Simulação de Requisitos de Alto Nível em Casos de Garantia de Software Aeroembarcado

Emerson José Porfírio, Fernando Junio Cunha e Sousa, Gislainy Crisostomo Velasco, Sérgio T. Carvalho

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 — 74001-970 — Goiânia — GO -– Brasil

{emerson.porfirio,fernandosousa,gislainyvelasco,sergio}@inf.ufg.br

Abstract. *Context: A Software Assurance Case is an audit instrument used to ensure that software for critical systems is safe in accordance with standards established by the responsible bodies. In the aeronautical software industry, for example, such cases are used to describe the activities prescribed by the Software Considerations in Airborne Systems and Equipment Certification (RCTA DO-178C) without which the software could be a contributing factor to catastrophic accidents when airborne. Problem: Due to new concepts and increasingly complex functionalities of avionics systems, such activities, inherent in Software Engineering, require constant research and updating of the processes and tools used. Method: This paper describes an academic work carried out by the authors, according to guidelines for Simulation-Based Studies (SBS). Results: The standard notation for Assurance Cases Unified Modeling Language (UML) and the Discrete-Events Specification (DEVS) formalism were combined in an example for modeling and simulation of high-level requirements of the aeronautical software Roll Rate Control (RRC). Conclusion: Through the simulation it was possible to achieve the certification objective served by the Assurance Case, which proved the viability of the tools for this purpose.*

Resumo. *Contexto: Um Caso de Garantia de Software é um instrumento de auditoria utilizado para garantir que o software para sistemas críticos seja seguro e esteja em conformidade com as normas estabelecidas pelos organismos responsáveis. Na indústria de software aeronáutico, por exemplo, ele é utilizado para descrever as atividades prescritas pela norma Software Considerations in Airborne Systems and Equipment Certification (RCTA DO-178C), sem as quais o software poderia ser um fator contribuinte para acidentes catastróficos quando aeroembarcado. Problema: Devido a novos conceitos e funcionalidades cada vez mais complexas dos sistemas aviônicos, tais atividades, inerentes à Engenharia de Software, necessitam de constante pesquisa e atualização dos processos e ferramentas utilizados. Método: Este artigo descreve um trabalho acadêmico de simulação segundo um guia para Simulation-Based Studies (SBS). Resultados: A notação padrão para Casos de Garantia Unified Modeling Language (UML) e o formalismo Discrete-Events Specification (DEVS) foram combinados em um exemplo para modelagem e simulação de requisitos de alto nível do software aeronáutico Roll Rate Control (RRC). Conclusão: Pela simulação foi possível realizar o objetivo de certificação atendido pelo Caso de Garantia, o que comprovou a viabilidade das ferramentas para esse propósito.*

1. Introdução

A intensiva utilização de software em sistemas aerotransportados e equipamentos usados em aeronaves e motores resultou na necessidade de orientação de conformidade aceita pela indústria aeronáutica [DO-178C 2011]. Para satisfazer os requisitos de aeronavegabilidade, sem os quais o software não pode ser aeroembarcado, a norma RCTA[1] *Software Considerations in Airborne Systems and Equipment Certification* (DO-178C) foi escrita para satisfazer essa conformidade com o objetivo de garantir que o software atenda aos conceitos e funcionalidades aplicáveis, focando principalmente em aspectos de segurança[2] [Rierson 2013].

Um Caso de Garantia de Software é um instrumento de auditoria utilizado para garantir que o software para sistemas críticos seja seguro e esteja em conformidade com as normas estabelecidas pelos organismos responsáveis [OMG 2018]. Na indústria de software aeronáutico, por exemplo, ele é utilizado na forma de documentos ou gráficos para descrever os argumentos e as evidências[3] necessários para que os objetivos do processo de desenvolvimento e certificação do software, como alegados pela norma DO-178C, sejam alcançados. Contudo, devido a novos conceitos e funcionalidades cada vez mais complexas dos sistemas aviônicos[4] atuais, tais atividades, inerentes à Engenharia de Software, necessitam de constante pesquisa e atualização dos processos e ferramentas utilizados. A questão emergente é: como pesquisadores e profissionais poderiam utilizar processos e ferramentas de modelagem e simulação, pouco aplicados a esse domínio, para desenvolver Casos de Garantia logo nas fases iniciais de conceituação e mais aderentes à realidade dos chamados Sistemas Aviônicos Intensivos em Software (SAIS[5])?

