UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Programa Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

UNiVErSE - Explorando Safety e Security em Veículos Aéreos não Tripulados

Matheus Lopes Franco

Dissertação de Mestrado

ICMC USP
SÃO CARLOS

Matheus Lopes Franco

UNiVErSE - Explorando *Safety* e *Security* em Veículos Aéreos não Tripulados

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco

USP – São Carlos
Agosto de 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Lopes Franco, Matheus
L864u UNiVErSE - Explorando Safety e Security em Veículos Aéreos não Tripulados / Matheus Lopes Franco; orientadora Kalinka Regina Lucas Jaquie Castelo Branco. -- São Carlos, 2019.
114 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2019.

1. Arvore de falhas. 2. Safety. 3. Security. 4. Veículo Aéreo Não tripulado. I. Regina Lucas Jaquie Castelo Branco, Kalinka , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Matheus Lopes Franco

UNiVErSE - Exploring Safety and Security in Unmanned Aerial Vehicle

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *EXAMINATION BOARD PRESENTATION COPY*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Kalinka Regina Lucas Jaquie Castelo Branco

USP – São Carlos
August 2019

Dedico este trabalho a todos os meus professores e a meus pais Sergio A. Franco e Miriam Cristina Lopes Franco, que desde sempre acreditaram em mim e me fizeram chegar até aqui.

AGRADECIMENTOS

Quero começar agradecendo a todos os meus professores e professoras, desde os do ensino fundamental até os da pós-graduação, todos tem participação direta ou indireta nesta tese. Dentre todos estes alguns se destacam, como exemplo a professora Náida e o professor Jorge Antonio Marinovic que sempre me ajudaram a superar minhas dificuldades durante o ensino fundamental e médio.

Seria impossível citar todos os professores que me influenciaram durante a graduação, mas em especial quero agradecer ao professor Luiz Henrique Andrade Correia, além de deixar aqui meu agradecimento a todos os meus professores da Universidade Federal de Lavras, pois sem a base sólida de conhecimento lá adquirida seria impossível a execução desse trabalho.

Agradeço também a todos meus colegas e amigos do Laboratório de Sistemas Embarcados Críticos, pois me ajudam diariamente, seja com elogios, críticas ou até mesmo com brincadeiras que tornaram o ambiente de trabalho descontraído, porém muito produtivo.

Agradeço também aos meus amigos que foram capazes de entender minhas ausências em determinados momentos e se fizeram presentes em outros, sempre me incentivando e apoiando.

Por último, mas de forma alguma menos importante, quero agradecer a minha orientadora Kalinka Regina Lucas Jaquie Castelo Branco, que desde antes de me ter como orientado já me ajudou e motivou, entendeu minhas dificuldades que vão desde a escrita até o conteúdo propriamente dito e mesmo assim me aceitou, hoje me auxilia não só com questões acadêmicas, mas com os problemas do dia a dia também.

Obrigado a todos vocês que tanto me ajudam.

"O insucesso é apenas uma oportunidade
para recomeçar com mais inteligência."
(Henry Ford)

RESUMO

FRANCO, MATHEUS LOPES. **UNiVErSE - Explorando *Safety* e *Security* em Veículos Aéreos não Tripulados**. 2019. 114 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Contexto: Sistemas críticos são sistemas computacionais nos quais uma falha pode levar a consequências catastróficas que variam de danos à propriedade, ao ambiente ou financeiros, à lesões e até mesmo a perda de vidas humanas. Sistemas autônomos automotivos e aeroespaciais são exemplos de sistemas críticos que usualmente operam em ambientes de rede compartilhadas em conjunto com outros sistemas. Em virtude da natureza críticas impostas por esses sistemas e seus ambientes de operação, o desenvolvimento de tais sistemas deve atender a requisitos de segurança (*security*) e de proteção (*safety*). Dessa formas, as propriedades de segurança e de proteção desses sistemas devem ser analisadas e demonstradas em diferentes níveis de abstração para a obtenção da aprovação e da certificação de tais sistemas. **Problema:** Os sistemas autônomos podem ser suscetíveis a ameaças de segurança em decorrência de falhas de proteção que podem causar danos ao ambiente, à propriedade, às finanças, lesões ou a perda de vidas humanas. Entretanto, as técnicas existentes na literatura para apoiar a engenharia de dependabilidade do sistema, como *Failure Propagation and Transformation Notation* (FPTN) e *Security HaZOP*, somente apoiam a análise de propriedades de *safety* e de *security* de forma isolada. **Questão de Pesquisa:** Neste contexto, esta dissertação fornece resposta à seguinte questão de pesquisa: como engenheiros podem analisar o impacto de ameaças de segurança e proteção sob a dependabilidade de sistemas autônomos de forma efetiva e sistemática? **Objetivo:** Nesta dissertação, é apresentada a UNiVErSE, uma abordagem composicional para a análise integrada de propriedades de segurança e de proteção e à geração automática de árvores de falhas e árvores de ataque de modo a apoiar a certificação de sistemas autônomos. **Resultados:** A validação da abordagem UNiVErSE na análise das propriedades de segurança e proteção e geração integrada de árvores de ataque e de falhas de *safety* para o sistema aeroespacial de piloto automático SLUGS. A aplicação da abordagem UNiVErSE contribuiu para reduzir a complexidade e o esforço em atividades de análise de segurança e proteção requiridas para a certificação de sistemas críticos.

