte é apresentada uma proposta de uma metodologia de teste de software embarcado crítico, tendo como base o modelo híbrido (apresentado na Figura 5.3) e os estudos primários selecionados na RS, visando definir uma metodologia de teste apoiada por ferramentas CASE (*Computer-Aided Software Engineering*) certificadas.

5.2 Metodologia de teste proposta

Dentre os estudos primários selecionados na RS, Morschhauser e Lindvall (2007) e Tudor et al. (2004) fazem uso da ferramenta MATLAB/SIMULINK, que disponibiliza uma linguagem (no caso, SIMULINK) para modelagem de sistemas reativos empregando o formalismo matemático (equações diferenciais, equações booleanas e etc).

Já os estudos Lakehal e Parissis (2007), Papailiopoulou (2008) e Do et al. (2006) propõem o uso da ferramenta SCADE (*Safety Critical Application Development Environment*) que permite a definição hierárquica dos componentes do sistema e a geração automática de código na linguagem C.

Ambas ferramentas CASE são certificadas, mas como o ambiente de desenvolvimento SCADE foi uma das ferramentas mais utilizadas pelos estudos primários e por permitir a construção de produtos de software no contexto da aviação (DO et al., 2006), optou-se por adotá-la nesta metodologia.

A Figura 5.4 apresenta uma visão gráfica da metodologia proposta, sendo que a partir de uma especificação de requisitos de um sistema, parte destes requisitos deverão ser alocados para a especificação de requisitos do software embarcado crítico, que servirão de insumo para a definição do projeto do software e consequentemente a codificação do mesmo. Conforme discutido na Seção 5.1, atividades de verificação e validação deverão ser empregadas paralelamente às atividades de desenvolvimento do sistema embarcado crítico (apresentadas no lado direito da Figura 5.4).

A atividade de teste de componentes objetiva verificar elementos de menor unidade do projeto do software, nesta atividade deverão ser atendidos os seguintes requisitos do processo de teste da DO-178B: Casos de teste de intervalo normal, Casos de teste de robustez e o Teste de baixo nível baseado em requisitos (método do requisito de teste MTR).

Já a atividade de teste de integração de componentes, objetiva assegurar que os componentes de software interagem satisfatoriamente entre si e atendem os requisitos de software e a arquitetura do software. Nesta atividade deverão ser atendidos os seguintes requisitos do processo de teste da DO-178B: Análise de cobertura de teste baseado em requisitos, Análise de cobertura estrutural e o Teste de integração de software baseado em requisitos (método do requisito de teste MTR).

Com relação à atividade de teste baseado em modelos, dado um modelo que captura o comportamento essencial do sistema, o mesmo pode ser extremamente importante para o teste, servindo tanto como oráculo como base para a geração de scripts e cenários de teste (DELAMARO et al., 2007).

A atividade de teste de integração hardware/software, visa integrar o sistema final e realizar os testes finais de aceitação, que devem ser definidos sob coordenação do cliente final (JúNIOR et al., 2009). Nesta atividade deverá ser atendido o seguinte requisito do processo de teste da DO-178B: Teste de integração de hardware/software baseado em requisitos (método do requisito de teste MTR).

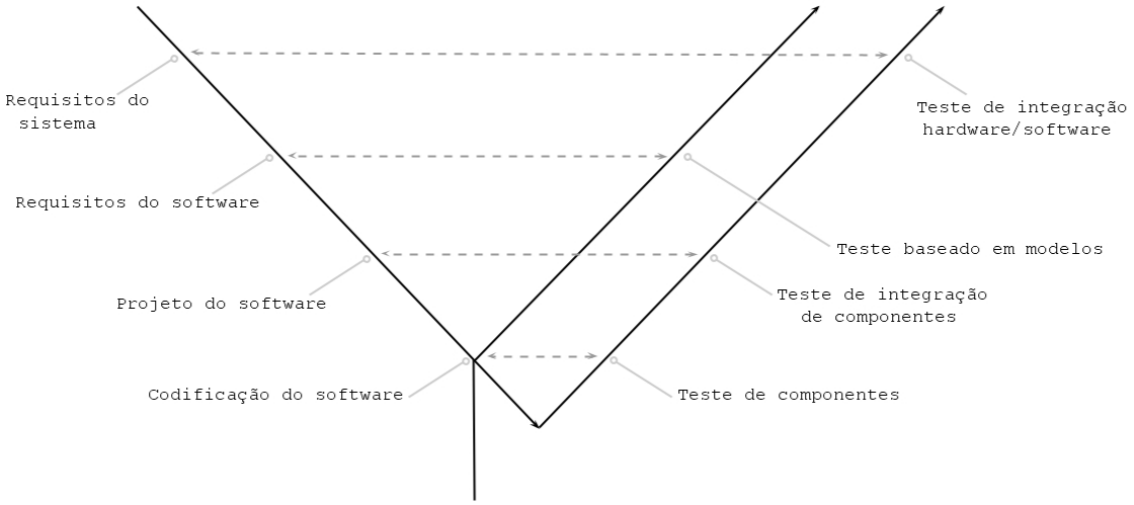

Figura 5.4: *Visão geral da metodologia de teste proposta.*

Dentre os estudos primários selecionados por meio da RS, apresentados na Tabela 4.1 localizada na Página 53, utilizou-se para realizar o mapeamento dos mesmos para a metodologia de teste proposta apenas os estudos que enfatizavam pelo menos um dos requisitos de teste da DO-178B (vide Tabela 5.1).

Tabela 5.1: *Mapeamento dos estudos primários para a metodologia de teste proposta*

Atividade de V&V	**Requisito da DO-178B**	**Estudo Primário**
Teste de Integração hardware/software	Teste de integração de hardware/software baseado em requisitos	(CHEON et al., 2004) (CLANCY et al., 2006) (COULTER, 1999) (LOUBACH et al., 2006) (JASKE et al., 1996) (PEREZ; KAISER, 2009) (WANG, 2004) (WANG; TAN, 2005)
Continua na próxima página...		

Tabela 5.1: *Mapeamento dos estudos primários para a metodologia de teste proposta*

Atividade de V&V	Requisito da DO-178B	Estudo Primário
Teste baseado em modelos	Verificação dos requisitos especificados utilizando métodos formais[1]	(ALAGAR et al., 2003) (ANDREWS; ZHANG, 2000) (AUGUSTON et al., 2005a) (AUGUSTON et al., 2005b) (BARBIER; BELLOIR, 2003) (BHATEJA, 2009) (DONG et al., 2009) (DRUSINSKY et al., 2007) (DRUSINSKY et al., 2006) (ESSER; STRUSS, 2007) (HAUSE et al., 2010) (HU, 2004) (KLOOS; ESCHBACH, 2010) (KLOOS; ESCHBACH, 2009) (KANDL et al., 2006) (JAW et al., 2008) (LöFFLER et al., 2010) (MORSCHHAUSER; LINDVALL, 2007) (NAKAO; ESCHBACH, 2008) (SANTIAGO et al., 2008b) (VIEIRA et al., 2008) (ZHENG et al., 2008) (YIN; LIU, 2009) (YIN et al., 2010) (TRAKHTENBROT, 2005) (TU; WU, 1999) (TUMMALA et al., 2006) (YU et al., 2010)
Continua na próxima página...		

[1] A norma DO-178B não menciona explicitamente a realização de teste baseado em modelos, mas como orienta o uso de métodos formais para especificar SEC, consequentemente os mesmos deverão ser avaliados considerando critérios de teste baseado em modelos.

Tabela 5.1: *Mapeamento dos estudos primários para a metodologia de teste proposta*

Atividade de V&V	Requisito da DO-178B	Estudo Primário
Teste de integração de componentes	Teste de integração de software baseado em requisitos	(CARPENTER, 1999) (CHANG et al., 1997) (CHEON et al., 2004) (CLANCY et al., 2006) (COULTER, 1999) (LOUBACH et al., 2006) (JASKE et al., 1996) (PEREZ; KAISER, 2009) (REZA et al., 2008) (TSAI et al., 2002) (WANG, 2004) (WANG; TAN, 2005)
	Análise de cobertura de teste baseado em requisitos	(ABDUL-BAKI et al., 2000) (BHOGARAJU et al., 2000) (BRITT, 1994) (CHEON et al., 2004) (COULTER, 1999) (DUPUY; LEVESON, 2000) (KUMAR et al., 2010) (LOUBACH et al., 2006) (MORTON, 1999) (HU, 2004) (MATHAIKUTTY et al., 2007) (MALEKZADEH; AINON, 2010) (NEBUT et al., 2006) (TSAI et al., 2002) (TSAI et al., 2005) (VIEIRA et al., 2008) (XU; WU, 1999) (YU et al., 2009) (TU; WU, 1999) (SANTO, 1988) (WU; HUANG, 2003) (WU; LI, 1999)

Continua na próxima página...

Tabela 5.1: *Mapeamento dos estudos primários para a metodologia de teste proposta*

Atividade de V&V	Requisito da DO-178B	Estudo Primário
	Análise de cobertura estrutural	(ANGELETTI et al., 2009) (AWEDIKIAN et al., 2009) (BELL; BRAT, 2008) (COULTER, 1999) (DUPUY; LEVESON, 2000) (KUMAR et al., 2010) (LOUBACH et al., 2006) (MORTON, 1999) (GOTLIEB, 2009) (GROSS et al., 2009) (GUAN et al., 2006) (GIANNAKOPOULOU, 2010) (KANDL et al., 2006) (JAW et al., 2008) (LAKEHAL; PARISSIS, 2005a) (LAKEHAL; PARISSIS, 2005b) (LAKEHAL; PARISSIS, 2007) (LöFFLER et al., 2010) (MORSCHHAUSER; LINDVALL, 2007) (MATHAIKUTTY et al., 2007) (MCDONALD et al., 2001) (NEBUT et al., 2006) (PăSăREANU et al., 2008) (PAPAILIOPOULOU, 2008) (SANTHANAM, 2001) (YU; LAU, 2006) (YU; LAU, 2004) (TRAKHTENBROT, 2005) (HAYHURST; VEERHUSEN, 2001) (RAYADURGAM; HEIMDAHL, 2001) (SMITH; URI, 2004) (WANG, 2004) (WANG; TAN, 2005)

Continua na próxima página...