Para contribuir com o esforço em responder a questão apresentada, este artigo descreve o trabalho acadêmico realizado pelos autores. Foram combinados a notação padrão para modelagem de Casos de Garantia *Unified Modeling Language* (UML) e o formalismo *Discrete-Events Specification* (DEVS) em um exemplo para modelagem e simulação de funcionalidades conceituais do software aeronáutico *Roll Rate Control* (RRC). Os experimentos foram realizados segundo o protocolo descrito na *guideline* para *Simulation-Based Studies* (SBS) proposta por França e Travassos [de França and Travassos 2012].

Este artigo está dividido da seguinte forma: a Seção 2 apresenta um *background* sobre o Caso de Garantia, as notações DEVS/UML e os trabalhos relacionados à questão de pesquisa; a Seção 3 apresenta o método utilizado e o resumo do experimento realizado com a ferramenta de simulação DEVS MS4 Me utilizada para comprovar, dinamicamente, que os requisitos de alto nível, expressos no modelo UML do software de exemplo, estão em conformidade com a Norma DO-178C; e, por fim, a Seção 4 apresenta a Conclusão e os Trabalhos Futuros.

---

[1] *Radio Technical Commission for Aeronautics*

[2] Segurança é traduzida em português a partir de dois atributos de qualidade em inglês: *safety* e *security*. No contexto deste artigo refere-se ao primeiro, isto é, o atributo no qual o software não provoque malefícios aos usuários de uma aeronave, como perdas materiais, ferimentos e mortes.

[3] Documentos, modelos, códigos, casos de teste, casos de simulação e etc.

[4] Sistemas eletrônicos embarcados em aeronaves.

[5] Sistemas aviônicos de informação ou de controle onde as funcionalidades operacionais são atendidas em sua maior parte pelo software, em detrimento aos sistemas mecânicos e ao hardware.

2. *Background* e trabalhos relacionados

Para Hawkins [Hawkins et al. 2015], um Caso de Garantia é um instrumento de evidências argumentativas que fornecem um caso convincente, compreensível e válido de que o software é seguro para ser utilizado em um determinado ambiente operacional. Esse pode ser estruturado a partir de cinco componentes essenciais:

- **ALEGAÇÃO** - Uma declaração sobre atributos desejados;
- **ARGUMENTO** - Explica como uma alegação é apoiada ou justificadamente inferida das evidências e das subalegações associadas. Outros termos por vezes utilizados para o mesmo conceito incluem estratégia, justificativa e razão;
- **EVIDÊNCIA** - Refere-se ao corpo disponível de fatos conhecidos, relacionados às propriedades do sistema ou aos seus processos de desenvolvimento. Dado, fato e solução são termos sinônimos. Exemplos de evidências incluem registros de perigos, análises de risco, resultados de testes, simulações e teoremas matemáticos;
- **CONTEXTO** - Qualquer informação que seja necessária para fornecer definições ou descrições de termos; ou para restringir a aplicação do Caso de Garantia a um ambiente particular ou conjunto de condições; e
- **SUPOSIÇÃO** - Declarações nas quais as alegações e argumentos se baseiam, mas que não são elaboradas ou demonstradas como verdadeiras em um Caso de Garantia.

A Tabela 1 mostra um Caso de Garantia em um contexto genérico.

Tabela 1. Os componentes de um caso de garantia e exemplos (Adaptado de [Hawkins et al. 2015]).

Componente	Exemplo
ALEGAÇÃO	O software desempenha sua função pretendida em um nível aceitável de segurança (nível X).
ARGUMENTO	É justificado pela exatidão do software em relação aos requisitos do sistema alocado (sistema maior) e requisitos derivados do processo de desenvolvimento.
EVIDÊNCIAS	Os Requisitos de Alto Nível (HLR) são satisfatórios para o refinamento em nível X dos requisitos do sistema alocado; o Código Objeto Executável é satisfatório para o refinamento de nível X dos requisitos de alto nível; e a evidência fornecida é adequada para justificar a confiança de que a exatidão do software foi demonstrada na extensão necessária para o nível X.

