Direcionadores para o Desenvolvimento Baseado em Modelos de Software Embarcado Aeronáutico

André Sarkis, Johnny Marques, Luiz Alberto Vieira Dias

Instituto Tecnológico de Aeronáutica (ITA) – São José dos Campos, SP – Brasil

sarkis.andre@gmail.com, {johnny, vdias}@ita.br

Abstract. *Model Based Development (MBD) can be used to specify requirements at various levels: product, system, or software. In the aviation area, the use of regulatory standards in the certification process is usually required. The DO-178C and its supplements define the needs of the software development process to allow the certification of the software product. In recent years, the growth in the use of modelling has been remarkable. Thus, the objective of this paper is to present a set of drivers for MBD in Embedded Aeronautical Software, ensuring compliance with the RTCA DO-178C and DO-331. In addition to the drivers, a case study is presented.*

Resumo. *O Desenvolvimento Baseado em Modelos (MBD, Model-Based Development) pode ser usado para especificar requisitos em vários níveis: produto, sistema, ou de software. Na área de aviação, o uso de normas regulatórias no processo de certificação é usualmente requerido. A RTCA DO-178C e seus suplementos legislam sobre as necessidades que o processo de desenvolvimento de software precisa atender para tornar o produto de software certificável. Nos últimos anos, é notável o crescimento no uso de modelagem. Assim, o objetivo deste artigo é apresentar um conjunto de direcionadores para o MDB em Software Embarcado Aeronáutico, garantindo o atendimento às normas DO-178C e DO-331. Além dos direcionadores, é apresentado um estudo de caso.*

1. Introdução

A segurança no desenvolvimento de software é fundamental para sistemas críticos, já que um mal funcionamento do mesmo poder causar a perda de vidas humanas. Exemplos de sistemas críticos incluem sistemas que controlam aeronaves, reatores nucleares e dispositivos médicos [Marques 2019]. As agências reguladoras em setores críticos para a segurança geralmente exigem que os desenvolvedores de sistemas com software atendam aos rigorosos requisitos de certificação, como a RTCA DO-178C [RTCA 2011a] na aviação.

As partes interessadas da indústria da aviação variam em seus campos específicos de especialização e em sua contribuição para a segurança. Por exemplo, existem colaboradores nas áreas de motores, estruturas, aviônicos, sistemas de cabine, sistemas elétricos, sistemas de pouso e direção, entre outros [Ferrell 2017].

De acordo com Kennedy e Towhidnejad (2017), a estratégia de definição de suplementos da RTCA DO-178C permite a adoção de novas tecnologias. Esses suplementos acomodam o uso crescente de novos métodos e técnicas, principalmente métodos formais, programação orientada a objetos e Desenvolvimento Baseado em Modelos (MBD) [Daniels 2011].

Para que isso seja adequado, Kennedy e Towhidnejad (2017) também destacaram que houve um esforço feito para garantir que os suplementos fossem lançados a tempo de apoiar adoção quando necessário. As questões de custo e complexidade em software de aviação também implicam que novos métodos e técnicas são desejáveis. No campo de MBD, o suplemento RTCA DO-331 foi definido para permitir o uso do desenvolvimento baseado em modelos.

O Desenvolvimento Baseado em Modelos (MBD) pode ser usado para especificar requisitos em vários níveis: aeronave, sistema, ou de software. O objetivo deste artigo é apresentar um conjunto de direcionadores para o MBD em software embarcado aeronáutico, visando o atendimento às normas RTCA DO-178C e DO-331. As Questões de Pesquisa identificadas pelos autores são apresentadas na Tabela 1.

Tabela 1. Questões de pesquisa

ID	Questão de Pesquisa	Justificativa
QP1	Como implementar a rastreabilidade dos modelos para outros artefatos de desenvolvimento?	É necessário a confirmar que os modelos foram desdobrados corretamente a partir dos requisitos e corretamente implementados no código-fonte.
QP2	Como diminuir a complexidade dos modelos?	Associado à QP 1, a rastreabilidade pode ser mais facilmente implementada pela diminuição da complexidade dos modelos com aumento dos níveis de abstração.

2. Trabalhos Correlatos

Apesar de ser uma tecnologia relativamente nova, com pouco mais de uma década de existência, a adoção do MBD pela indústria de software é notável. Em uma pesquisa realizada na indústria italiana [Tomassetti et al. 2012] foi identificado que mais de dois terços dos entrevistados utilizam modelos em seu desenvolvimento, sendo que a metade também adotavam alguma técnica do MBD.

Em outro estudo sobre como o MBD é utilizado na indústria [Hutchinson et al. 2011], descobriu-se que 83% dos entrevistados identificavam o Desenvolvimento Baseado em Modelos como algo positivo e benéfico. Um terceiro estudo consistiu em um experimento desenvolvido com estudantes de pós-graduação, visando comparar o uso do MBD com métodos tradicionais [20]. Os resultados da pesquisa mostraram que os estudantes de pós-graduação, potenciais engenheiros júnior de software, consideram o uso de modelos para geração de código como a metodologia mais útil, e estariam dispostos a continuar usando este método.

As três pesquisas também mostraram algumas barreiras que ainda impedem que o uso do MBD seja mais difundido. Falta de treinamento e conhecimento dos designers e ferramentas ainda imaturas são apontados como principais empecilhos para a adoção do MBD [Tomassetti et al. 2012] [Hutchinson et al. 2011]. Os estudantes de pós-graduação também apontaram o MBD como uma experiência diferente do que eles tiveram durante a graduação, o que ocasionou que a curva de aprendizagem fosse maior que o necessário [Martínez et al. 2013]. Isto indica uma necessidade universidades adotarem os novos paradigmas de engenharia de software em sua grade curricular.