Tabela 5.1: *Mapeamento dos estudos primários para a metodologia de teste proposta*

Atividade de V&V	Requisito da DO-178B	Estudo Primário
Teste de componentes	Casos de teste de intervalo normal	(BRITT, 1994) (CARPENTER, 1999) (CHANG et al., 1997) (COULTER, 1999) (DUPUY; LEVESON, 2000) (KUMAR et al., 2010) (LOUBACH et al., 2006) (JASKE et al., 1996) (KROPP et al., 1998) (LI et al., 1999) (NADEEM et al., 2006) (NEBUT et al., 2006) (TOMMASO et al., 2005) (TSAI et al., 2000) (TU et al., 1998) (XU; WU, 1999) (YU et al., 2009) (WANG, 2004) (WANG; TAN, 2005)
	Casos de teste de robustez	(CARPENTER, 1999) (CHANG et al., 1997) (COULTER, 1999) (DUPUY; LEVESON, 2000) (KUMAR et al., 2010) (LOUBACH et al., 2006) (JASKE et al., 1996) (KROPP et al., 1998) (LI et al., 1999) (NADEEM et al., 2006) (TOMMASO et al., 2005) (TSAI et al., 2000) (TU et al., 1998) (XU; WU, 1999) (YU et al., 2009) (WANG, 2004) (WANG; TAN, 2005)

Continua na próxima página...

Tabela 5.1: *Mapeamento dos estudos primários para a metodologia de teste proposta*

Atividade de V&V	Requisito da DO-178B	Estudo Primário
	Teste de baixo nível baseado em requisitos	(CARPENTER, 1999) (CHANG et al., 1997) (CHEON et al., 2004) (CLANCY et al., 2006) (COULTER, 1999) (DENG; SANG, 2008) (LOUBACH et al., 2006) (JASKE et al., 1996) (PEREZ; KAISER, 2009) (REZA et al., 2008) (TSAI et al., 2002) (XU; WU, 1999) (YU et al., 2009) (WANG, 2004) (WANG; TAN, 2005)

Considerando os estudos selecionados, a partir de um subconjunto dos estudos identificados na Tabela 5.1 definiu-se uma metodologia de teste aderente ao Nível A da DO-178B, discutida com mais detalhes na próxima seção.

5.2.1 Metodologia de teste proposta aderente ao Nível A da DO-178B

A Figura 5.5 apresenta o relacionamento do subconjunto de estudos primários selecionados com as correspondentes atividades de V&V da metodologia proposta. Para selecionar o correspondente subconjunto de estudos, foi considerado o grau de exigência de cobertura do Nível A da DO-178B, a compatibilidade com o ambiente de desenvolvimento SCADE e estudos que foram classificados como “Muito Bom” segundo os critérios de avaliação de qualidade dos estudos primários.

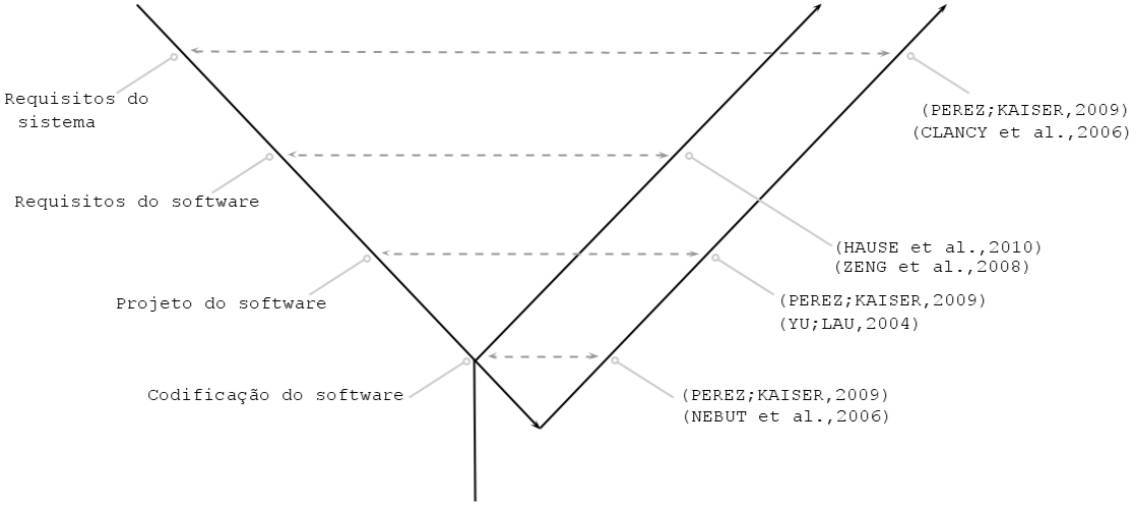

Figura 5.5: *Mapeamento de estudos primários para as atividades de V&V da metodologia.*

O SCADE Suite suporta o paradigma de desenvolvimento baseado em modelo, para modelar (projetar) comportamentos. O referido ambiente utiliza a combinação de dois formalismos para especificação: máquinas de estado (para especificar modos e transições em uma determinada aplicação) e diagramas de fluxo de dados (para especificar o controle de algoritmos) (TECHNOLOGIES, 2010).

Para a definição destes respectivos comportamentos, o SCADE Suite pode receber como insumo a arquitetura do sistema definida em SysML/UML (*Systems Modeling Language/Unified Modeling Language*), que se trata de uma linguagem de modelagem de sistemas que facilita o entendimento da estrutura de um sistema por meio da definição de blocos, pacotes e outros recursos específicos da linguagem. O estudo primário de Hause et al. (2010) apresenta uma metodologia que utiliza a SysML/UML para apoiar a realização do teste em um sistema crítico.

Já o estudo primário de Zheng et al. (2008) apresenta um estudo de caso que utiliza método formal para especificar sistemas e a partir do mesmo gera casos de teste e utiliza um algoritmo para selecionar os casos de teste baseado em métrica. Os algoritmos de geração de casos de teste e execução são implementados utilizando o TROMLAB, framework para especificação formal de sistemas reativos de tempo real (ALAGAR et al., 2001).

A partir de um modelo é possível o SCADE gerar código por meio do KCG (*Qualified Code Generator*) de forma automática, em virtude disso não é necessário a construção de casos de teste unitários para este código, mas é necessária a verificação da ação do compilador C por meio do CVK (*Compiler Verification Kit*), ou seja, verificar se foi compilado de forma correta o respectivo código C, resultando assim em um código objeto equivalente.

Concluída a geração do código objeto, deve-se aplicar os Métodos de teste baseado em requisitos previstos na DO-178B. O estudo primário de Perez e Kaiser (2009) apresenta uma hierarquia de teste baseada no Modelo V para sistemas embarcados críticos que se inicia no teste de componente, ou seja, teste unitário, necessário apenas quando é realizada adaptações manualmente do código de interface antes da geração do código objeto. No segundo momento deve ser realizado o teste de integração de componentes de software baseado em requisitos, com o objetivo de assegurar que os componentes de sofware interagem corretamente com os outros e satisfazem os requisitos de software e a arquitetura de software. Yu e Lau (2004) apresentam uma discussão teórica e empírica sobre os critérios de teste estruturais, enfatizando critérios de teste mais rigorosos, tais como: MC/DC, MUMCUT, MUTP, MNFP e CUTPNFP. Já o estudo de Nebut et al. (2006) apresentam uma discussão sobre a geração de casos de teste a partir de casos de uso no qual são apresentados três estudos de caso que são avaliados em relação aos seguintes critérios de teste: Todos-Nós e Todas-Arestas.

Deve ser realizado também, teste de integração de hardware/software baseado em requisitos com o objetivo de identificar defeitos que estejam relacionados com a operação do software no ambiente de implantação e com as funcionalidades de alto nível. Por fim deve ser realizado o teste de sistema, no qual os casos de teste são exercitados no ambiente de implantação considerando as condições reais do sistema.

Clancy et al. (2006) apresentam quatro estudos de caso relacionados com a definição e melhoria do processo de desenvolvimento e de teste de software de vôo no contexto de missões envolvendo aeronaves espaciais. A partir deste estudo constatou-se que em média 25% do ciclo de vida do software de vôo foi consumido para a realização do teste de hardware/software, sendo recomendado então a definição de *scripts* para configuração do sistema.

5.3 Considerações finais

Neste capítulo buscou-se discutir algumas alternativas de modelos de ciclo de vida para software embarcado crítico. Uma proposta de metodologia de teste de software embarcado crítico foi apresentada, detalhando as respectivas atividades de V&V, que foram apoiadas pelos estudos primários selecionados na RS.

Com o intuito de exemplificar melhor a correspondente metodologia de teste, a partir de um subconjunto dos estudos foi apresentada uma instância da metodologia que é aderente ao Nível A da DO-178B. É importante ressaltar que a automatização da metodologia é interessante, mas isso demanda padronizar cada um dos modelos apresentados nos trabalhos referenciados na Figura 5.5.

CAPÍTULO 6

Conclusão

A construção de um software de qualidade não é uma tarefa trivial, pelo contrário, dependendo do nível de qualidade e segurança que o software exige, pode se tornar uma tarefa bastante complexa. Como por exemplo, o software para sistemas embarcados críticos. Defeitos não identificados neste tipo de sistema podem provocar falhas que resultam em sérios problemas de segurança, podendo ocasionar desde um grave prejuízo financeiro até a perda de vidas humanas.

Considerando este contexto, um dos atuais objetivos do INCT-SEC é pesquisar diretrizes para a certificação da inclusão de Veículos Aéreos Não Tripulados (VANTs) no espaço aéreo controlado. No presente trabalho procurou-se definir um protocolo de uma revisão sistemática no contexto de teste de software embarcado crítico.

A partir das evidências coletadas por meio da RS, procurou-se propor uma metodologia de teste de software para o contexto do INCT-SEC, que foi inspirada no Modelo-YV, sendo que as atividade de V&V foram apoiadas por estudos primários selecionados durante a condução da RS.

Neste capítulo, inicialmente na Seção 6.1 é apresentado os resultados obtidos em relação ao que foi proposto para este trabalho. Enquanto a Seção 6.2 expõe as limitações do trabalho realizado e propõe trabalhos futuros relacionados a este.

6.1 Resultados Obtidos

Durante a condução da RS foram identificados 97 estudos primários relacionados com o assunto, os quais foram analisados visando responder às 5 questões de pesquisa propostas no planejamento da RS (apresentadas no Capítulo 3). Por meio da análise dos estudos, constatou-se que a norma mais utilizada neste contexto foi a DO-178B, sendo o MC/DC (*Modified Condition/Decision Coverage*) o critério de teste estrutural mais explorado. Uma das contribuições deste trabalho, portanto foi a reunião e análise dos trabalhos que tratam de teste de software em sistemas embarcados críticos (apresentados no Capítulo 4).

Além disso, os tipos de teste de software foram caracterizados e mapeados para os correspondentes requisitos do processo de teste da DO-178B. Constatou-se que o teste funcional foi o mais utilizado (36.77%), em seguida tem-se o teste baseado em modelos (28.36%) e o teste estrutural (24.4%). Observa-se que mesmo em sistemas embarcados críticos o teste funcional é bastante empregado, isto se deve ao fato de que a referida técnica é bastante importante para identificar defeitos básicos (que não estão em conformidade com a especificação de requisitos do software) na primeira fase do teste do software, enquanto as outras técnicas de teste de software (utilizadas em geral nas fases seguintes do processo de teste) objetivam complementar esta atividade, detectando defeitos não identificados nas fases anteriores.