Segundo De la Vara e colaboradores [de la Vara et al. 2016], a utilização de modelos é amplamente realizada nos processos de desenvolvimento de Casos de Garantia no domínio aeronáutico. Todavia, há uma necessidade em transcender o conhecimento entre as funcionalidades conceituais dos sistemas e os modelos para que explicitem os objetivos da DO-178C de forma clara e, ao mesmo tempo, reflitam a real e emergente natureza do software [Hawkins et al. 2013].

A UML é uma linguagem de modelagem gráfica universal para inúmeros domínios e foi escolhida pelo *Object Management Group* (OMG) como notação padrão utilizada em Casos de Garantia, segundo o metamodelo que referencia seu desenvolvimento na indústria, o *Structured Assurance Case Metamodel Version 2* (SACM 2.0) [OMG 2018].

Por sua vez, o *Discrete Event System Specification* (DEVS) é um formalismo projetado para descrever sistemas de estados discretos e contínuos, como os embarcados em aeronaves. É uma notação matemática abstrata que pode descrever as funcionalidades de um sistema crítico desde sua conceituação [Zeigler et al. 2018]. No entanto, faltava-lhe uma representação gráfica adequada em termos de software, o que tornava a simulação computacional dos modelos DEVS uma questão desafiadora [Zeigler et al. 2017]. Contudo, após vários estudos, surgiram ferramentas e ambientes de desenvolvimento capazes de diminuir essa lacuna.

Alguns autores vêm utilizando ferramentas DEVS na simulação de conceitos e requisitos em aplicações de sistemas aeroembarcados. Por exemplo, Buzdalov e Khoroshilov [Buzdalov 2014, Buzdalov and Khoroshilov 2014] sugerem que os simuladores de código aberto existentes não são aplicáveis ou não são eficazes na aplicação em modelos de alto nível. Já Robati e colaboradores [Robati et al. 2015], apresentam uma verificação de uma Arquitetura Modular Integrada para aviônica, baseada em uma extensão do DEVS (DEVS Suite) e mostram como ela pode ser simulada usando esse ambiente de simulação de evento discreto.

Pelo exposto, a utilização de DEVS no desenvolvimento de software para aviônica vem sendo considerada por alguns autores. Contudo, eles utilizam o formalismo independentemente do padrão UML. A transformação e agregação de modelos UML/DEVS auxilia os Estudos Baseados em Simulação na verificação e no acompanhamento dinâmico das interações entre os conceitos iniciais do software e as funcionalidades alocadas do sistema maior, permitindo a evolução de Casos de Garantia mais eficientes. Eles contribuem para o desenvolvedor realizar as simulações do software antes mesmo de simular o sistema onde será embarcado, seguindo diretamente as prescrições da DO-178C e a um custo reduzido em relação aos processos tradicionais.

3. Metodologia e experimento realizado

O método utilizado neste estudo foi baseado no protocolo de simulação para Engenharia de Software proposto por França e Travassos [de França and Travassos 2012] e que consiste na condução de experimentos *in silico*, isto é, utilizado para simulações computacionais que modelam um processo natural ou de laboratório e não para cálculos computacionais genéricos. O experimento foi realizado utilizando as 33 *Report Guidelines* (SG01 - SG33), nas quais foram observados os seguintes aspectos:

- Planejamento:
  - Identificação do relatório (SG01);
  - Do contexto às questões de pesquisa (SG02 a SG06);
  - Viabilidade da simulação (SG07);
  - *Background* e trabalhos relacionados (SG08);
  - Modelo de simulação e validação (SG09 e SG10);
  - Assuntos envolvidos (SG11); e
  - Design experimental 1 (SG12 a SG14).
- Condução:
  - Ensaio experimental intermediário (SG15);
  - Dados de suporte (SG16); e
  - Ambiente de suporte à simulação (SG17).