Aproximando mais da indústria aeronáutica, temos como exemplo de aplicação de MBD um projeto desenvolvido pela Lockheed Martin [OMG 2020]. A empresa usou a metodologia para o desenvolvimento do software do MMC (Modular Mission Computer, ou Computador de Missão Modular, em uma tradução livre) para o F-16. Os modelos gerados pela companhia foram utilizados para geração automática de código e validação do design.

Segundo a Lockheed, a utilização do MBD reduziu em 20% o tempo de desenvolvimento do software. Aliado ao ganho de desempenho, outros benefícios são citados, como a capacidade de simular os modelos; a independência do modelo em relação à plataforma, promovendo a reusabilidade; a eliminação da codificação manual e, com ela, a eliminação dos erros inerentes desta atividade e o aumento do nível de abstração, permitindo aos desenvolvedores focar no design do software e não em questões específicas de programação.

O exemplo da Lockheed Martin é muito valioso para a indústria aeronáutica, porém, não pode ser esquecido que o F-16 é uma aeronave militar, e como tal, não está sujeita as normas de certificação aeronáutica que as aeronaves civis estão submetidas. Neste caso é importante tentar encaixar o MBD dentro do contexto das normas DO-178C e DO-331.

Marques et al. (2013) apresentaram como que o MBD poderia ser utilizado com a RTCA DO-178B, de 1992. Este trabalho apresentou as perspectivas futuras da época, antes da definição das recentes normas RTCA DO-178C e RTCA DO-331 [RTCA 2011b]. Este trabalho apresentou os impactos no planejamento, desenvolvimento e verificação de software pelo uso de MBD.

Honda e Dias (2013) apresentaram uma visão geral do processo de aprovação de software embarcado aeronáutico, com foco no MBD, além de introduzir uma versão preliminar do desenvolvimento do *Range Analyzer*, uma ferramenta com a capacidade de detectar ocorrências de estouro aritmético.

Marques e Cunha (2018) apresentaram como podem ser adaptados quatro ciclos de vida de desenvolvimento de software tradicionais (em cascata, V, em espiral e incremental), quando o PDM é usado. Este documento fornece uma maneira genérica de organizar esses ciclos de vida, satisfazendo os objetivos RTCA DO-178C e DO-331 para um software de Nível A.

Hochstrasser et al. (2018) apresentaram uma ferramenta de construção monolítica, orientada a processos, para desenvolvimento baseado em modelo no MATLAB, Simulink e Stateflow.

Porfírio et al. (2019) descrevem um trabalho acadêmico de simulação de requisitos de HLR em casos de garantia de software embarcado aeronáutico, segundo um guia para *Simulation-Based Studies* (SBS).

Marques et al. (2020) apresentaram um arcabouço adaptado do Scrum (Q-Scrum), integrando o MBD para desenvolvimento, aferição de qualidade, confiabilidade e segurança de software em sistemas críticos, usando a ferramenta ANSYS Scade Suite.

3. Normas para o Desenvolvimento de Software Embarcado Aeronáutico

Usualmente, as normas não definem como o processo de desenvolvimento de software deve ser estabelecido e sim quais necessidades ele precisa atender. No caso do software embarcado aeronáutico, a DO-178C apresenta um conjunto de objetivos que o processo de desenvolvimento de software deverá atender.

A DO-178C define que os requisitos de software sejam desenvolvidos em dois níveis: os requisitos de alto nível (HLR, *High Level Requirements*) e requisitos de baixo nível (LLR, *Low Level Requirements*).

Os HLR são os requisitos de software desenvolvidos diretamente a partir dos requisitos e da arquitetura do sistema. Os HLR descrevem as funcionalidades, as interfaces e os atributos de performance [Rierson 2013].

Os LLR são os requisitos desdobrados a partir dos HLR, portanto, devem descrever os detalhes da implementação dos HLR. Os LLR devem ser detalhados a ponto que o código possa ser gerado a partir deles sem necessidade de informação adicional.

A DO-178C define cinco níveis para desenvolvimento de software. Os níveis vão de A até E, determinando diferentes quantidades de objetivos. Os níveis de software são estabelecidos em correlação com o nível de severidade da condição de falha que um erro de software pode causar. A Tabela 2 traz a correlação entre o nível de software e a severidade da condição de falha, além da quantidade de objetivos associados a cada nível de software.

Tabela 2. Níveis de software

Severidade da condição de falha	Nível de software	Nº de objetivos
Catastrófica	A	71
Perigosa	B	69
Maior	C	62
Menor	D	26
Sem impacto em segurança	E	0

A DO-331 [RTCA 2011b], é um documento produzido pelo mesmo comitê que realizou a revisão da norma DO-178C. Ela é considerada um suplemento da DO-178C

quando o MBD é utilizado. A DO-331 proporciona flexibilidade no uso de modelos ao tendo em seus objetivos, mitigações para propiciar que o software produzido cumpra a função desejada e não tenha nenhuma função indesejável.

A DO-331 define dois tipos de modelos:

- Modelo de Especificação (ME): representa os HLR, provendo uma representação abstrata de características funcionais, de performance, de interface ou de safety de componentes de software. O ME deve somente conter detalhes que permitam o entendimento da funcionalidade, mas não deve guiar uma implementação ou arquitetura específica; e
- Modelo de Design (MD): define a estrutura interna de dados, fluxo de dados e/ou fluxo de controle do componente de software, além de poder definir LLR e/ou arquitetura. O modelo de design também contempla modelos usados para gerar código.