Com relação aos requisitos do processo de teste de software da DO-178B, constatou-se que a análise de cobertura estrutural foi a mais referenciada pelos estudos primários (27.56%), isso se deve ao grau de complexidade para se aplicar os critérios de exigência de cobertura estrutural e também ao impacto da aplicação dos mesmos na qualidade final do produto de software embarcado crítico. Dentre os estudos primários selecionados, constatou-se a existência de estudos experimentais que revelaram critérios de testes estruturais mais rigorosos do que o MC/DC, tais como o M-CC, o MUMCUT e o CUTPNFP.

A partir de um subconjunto de estudos primários, definiu-se uma metodologia de teste aderente ao Nível A da DO-178B. Constatou-se que alguns dos estudos são incipientes, necessitando assim de mais estudos e experimentos para avaliar com mais detalhes as respectivas hipóteses/lições aprendidas. Acredita-se que com a atualização do protocolo da RS, ou seja, adoção de mais base de dados consiga-se obter estudos primários de boa qualidade e que não sejam incipientes.

6.2 Limitações e Trabalhos Futuros

No presente trabalho utilizou-se apenas duas bases eletrônicas indexadas para a condução da RS. Desse modo, visualiza-se como uma oportunidade de melhoria deste a atualização do protocolo da revisão sistemática definida neste visando coletar outras evidências, com adoção de outras bases de dados eletrônicas indexadas, tais como: ScienceDirect e Springer.

Como toda metodologia proposta, faz-se necessário a validação empírica da mesma. Nesse intuito, faz-se necessário executar um estudo de caso com o objetivo de avaliar e validar a metodologia de teste de software definida para o contexto de sistemas embarcados críticos.

Outro ponto a ser explorado consiste na automatização da metodologia proposta, objetivando identificar/construir, avaliar e integrar ferramentas para apoiar a metodologia

e também é uma oportunidade de investigação a padronização dos modelos apresentados nos estudos primários selecionados para apoiar a metodologia.

Bibliografia

ABDUL-BAKI, B.; BALDWIN, J.; RUDEL, M.-P. Independent validation and verification of the tcas ii collision avoidance subsystem. *Aerospace and Electronic Systems Magazine, IEEE*, v. 15, n. 8, p. 3 –21, ago. 2000. ISSN 0885-8985.

ALAGAR, V.; CHEN, M.; ORMANDJIEVA, O.; ZHENG, M. Automated test generation from object-oriented specifications of real-time reactive systems. In: *Software Engineering Conference, 2003. Tenth Asia-Pacific*. [S.l.: s.n.], 2003. p. 406 – 414.

ALAGAR, V. S.; ACHUTHAN, R.; MUTHIAYEN, D. *TROMLAB: A Software Development Environment for Real-Time Reactive Systems*. Montreal, Canada, 2001.

ALI, M. S.; BABAR, M. A.; CHEN, L.; STOL, K.-J. A systematic review of comparative evidence of aspect-oriented programming. *Inf. Softw. Technol.*, Butterworth-Heinemann, Newton, MA, USA, v. 52, p. 871–887, September 2010. ISSN 0950-5849. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2010.05.003>.

ALVER, M. O. et al. *JabRef*. [S.l.], 2008. Disponível em: <http://jabref.sourceforce.net>.

AMEY, P.; DION, B. Combining model-driven design with diverse formal verification. In: *III European Congress of the Embedded Real Time Software – ERTS'2006*. Toulouse, France: [s.n.], 2006. p. 1–8. Disponï¿½vel on-line: http://www.esterel-technologies.com/technology/WhitePapers/. Acesso em: 09/08/2009.

AMMANN, P.; OFFUTT, J.; HUANG, H. Coverage criteria for logical expressions. In: *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*. [S.l.: s.n.], 2003. p. 99 – 107. ISSN 1071-9458.

ANDREWS, J.; ZHANG, Y. Broad-spectrum studies of log file analysis. In: *Software Engineering, 2000. Proceedings of the 2000 International Conference on*. [S.l.: s.n.], 2000. p. 105 –114.

ANGELETTI, D.; GIUNCHIGLIA, E.; NARIZZANO, M.; PUDDU, A.; SABINA, S. Automatic test generation for coverage analysis of ertms software. In: *Proceedings of the 2009*

International Conference on Software Testing Verification and Validation. Washington, DC, USA: IEEE Computer Society, 2009. p. 303–306. ISBN 978-0-7695-3601-9. Disponível em: <http://portal.acm.org/citation.cfm?id=1547558.1548216>.

AUGUSTON, M.; MICHAEL, J. B.; SHING, M.-T. Environment behavior models for scenario generation and testing automation. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 30, p. 1–6, May 2005. ISSN 0163-5948. Disponível em: <http://doi.acm.org/10.1145/1082983.1083284>.

AUGUSTON, M.; MICHAEL, J. B.; SHING, M.-T. Test automation and safety assessment in rapid systems prototyping. In: *Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping*. Washington, DC, USA: IEEE Computer Society, 2005. (RSP '05), p. 188–194. ISBN 0-7695-2361-7. Disponível em: <http://dx.doi.org/10.1109/RSP-.2005.49>.

AWEDIKIAN, Z.; AYARI, K.; ANTONIOL, G. Mc/dc automatic test input data generation. In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2009. (GECCO '09), p. 1657–1664. ISBN 978-1-60558-325-9. Disponível em: <http://doi.acm.org/10.1145/1569901.1570123>.

BARBIER, F.; BELLOIR, N. Component behavior prediction and monitoring through built-in test. In: *Engineering of Computer-Based Systems, 2003. Proceedings. 10th IEEE International Conference and Workshop on the*. [S.l.: s.n.], 2003. p. 17 – 22.

BARBOSA, J. R.; DELAMARO, M. E.; MALDONADO, J. C.; VINCENZI, A. M. R. Software testing in critical embedded systems: a systematic review of adherence to the do-178b standard. In: *The Third International Conference on Advances in System Testing and Validation Lifecycle. VALID 2011.* Barcelona, Spain: IARIA, 2011. p. 126–130. ISBN 978-1-61208-168-7. Disponível em: <http://www.iaria.org/conferences2011/VALID11.html>.

BARBOSA, J. R.; DELAMARO, M. E.; MALDONADO, J. C.; VINCENZI, A. M. R. Teste de software no contexto de sistemas embarcados críticos. In: *Computer on the Beach 2011*. Florianópolis, SC, Brasil: UNIVALI, 2011. p. 339–340. ISBN 978-85-7696-077-5. Disponível em: <http://www.computeronthebeach.com.br/2011% -/>.

BELL, D.; BRAT, G. Automated software verification x00026; validation: An emerging approach for ground operations. In: *Aerospace Conference, 2008 IEEE*. [S.l.: s.n.], 2008. p. 1 –8. ISSN 1095-323X.

BERTRAND, C.; FUHRMAN, C. P. Towards defining software development processes in do-178b with openup. *Proceedings of the Canadian Conference on Eletrical and Computer Engineering*, 2008.

BHATEJA, P. Grammar based asynchronous testing. In: *Proceedings of the 2nd India software engineering conference*. New York, NY, USA: ACM, 2009. (ISEC '09), p. 105–110. ISBN 978-1-60558-426-3. Disponível em: <http://doi.acm.org/10.1145-/1506216.1506237>.

BHOGARAJU, S.; SINGH, G.; EDWARDS, G.; LIMBERG, J.; WATSON, M.; GOBROGGE, S. An information system for systematic validation of the software used in vehicular microcontrollers. In: *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. [S.l.: s.n.], 2000. p. 104 –109.

BIOLCHINI, J. C. de A.; MIAN, P. G.; NATALI, A. C. C.; CONTE, T. U.; TRAVASSOS, G. H. Scientific research ontology to support systematic review in software engineering. *Adv. Eng. Inform.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, p. 133–151, April 2007. ISSN 1474-0346. Disponível em: <http://portal.acm.org/citation.cfm?id=1247763.1248314>.

BLACKBURN, M.; BUSSER, R. T-vec: a tool for developing critical systems. In: *Computer Assurance, 1996. COMPASS '96, 'Systems Integrity. Software Safety. Process Security'. Proceedings of the Eleventh Annual Conference on*. [S.l.: s.n.], 1996. p. 237 –249.

BRITT, J. Case study: Applying formal methods to the traffic alert and collision avoidance system (tcas) ii. In: *Computer Assurance, 1994. COMPASS '94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security. Proceedings of the Ninth Annual Conference on*. [S.l.: s.n.], 1994. p. 39 –51.

CARPENTER, P. B. Verification of requirements for saftey-critical software. *Ada Lett.*, ACM, New York, NY, USA, XIX, p. 23–29, September 1999. ISSN 1094-3641. Disponível em: <http://doi.acm.org/10.1145/319295.319299>.

CHANG, T.-F.; DANYLYZSN, A.; NORIMATSU, S.; RIVERA, J.; SHEPARD, D.; LATTANZE, A.; TOMAYKO, J. ldquo;continuous verification rdquo; in mission critical software development. In: *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*. [S.l.: s.n.], 1997. v. 5, p. 273 –284 vol.5. ISSN 1060-3425.

CHEON, S.; LEE, J.; KWON, K.; KIM, D.; KIM, H. The software verification and validation process for a plc-based engineered safety features-component control system in nuclear power plants. In: *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*. [S.l.: s.n.], 2004. v. 1, p. 827 – 831 Vol. 1.

CHEON, S.; PARK, G.; CHA, K.; LEE, J.; KWON, K. The software v v tasks for a safety-critical software based protection system in nuclear power plants. In: *Industrial*

Technology, 2005. ICIT 2005. IEEE International Conference on. [S.l.: s.n.], 2005. p. 302 – 307.

CLANCY, D. A.; CLYDE, B. A.; MIRANTES, M. A. The evolution of a test process for spacecraft software. In: *Proceedings of the 2nd IEEE International Conference on Space Mission Challenges for Information Technology*. Washington, DC, USA: IEEE Computer Society, 2006. p. 425–434. ISBN 0-7695-2644-6. Disponível em: <http://portal.acm.org/citation.cfm?id=1158336.1158613>.

CONRAD, M.; FEY, I.; SADEGHIPOUR, S. Systematic model-based testing of embedded automotive software. *Electron. Notes Theor. Comput. Sci.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 111, p. 13–26, January 2005. ISSN 1571-0661. Disponível em: <http://dx.doi.org/10.1016/j.entcs.2004.12.005>.