- *Reporting*:
  - Análise de saída 1 (SG18);
  - Ameaças à validade 1 (SG19);
  - Conclusões e trabalhos futuros (SG20 a SG22);
  - Validação do modelo (SG23 a SG25);
  - Design experimental 2 (SG26 a SG28);
  - Coleta e uso de dados de planejamento (SG29 e SG30);
  - Análise de saída 2 (SG31); e
  - Ameaças à validade 2 (SG32 e SG33).

As atividades realizadas durante o experimento de modelagem e simulação são resumidas a seguir. Contudo, o relatório completo da simulação descreve todas as 33 *Report Guidelines* efetuadas e pode ser visualizado em um repositório de pesquisa[6] disponibilizado pelos autores. O relatório foi identificado como "RELATÓRIO DE SIMULAÇÃO: Caso de Garantia DO-178C 6.3.1.a Roll Rate Control (Piloto)" e as palavras-chaves "Casos de Garantia", "Modelagem e Simulação", "DEVS" e "UML" fazem parte de sua identificação (SG01). Os experimentos de simulação foram realizados em um contexto acadêmico (SG02), proposto e supervisionado pelo professor da disciplina Simulação de Sistemas, de um Programa de Pós Graduação em Ciência da Computação.

Para fins de exemplo da utilização de UML/DEVS os autores escolheram uma funcionalidade de controle comum a vários tipos de aeronaves, independente do tamanho ou função: o movimento aerodinâmico chamado Rolagem. A Rolagem (do inglês, *Banking*) é o movimento que uma aeronave realiza em torno do seu eixo longitudinal de voo. Para tal, esta possui superfícies de comando chamadas *Ailerons* que são acionadas pelo Piloto ao comandar uma alavanca chamada manche (ou *joystick*, como no caso deste experimento) ou pelo sistema automatizado do tipo *Fly-By-Wire*, o *Flight Management System* (FMS). A Figura 1 mostra um modelo de movimento de Rolagem. Abstrai-se pela mesma que existe uma força de ascensão sob asa esquerda, provocada pelo chamado "vento relativo", que impele o giro do avião. Este fenômeno é uma reação adversa e na direção oposta ao comando do manche.

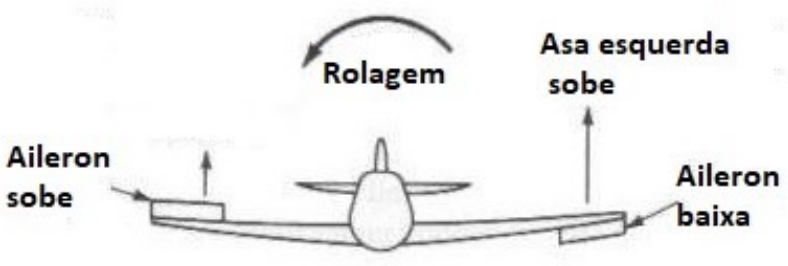

Figura 1. Modelo do Movimento de Rolagem (Vista frontal).

A DO-178C possui 71 objetivos de Conformidade relacionados aos níveis de falhas que o software desenvolvido pode causar ao ser embarcado em um sistema aeronáutico. Um dos objetivos alegados pela DO-178C, o 6.3.1, trata de Revisões e Análises

[6]https://www.dropbox.com/sh/6ehpbt6gjga9tmu/AAAcMwtrybuHtqlwJpYY9QT2a?dl=0

dos Requisitos de Alto Nível (*High Level Requirements* (HLR)) [DO-178C 2011]. A norma alega que essas atividades de revisões e análises detectam e relatam erros que podem ter sido introduzidos durante o Processo de Requisitos de Software. As atividades de revisão e análise confirmam que os HLRs atendem à Norma (SG03). Portanto, o objetivo da simulação foi comprovar o argumento estabelecido pelo Caso de Garantia DO-178C 6.3.1.a, realizando uma simulação de um HLR em ambiente DEVS, para comprovar que seu modelo UML demonstra a conformidade com os requisitos funcionais do sistema (SG04).