Palavras-chave: Veículo Aéreo Não Tripulado, Segurança, Proteção, Árvore de Falhas, Dependabilidade.

ABSTRACT

FRANCO, MATHEUS LOPES. **UNiVErSE - Explorando *Safety* e *Security* em Veículos Aéreos não Tripulados**. 2019. 114 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Context: Critical-systems are systems in which a failure may lead to catastrophic consequences, ranging from damage to the property, environment, or finances, to injuries and even loss of human's life. Aerospace and automotive autonomous systems are examples of critical systems that usually operate in shared and networked environments together with other systems. Due the criticality posed by these systems and their environment, their development should address safety and security requirements. Thus, safety and security properties of these systems should be analysed and demonstrated at different levels of abstraction to achieve approval and certification. **Problem**: Autonomous systems can be susceptible to security threats due the occurrence of safety failures leading the system to damages to the environment, property, finance, injuries, or loss of human's life. However, existing techniques to support system dependability engineering, e.g., Failure Propagation and Transformation Notation (FPTN) and Security HaZOP, only support the analysis safety and security properties in isolation. Research Question: In this context, this dissertation address the following research question: how engineers can effectively and systematically analyse the impact of safety and security threats on the overall system dependability? **Objective**: In this dissertation is presented the proposal of UNiVErSE, a compositional approach to support the integrated analysis of system safety and security properties, and automatic generation of fault and attack trees to support the certification of autonomous systems. **Results**: The validation of the proposed approach on the analysis of safety and security properties and generation of integrated fault and attack trees for the SLUGS open source aerospace autonomous pilot. Applying UNiVErSE contributed to reduce the complexity and effort of performing safety and security analysis required for certifying critical systems.

Keywords: Unmanned Aerial Vehicle, Security, Safety, Fault Tree, Dependability.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AADL	*Architecture Analysis and Design Language*
ACM	*Association for Computing Machinery*
ALARP	*As Low As Reasonably Practicable*
CAA	*Civil Aviation Authority*
CAN	*Controller Area Network*
DALs	Development Assurance Levels
DoS	Denail of Service
DSL	*Domain-Specific Language*
EASA	*European Aviation Safety Agency*
FAA	*Federal Aviation Administration*
FFA	*Functional Failure Analysis*
FMEA	*Failure Mode and Effect Analysis*
FMEAs	*Failure Models and Effects Analyses*
FMECA	*Failure Modes, Effects and Criticality Analysis*
FPTC	*Failure Propagation and Transformation Calculus*
FPTN	*Failure Propagation and Transformation Notation*
FTA	*Fault Tree Analysis*
GCS	*Ground Control Station*
GPS	*Global Position System*
HALE	*High altitude long endurance*
HAZOP	*HAZard and OPerability studies*
HIL	*Hardware-in-the-Loop*
HiP-HOPS	*Hierarchically Performed Hazard Origin and Propagation Studies*
IC	*Input Capture*
IoT	*Internet of Things*
MALE	*Medium altitude long endurance*
MANET	*Mobile Ad hoc Network*
MDA	*Model Driven Architecture*
MDD	*Model Driven Development*
MDE	*Model Driven Engineering*
MDT	*Model Driven Test*