A DO-331 apresenta alguns possíveis cenários do uso de modelos no desenvolvimento de software crítico. Os cenários contemplam o uso de ME, MD ou ambos. Neste trabalho, os autores apenas consideram o uso de MD, realizado a partir de requisitos HLR textuais.

4. Direcionadores para o Desenvolvimento Baseado em Modelos

No escopo deste trabalho foi utilizado o seguinte cenário de MBD:

- Existem requisitos textuais que descrevem os HLR, não usando o ME; e
- O MD satisfaz os HLR.

Esse cenário permite que seja explorado a aplicação de Modelos de Design, a relação entre requisitos textuais e modelos, além da transformação dos modelos em código-fonte, conforme apresentado Figura 1.

Os motivos para limitar a este único cenário são:

- Representatividade: possibilidade de explorar a interface entre HLR textuais e Modelos de Design; e
- Abordar os objetivos de Design, que são em maior número na DO-178C e DO-331.

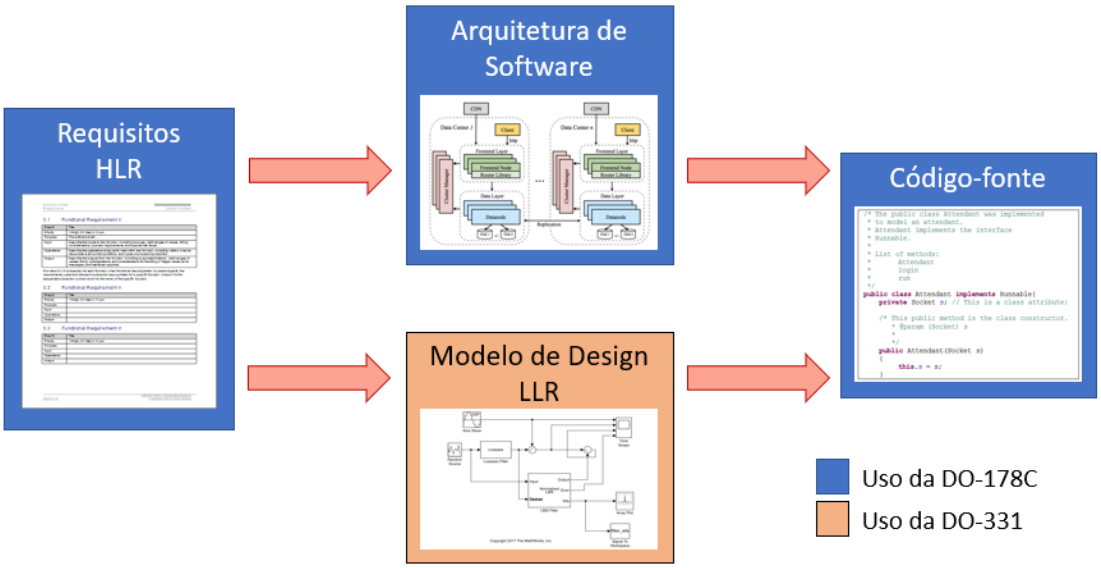

Figura 1: Cenário de aplicação de Modelos de Design (MD)

O conjunto de direcionadores é livre de uma ferramenta específica e foi elaborado de forma a ser independente das etapas anteriores e posteriores ao desenvolvimento dos modelos. Ao todo este trabalho apresenta 3 direcionadores iniciais:

- Direcionador D1: Definir um Padrão de Definição dos Modelos (PDM);
- Direcionador D2: Usar blocos funcionais para agrupar blocos fundamentais;
- Direcionador D3: Mapear a rastreabilidade dentro do próprio modelo; e
- Direcionador D4: Separar a arquitetura de software dos requisitos.

4.1. Definir um Padrão de Definição dos Modelos (PDM) – Direcionador D1

O PDM, também conhecido como *Software Model Standard*, e é um documento que deve conter as restrições e regras que devem ser seguidas pelos desenvolvedores para a confecção dos modelos, permitindo que os modelos gerados pelo processo de desenvolvimento sejam uniformes e não-ambíguos.

Algumas das informações que devem estar presente no PDM são:

- Ferramentas e linguagens de modelagem a serem utilizadas;
- Regras de estilo a serem adotadas, como a convenção de nomes, o número máximo de níveis do modelo, quantidade máxima de elementos por diagrama, entre outros; e
- Meios de identificar os blocos que não estão contribuindo para os requisitos e/ou a arquitetura como, por exemplo, um bloco de comentários. É recomendado que haja uma distinção visual entre um bloco comum, que está implementando alguma funcionalidade, e um de comentário.

É importante que sejam consideradas as especificidades de cada ferramenta de modelagem selecionada. Os formatos dos blocos, a maneira como os parâmetros de configuração são usados, como as conexões entre os blocos são feitas, devem estar definidos no PDM. Outro aspecto importante que deve estar refletido é como as bibliotecas de modelagem podem ser utilizadas. Caso algum elemento da biblioteca não seja utilizado, seja por preferência do desenvolvedor, seja por limitação do sistema, o mesmo deve ser marcado como proibido.

O PDM deve também conter os blocos criados especialmente para o desenvolvimento. Por exemplo, suponha que um bloco de filtro seja definido usando blocos fundamentais da ferramenta e esse bloco seja reutilizado diversas vezes durante a modelagem. Neste caso, é interessante que este bloco esteja definido no PDM, para deste modo possibilitar que ele seja utilizado corretamente e consistentemente ao longo de toda a modelagem.