Um modelo de Sistema de Controle de Rolagem de uma Aeronave Conceitual foi utilizado como exemplo para a simulação e validação (SG09). Os fundamentos teóricos do experimento foram retirados dos estudos de Heim [Heim et al. 2016], Soubra e colaboradores [Soubra et al. 2015] e Le Sergent e colaboradores [Le Sergent et al. 2016]. Para realização dos experimentos também foram necessários alguns conhecimentos prévios sobre o domínio apresentado, as ferramentas utilizadas e o processo estabelecido (SG11). Alguns dos assuntos discutidos são: Casos de Garantia, software aeroembarcado, DEVS, UML e certificação pela DO-178C.

O projeto experimental (SG12), o cenário de simulação (SG13) empregado e a possibilidade de um número satisfatório de execuções (SG14) foram decisivos na escolha do RRC. O sistema RRC é composto por três módulos denominados *RollControl*, *RollMode* e *RollRate*. O módulo raiz do sistema é chamado de *RollControl* [Soubra et al. 2015].

O **Controle de Rolagem (*RollControl*)** é composto por dois sub-módulos: o *RollControl*, que é composto por 2 funções (*RollRateCalculate* e *RollRateWarning*); e o submódulo *RollMode*. O *RollControl* possui quatro entradas e quatro saídas:

- ***leftWarning***: uma saída do tipo de dado *Boolean* que informa, graças a um alarme, que a taxa de rotação da aeronave é superior a mais ou menos 15º/s na asa esquerda;
- ***rightWarning***: uma saída do tipo de dado *Boolean* que informa, graças a um alarme, que a taxa de rolagem da aeronave é superior a mais ou menos 15º/s na asa direita;
- ***mode***: uma saída do tipo de dados *TRollmode* que possui 3 valores diferentes: OFF, NOMINAL (funcionamento normal do software) e FAILSOFT (rotina de tolerância a falhas do software).

Para o experimento piloto foram escolhidas apenas algumas funcionalidades conceituais do modelo, o que permitiu contar com dados de suporte (SG16) suficientes para a execução dos experimentos. O ensaio experimental (SG15) sugerido pode ser acompanhado segundo o *storytelling* a seguir:

- O Piloto comanda o *Joystick* para fazer um movimento de rolagem à direita;
- O *Joystick* recebe o comando (*cmdJoystick*), calcula a força, a direção e o tempo de comando e retorna um comando de entrada (*retJoystick*) ao Piloto/FMS;
- Se o sistema *Fly-By-Wire* estiver em funcionamento (Piloto/FMS habilitado em FMS), ele recebe o comando de entrada, transforma em um sinal discreto e informa ao *RateControl*, que realizará suas funções (ver *Storytelling* do "loop fechado");

- O sinal discreto *cmdJoystick* é transformado em um sinal de servocomando (*cmdServo::ServoAileronEsquerdo*) ao aileron esquerdo para que este desça (*cmdServo::ServoAileronEsquerdo*) e, em contrapartida, uma força de reação causada pelo vento relativo incidente no *aileron* provoca um sinal de *feedback* (*guinadaAdversa*) ao servocomando;
- Em paralelo ao evento anterior, o sinal discreto *cmdJoystick* é transformado em um sinal de servocomando (*cmdServo::ServoAileronDireito*) ao *aileron* direito para que este suba (*cmdServo::ServoAileronDireito*) e, em contrapartida, uma força de reação causada pelo vento relativo incidente no *aileron* provoca um sinal de *feedback* (*guinadaAdversa*) ao servocomando;
- Os respectivos servocomandos recebem os sinais de guinada adversa e repassam ao sistema de controle de rolagem (*RollRateControl*) para os devidos processamentos (ver *Storytelling* do "loop fechado");
- O Sistema de Controle de Rolagem (*RollRateControl*) envia ao FMS o sinal de taxa de rolagem (*rollRate*), o modo de funcionamento atual do sistema (*rollMode*) e o sinal de alerta (*rollWarning*) em caso de falhas; e
- O Piloto/FMS recebe as informações de falhas e informa ao Sistema Aural (*feedback::SistemaAural*), pelo painel de avisos (*interface::DisplayR*) e pelos fones de ouvido do Piloto (*som::SistemaAural*).