MPCS	*Mission Planning and Control Stations*
MUAV	*Mini* UAV
NAV	*Nano Air Vehicles*
PID	*Proportional Integral Derivative*
PWM	*Pulse Width Modulation*
SANT	Sistema de Aeronaves Não Tripulada
SILs	Safety Integrity Levels
SILs	Safety Integrity Levels
SLUGs	*Santa Cruz Low-Cost UAV GNC Subsystem*
SPI	*Serial Peripheral Interface*
SQL	*Structured Query Language*
TI	Tecnologia de Informação
TUAV	*Medium Range or Tactical* UAV
UART	*Universal Asynchronous Receiver/Transmitter*
UAS	*Unmanned Aerial Systems*
UNiVErSE	*UNmanned Vehicle safEty Security improvEr*
UNiVErSE	***U**nmanned**VE**hicle **SAF**ety improv**ER***
Validação & Verificação	V&V
VANTs	Veículos Aéreos Não Tripulados

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Contextualização

Um sistema embutido ou embarcado é um conjunto de hardware e software, fortemente integrados, projetado para realizar uma função específica. O seu desenvolvimento é marcado pelo desafio de superar suas restrições características, como fontes de alimentação limitadas ou capacidade reduzida de memória. É esperado que o sistema seja autônomo, funcionando sem modificações e com pouca ou nenhuma intervenção humana por um grande período de tempo (JIMNEZ; PALOMERA; COUVERTIER, 2013). Estes sistemas são considerados críticos quando eventos de falha podem acarretar perdas de vidas humanas ou perdas de ativos de alto valor. Em algumas aplicações, como na aviação, sistemas embarcados críticos devem apresentar taxas de falha baixas como uma falha grave a cada 10^5 até 10^9 horas de operação (MICA; COSTELLO, 2008).

O estudo da segurança em sistemas embarcados críticos vem ganhando importância, tanto pelo número crescente de equipamentos quanto pelos desafios apresentados nestas aplicações. Solucionar questões sobre confiabilidade, robustez, segurança e disponibilidade é imprescindível para garantir o aspecto crítico desses sistemas.

Nos últimos anos, os Veículos Aéreos Não Tripulados (VANTs) têm recebido um número crescente de componentes eletrônicos, executando software embutido e conectados por meio de canais de comunicação sem fio. Além disso, à medida que o poder de processamento aumenta e o software se torna mais sofisticado, esses veículos ganham a capacidade de realizar operações complexas, tornando-se mais autônomo, eficiente, adaptável, seguro e utilizável. Esses veículos autônomos operam sem qualquer envolvimento humano significativo, em vez disso, dependem de comunicação e sensores/atuadores para navegar ou executar manobras específicas, como pouso, decolagem ou desvio de obstáculos. Na prática, isso traz maior flexibilidade e permite um número cada vez maior de aplicação e uso para esses veículos. Os VANTs podem ser implantados

em ambientes onde a acessibilidade física é difícil ou muitas vezes até impossível.

Sendo classificados como sistemas críticos de segurança, uma vez que eventos de falha podem causar lesões ou a perda de ativos de alto valor, a segurança física ou *safety* é uma das principais preocupações para desenvolvedores e usuários. No entanto, a combinação de alta mobilidade e comunicação sem fio aumentou ainda mais a exposição desses sistemas a ameaças mal-intencionadas e a falhas decorrentes de conectividade incerta ou tempo de comunicação.

Os requisitos não funcionais, como a segurança da informação ou *security*, tornam-se cada vez mais difíceis de serem cumpridos, criando novos desafios para esses sistemas embarcados críticos de segurança (*safety-critical embedded systems*) (BLOOMFIELD; LALA, 2013). Na verdade, para que esses VANTs se tornem realidade e possam ser inseridos ao espaço aéreo não segregado, devem ser realizadas mais pesquisas sobre o desenvolvimento e garantia de segurança (tanto com relação à *security* quando à *safety*). Isso requer abordagens multidisciplinares para projetar e implementar as funcionalidades necessárias, desde sistemas de controle inteligentes, comunicações e redes, segurança, técnicas de aprendizado de máquina, até tolerância a falhas (KOOPMAN; WAGNER, 2017), uma vez que até agora, esses veículos foram considerados sistemas isolados (sendo utilizados normalmente em um espaço segregado).