Ter esses blocos especiais definidos e padronizados auxilia na atividade de validação do modelo, uma vez que aumenta a certeza de que o bloco está sendo utilizado corretamente. A criação, internalização e adoção do PDM contribui diretamente para o sucesso em atender os objetivos da DO-178C e DO-331, respectivamente.

4.2. Usar blocos funcionais para agrupar blocos fundamentais – Direcionador D2

Durante o desenvolvimento do modelo, é comum que os blocos fundamentais, disponibilizados pelas bibliotecas da ferramenta, sejam usados para gerar funcionalidades mais abrangentes como, por exemplo, filtros, contadores e conversores de dados. Normalmente, essas funcionalidades são reusadas várias vezes ao longo do modelo.

A funcionalidade poderia ser identificada usando linhas tracejadas ao redor do conjunto de blocos fundamentais e usando um marcador próximo a eles. Porém, várias ferramentas possibilitam a criação de blocos personalizados pelo usuário. Neste caso, é sugerido o uso de um agrupamento dentro do bloco personalizado.

A adoção de blocos funcionais auxilia na leitura do modelo, já que abstrai uma grande quantidade de blocos fundamentais em um bloco que representa uma funcionalidade tangível. É importante que o bloco funcional tenha significado por si só. Portanto, nomeie-os de forma significativa, que sintetizem a função principal do bloco, evitando nomes genéricos como “Meu bloco 1”, por exemplo. Também é recomendável que os blocos criados pelo usuário sejam incorporados ao PDM, promovendo a consistência no reuso dos blocos funcionais ao longo do modelo.

O uso de blocos funcionais auxilia na consistência do modelo, facilita a leitura do mesmo e colabora com a demonstração de cumprimento com os HLR, uma vez que eles estão mais próximos do nível de detalhe que os requisitos textuais possuem.

4.3. Mapear a rastreabilidade dentro do próprio modelo – Direcionador D3

Segundo a DO-178C, deve haver rastreabilidade bidirecional para os requisitos LLR, que envolve o rastreio tanto para os HLR, como para o código fonte, além de rastreio para os

casos de teste relacionados com os requisitos. Num primeiro momento, duas alternativas surgem para registrar a rastreabilidade: criar uma matriz em um documento à parte ou mantê-la dentro do próprio modelo.

O uso de uma matriz permite uma rastreabilidade mais limpa e organizada, porém torna-se mais um artefato incorporado aos dados de ciclo de vida do software, gerando necessidade de mantê-lo sobre controle de configuração e constante revisão em cada modificação do modelo. Assim, este trabalho sugere a segunda opção que é manter a rastreabilidade dentro do próprio modelo.

Manter a rastreabilidade incorporada ao modelo, por outro lado, pode a princípio deixá-lo mais poluído, porém oferece vantagens, como permitir identificar a rastreabilidade de um modo visual e intuitivo, eliminar a necessidade de um artefato extra e propiciar melhor entendimento entre o modelo e o requisito e código ao qual ele está relacionado. Por isso, é recomendável que se mantenha as informações de rastreabilidade dentro do próprio modelo.

A Figura 2 mostra um exemplo de rastreabilidade feita no próprio modelo, onde os elementos em ciano, apresentam a rastreabilidade do modelo para o código-fonte. Já os elementos em verde apresentam a rastreabilidade para os HLRs e os elementos em rosa, a rastreabilidade para os casos de teste.

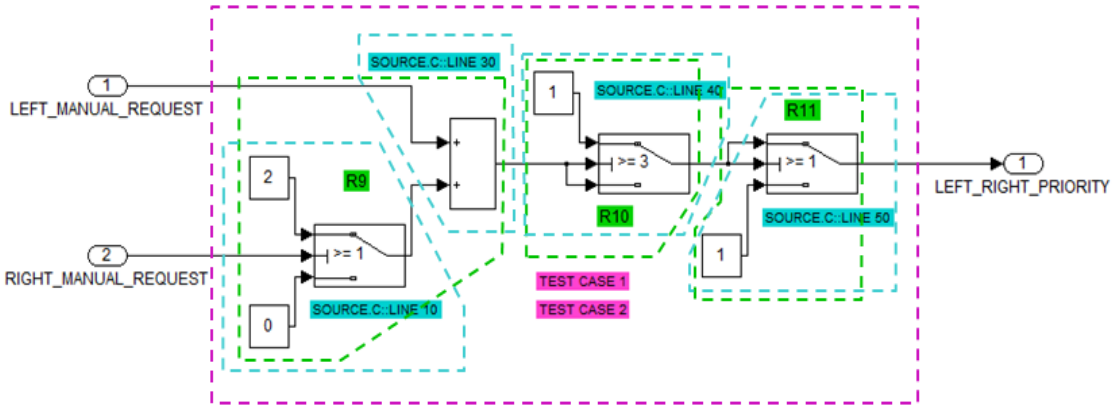

Figura 2: Exemplo de rastreabilidade

4.4. Separar a arquitetura de software dos requisitos – Direcionador D4

Para uma melhor compreensão do modelo, é adequado que haja uma separação entre a arquitetura de software e os requisitos de baixo nível. A separação traz maior clareza do cumprimento dos objetivos aplicáveis a ambos.

É recomendável que o primeiro nível do modelo represente a arquitetura, trazendo os componentes de software e a interrelação de dados e controle entre eles. Provavelmente, o modelo por si só não tem capacidade de representar completamente o fluxo de dados e o de controle. Neste caso, pode-se usar blocos adicionais, ou etiquetas, para descrever a sequência de como os controles e os dados são transmitidos entre os diversos componentes. Caso haja muitas interações, é recomendável que um diagrama de fluxo de dados e de controle seja elaborado conjuntamente com o modelo.