A Figura 2 a seguir, ilustra o modelo em UML das respectivas Classes, para uma visão em "*loop* aberto" (menor granularidade) das funcionalidades explicadas anteriormente. Nota-se que a Classe *RollRateControl* agrega as Classes *RollRate*, *RollMode* e *RollWarning*, que foram os sujeitos principais para a simulação proposta.

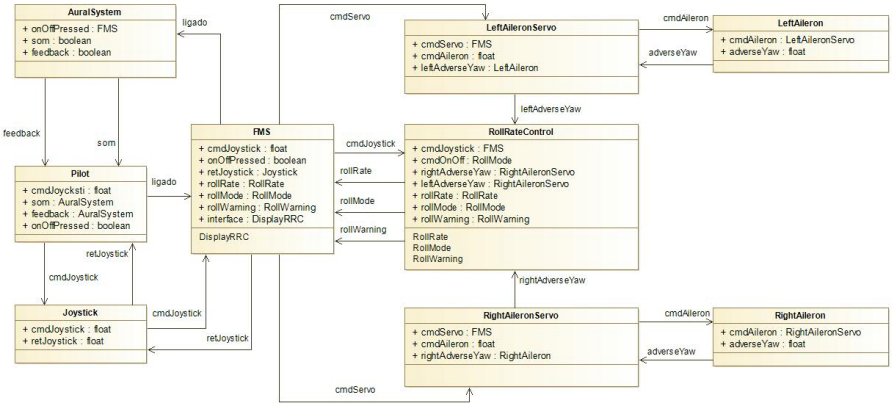

Figura 2. Modelo UML em "*Loop* Aberto" do RRC.

O ambiente DEVS selecionado (SG17) para os experimentos foi o MS4 Me *Systems DEVS Modeling Environment*, que apesar de não atender plenamente aos objetivos de desenvolvimento de software crítico, segundo a norma complementar *Software Tool Qualification Considerations* (DO-333)[7], foi disponibilizado de forma gratuita para utilização nos experimentos e atende aos requisitos acadêmicos considerados.

[7]A norma *Software Tool Qualification Considerations* (DO-333) esclarece os objetivos de qualificação necessários para as ferramentas que podem ser utilizadas no desenvolvimento certificado de software para aviônica certificada e fornece uma orientação específica para a qualificação da ferramenta [Rierson 2013].

Os experimentos, os dados qualitativos, quantitativos e de calibração (SG29 e SG30) e demais insumos são descritos no relatório e vídeo disponibilizados no repositório. Os requisitos conceituais do RRC foram desenvolvidos para gerar as entradas de simulação necessárias. Os modelos foram gerados de acordo com o entendimento dos autores sobre o sistema. Contudo, a validação foi acompanhada por especialistas de domínio (SG23). Algumas evidências empíricas (SG24) dos modelos de Soubra e colaboradores [Soubra et al. 2015] e os pressupostos (SG25) foram analisados também por estudantes e profissionais[8] da área. Os insumos utilizados, o processo, as ferramentas e os relatórios gerados foram disponibilizados no repositório da pesquisa.

O design experimental foi refinado e analisado (SG26) durante os experimentos e vários parâmetros foram remodelados, como os níveis de granularidade (SG27) e a quantidade de execuções (SG28) necessárias a cada cenário. A Figura 3 ilustra o modelo UML em "*Loop* Fechado" do RRC (maior granularidade proposta) para a transformação e consequente simulação utilizando o ambiente MS4 Me - DEVS.

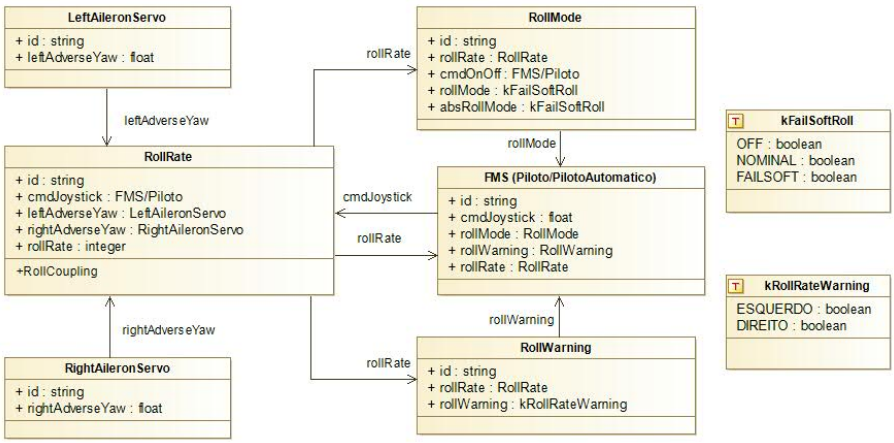

Figura 3. Modelo UML em "*Loop* Fechado" do RRC.