As únicas preocupações de segurança eram com assaltantes ou sequestradores. Agora, com redes, o cenário mudou drasticamente. Por um lado, a própria rede pode ser direcionada com intenções maliciosas, por exemplo, por meio de ataques de Negação de Serviço (DoS) (KASHIKAR; NIMBHORKAR, 2013), tornando impossível comunicar-se oportunamente, coordenar, tornando mais difícil garantir que esses veículos realizem suas funções de forma segura e eficiente. Por outro lado, os sistemas internos dos veículos em rede tornam-se acessíveis a partir do exterior e, portanto, são potencialmente vulneráveis ao sequestro remoto, com graves consequências sobre a segurança. Esta possibilidade foi discutida para o caso de VANTs em (FAUGHNAN *et al.*, 2013). Kashikar e Nimbhorkar (KASHIKAR; NIMBHORKAR, 2013) estudaram a troca de mensagens entre nós em um *Mobile Ad hoc Network* (MANET). Uma vez que a troca é feita por meio de um pacote de dados por vez, essas redes são afetadas pelos ataques DoS. O método proposto pelos autores visa bloquear nós maliciosos no acesso à rede. Se o nó alvo do ataque começar a receber pacotes em quantidades maiores do que a rede, esse nó solicitará ao nó atacante que reduza sua transmissão. Caso contrário, a comunicação entre os nós é interrompida e o nó alvo colocará o nó atacante em uma lista de identificações não confiáveis. À medida que um nó tenta juntar-se à rede, os nós pertencentes à rede verificam suas listas de identificações não confiáveis. Uma vez que este nó esteja em qualquer uma dessas listas, ele não terá acesso acesso à rede. Faughnan et al. (FAUGHNAN *et al.*, 2013) e Kashikar e Nimbhorkar (KASHIKAR; NIMBHORKAR, 2013) mostraram os possíveis ataques que um VANT pode enfrentar durante as operações.

Javaid et al. (JAVAID *et al.*, 2012) visavam a criação de canais seguros para comunicação entre sistemas de VANTs, satélites e estações de base. Após testes com ataques, o sistema

apresentou falhas em alguns componentes, especialmente após o ataque DoS. Além disso, esses veículos são muito dependentes da detecção precisa do meio ambiente, por exemplo, visão por computador, de modo que a interferência intencional com os sensores também representa um problema de *safety*. Em resumo, existem problemas significativos resultantes de falhas ou violações de *safety*/*security* que precisam ser abordados (SAMPIGETHAYA; POOVENDRAN, 2013).

Para que sistemas criticos tenham permissão de operar é comum que tenham que obedecer certos padrões mínimos de segurança específicos para cada tipo de operação e ambiente em que é aplicado. Muitas vezes um mesmo sistema crítico possui restrições diferentes de uso dependendo de onde e para o que é aplicado, para identificar e regular o uso de tais sistemas são elaborados os *Safety standards*, ou padrões de segurança.

Já existem padrões de segurança funcionais bem estabelecidos para aviônicos (por exemplo, DO-178C (167, 1992) e DO-254 (DO, 2000)). No entanto, os padrões de segurança cibernética neste domínio foram publicados recentemente, como DO-326A (DO, 2010). Na prática, existe uma clara necessidade de investigar o quão bem as atuais e novas iniciativas de *security* e *safety* atendem aos requisitos para futuros VANTs seguros.

Padrões/considerações de segurança, por exemplo, SAE 4754A no domínio aeroespacial e ISO 26262 no automotivo, estabelecem que as propriedades de confiabilidade do sistema, neste caso, segurança e proteção, de um sistema crítico (por exemplo, autônomo) devem ser analisadas e demonstradas em diferentes níveis de abstração para obter aprovação e certificação de segurança.

Um desafio fundamental é que as soluções existentes para garantir a *safety* podem abrir mais pontos de vulnerabilidades do ponto de vista da *security*. Por outro lado, as deficiências de segurança computacional podem levar, se exploradas por atacantes, a violações de segurança física. Nos poucos casos em que a segurança é levada em consideração, o único problema abordado é a comunicação de rede aberta, como sistemas sem fio. A segurança em geral não é tratada (ou é tratada de forma superficial), sem um suporte total para identificar e mitigar ameaças existentes. Portanto, a relação entre segurança física e segurança computacional ainda é um problema aberto na comunidade.