COULTER, A. Graybox software testing methodology: embedded software testing technique. In: *Digital Avionics Systems Conference, 1999. Proceedings. 18th*. [S.l.: s.n.], 1999. v. 2, p. 10.A.5–1 –10.A.5–8 vol.2.

CULLYER, W.; STOREY, N. Tools and techniques for the testing of safety-critical software. *Computing Control Engineering Journal*, v. 5, n. 5, p. 239 –244, out. 1994. ISSN 0956-3385.

DAVIDSON, J. L.; AMOUSSOU, G.-A. Testing to certify an embedded software system. *J. Comput. Small Coll.*, Consortium for Computing Sciences in Colleges, USA, v. 25, p. 83–90, April 2010. ISSN 1937-4771. Disponível em: <http://portal.acm.org/citation-.cfm?id=1734797.1734815>.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao teste de software*. 1. ed. [S.l.]: Campus, 2007. 394 p.

DENG, Z.-Y.; SANG, N. Pattern verification-based increment memory testing method for safety-critical system. In: *Proceedings of the 2008 International Conference on Embedded Software and Systems Symposia*. Washington, DC, USA: IEEE Computer Society, 2008. p. 120–125. ISBN 978-0-7695-3288-2. Disponível em: <http://portal.acm.org/citation.cfm?id=1444447.1445059>.

DO, H. V.; ROBACH, C.; DELAUNAY, M. Mutation analysis for reactive system environment properties. In: *Proceedings of the Second Workshop on Mutation Analysis*. Washington, DC, USA: IEEE Computer Society, 2006. (MUTATION '06), p. 2–. ISBN 0-7695-2897-X. Disponível em: <http://dx.doi.org/10.1109/MUTATION.2006.9>.

DONG, Y. wei; WANG, G.; ZHAO, H. bing. A model-based testing for aadl model of embedded software. In: *Quality Software, 2009. QSIC '09. 9th International Conference on*. [S.l.: s.n.], 2009. p. 185 –190. ISSN 1550-6002.

DOWNING, C. *A Primer on Software Safety Certification*. 2002. 1–7 p.

DRUSINSKY, D.; SHING, M.-T.; DEMIR, K. A. Creation and validation of embedded assertion statecharts. In: *Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping*. Washington, DC, USA: IEEE Computer Society, 2006. p. 17–23. ISBN 0-7695-2580-6. Disponível em: <http://portal.acm.org/citation-.cfm?id=1136646.1136924>.

DRUSINSKY, D.; SHING, M.-T.; DEMIR, K. A. Creating and validating embedded assertion statecharts. *IEEE Distributed Systems Online*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 8, p. 3–, May 2007. ISSN 1541-4922. Disponível em: <http://portal.acm.org/citation.cfm?id=1271920.1271983>.

DUPUY, A.; LEVESON, N. An empirical evaluation of the mc/dc coverage criterion on the hete-2 satellite software. In: *Digital Avionics Systems Conferences, 2000. Proceedings. DASC. The 19th*. [S.l.: s.n.], 2000. v. 1, p. 1B6/1 –1B6/7 vol.1.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 2008.

DYBÅ, T.; DINGSØYR, T. Strength of evidence in systematic reviews in software engineering. In: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. New York, NY, USA: ACM, 2008. (ESEM '08), p. 178–187. ISBN 978-1-59593-971-5. Disponível em: <http://doi.acm.org-/10.1145/1414004.1414034>.

EN-NOUAARY, A.; KHENDEK, F.; DSSOULI, R. Testing embedded real-time systems. In: *Proceedings of the Seventh International Conference on Real-Time Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 2000. (RTCSA '00), p. 417–. ISBN 0-7695-0930-4. Disponível em: <http://portal.acm.org/citation.cfm?id=580571-.828835>.

ESSER, M.; STRUSS, P. Fault-model-based test generation for embedded software. In: *Proceedings of the 20th international joint conference on Artifical intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. p. 342–347. Disponível em: <http://portal.acm.org/citation.cfm?id=1625275.1625329>.

FERRARI, F. C.; MALDONADO, J. C. Teste de software orientado a aspectos: Uma revisão sistemática. *Relatórios Técnicos do ICMC*, n. 291, 2007.

FRANKL, P. G.; WEYUKER, E. J. A formal analysis of the fault-detecting ability of testing methods. v. 19, n. 3, p. 202–213, mar. 1993.

GIANNAKOPOULOU, D. "fly me to the moon": Verification of aerospace systems. In: *Software Engineering and Formal Methods (SEFM), 2010 8th IEEE International Conference on*. [S.l.: s.n.], 2010. p. 5 –11.

GOTLIEB, A. Euclide: A constraint-based testing framework for critical c programs. In: *Proceedings of the 2009 International Conference on Software Testing Verification and Validation*. Washington, DC, USA: IEEE Computer Society, 2009. p. 151–160. ISBN 978-0-7695-3601-9. Disponível em: <http://portal.acm.org/citation.cfm?id=1547558-.1548200>.

GROSS, H.; KRUSE, P. M.; WEGENER, J.; VOS, T. Evolutionary white-box software test with the evotest framework: A progress report. In: *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops*. Washington, DC, USA: IEEE Computer Society, 2009. p. 111–120. ISBN 978-0-7695-3671-2. Disponível em: <http://portal.acm.org/citation.cfm?id=1547559.1548243>.

GUAN, J.; OFFUTT, J.; AMMANN, P. An industrial case study of structural testing applied to safety-critical embedded software. In: *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. New York, NY, USA: ACM, 2006. (ISESE '06), p. 272–277. ISBN 1-59593-218-6. Disponível em: <http://doi.acm.org/10.1145-/1159733.1159774>.

HAUSE, M.; STUART, A.; RICHARDS, D.; HOLT, J. Testing safety critical systems with sysml/uml. In: *Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on*. [S.l.: s.n.], 2010. p. 325 –330.

HAYHURST, K.; VEERHUSEN, D. A practical approach to modified condition/decision coverage. In: *Digital Avionics Systems, 2001. DASC. The 20th Conference*. [S.l.: s.n.], 2001. v. 1, p. 1B2/1 –1B2/10 vol.1.

HEIMDAHL, M. Safety and software intensive systems: Challenges old and new. In: *Future of Software Engineering, 2007. FOSE '07*. [S.l.: s.n.], 2007. p. 137 –152.

HOTE, C. Extension of static verification techniques by semantic analysis. In: *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*. [S.l.: s.n.], 2005. v. 2, p. 9 pp. Vol. 2.

HU, X. *A simulation-based software development methodology for distributed real-time systems*. Tese (Doutorado) — The University of Arizona, Arizona, EUA, 2004. AAI3131605.

IEEE. *IEEE Std 610: Computer dictionary a compilation of IEEE Standard Computer Glossaries*. [S.l.]: Institute of Eletrical and Eletronics Engineers, 1990.

JASKE, U.-M.; ALAKUIJALA, J.; LAITINEN, J. Testing brachytherapy treatment planning software. In: *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*. [S.l.: s.n.], 1996. v. 5, p. 2030 –2032 vol.5.

JAW, L.; HOMAN, D.; CRUM, V.; CHOU, W.; KELLER, K.; SWEARINGEN, K.; SMITH, T. Model-based approach to validation and verification of flight critical software. In: *Aerospace Conference, 2008 IEEE*. [S.l.: s.n.], 2008. p. 1 –8. ISSN 1095-323X.

JúNIOR, O. T.; BRAGA, R. T. V.; NERIS, L. de O.; BRANCO, K. R. L. J. C. Uma metodologia para desenvolvimento de sistemas embarcados críticos com vistas a certificação. In: *IX Simpósio de Automação Inteligente – IX SBAI*. Brasília, DF: [s.n.], 2009.

JOUNG, E.; LEE, C.; LEE, H.; KIM, G. Software safety criteria and application procedure for the safety critical railway system. In: *Transmission Distribution Conference Exposition: Asia and Pacific, 2009*. [S.l.: s.n.], 2009. p. 1 –4.

KANDL, S.; KIRNER, R.; PUSCHNER, P. Development of a framework for automated systematic testing of safety-critical embedded systems. In: *Intelligent Solutions in Embedded Systems, 2006 International Workshop on*. [S.l.: s.n.], 2006. p. 1 –13.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. Keele, Staffs, ST5 5BG, UK, July 2004.

KITCHENHAM, B.; CHARTERS, S.; BUDGEN, D.; BRERETON, P.; TURNER, M.; LINKMAN, S.; JØRGENSEN, M.; MENDES, E.; VISAGGIO, G. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Keele, Staffs, ST5 5BG, UK, July 2007.

KLOOS, J.; ESCHBACH, R. Generating system models for a highly configurable train control system using a domain-specific language: A case study. In: *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops*. Washington, DC, USA: IEEE Computer Society, 2009. p. 39–47. ISBN 978-0-7695-3671-2. Disponível em: <http://portal.acm.org/citation.cfm?id=1547559.1548236>.

KLOOS, J.; ESCHBACH, R. A systematic approach to construct compositional behaviour models for network-structured safety-critical systems. *Electron. Notes Theor. Comput. Sci.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands,

The Netherlands, v. 263, p. 145–160, June 2010. ISSN 1571-0661. Disponível em: <http://dx.doi.org/10.1016/j.entcs.2010.05.009>.

KROPP, N.; KOOPMAN, P.; SIEWIOREK, D. Automated robustness testing of off-the-shelf software components. In: *Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on*. [S.l.: s.n.], 1998. p. 230 –239. ISSN 0731-3071.

KUMAR, S.; RAMAIAH, P.; KHANAA, V. A methodology for building safer software based critical computing systems. In: *Advance Computing Conference (IACC), 2010 IEEE 2nd International*. [S.l.: s.n.], 2010. p. 422 –429.

LAKEHAL, A.; PARISSIS, I. Lustructu: A tool for the automatic coverage assessment of lustre programs. In: *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*. Washington, DC, USA: IEEE Computer Society, 2005. p. 301–310. ISBN 0-7695-2482-6. Disponível em: <http://portal.acm.org/citation.cfm?id=1104997-.1105258>.

LAKEHAL, A.; PARISSIS, I. Structural test coverage criteria for lustre programs. In: *Proceedings of the 10th international workshop on Formal methods for industrial critical systems*. New York, NY, USA: ACM, 2005. (FMICS '05), p. 35–43. ISBN 1-59593-148-1. Disponível em: <http://doi.acm.org/10.1145/1081180.1081186>.

LAKEHAL, A.; PARISSIS, I. Automated measure of structural coverage for lustre programs: a case study. In: *Proceedings of the 29th International Conference on Software Engineering Workshops*. Washington, DC, USA: IEEE Computer Society, 2007. p. 209–. ISBN 0-7695-2830-9. Disponível em: <http://portal.acm.org/citation.cfm?id=1260984-.1261338>.