Os níveis abaixo da arquitetura devem detalhar como as funções alocadas para aquele componente estão sendo implementadas e devem representar requisitos, que são entrada para a geração de código, quer ele seja manual ou automático.

Esse direcionador auxilia na demonstração que tanto a arquitetura quantos os requisitos são compatíveis com os HLR e que são consistentes e acurados, uma vez que são representados separadamente.

5. Exemplo no Uso dos Direcionadores

Aplicamos os direcionadores em um sistema de gerenciamento de displays aviônicos. Neste exemplo, utilizou-se a ferramenta de modelagem Mathworks Simulink (versão R2019b) para gerar o modelo sendo representado nas figuras 2, 4, 5 e 6. A Figura 3 mostra uma arquitetura do sistema simplificado, contendo duas Unidades de Display (DU). Elas estão conectados a dois sistemas computacionais redundantes, que executam três aplicações: PFD (*Primary Flight Display*) contendo informações básicas do voo, como velocidade, altitude e atitude da aeronave, EICAS (*Engine-Indicating and Crew-Alerting System),* que apresenta informações sobre os motores, além de mensagens sobre a saúde da aeronave e MAP (Mapa), usado para a navegação. Os computadores também executam a aplicação WM (*Window Manager*), responsável pelo gerenciamento de displays.

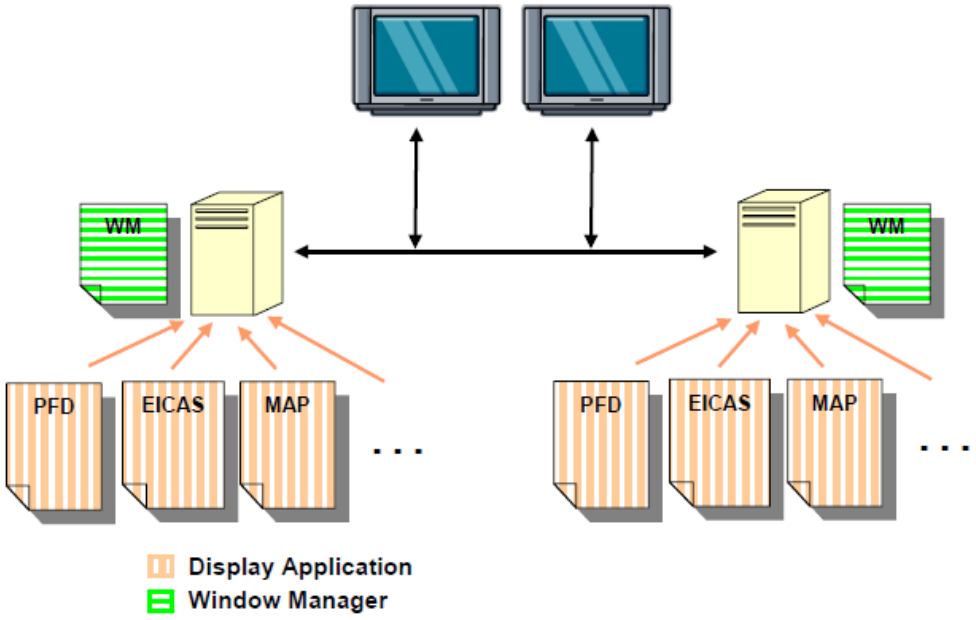

Figura 3: Arquitetura do sistema simplificado de gerenciamento dos displays aviônicos [RTCA 2011c]

O software de WM possui diversas funcionalidades. A Tabela 3 apresenta os requisitos os High-Level Requirements. Esses requisitos foram retirados do trabalho de Sarkis (2014), que foram adaptados de RTCA DO-333 [RTCA 2011c].

Tabela 3: HLR para o WM

Identificação do Requisito	Texto do Requisito (*em inglês*)

Req1	If a DU is available, then the system shall set one of the applications to that DU.
Req2	If a DU is unavailable, then the system shall set DU to blank.
Req9	Case both DU have MAP displayed, the system shall set the cursor to the DU that requested priority over the cursor.
Req10	Case both DU have requested priority over the cursor, the system shall set the cursor to the left DU.
Req11	Case none DU have requested priority over the cursor, the system shall set the cursor to the left DU.

Com o conjunto de requisitos HLR definido, o MD pode ser desenvolvido. A Figura 4 contém o primeiro nível do MD. Neste primeiro nível, temos a representação arquitetônica do sistema, que possui dois componentes: o DU_APPLICATION_SELECTION, responsável por indicar para cada DU qual a aplicação que ela deve exibir, e o CURSOR_PLACEMENT, responsável por indicar em qual DU o cursor deve ser posicionado.