1.2 Motivação e Caracterização do Problema

O estudo da segurança pode ser dividido em duas áreas *safety*, responsável pela segurança física dos componentes que compõem o sistema e a *security*, responsável pela segurança da informação dos softwares que atuam no sistema. Solucionar questões sobre confiabilidade, robustez, segurança e disponibilidade dos canais de comunicação é imprescindível para garantir o aspecto crítico desses sistemas.

Ataques a VANTs podem colocar em grave perigo a segurança da operação e, portanto, são vistos como uma séria preocupação social. As motivações para realizar tais ataques podem ser variadas, como é típico com outros tipos de ataques, desde a simples diversão de atacantes até razões políticas ou financeiras. Lidar com esses ataques é um problema complexo e difícil, diferente do problema de garantir computadores conectados à internet, devido a algumas características prevalentes de cenários envolvendo veículos autônomos: conectividade externa e desempenho computacional geralmente são limitados, é necessária operação em tempo real, componentes do veículo são normalmente fornecidos por diferentes fornecedores e podem ser heterogêneos, é impossível antecipar todos os possíveis cenários e ameaças de ataque, e o emprego de mecanismos de segurança pode introduzir riscos de segurança (por exemplo, diminuindo o tempo de resposta de algum sistema de controle). Portanto, soluções de última geração para garantir sistemas baseados em computador são úteis apenas em uma extensão limitada e ainda existem lacunas significativas, exigindo esforços específicos de pesquisa.

Técnicas de engenharia de confiabilidade disponíveis na literatura, por exemplo,(LEVESON, 1995), fornecem apenas suporte para analisar as propriedades de *safety* do sistema de forma isolada ignorando *security*.Algumas dessas técnicas são composicionais e baseadas em modelos, por exemplo, HiP-HOPS(PAPADOPOULOS *et al.*, 2011), CHESS(MAZZINI S., 2016), AADL Error Annex (DELANGE; FEILER, 2014).

Técnicas composicionais permitem realizar engenharia de *safety* em paralelo ao projeto do sistema. Assim, suposições sobre possíveis falhas no nível do sistema, identificadas durante a análise de riscos, e como essas falhas podem ser propagadas por meio de subsistemas e componentes podem ser introduzidas na forma de anotações no modelo de arquitetura do sistema.

As técnicas existentes de análise de *security*, por exemplo, *Security HaZOP* e *Architecture Analysis and Design Language (AADL) Security Annex* (DELANGE; FEILER, 2014) fornecem suporte para analisar o impacto de ameaças externas, por exemplo, ataques e injeção de *Structured Query Language* (SQL), na segurança geral do sistema. O *Security HaZOP* fornece suporte manual para especificar riscos relacionados à *sacurity* e o risco que eles representam para a segurança geral do sistema. O *AADL Security Annex* é uma técnica de composição que da suporte a engenheiros na especificação de potenciais ameaças de *security*, suas causas e efeitos, por meio de anotações em um modelo de arquitetura AADL. Entretanto, realizar uma análise conjunta das propriedades de segurança de um sistema crítico sistematicamente é uma tarefa demorada e propensa a erros, afetando, assim, negativamente o desenvolvimento do sistema e os custos de certificação.

Sistemas autônomos podem ser suscetíveis a ameaças de *security* devido à ocorrência de falhas relacionadas à *safety*, o que pode tornar o sistema vulnerável a essas ameaças. Por outro lado, a ocorrência de ameaças de *security* também pode apresentar ameaças de *safety* que causam uma falha no sistema, implicando em danos ao meio ambiente, propriedade ou finanças,

ferimentos ou perda de vidas humanas. Assim, esta pesquisa aborda as seguintes questões de pesquisa:

1. Como é possível efetiva e sistematicamente analisar (prever) o impacto da ocorrência de ameaças de *safety*/*security* durante a engenharia de confiabilidade do sistema?
2. Como é possível reduzir o número de erros e custos durante a análise das propriedades de *safety*/*security* e proteção do sistema?