LI, W.; XU, Z.; JIN, Y. An approach for testing safety-critical software. In: *VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on*. [S.l.: s.n.], 1999. p. 180 –183.

LöFFLER, R.; MEYER, M.; GOTTSCHALK, M. Formal scenario-based requirements specification and test case generation in healthcare applications. In: *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care*. New York, NY, USA: ACM, 2010. (SEHC '10), p. 57–67. ISBN 978-1-60558-973-2. Disponível em: <http://doi.acm.org/10.1145/1809085.1809093>.

LOUBACH, D.; NOBRE, J.; CUNHA, A. da; DIAS, L.; NASCIMENTO, M.; SANTOS, W. D. Testing critical software: A case study for an aerospace application. In: *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*. [S.l.: s.n.], 2006. p. 1 –9.

MALDONADO, J. C. Sistemas embarcados críticos: Aplicações em segurança e agricultura. *Projeto de Pesquisa - Institutos Nacionais de Ciência e Tecnologia - CNPq*, 2008. Disponível em: <http://www.inct-sec.org/>.

MALEKZADEH, M.; AINON, R. An automatic test case generator for testing safety-critical software systems. In: *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*. [S.l.: s.n.], 2010. v. 1, p. 163 –167.

MATHAIKUTTY, D. A.; AHUJA, S.; DINGANKAR, A.; SHUKLA, S. Model-driven test generation for system level validation. In: *Proceedings of the 2007 IEEE International High Level Design Validation and Test Workshop*. Washington, DC, USA: IEEE Computer Society, 2007. p. 83–90. ISBN 978-1-4244-1480-2. Disponível em: <http://portal.acm.org/citation.cfm?id=1546679.1546852>.

MCDONALD, J.; MURRAY, L.; LINDSAY, P.; STROOPER, P. Module testing embedded software–an industrial pilot project. In: *Proceedings of the Seventh International Conference on Engineering of Complex Computer Systems*. Washington, DC, USA: IEEE Computer Society, 2001. p. 233–. Disponível em: <http://portal.acm.org/citation-.cfm?id=876906.881543>.

MORSCHHAUSER, I.; LINDVALL, M. Model-based validation verification integrated with sw architecture analysis: A feasibility study. In: *Aerospace Conference, 2007 IEEE*. [S.l.: s.n.], 2007. p. 1 –18. ISSN 1095-323X.

MORTON, S. Enhanced software design reusability in safety-critical embedded applications using automated structural test generation. In: *AUTOTESTCON '99. IEEE Systems Readiness Technology Conference, 1999. IEEE*. [S.l.: s.n.], 1999. p. 755 –767.

NADEEM, A.; MALIK, Z.; LYU, M. An automated approach to inheritance and polymorphic testing using a vdm++ specification. In: *Multitopic Conference, 2006. INMIC '06. IEEE*. [S.l.: s.n.], 2006. p. 224 –230.

NAKAO, H.; ESCHBACH, R. Strategic usage of test case generation by combining two test case generation approaches. In: *Secure System Integration and Reliability Improvement, 2008. SSIRI '08. Second International Conference on*. [S.l.: s.n.], 2008. p. 213 –214.

NASCIMENTO, H. A. D. do. *Hints for Graph Drawing*. [S.l.], 2003. Disponível em: <http://www.inf.ufg.br/~hadn/>.

NEBUT, C.; FLEUREY, F.; TRAON, Y. L.; JEZEQUEL, J.-M. Automatic test generation: A use case driven approach. *IEEE Trans. Softw. Eng.*, IEEE Press, Piscataway, NJ, USA,

v. 32, p. 140–155, March 2006. ISSN 0098-5589. Disponível em: <http://portal.acm.org-/citation.cfm?id=1128600.1128812>.

NICHOLSON, M. Supporting design synthesis for safety-critical systems. In: *Genetic and Evolutionary Computation Conference – GECCO '03*. Chicago, IL: [s.n.], 2003. Disponï¿½vel on-line: http://www-users.cs.york.ac.uk/~mark/papers/GECCO03-synthesis.pdf. Acesso em: 16/08/2009.

PAPAILIOPOULOU, V. Automatic test generation for lustre/scade programs. In: *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2008. (ASE '08), p. 517–520. ISBN 978-1-4244-2187-9. Disponível em: <http://dx.doi.org/10.1109/ASE-.2008.96>.

PEREZ, A. M.; KAISER, S. Integrating test levels for embedded systems. In: *Testing: Academic and Industrial Conference - Practice and Research Techniques, 2009. TAIC PART '09.* [S.l.: s.n.], 2009. p. 184 –193.

PăSăREANU, C. S.; MEHLITZ, P. C.; BUSHNELL, D. H.; GUNDY-BURLET, K.; LOWRY, M.; PERSON, S.; PAPE, M. Combining unit-level symbolic execution and system-level concrete execution for testing nasa software. In: *Proceedings of the 2008 international symposium on Software testing and analysis*. New York, NY, USA: ACM, 2008. (ISSTA '08), p. 15–26. ISBN 978-1-60558-050-0. Disponível em: <http://doi.acm.org/10.1145/1390630.1390635>.

QIAN, H. ming; ZHENG, C. A embedded software testing process model. In: *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*. [S.l.: s.n.], 2009. p. 1 –5.

RAYADURGAM, S.; HEIMDAHL, M. Coverage based test-case generation using model checkers. In: *Engineering of Computer Based Systems, 2001. ECBS 2001. Proceedings. Eighth Annual IEEE International Conference and Workshop on the*. [S.l.: s.n.], 2001. p. 83 –91.

REZA, H.; BUETTNER, S.; KRISHNA, V. A method to test component off-the-shelf (cots) used in safety critical systems. In: *Proceedings of the Fifth International Conference on Information Technology: New Generations*. Washington, DC, USA: IEEE Computer Society, 2008. p. 189–194. ISBN 978-0-7695-3099-4. Disponível em: <http://portal.acm.org/citation.cfm?id=1396808.1397552>.

RTCA/EUROCAE. *Software Considerations in Airborne Systems and Equipment Certification*. Washington, D.C., EUA, dez. 1992.

RYU, H.; RYU, D.-K.; BAIK, J. A strategic test process improvement approach using an ontological description for mnd-tmm. In: *Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. Washington, DC, USA: IEEE Computer Society, 2008. p. 561–566. ISBN 978-0-7695-3131-1. Disponível em: <http://portal.acm.org/citation.cfm?id=1396380.1396519>.

SANTHANAM, U. Automating software module testing for faa certification. *Ada Lett.*, ACM, New York, NY, USA, XXI, p. 31–38, September 2001. ISSN 1094-3641. Disponível em: <http://doi.acm.org/10.1145/507546.507582>.

SANTIAGO, V.; SILVA, W. P.; VIJAYKUMAR, N. L. Shortening test case execution time for embedded software. In: *Proceedings of the 2008 Second International Conference on Secure System Integration and Reliability Improvement*. Washington, DC, USA: IEEE Computer Society, 2008. p. 81–88. ISBN 978-0-7695-3266-0. Disponível em: <http://portal.acm.org/citation.cfm?id=1446296.1446510>.

SANTIAGO, V.; VIJAYKUMAR, N. L.; aES, D. G.; AMARAL, A. S.; FERREIRA, E. An environment for automated test case generation from statechart-based and finite state machine-based behavioral models. In: *Proceedings of the 2008 IEEE International Conference on Software Testing Verification and Validation Workshop*. Washington, DC, USA: IEEE Computer Society, 2008. p. 63–72. ISBN 978-0-7695-3388-9. Disponível em: <http://portal.acm.org/citation.cfm?id=1439281.1440221>.

SANTO, B. D. A methodology for analyzing avionics software safety. In: *Computer Assurance, 1988. COMPASS '88*. [S.l.: s.n.], 1988. p. 113 –118.

SMITH, T.; URI, C. Ieee scan-like interface for air traffic control software testing. In: *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*. [S.l.: s.n.], 2004. v. 2, p. 10.C.5 – 10.1–9 Vol.2.

SOUZA, M. da S.; DIAS, L. A. V. A gestção de projetos existente na norma do-178b. *Brazilian Symposium on Aerospace Eng. & Applications*, 09 2009.

SWARUP, M. B.; RAMAIAH, P. S. An approach to modeling software safety. In: *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on*. [S.l.: s.n.], 2008. p. 800 –806.

TECHNOLOGIES, E. *Methodology Handbook - Efficient Development of Safe Avionics Software with DO-178B Objectives Using SCADE Suite*. 5. ed. [S.l.]: Esterel Technologies, 2010. 110 p.

TIAN, P.; WANG, J.; LENG, H.; QIANG, K. Construction of distributed embedded software testing environment. In: *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC '09. International Conference on*. [S.l.: s.n.], 2009. v. 1, p. 470 –473.

TOMMASO, P. di; FLAMMINI, F.; LAZZARO, A.; PELLECCHIA, R.; SANSEVIERO, A. The simulation of anomalies in the functional testing of the ertms/etcs trackside system. In: *Proceedings of the Ninth IEEE International Symposium on High-Assurance Systems Engineering*. Washington, DC, USA: IEEE Computer Society, 2005. p. 131–139. ISBN 0-7695-2377-3. Disponível em: <http://portal.acm.org/citation.cfm?id=1111687.1112122>.

TRACEY, N.; CLARK, J.; MCDERMID, J.; MANDER, K. A search-based automated test-data generation framework for safety-critical systems. In: ____. New York, NY, USA: Springer-Verlag New York, Inc., 2002. p. 174–213. ISBN 1-85233-399-5. Disponível em: <http://portal.acm.org/citation.cfm?id=763012.763025>.

TRAKHTENBROT, M. Use of verification for testing and debugging of complex reactive systems. In: *Software Engineering and Formal Methods, 2005. SEFM 2005. Third IEEE International Conference on*. [S.l.: s.n.], 2005. p. 13 – 22.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. do. Introdução à engenharia de software experimental. *Relatório Técnico - COPPE - UFRJ*, n. RT-ES-590/02, 2002.

TSAI, W.; YU, L.; ZHU, F.; PAUL, R. Rapid verification of embedded systems using patterns. In: *Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International*. [S.l.: s.n.], 2003. p. 466 – 471. ISSN 0730-3157.

TSAI, W. T.; AGARWAL, V.; HUANG, B.; PAUL, R. Augmenting sequence constraints in z and its application to testing. In: *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET'00)*. Washington, DC, USA: IEEE Computer Society, 2000. (ASSET '00), p. 41–. ISBN 0-7695-0559-7. Disponível em: <http://portal.acm.org/citation.cfm?id=786772.787147>.