A transformação dos modelos não foi realizada diretamente, dos Diagramas de Classes para os *Atomic Models* ou *Component Models* do DEVS. Por exemplo, o *Component Model Flightdeck* da Figura 4 não foi realizado dos modelos em UML disponibilizados (Figuras 2 e 3). Esta escolha, apesar de tecnicamente adequada em relação aos sistemas reais, pode injetar ameaças à validação dos experimentos (SG19), já que as classes em UML foram mapeadas dos HLR de acordo com o visão arquitetural dos autores sobre o sistema RRC. Apesar de realizada em um ambiente acadêmico (**CONTEXTO**), esta atividade foi realizada sem o prévio escrutínio de profissionais (**SUPOSIÇÃO**). Portanto, isso poderia enviesar os experimentos ao replicá-los no desenvolvimento de Casos de Garantia industriais. Todavia, as ameaças à viabilidade do experimento (SG07) foram consideradas sob controle pelas condições *in silico* do ambiente em que foi realizado e pela verificação criteriosa do processo pela aplicação das *guidelines*.

Foram utilizadas várias das possibilidades oferecidas pela ferramenta MS4 Me com o intuito de simplificar a construção da simulação, reaproveitar código e reduzir

[8]Professores e alunos pilotos de uma faculdade de Ciências Aeronáuticas foram convidados e contribuíram com as explicações técnicas.

o impacto das alterações e correções futuras. Procurou-se explorar a maior parte dos exemplos fornecidos no manual da própria ferramenta buscando adequá-la à realidade do experimento. A MS4 Me fornece uma ampla estrutura de projeto para que grupos futuros possam realizar alterações ou novas implementações, ou ainda, que possam realizar críticas sobre a proposta, amadurecendo e formalizando padrões para o desenvolvimento de novas simulações.

Para comprovar o argumento do Caso de Garantia, realizou-se vários experimentos intermediários de simulação (SG15) utilizando o ambiente DEVS MS4 Me. Para tais experimentos, um arquivo **.txt** foi criado para gerar as entradas aleatórias dos comandos do piloto e as respectivas saídas da simulação. A Figura 4 mostra os principais componentes do *Aircraft*, seus *components* e *atomic models*, portas de entradas e saídas, e as interações entre os "nós", momentos antes da execução. Nota-se que o *RollRateControl* não pertence a um *component model* interno e que *Wings* e *Flightdeck* não foram previamente projetados como "classes" nos modelos em UML. O que revela uma transformação indireta do modelo base e sua independência na construção do design final para a simulação.

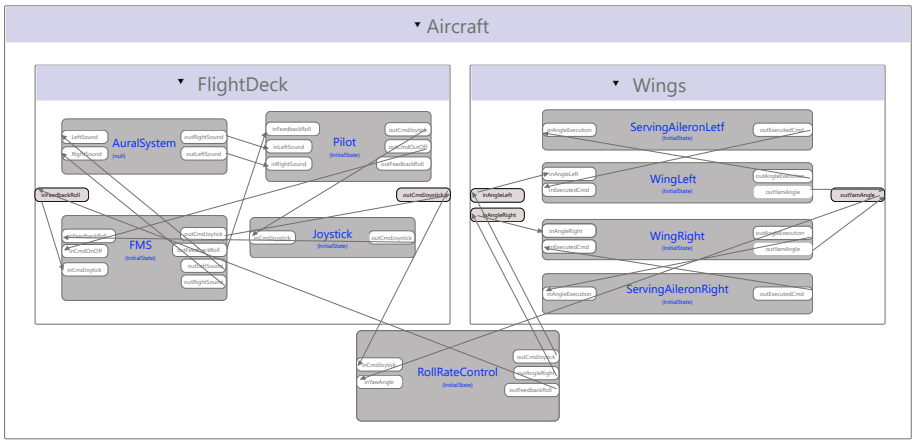

Figura 4. Modelo de simulação do RRC utilizando o ambiente MS4 Me DEVS.