Este trabalho contribuirá para avançar o estado da arte por meio do desenvolvimento de uma identificação sistemática de ameaças de segurança a sistemas autônomos e da identificação de interdependências entre *safety* e *security* no contexto de sistemas embarcados criticos. Devem ser consideradas análises entre *safety* do sistema e engenharia de *security* do sistema, como árvore de ataque e árvore de falhas, e os aspectos exclusivos como falhas versus ataques maliciosos. O projeto também investiga a incorporação da segurança como um conceito unificado no desenvolvimento de sistemas embarcados criticos.

1.3 Objetivos

Neste trabalho é apresentada a abordagem ***UnmannedVEhicle SAFety improvER*** (UNiVErSE) para a análise de propriedades de *safety* e *security* em modelos composicionais[1].

O UNiVErSE compreende um conjunto de atividades para apoiar a análise do impacto de falhas de *safety*/*security* na confiabilidade do sistema e um catálogo/taxonomia de *guide words* de falha de *safety* e sua relação com falhas de *security*. Estas palavras-guia e suas relações são consideradas para realizar e integrar a análise das propriedades de segurança do sistema usando técnicas de engenharia de confiabilidade de composição existentes, por exemplo, HiP-HOPS e AADL Error Annex.

A abordagem UNiVErSE foi validada em um estudo de caso no domínio de sistemas de aeronaves não tripuladas, *Santa Cruz Low-Cost UAV GNC Subsystem* (SLUGs), que foi escolhido pois é um sistema de código aberto realista desenvolvido no MATLAB® Simulink. O HiP-HOPS foi estendido com *guide words* de *security* para permitir suporte para análise integrada de propriedades de proteção e *security*.

Os objetivos específicos dessa dissertação são:

[1] Representar o conhecimento de um engenheiro exige teorias de domínio que são de ordens de magnitude maiores do que as teorias existentes atualmente, que descrevem fenômenos em vários níveis de granularidade e incorporam múltiplas perspectivas. Para construir e usar tais teorias efetivamente estratégias são requeridas para organizar modelos de domínio e técnicas para determinar qual subconjunto de conhecimento aplicar para uma dada tarefa. A modelagem composicional é uma técnica que aborda esses problemas. A modelagem composicional usa premissas explícitas de modelagem para decompor o conhecimento de domínio em fragmentos de modelo semi-independentes, cada um descrevendo vários aspectos de objetos e processos físicos (FALKENHAINER; FORBUS, 1991)

1. **Investigar a interação entre *safety* e *security* no contexto de VANTs**. Tradicionalmente, a principal preocupação no projeto e desenvolvimento de sistemas de VANTs tem sido garantir requisitos de *safety*. Entretanto, já que estes veículos se comunicam com entidades externas, algumas arquiteturas que foram concebidas para assegurar requisitos de *safety* podem apresentar falhas de *security* e vice-versa.

2. **Integração de requisitos de *safety* e *security* no ciclo de vida do desenvolvimento de VANTs**. Sistemas críticos são usualmente concebidos para atender aspectos de *safety*, mas não totalmente direcionados para aspectos de *security*. Desta forma, faz-se necessária a integração entre aspectos de *safety* e *security*.

3. **Avaliação de sistemas com requisitos de *safety* e *security***. *Safety* é o principal requisito de um sistema crítico e técnicas de V&V (Validação & Verificação) que visam estes requisitos são bem conhecidas e maduras. Por outro lado, aspectos de *security* são amplamente explorados em sistemas de informação convencionais. Desta forma, faz-se necessária a avaliação de requisitos de *security* juntamente com o atendimento de requisitos de *safety*.

1.4 Estrutura do Texto

Este texto está estruturado da seguinte maneira: no Capítulo 2 são descritos os principais conceitos relacionados a VANTs e SANTs, elucidando suas principais aplicações. É ainda apresentado neste capítulo o piloto automático SLUGs bem como o conceito de engenharia dirigida a modelos. O Capítulo 3 apresenta os conceitos de Engenharia de Segurança de Sistemas, tanto física quando computacional. Estes capítulos são necessários para o entendimento do cenário de aplicação do trabalho.

No Capítulo 4 é apresentada a UNiVErSE, uma nova abordagem para a análise de propriedades de *safety* e *security* em modelos composicionais focada em VANTs, enquanto que o Capítulo 5 apresenta os resultados com base no estudo de caso do SLUGS e a validação do UNiVErSE.

Por fim, o Capítulo 6 traz a conclusão da dissertação, assim como as produções bibliográficas e proposta para futuros trabalhos.