TSAI, W.-T.; NA, Y.; PAUL, R. J.; LU, F.; SAIMI, A. Adaptive scenario-based object-oriented test frameworks for testing embedded. In: *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*. Washington, DC, USA: IEEE Computer Society, 2002. (COMPSAC '02), p. 321–326. ISBN 0-7695-1727-7. Disponível em: <http://portal.acm.org/citation-.cfm?id=645984.673672>.

TSAI, W.-T.; YU, L.; ZHU, F.; PAUL, R. Rapid embedded system testing using verification patterns. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 22,

p. 68–75, July 2005. ISSN 0740-7459. Disponível em: <http://portal.acm.org/citation-.cfm?id=1079836.1079969>.

TU, H.; WU, F. How to design an environment simulator for safety critical software testing. In: *Test Symposium, 1999. (ATS '99) Proceedings. Eighth Asian*. [S.l.: s.n.], 1999. p. 256 –260.

TU, H.; WU, F.; REN, X. Rough-hierarchical testing for safety critical software. In: *Proceedings of the 7th Asian Test Symposium*. Washington, DC, USA: IEEE Computer Society, 1998. (ATS '98), p. 126–. ISBN 0-8186-8277-9. Disponível em: <http://portal.acm.org/citation.cfm?id=783735.783909>.

TUDOR, N.; ADAMS, M.; CLAYTON, P.; O'HALLORAN, C. Auto-coding/auto-proving flight control software. In: *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*. [S.l.: s.n.], 2004. v. 2, p. 6.E.4 – 61–11 Vol.2.

TUMMALA, H.; AUGUSTON, M.; MICHAEL, J.; SHING, M.-T.; LITTLE, D.; PACE, Z. Implementation and analysis of environment behavior models as a tool for testing real-time, reactive systems. In: *System of Systems Engineering, 2006 IEEE/SMC International Conference on*. [S.l.: s.n.], 2006. p. 5 pp.

VIEIRA, M.; SONG, X.; MATOS, G.; STORCK, S.; TANIKELLA, R.; HASLING, B. Applying model-based testing to healthcare products: preliminary experiences. In: *Proceedings of the 30th international conference on Software engineering*. New York, NY, USA: ACM, 2008. (ICSE '08), p. 669–672. ISBN 978-1-60558-079-1. Disponível em: <http://doi.acm.org/10.1145/1368088.1368183>.

VINCENZI, A. M. R. *Processo e Técnicas de Teste no Contexto de Sistemas Embarcados Críticos*. [S.l.], 2009.

WANG, L. Issues on software testing for safety-critical real-time automation systems. In: *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*. [S.l.: s.n.], 2004. v. 2, p. 101–12 Vol.2.

WANG, L.; TAN, K. Software testing for safety critical applications. *Instrumentation Measurement Magazine, IEEE*, v. 8, n. 2, p. 38 – 47, jun. 2005. ISSN 1094-6969.

WU, F.; HUANG, L. Efficiency analysis safety assessment of automatic testing for safety-critical software [railway control]. In: *Test Symposium, 2003. ATS 2003. 12th Asian*. [S.l.: s.n.], 2003. p. 106 – 109. ISSN 1081-7735.

WU, F.; LI, M. Railway signaling safety-critical software testing based on dynamic decision table. In: *Test Symposium, 1999. (ATS '99) Proceedings. Eighth Asian*. [S.l.: s.n.], 1999. p. 247 –250.

XIE, G. Decompositional verification of component-based systems-a hybrid approach. In: *Automated Software Engineering, 2004. Proceedings. 19th International Conference on*. [S.l.: s.n.], 2004. p. 414 –417. ISSN 1068-3062.

XU, Z.; WU, F. A novel testing approach for safety-critical software. In: *Proceedings of the 8th Asian Test Symposium*. Washington, DC, USA: IEEE Computer Society, 1999. (ATS '99), p. 251–. ISBN 0-7695-0315-2. Disponível em: <http://portal.acm.org/citation-.cfm?id=783328.783555>.

YIN, Y.; LI, Z.; LIU, B. Real-time embedded software test case generation based on time-extended efsm: A case study. In: *Information Engineering (ICIE), 2010 WASE International Conference on*. [S.l.: s.n.], 2010. v. 2, p. 272 –275.

YIN, Y.; LIU, B. A method of test case automatic generation for embedded software. In: *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*. [S.l.: s.n.], 2009. p. 1 –5.

YOON, K.-A.; PARK, S.-H.; BAE, D.-H.; CHANG, H.-S.; JUNG, J.-C. A framework for the vamp;v capability assessment focused on the safety-criticality. In: *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on*. [S.l.: s.n.], 2005. p. 17 –24.

YU, G.; XU, Z. w.; DU, J. w. An approach for automated safety testing of safety-critical software system based on safety requirements. In: *Proceedings of the 2009 International Forum on Information Technology and Applications - Volume 03*. Washington, DC, USA: IEEE Computer Society, 2009. (IFITA '09), p. 166–169. ISBN 978-0-7695-3600-2. Disponível em: <http://dx.doi.org/10.1109/IFITA.2009.18>.

YU, G.; XU, Z. w.; XIONG, J. Modeling and safety test of safety-critical software. In: *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*. [S.l.: s.n.], 2010. v. 2, p. 580 –583.

YU, Y. T.; LAU, M. F. Comparing several coverage criteria for detecting faults in logical decisions. In: *Proceedings of the Quality Software, Fourth International Conference*. Washington, DC, USA: IEEE Computer Society, 2004. (QSIC '04), p. 14–21. ISBN 0-7695-2207-6. Disponível em: <http://dx.doi.org/10.1109/QSIC.2004.13>.

YU, Y. T.; LAU, M. F. A comparison of mc/dc, mumcut and several other coverage criteria for logical decisions. *J. Syst. Softw.*, Elsevier Science Inc., New York, NY, USA, v. 79, p. 577–590, May 2006. ISSN 0164-1212. Disponível em: <http://dx.doi.org/10.1016/j.jss-.2005.05.030>.

ZELKOWITZ, M.; WALLACE, D. Experimental models for validating technology. *Computer*, v. 31, n. 5, p. 23 –31, may 1998. ISSN 0018-9162.

ZHENG, M.; ALAGAR, V.; ORMANDJIEVA, O. Automated generation of test suites from formal specifications of real-time reactive systems. *J. Syst. Softw.*, Elsevier Science Inc., New York, NY, USA, v. 81, p. 286–304, February 2008. ISSN 0164-1212. Disponível em: <http://portal.acm.org/citation.cfm?id=1326359.1326427>.

ZHU, H. A formal analysis of the subsume relation between software test adequacy criteria. v. 22, n. 4, p. 248–255, abr. 1996.

APÊNDICE A

Glossário

Alguns termos mencionados neste trabalho são apresentados na Tabela A.1 seguido do respectivo nome (ou sigla) e da definição.

Tabela A.1: *Glossário*

Termo	**Definição**
Branch/Decision coverage (B/DC)	Se trata de um critério de teste de software estrutural que tem como objetivo avaliar se cada um dos termos de uma dada condição assumiram os possíveis valores booleanos (*true* e *false*) (MCDONALD et al., 2001).
Caso de teste	Um conjunto de entradas de teste, condições de execução e resultados esperados desenvolvidos para atender um objetivo específico, como por exemplo exercitar um caminho do programa em particular ou para verificar inconsistência com um requisito específico (IEEE, 1990).
Condição (*Condition*)	É uma expressão booleana sem operadores booleanos (YU; LAU, 2006).
Condition Coverage (CC)	Cada condição tem que assumir todos os possíveis valores pelo menos uma vez (YU; LAU, 2006).
Condition/Decision Coverage (C/DC)	Cada condição em uma decisão tem que assumir todos os possíveis valores pelo menos uma vez e, cada decisão em um programa tem que assumir todos os possíveis valores pelo menos uma vez (YU; LAU, 2006).
Critério de teste	Se trata de um critério que um sistema ou componente deve encontrar e passar para exercitar um teste específico (IEEE, 1990).
Continua na próxima página...	

Tabela A.1: *Glossário*

Termo	**Definição**
Critério de teste de intensidade de segurança (*safety depth of testing*)	É considerado um critério de teste estrutural, que tem como objetivo avaliar o número de caminhos em um módulo que foram exercitadas, o número de instruções exercitadas no módulo e o número de parâmetros de entrada (de um método ou de uma função) com pelo menos um valor válido e um inválido dentre todos os parâmetros de entrada que foram exercitados (KUMAR et al., 2010).
Decisão (*Decision*)	É uma expressão booleana composta de condições com zero ou mais operadores booleanos (YU; LAU, 2006).
Decision Coverage (DC)	Cada decisão tem que assumir todos os possíveis valores pelo menos uma vez (YU; LAU, 2006).
Estudo de caso	Visa monitorar um projeto específico em profundidade (ZELKOWITZ; WALLACE, 1998) , (TRAVASSOS et al., 2002).
Estudos de caso	São estudos que apresentam mais de um caso e em geral realizam comparações entre os diferentes resultados dos projetos envolvidos (ZELKOWITZ; WALLACE, 1998).
Experimento	Geralmente é realizado em laboratório e permite um maior controle, tendo assim como alvo manipular uma ou algumas variáveis e manter as outras fixas medindo o efeito do resultado (TRAVASSOS et al., 2002).
Modified Condition/Decision Coverage (MC/DC)	Cada condição em uma decisão tem assumir todos os possíveis valores pelo menos uma vez e, cada decisão tem que assumir todos os possíveis valores pelo menos uma vez e cada condição tem que ser atribuída para afetar de forma independente o resultado da decisão. Uma condição é atribuída para afetar de forma independente o resultado de uma decisão, variando apenas uma delas e mantendo as outras condições fixas (YU; LAU, 2006).
Multiple-Condition Coverage (M-CC)	Todas as possíveis combinações de resultados das condições dentro de cada decisão tem que ser exercitada pelo menos uma vez (YU; LAU, 2006).
(MUMCUT)	Um conjunto de casos de teste T satisfaz todos os seguintes critérios: MUTP, MNFP e CUTPNFP (YU; LAU, 2006).
Continua na próxima página...	

Tabela A.1: *Glossário*

Termo	**Definição**
Requisitos de alto nível	São construídos diretamente a partir da análise dos requisitos do sistema e da arquitetura do sistema (RTCA/EUROCAE, 1992).
Requisitos de baixo nível	São requisitos de software que permitem a construção direta de código fonte sem a necessidade de informações adicionais (RTCA/EUROCAE, 1992).
Survey	Se trata de uma investigação realizada em retrospectiva, em geral é conduzida quando algumas técnicas ou ferramentas já tenham sido utilizadas (TRAVASSOS et al., 2002).
Técnica de teste	O processo de operação de um sistema ou componente durante condições específicas, observações ou registro de resultados e avaliação de alguns aspectos do sistema ou componente (IEEE, 1990).