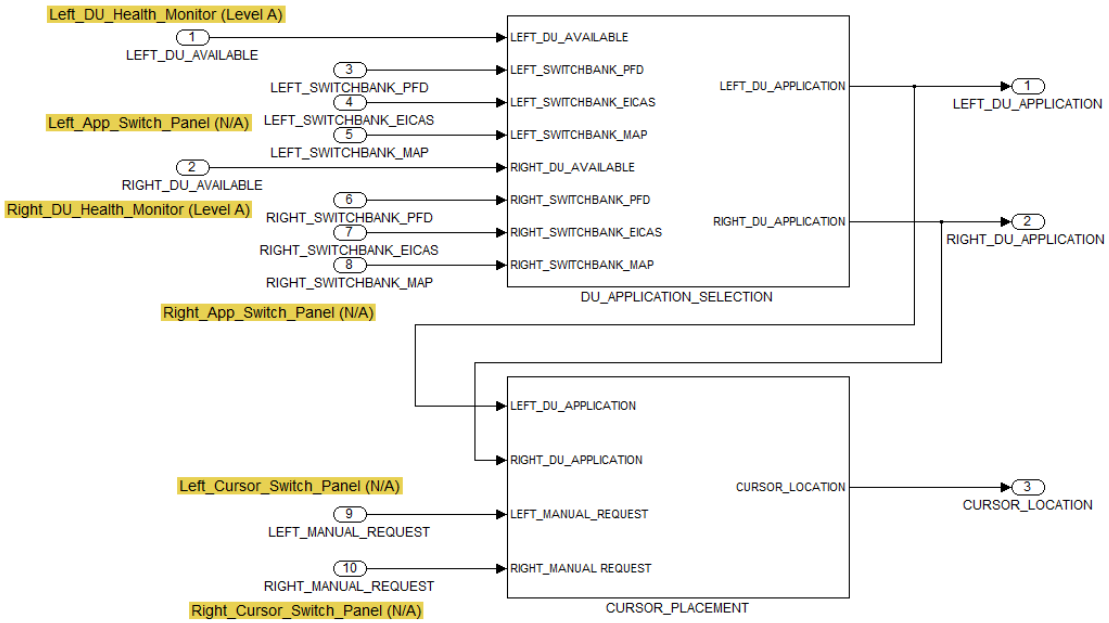

Figura 4: Modelo de Design - 1º nível do modelo (representação arquitetônica)

A Figura 5 exibe o segundo nível de software, detalhando a lógica do componente DU_APPLICATION_SELECTION.

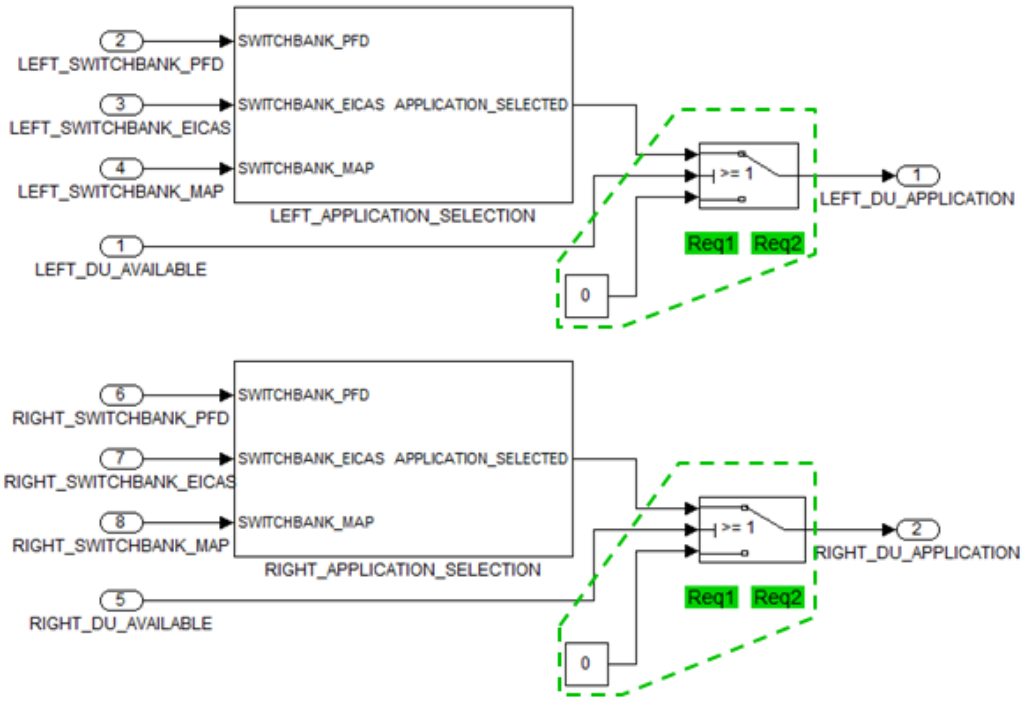

Figura 5: Modelo de design - 2º nível do modelo - componente DU_APPLICATION

A Figura 6 mostra o terceiro nível do modelo, detalhando a funcionalidade CURSOR_PRIORITY dentro do componente CURSOR_PLACEMENT. Esta funcionalidade implementa parte do requisito Req9, além dos Req10 e Req11. De forma geral, a prioridade sobre o cursor é dada a DU que a requisitou. Caso ambas as Unidades de Display (DUs), ou nenhuma delas, tenham pedido prioridade, então a DU da esquerda é indicada como a prioritária.

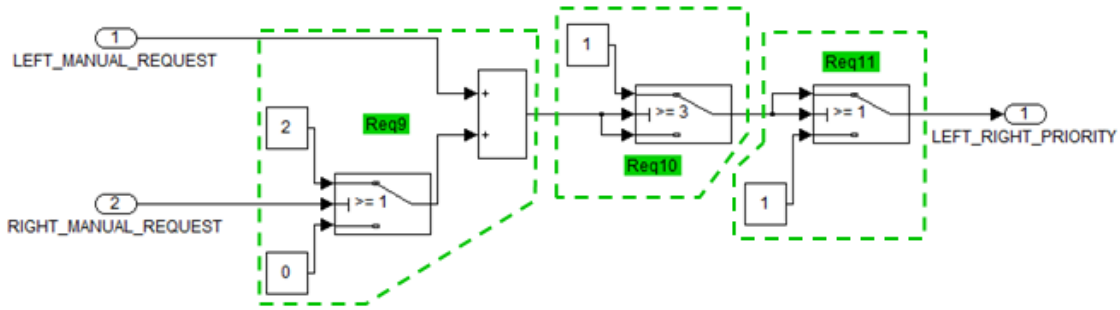

Figura 6: Modelo de design - 3º nível do modelo - funcionalidade CURSOR_PRIORITY

5.1. Avaliação dos Direcionadores no Estudo de Caso

O PDM (direcionador D1) define regras para os nomes das variáveis e funções utilizadas, quais blocos podem e não podem ser usados e quais tipos de variáveis (inteiro, real, booleana) são aceitáveis. Essa padronização provida pelo PDM melhora a leitura e compreensão do modelo, favorecendo a compatibilidade do modelo criado com outros do mesmo sistema.