4. Conclusões e trabalhos Futuros

Neste artigo foi apresentado um exemplo de experimento realizado sobre a utilização em conjunto da notação UML e do formalismo DEVS na modelagem e simulação de Requisitos de Alto Nível (HLRs), para auxiliar as equipes de pesquisa e desenvolvimento em Casos de Garantia. Para tal, utilizou-se a transformação indireta de modelos UML estáticos em componentes dinâmicos do ambiente DEVS MS4 Me, permitindo a simulação de funcionalidades de um SAIS, o RRC, logo nas fases iniciais de sua conceituação.

As ferramentas foram exploradas para extrair uma ampla base de possibilidades em simulação. Os resultados do experimento comprovaram a possibilidade de aplicação. No futuro, os autores pretendem realizar outros experimentos com o formalismo DEVS e desenvolver os demais Casos de Garantia para a DO-178C e outras normas relacionadas. Com isto, pretende-se contribuir com a Modelagem & Simulação de produtos e processos da Engenharia de Software para aplicação em domínios críticos, como o da Aviônica.

Referências

Buzdalov, D. (2014). An architecture of effective discrete-event simulation engine for early validation of avionics systems. In *SYRCoSE-Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering*, number 8. ACM/IEEE.

Buzdalov, D. and Khoroshilov, A. (2014). A discrete-event simulator for early validation of avionics systems. In *ACVI 2014–Architecture Centric Virtual Integration Workshop Proceedings*, pages 28–37. ACM/IEEE.

de França, B. N. and Travassos, G. H. (2012). Reporting guidelines for simulation-based studies in software engineering.

de la Vara, J. L., Ruiz, A., Attwood, K., Espinoza, H., Panesar-Walawege, R. K., López, Á., del Río, I., and Kelly, T. (2016). Model-based specification of safety compliance needs for critical systems: A Holistic Generic Metamodel. *Information and Software Technology*, 72:16–30.

DO-178C (2011). *Software Considerations in Airborne Systems and Equipment Certification*. RTCA, Inc. 1150 18th Street, NW, Suite 910 Washington, DC 20036-3816 USA.

Hawkins, R., Habli, I., Kelly, T., and McDermid, J. (2013). Assurance Cases and prescriptive software safety certification: A Comparative Study. *Safety science*, 59:55–71.

Hawkins, R., Habli, I., Kolovos, D., Paige, R., and Kelly, T. (2015). Weaving an Assurance Case from Design: A Model-Based Approach. In *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*, pages 110–117. IEEE.

Heim, S., Dumas, X., Bonnafous, E., Dhaussy, P., Teodorov, C., and Leroux, L. (2016). Model Checking of SCADE Designed Systems. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.

Le Sergent, T., Dormoy, F.-X., and Le Guennec, A. (2016). Benefits of Model Based System Engineering for Avionics Systems. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.

OMG (2018). Structured Assurance Case Metamodel Specification Version 2.0. url https://www.omg.org/spec/SACM/2.0/. [Online; acesso em 9 de Junho 2019].

Rierson, L. (2013). *Developing Safety-Critical Software: A practical guide for aviation software and DO-178C compliance*. CRC Press.

Robati, T., El Kouhen, A., Gherbi, A., and Mullins, J. (2015). Simulation-based verification of avionic systems deployed on ima architectures.

Soubra, H., Jacot, L., and Lemaire, S. (2015). Manual and Automated Functional Size Measurement of an Aerospace Realtime Embedded System: A Case Study based on SCADE and on COSMIC ISO 19761.

Zeigler, B. P., Muzy, A., and Kofman, E. (2018). *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Academic Press.

Zeigler, B. P., Sarjoughian, H. S., Duboz, R., and Soulié, J.-C. (2017). *Guide to modeling and simulation of systems of systems*. Springer.