APÊNDICE B

Resultados da Condução da Revisão Sistemática

B.1 Condução

A revisão sistemática foi conduzida por um período de 4 meses (09/2010 a 12/2010), de acordo com o planejamento apresentado nas seções anteriores. Ao todo, foram recuperados 872 trabalhos, que foram submetidos para as etapas de seleção preliminar, seleção final e extração de resultados. Nas próximas seções são apresentados mais detalhes das atividades realizadas, incluindo a estratégia adotada para construção das strings de busca e os resultados das buscas para cada uma das fontes selecionadas.

B.1.1 Seleção Preliminar

A seleção preliminar foi conduzida em três etapas:

1.Construção das Strings de busca;
2.Realização das buscas; e
3.Seleção preliminar de trabalhos.

Essas três etapas são detalhadas nas próximas seções.

Construção das Strings de Busca

Para se definir as *strings* de busca, foram utilizadas as palavras-chaves e sinônimos identificados na Seção 3.2.2 do Capítulo 3 localizado na Página 42. Utilizando o operador lógico "ou" (OR) para integrar os termos-chave e seus respectivos sinônimos, e o operador "e" (AND) para integrar termos-chave diferentes, conforme é apresentado a seguir:

(critical embedded OR safety-critical OR mission-critical OR embedded software) AND (test) AND (software OR system)

Buscas Realizadas

Nesta seção são apresentadas os detalhes das buscas realizadas em cada uma das fontes escolhidas.

Busca no IEEE

A *string* definida na etapa de Construção das *Strings*, foi alterada para ser submetida à máquina no modo "busca avançada", e para atender à respectiva sintaxe da máquina. A busca foi realizada em 01/12/2010, na página de busca avançada (http://ieeexplore.ieee.org/Xplore/dynhome.jsp). A primeira busca foi restringida aos resumos das publicações, a *string* a seguir retornou um total de 593 trabalhos:

```
(("Abstract": "critical embedded") OR ("Abstract": "safety-critical") OR
 ("Abstract": "mission-critical") OR ("Abstract": "embedded software")) AND
 ("Abstract": test) AND (("Abstract": software) OR ("Abstract": system))
```

Figura B.1: *Primeira consulta na máquina de busca do IEEE.*

Realizou-se também uma busca avançada considerando a possibilidade de identificar palavras-chave no título da publicação (revista, jornal ou congresso), utilizando a *string* abaixo, não retornou nenhum trabalho:

```
(("Publication Title":"critical embedded") OR
 ("Publication Title":"safety-critical") OR
 ("Publication Title":"mission-critical") OR
 ("Publication Title":"embedded software")) AND
 ("Publication Title":"test") AND
 (("Publication Title":"software") OR
 ("Publication Title":"system") )
```

Figura B.2: *Segunda consulta na máquina de busca do IEEE.*

Realizou-se também uma terceira busca avançada considerando a possibilidade de identificar palavras-chave no título do artigo, utilizando a *string* abaixo, que retornou um total de 77 trabalhos, dos quais 57 eram artigos já obtidos a partir da consulta anterior:

```
(("Document Title":"critical embedded") OR
 ("Document Title":"safety-critical") OR
 ("Document Title":"mission-critical") OR
 ("Document Title":"embedded software"))
 AND ("Document Title":test) AND
 (("Document Title":software) OR
 ("Document Title":system))
```

Figura B.3: *Terceira consulta na máquina de busca do IEEE.*

Busca na ACM

Na máquina de busca da biblioteca digital da ACM (http://portal.acm.org/advsearch.cfm?coll=GUIDE&dl=GUIDE&query=Owner%3AGUIDE&qrycnt=1274481&since_month=&since_year=&before_month=&before_year=&CFID=31154586&CFTOKEN=88703848) também é possível realizar buscas avançadas, a busca foi realizada no dia 25/11/2010 no modo avançado, a string abaixo retornou 264 trabalhos:

```
((Abstract:"critical embedded" OR Abstract:"safety-critical" OR
 Abstract:"mission-critical" OR Abstract:"embedded software") AND
 (Abstract: "test" ) AND (Abstract: "software" OR Abstract: "system" ))
```

Figura B.4: *Primeira consulta na máquina de busca da ACM.*

Realizou-se uma segunda busca avançada, a busca foi realizada no dia 25/11/2010 no modo avançado, a string abaixo retornou 19 trabalhos, dos quais 12 eram artigos já obtidos a partir da consulta anterior:

```
((Title:"critical embedded" OR Title: "safety-critical" OR
 Title:"mission-critical" OR Title:"embedded software") AND
 (Title: "test" ) AND ( Title: "software" OR Title: "system"))
```

Figura B.5: *Segunda consulta na máquina de busca da ACM.*

Seleção Preliminar de trabalhos

Nesta seção são apresentadas os detalhes da seleção preliminar em cada uma das fontes escolhidas.

B.1.2 Seleção Final

Nesta seção são apresentadas os detalhes da seleção final em cada uma das fontes escolhidas.

Base eletrônica indexada IEEE

Na Figura B.6, a Fase 1 corresponde ao total de estudos primários retornados da base eletrônica ACM após a submissão das respectivas string de consulta (n=601). A Fase 2 corresponde ao total de estudos resultantes do processo de seleção preliminar (n=166), sendo n=435 excluídos pois o título ou resumo não atendiam o escopo das questões de pesquisa da RS. A Fase 3 corresponde ao total de estudos resultantes do processo de seleção final (n=53), sendo n=113 excluídos uma vez que após a leitura completa dos referidos estudos, identificou-se que os mesmos não atendiam o escopo das questões de

pesquisa da RS . Finalmente, na Fase 4 foram eliminados ainda *n*=1 visto que após a avaliação dos estudos segundo os critérios de qualidade dos estudos primários definidos no planejamento da RS, foram considerados de baixa qualidade e, desse modo, restaram *n*=52 estudos primários selecionados para extração e sumarização dos resultados.

Figura B.6: *Fases da seleção final - IEEE*

Base eletrônica indexada ACM

Na Figura B.7, a Fase 1 corresponde ao total de estudos primários retornados da base eletrônica ACM após a submissão das respectivas string de consulta (*n*=271). A Fase 2 corresponde ao total de estudos resultantes do processo de seleção preliminar (*n*=119), sendo *n*=152 excluídos pois o título ou resumo não atendiam o escopo das questões de pesquisa da RS. A Fase 3 corresponde ao total de estudos resultantes do processo de seleção final (*n*=47), sendo *n*=72 excluídos uma vez que após a leitura completa dos referidos estudos, identificou-se que os mesmos não atendiam o escopo das questões de pesquisa da RS . Finalmente, na Fase 4 foram eliminados ainda *n*=2 visto que após a avaliação dos estudos segundo os critérios de qualidade dos estudos primários definidos no planejamento da RS, foram considerados de baixa qualidade e, desse modo, restaram *n*=45 estudos primários selecionados para extração e sumarização dos resultados.

Figura B.7: *Fases da seleção final - ACM*

Avaliação da qualidade dos estudos primários

Na Tabela B.1 é apresentado a avaliação individual de cada estudo primário selecionado. As oito primeiras questões dizem respeito à qualidade e o rigor dos estudos, já as questões 9 a 11 dizem respeito à credibilidade das evidências e limitações apresentadas no estudo primário (ALI et al., 2010).

Tabela B.1: *Avaliação da Qualidade dos Estudos*

Estudo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Subtotal(1-8)	Q9	Q10	Q11	Subtotal(9-11)	Total
(RYU et al., 2008)	1	1	1	0	0	0	1	1	5	0	1	0	1	6
(AUGUSTON et al., 2005a)	1	1	0.5	0	0	0	1	0.5	4	0	1	0	0	5
(AUGUSTON et al., 2005b)	1	1	0.5	0	0	0	1	0.5	4	0	1	0	0	5
(BHATEJA, 2009)	1	0.5	1	0	0	0.5	1	0	4	0	1	0	0	5
(DRUSINSKY et al., 2007)	1	1	0.5	0	0.5	0	0.5	0.5	4	0	0.5	0	0.5	4.5
(DRUSINSKY et al., 2006)	1	1	0.5	0	0.5	0	0.5	1	4	0	0	0	0	4.5
(ESSER; STRUSS, 2007)	1	1	0.5	0.5	0.5	0	1	1	5.5	0	1	0.5	1.5	7
(HU, 2004)	1	1	1	0	0.5	0	1	1	5.5	0	1	1	2	7.5
(KLOOS; ESCHBACH, 2010)	1	1	1	0.5	0.5	0	0.5	0	4.5	0	1	0.5	1.5	6
(KLOOS; ESCHBACH, 2009)	1	1	1	1	0	0	1	1	6	0	1	0	1	7
(LöFFLER et al., 2010)	1	1	1	0	1	0	1	0.5	5.5	0	1	1	2	7.5
(SANTIAGO et al., 2008b)	1	1	0.5	0.5	0.5	0	1	1	5.5	0	1	1	2	7.5
(VIEIRA et al., 2008)	1	1	1	0.5	0.5	0	0.5	0.5	5	0	1	1	2	7
(ZHENG et al., 2008)	1	1	1	0.5	0.5	1	1	1	7	0.5	1	1	2.5	9.5
(DO et al., 2006)	1	1	1	0	0.5	0	0.5	0	4	0	1	1	2	6
(ANGELETTI et al., 2009)	1	1	1	0	1	0.5	0.5	1	6	0	1	0.5	1.5	7.5
(AWEDIKIAN et al., 2009)	1	0.5	1	0.5	0.5	0.5	0.5	1	5.5	0	1	1	2	7.5
(GOTLIEB, 2009)	1	1	0	0.5	0.5	0.5	1	1	5.5	0	1	1	2	7.5
(GROSS et al., 2009)	1	1	0	0.5	0.5	0.5	1	1	4.5	0	1	1	2	6.5
(GUAN et al., 2006)	1	1	0.5	0.5	1	0.5	1	1	6.5	0	1	1	2	8.5
(LAKEHAL; PARISSIS, 2005a)	1	1	0.5	0.5	0.5	0	0.5	1	5	0	1	1	2	7
(LAKEHAL; PARISSIS, 2005b)	1	0.5	1	0.5	0.5	0	0.5	0.5	4.5	0	1	1	2	6.5
(MATHAIKUTTY et al., 2007)	1	1	1	0	1	0	0.5	1	5.5	0	1	1	2	7.5
(PăSăREANU et al., 2008)	1	1	1	0.5	0.5	0	1	1	6	0	1	1	2	8
Continua na próxima página...														