O uso de blocos funcionais para agrupar blocos fundamentais (direcionador D2) foi utilizado no exemplo da aplicação WM, onde foram usados dois blocos funcionais, o bloco LEFT/RIGHT_DU_APPLICATION_SELECTION, usados dentro do componente DU_APPLICATION e o bloco CURSOR_PRIORITY, parte do componente CURSOR_PLACEMENT. Esses blocos permitiram o reuso e melhoram a leitura do modelo, reunindo uma sequência de switches em um bloco funcional, que indica através de seu nome qual sua função. A aplicação deste direcionador D2 facilita a análise do modelo quanto a consistência e acurácia.

O mapeamento da rastreabilidade dentro do próprio modelo (direcionador D3) é percebido nos requisitos identificados no próprio modelo. Desta forma, é simples observar se todos os blocos têm ao menos um requisito associado. A Figura 2 mostra um exemplo de rastreabilidade completa.

A separação da arquitetura de software dos requisitos (direcionador D4) é percebida no exemplo da aplicação WM, onde a arquitetura do sistema foi definida no primeiro nível, enquanto os níveis inferiores focaram na implementação dos requisitos de baixo nível. Desta forma, as questões de arquitetura, como quais componentes formam o sistema e a relação entre eles fica separada da implementação dos requisitos, permitido uma análise mais focada. Dessa forma, o cumprimento com os objetivos específicos de arquitetura é mais direto.

5.2. Avaliação dos Direcionadores quanto às normas RTCA DO-178C e DO-331

A Tabela 4 apresenta a rastreabilidade entre os direcionadores com os objetivos aplicáveis das normas DO-178C e DO-331.

Tabela 4: Matriz objetivos das normas e direcionadores

Norma	Objetivo da Normas	D1	D2	D3	D4
DO-178C	A2-3: Arquitetura de software desenvolvida				X
DO-178C	A2-5: Os requisitos de baixo nível derivados são definidos.			X	
DO-178C	A4-8: Arquitetura de software é compatível com os HLRs.				X
DO-178C	A4-9: Arquitetura de software é consistente.				X
DO-178C	A4-12: Arquitetura de Software está em conformidade com o PDM.	X			
DO-178C	A4-1: Os requisitos de baixo nível estão em conformidade com os requisitos de alto nível.			X	
DO-178C	A4-2: Os requisitos de baixo nível são precisos e consistentes.		X		

DO-178C	A4-5: Os requisitos de baixo nível estão em conformidade com o PDM.	X			
DO-178C	A4-6: Os requisitos de baixo nível são rastreáveis aos requisitos de alto nível.			X	
DO-331	MB A2-10: Os elementos do modelo de design que não contribuem para a implementação ou realização de qualquer requisito de baixo nível são identificados	X		X	

6. Conclusão

O objetivo deste artigo foi apresentar um conjunto de direcionadores para o MBD em software embarcado aeronáutico, visando o atendimento às normas RTCA DO-178C e DO-331. As Questões de Pesquisa (QP) identificadas foram respondidas conforme a Tabela 5.

Tabela 5. Questões de pesquisa e respostas

ID	Questão de Pesquisa	Resposta
QP1	Como implementar a rastreabilidade dos modelos para outros artefatos de desenvolvimento?	A rastreabilidade para os requisitos HLR foram implementadas a partir da identificação das partes dos modelos que atendem cada requisito identificado. Respeitando o direcionador D3, as Figuras 4 e 5 apresentam um exemplo de rastreabilidade dos modelos para os requisitos HLR.
QP2	Como diminuir a complexidade dos modelos?	A redução de complexidade dos modelos foi tratada com o aumento dos níveis de abstração. Respeitando o direcionador D2, as Figuras 3, 4 e 5 apresentaram os três níveis de abstração avaliados no estudo de caso. O direcionador D4, também é importante nessa diminuição, já que as necessidades de arquitetura ficam expressas no primeiro nível de modelagem, cabendo aos dois níveis seguintes o detalhamento do Design.

Para o direcionador D1, o Padrão de Definição dos Modelos (PDM) pode identificar regras específicas que oriente para como estabelecer a rastreabilidade, logo definindo regras específicas que podem especializar o direcionador D3. O mesmo acontece com o direcionador D2, ao se estabelecer, por exemplo, regras que definam um número máximo de blocos que um modelo pode ter e qual o número máximo de níveis de abstração permitido. Finalmente, o direcionador D4 auxilia na demonstração que tanto a arquitetura quantos os requisitos são compatíveis com os HLR e que são consistentes e acurados, uma vez que são representados separadamente.

Os autores concluíram que os 4 direcionadores aqui apresentados são úteis na integração dos modelos ao ciclo de vida do software, mantendo ainda compatibilidade e aderência com as normas RTCA DO-178C e RTCA DO-331.