CAPÍTULO

2

VEÍCULOS AÉREOS NÃO TRIPULADOS

2.1 Considerações Iniciais

Neste capítulo é descrita a fundamentação teórica necessária para o entendimento desta dissertação de mestrado. É possível observar que, por essa dissertação tratar de um tema multidisciplinar, esse capítulo apresenta uma diversidade de informações de diferentes áreas de pesquisa. Sendo assim, este capítulo aborda os conceitos fundamentais relacionados a VANTs, piloto automático e a engenharia dirigida a modelos.

2.2 Veículos Aéreos Não Tripulados

Em 1990 o termo robô aéreo foi definido como uma classe de máquinas voadoras pequenas e inteligentes, não importando seu nível de autonomia (nenhuma, parcial ou total). Neste grupo encontram-se os helicópteros, aviões, dirigíveis e aeronaves também chamadas de não convencionais (DUDEK, 2010), (MATARIC, 2007). Não se pode confundir um VANT com um modelo controlado por rádio. Diferente dos modelos controlados por rádio que são totalmente dependentes de comandos enviados por um piloto humano, um VANT possui certo grau de inteligência e diversos sensores que o auxiliam durante o voo e concedem parcial ou total autonomia para tomada de decisões (NONAMI *et al.*, 2010).

Um VANT (seja ele de asa fixa como o Hermes ilustrado na Figura 1, ou não, como o quadrotor Solo3DR ilustrado na Figura 2) é uma aplicação de um sistema embarcado crítico complexo. O termo VANT foi adotado tanto pela *Federal Aviation Administration* (FAA) quanto pela comunidade acadêmica internacional para designar sistemas que incluem não apenas os aviões, mas sim todo o conjunto: (*payload*), estação de controle terrestre e *links* de comunicação (TRINDADE *et al.*, 2010).

Comumente um VANT faz parte de um Sistema de Aeronaves Não Tripulada (SANT)

Figura 1 – VANT de asa fixa Hermes 450

Fonte: Imagem de domínio público

Figura 2 – VANT quadrotor Solo3DR

Fonte: Site do fornecedor
<https://www.bhphotovideo.com/c/product/1133723-REG/3d_robotics_solo.html>

sendo este composto tipicamente por um ou mais VANTs, e por uma ou mais estações de controle terrestre *Ground Control Station* (GCS) e/ou estações de controle e planejamento de missões *Mission Planning and Control Stations* (MPCS), *payload* (carga útil) e de *links* de comunicação.

Alguns SANTs podem incluir também subsistemas de lançamento e recuperação, subsistemas de transporte de aeronaves e outros equipamentos de assistência e manutenção de equipamentos em solo, subsistemas de comunicação, entre outros.

Neste capítulo são abordados os conceitos básicos de VANTs, mais especificamente o conceito de piloto automático, sendo que é descrito em detalhes o piloto automático *Santa Cruz*

Low-cost Unmanned Aerial Vehicle (UAV) - (SLUGs), alvo deste projeto. É também descrito o desenvolvimento orientado a modelos.

2.2.1 Composição de um SANT

Os VANTs podem, de forma bem básica, ser considerados veículos aéreos que não possuem um ser humano pilotando a aeronave da forma convencional, de modo que pode ser pilotado à distância por meio de uma estação de controle terrestre ou realizar missões pré-programadas de forma completamente autônoma (TRINDADE *et al.*, 2002). Podem ser compostos por aeronaves das mais diversas formas Trindade *et al.* (2010) e Austin (2010), como já mencionado anteriormente, tais como: asas fixas (aviões), asas rotativas (helicópteros), quadricópteros, balões dirigíveis e variantes de todos estes. Além disso, essas aeronaves podem fazer uso dos mais diferentes meios de propulsão: combustão interna, turbinas, motores elétricos, gases mais leves que o ar, entre outros. Os VANTs podem ser empregados tanto para uso militar quanto civil, sendo que para fins militares podem variar desde um alvo móvel para treinamento de artilharia antiaérea, vigilância, reconhecimento, monitoramento até as versões mais avançadas como aeronaves armadas para ataque. Já para fins civis são utilizados para fotografia aérea, filmagem, busca e salvamento, monitoramento ambiental e recreação entre várias outros fins que surgem a cada dia, como é o caso de VANTs para transporte de passageiros, proposto pela UBER[1,2].