Tabela B.1: *Avaliação da Qualidade dos Estudos*

Estudo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Subtotal(1-8)	Q9	Q10	Q11	Subtotal(9-11)	Total
(PAPAILIOPOULOU, 2008)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(SANTHANAM, 2001)	1	1	1	0	1	0.5	0.5	1	6	0	1	1	2	8
(YU; LAU, 2006)	1	1	1	0.5	1	1	1	1	7.5	0	1	1	2	9.5
(YU; LAU, 2004)	1	1	1	0.5	0.5	1	1	1	7	0	1	1	2	9
(LAKEHAL; PARISSIS, 2007)	1	1	0.5	0	1	0	1	1	5.5	0	1	1	2	7.5
(CARPENTER, 1999)	1	1	1	0	1	0	0.5	0.5	5	0	1	0	1	6
(CLANCY et al., 2006)	1	1	1	0.5	0.5	0.5	1	1	6.5	0	1	1	2	8.5
(DAVIDSON; AMOUSSOU, 2010)	1	1	0.5	0	0.5	0	0.5	1	4.5	0	1	1	2	6.5
(DENG; SANG, 2008)	1	1	0.5	0	0.5	0.5	0.5	1	5	0	1	1	2	7
(REZA et al., 2008)	1	1	1	0	0.5	0	1	0.5	5	0	1	1	2	7
(SANTIAGO et al., 2008a)	1	1	0.5	1	1	0.5	1	1	7	0	1	1	2	9
(TOMMASO et al., 2005)	1	1	0.5	0.5	0.5	0	1	0.5	5	0	1	1	2	7
(TSAI et al., 2000)	1	1	1	0	0.5	0	1	1	5.5	0	1	1	2	7.5
(TSAI et al., 2002)	1	1	0.5	0	1	0.5	0.5	1	5.5	0	1	1	2	7.5
(TSAI et al., 2005)	1	1	1	0	1	0.5	1	1	6.5	0	1	1	2	8.5
(TU et al., 1998)	1	1	0.5	0	0.5	0	1	1	5	0	1	0.5	1.5	6.5
(XU; WU, 1999)	1	1	1	0	0.5	0	1	1	5.5	0	1	1	2	7.5
(YU et al., 2009)	1	1	1	0	0.5	0	0.5	0	4	0	1	1	2	6
(MCDONALD et al., 2001)	1	1	1	0.5	1	0.5	1	1	7	0	1	1	2	9
(NEBUT et al., 2006)	1	1	1	0.5	0.5	1	1	1	7	0	1	1	2	9
(TRACEY et al., 2002)	1	1	1	1	1	0	1	1	7	0	1	1	2	9
(SWARUP; RAMAIAH, 2008)	1	1	1	0	0.5	0.5	0.5	0.5	5	0	1	1	2	7
(CHEON et al., 2005)	1	1	1	0	0.5	0	1	0	4.5	0	1	1	2	6.5
(CULLYER; STOREY, 1994)	1	1	1	0	1	0.5	0.5	0.5	5.5	0	1	1	2	7.5
Continua na próxima página...														

Tabela B.1: *Avaliação da Qualidade dos Estudos*

Estudo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Subtotal(1-8)	Q9	Q10	Q11	Subtotal(9-11)	Total
(HEIMDAHL, 2007)	1	1	0.5	0	0.5	0	1	0.5	4.5	0	1	1	2	6.5
(JOUNG et al., 2009)	1	1	1	0	0.5	0.5	1	0.5	5.5	0	1	1	2	7.5
(QIAN; ZHENG, 2009)	1	1	1	0	0.5	0	1	1	5.5	0	1	1	2	7.5
(YOON et al., 2005)	1	1	0.5	0	0.5	1	1	0.5	5.5	0	1	1	2	7.5
(ALAGAR et al., 2003)	1	1	1	0	1	0.5	1	1	6.5	0	1	1	2	8.5
(ANDREWS; ZHANG, 2000)	1	1	1	0	1	0	1	0.5	5.5	0	1	1	2	7.5
(BARBIER; BELLOIR, 2003)	1	1	1	0	0.5	0	1	0.5	5	0	1	1	2	7
(DONG et al., 2009)	1	1	1	0.5	0.5	0	1	1	6	0	1	1	2	8
(HAUSE et al., 2010)	1	1	1	0	0	0	1	0.5	4.5	0	1	0.5	1.5	6
(JAW et al., 2008)	1	1	1	0.5	0.5	0.5	1	1	6.5	0	1	1	2	8.5
(MORSCHHAUSER; LINDVALL, 2007)	1	1	0.5	0	1	0	1	1	5.5	0	1	1	2	7.5
(NAKAO; ESCHBACH, 2008)	1	1	1	0	1	0.5	0.5	1	6	0	1	1	2	8
(TRAKHTENBROT, 2005)	1	1	1	0	0.5	0	1	0.5	5	0	1	1	2	7
(TU; WU, 1999)	1	1	1	0	0.5	0	1	0.5	5	0	1	1	2	7
(TUMMALA et al., 2006)	1	1	1	0	0.5	0	1	1	5.5	0	1	0.5	1.5	7
(YIN et al., 2010)	1	1	1	1	0.5	0	1	1	6.5	0	1	1	2	8.5
(YIN; LIU, 2009)	1	1	1	0	0.5	0	1	0.5	5	0	1	0	1	6
(YU et al., 2010)	1	1	1	0.5	0	0	0.5	0	4	0	1	0	1	5
(KANDL et al., 2006)	1	1	1	0	0.5	0	1	0.5	5	0	1	1	2	7
(BELL; BRAT, 2008)	1	1	1	0.5	0.5	0.5	1	1	6.5	0.5	1	1	2.5	9
(GIANNAKOPOULOU, 2010)	1	1	0.5	0	0.5	0	1	0.5	4.5	0	1	0.5	1.5	6
(HAYHURST; VEERHUSEN, 2001)	1	1	1	0	1	0.5	1	1	6.5	0	1	1	2	8.5
(RAYADURGAM; HEIMDAHL, 2001)	1	1	1	0	0.5	0	1	0.5	5	0	1	1	2	7
(SMITH; URI, 2004)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5

Continua na próxima página...

Tabela B.1: *Avaliação da Qualidade dos Estudos*

Estudo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Subtotal(1-8)	Q9	Q10	Q11	Subtotal(9-11)	Total
(ABDUL-BAKI et al., 2000)	1	1	1	0	0.5	0	1	1	5.5	0	1	1	2	7.5
(BHOGARAJU et al., 2000)	1	1	1	0	0	0	1	1	5	0	1	1	2	7
(BLACKBURN; BUSSER, 1996)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(BRITT, 1994)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(CHANG et al., 1997)	1	1	1	0	0	0	1	1	5	0	1	1	2	7
(CHEON et al., 2004)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(SANTO, 1988)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(JASKE et al., 1996)	1	1	1	0	1	0	1	1	6	0	1	0	1	7
(KROPP et al., 1998)	1	1	1	0	1	0.5	1	1	6.5	0	1	1	2	8.5
(LI et al., 1999)	1	1	1	0	0.5	1	1	0.5	6	0	1	0	1	7
(MALEKZADEH; AINON, 2010)	1	1	1	0	0.5	0	1	1	5.5	0	1	0.5	1.5	7
(PEREZ; KAISER, 2009)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(NADEEM et al., 2006)	1	1	1	0	0	0	1	0	4	0	1	0.5	1.5	5.5
(TSAI et al., 2003)	1	1	1	0	0.5	0	1	0.5	5	0	1	0	1	6
(TUDOR et al., 2004)	1	1	1	0	0	0	1	0.5	4.5	0	1	1	2	6.5
(WU; HUANG, 2003)	1	1	1	0	0	0	1	1	5	0	1	0	1	6
(WU; LI, 1999)	1	1	1	0	0	0	1	1	5	0	1	0	1	6
(XIE, 2004)	1	1	1	0	0	0	1	0.5	4.5	0	1	0	1	5.5
(DUPUY; LEVESON, 2000)	1	1	1	0.5	0	0	1	0.5	5	0	1	0.5	1.5	6.5
(KUMAR et al., 2010)	1	1	1	0	1	0.5	1	1	6.5	0	1	0.5	1.5	8
(LOUBACH et al., 2006)	1	1	1	0	0.5	0.5	1	0.5	5.5	0	1	0	1	6.5
(MORTON, 1999)	1	1	1	0	0	0	1	0	4	0	1	0	1	5
(WANG, 2004)	1	1	1	0	0	0	1	0	4	0	1	0	1	5
(WANG; TAN, 2005)	1	1	1	0	0	0	1	0	4	0	1	0	1	5
Continua na próxima página...														

Tabela B.1: *Avaliação da Qualidade dos Estudos*

Estudo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Subtotal(1-8)	Q9	Q10	Q11	Subtotal(9-11)	Total
(COULTER, 1999)	1	1	1	0	0	0	1	0	4	0	1	0	1	5

Na Tabela B.2 é apresentado de forma agrupada o resultado da avaliação das oito primeiras questões do questionário de avaliação da qualidade dos estudos. Sendo que foram classificados segundo a pontuação total obtida nas respectivas questões.

Tabela B.2: *Nível de qualidade da estrutura e rigor dos estudos*

Índice de qualidade	**Fraco(0-4)**	**Regular(4.5-6.5)**	**Bom(7-8)**
IEEE	6	46	0
ACM	7	31	7
Número total de estudos	13	77	7
Percentual	13.4	79.38	7.22
Média	6.5	38.5	3.5
Desvio padrão	0.71	10.61	4.95

Na Tabela B.3 é apresentado de forma agrupada o resultado da avaliação das três últimas questões do questionário de avaliação da qualidade dos estudos. Sendo que foram classificados segundo a pontuação total obtida nas respectivas questões.

Tabela B.3: *Nível de credibilidade das evidências dos estudos*

Índice de qualidade	**Fraco(0-1)**	**Regular(1.5-2)**	**Bom(2.5-3)**
IEEE	13	39	0
ACM	8	36	1
Número total de estudos	21	75	1
Percentual	21.65	77.32	1.03
Média	10.5	37.5	0.5
Desvio padrão	3.54	2.12	0.71

Estudos primários excluídos após a realização da avaliação da qualidade

Na Tabela B.4 é apresentado a avaliação individual de cada estudo primário excluído após a conclusão da avaliação da qualidade dos estudos primários.

Tabela B.4: *Avaliação da Qualidade dos Estudos Excluídos*

Estudo	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Subtotal(1-8)	Q9	Q10	Q11	Subtotal(9-11)	Total
(CONRAD et al., 2005)	1	0.5	0.5	0.5	0	0	0.5	0.5	3.5	0	0	0	0	3.5
(EN-NOUAARY et al., 2000)	1	0.5	0	0	0	0.5	0.5	0	2.5	0	0	0	0	2.5
(HOTE, 2005)	1	0.5	0.5	0	0	0.5	0.5	0	3	0	0	0	0	3