Uma primeira limitação deste trabalho foi a escolha do cenário explorado, que descartou o uso de Modelos de Especificação (ME), focando essencialmente nos Modelos de Design, onde prevaleceu o foco na arquitetura (1º nível de modelagem) e requisitos de baixo nível (2º e 3º nível de modelagem).

Uma segunda limitação foi a manutenção da rastreabilidade incorporada ao modelo, que o deixa mais poluído, porém oferece vantagens, como permitir identificar a rastreabilidade de um modo visual e intuitivo, eliminar a necessidade de um artefato extra e propiciar melhor entendimento entre o modelo e o requisito e código ao qual ele está relacionado. Ainda relacionado à rastreabilidade, a seção 4.3 apresentou um exemplo de rastreabilidade do modelo para requisitos de alto-nível, código-fonte e testes, no entanto, o exemplo apresentado na seção 5 focou apenas na rastreabilidade para requisitos.

Como um primeiro trabalho futuro, os autores pretendem através dos direcionadores montar um arcabouço operacional com uma proposta de PDM para uso com o Mathworks Simulink e integrar um procedimento operacional de como utilizar esses direcionadores em um formato que possa ser de interesse das empresas do setor de aviação, em especial que desenvolvem software dessa natureza.

Em um segundo trabalho futuro, os autores pretendem conduzir uma investigação mais profunda de como os direcionadores podem ser usados e adaptados para o uso em cenários que envolvam o uso de Modelos de Especificação (ME).

Finalmente, um terceiro trabalho futuro é a integração desses direcionadores com o trabalho de Marques e Cunha (2018), que apresenta uma customização de ciclos de vida tradicionais, com o uso de Desenvolvimento Baseado em Modelos para software embarcado aeronáutico. Este futuro trabalho permitirá a incorporação dos direcionadores em um processo de software estabelecido sobre estes ciclos.

Referências

Daniels, D. (2011) “Thoughts from the DO-178C committee”, 6th IET International Conference on System Safety.

Ferrell, U. (2018) “Mindful Application of Standards for Avionics - An Intentional, Systematic, and Measurable Transformation”, 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC).

Hochstrasser, M., Myschik, S. e Holzapfel, F. (2018). “Application of a Process-Oriented Build Tool for Flight Controller Development Along a DO-178C/DO-331 Process”, International Conference on Model-Driven Engineering and Software Development (Modelsward 2018).

Honda, R. M. e Dias, L. A. V. (2013) “Range Analyzer: An Automatic Tool for Arithmetic Overflow Detection in Model-Based Development”, 10th International Conference on Information Technology: New Generations (ITNG 2013).

Hutchinson, J. Whittle, J., Rouncefield, M., Kristoffersen, S. (2011) “Empirical Assessment of MDE in Industry”. 33rd International Conference on Software Engineering (ICSE 2011).

Kennedy J.D., Towhidnejad, M. (2017) “Innovation and Certification in Aviation Software”, 2017 Integrated Communications, Navigation and Surveillance Conference (ICNS 2017).

Marques, J. C. e Cunha, A. M. (2018). “Tailoring Traditional Software Life Cycles to Ensure Compliance of RTCA DO-178C and DO-331 with Model-Driven Design”, 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC).

Marques, J. C. e Cunha, A. M. (2019). “ARES: An Agile Requirements Specification Process for Regulated Environments”, International Journal of Software Engineering and Knowledge Engineering, Vol.29, No. 10, pp-1403-1438.

Marques, J. C., Yelisetty, S. M. H., Cunha, A. M. e Dias, L. A. V. (2012) “Using Model-Based Development as Software Low-Level Requirements to Achieve Airborne Software Certification”, 9th International Conference on Information Technology: New Generations (ITNG 2012).

Marques, J., Cunha, A. M. e Dias, L. A. V. (2020). “Q-Scrum: A Framework for Quality in Safety-Critical Development”, 13th International Conference on the Quality of Information and Communications Technology (QUATIC 2020).

Martínez, Y., Cachero, C.; Meliá, S. (2013) “MDD vs. traditional software development: A practitioner’s subjective perspective”, Information and Software Technology, v. 55, n. 2, p. 189-200.

OMG. (2020) Lockheed Martin (MDA SUCCESS STORY). Success Stories - OMG. Disponível em: <http://www.omg.org/mda/mda_files/LockheedMartin.pdf>. Acesso em: 24 de setembro de 2020.

Porfírio, E. J., Sousa, F. J. C., Velasco, G. C. e Carvalho, S.T (2019) "Simulação de Requisitos de Alto Nível em Casos de Garantia de Software Aeroembarcado". I Workshop em Modelagem e Simulação de Sistemas Intensivos em Software (MSSiS 2019).

Rierson, L. (2013). Developing Safety-Critical Software: A Pratical Guide for Aviation Software and DO-178C Compliance, CRC Press.

RTCA (2011a). DO-178C Software Considerations in Airborne Systems and Equipment Certification.

RTCA (2011b). DO-331 Model-Based Development and Verification Supplement to DO-178C and DO-278A.

RTCA (2011c). DO-333 Formal Methods Supplement to DO-178C and DO-278A.

Sarkis, A. (2014) “Um Conjunto de Regras para a Produção de Modelos de Design em Observância com as Normas DO-178C e DO-331.”. Instituto Tecnológico de Aeronáutica.

Tomassetti, F., Torchiano, M., Tiso, A., Ricca, F., Reggio, G. (2012) “Maturity of Software Modelling and Model Driven Engineering: a Survey in the Italian Industry”, 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012).