Um SANT, do Inglês *Unmanned Aerial Systems* (UAS), pode ser dividido em (AUSTIN, 2010):

- **Aeronave**, que abrange fuselagem e estrutura física, sistema de propulsão, sistema de controle do voo, sistema de navegação, suprimentos de comunicação, sistema de prevenção de colisão, sistema elétrico e sistema de recuperação de voo. Os principais requisitos para seu projeto são o alcance, a velocidade e a autonomia necessária para a realização da missão. O alcance e a autonomia determinam o quanto de combustível ou bateria devem ser carregados, enquanto que a velocidade determina o tipo de configuração, como o uso de asas fixas ou de asas rotativas;

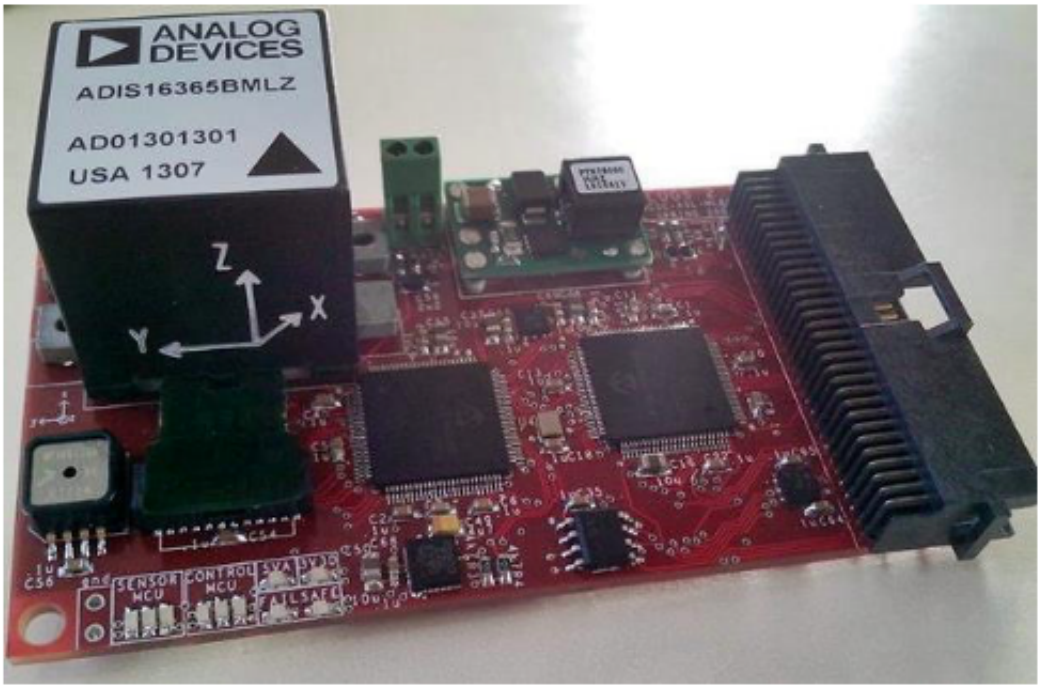
- **Piloto Automático**(exemplificado na Figura 3 por meio do piloto automático do SLUGS), que é o responsável por gerenciar a carga útil, realizar a transmissão de dados para a estação de base, executar os comandos enviados pela estação de base e manter a aeronave estável e em uma trajetória definida pelo plano de voo. Entre essas funções, a última se destaca por sua dificuldade e complexidade, sendo gerenciada por dois subsistemas diferentes, de

[1] <https://www.techtudo.com.br/noticias/2018/05/tudo-sobre-o-uberair-saiba-como-vai-funcionar-o-taxi-voador-da-u
ghtml>

[2] <https://g1.globo.com/sp/vale-do-paraiba-regiao/noticia/2019/06/11/
com-novo-design-carro-voador-da-embraer-vai-fazer-voos-de-teste-em-2020-nos-eua-e-australia.
ghtml>

controle e de navegação. O **Subsistema de Controle**, está relacionado à estabilidade da aeronave, ou seja, mantê-la em um estado de voo exequível determinado por sua posição, orientação e velocidade. Justamente por controlar estruturas físicas, esse subsistema depende da configuração da aeronave e de suas características, que são utilizadas juntamente com os efeitos aerodinâmicos para a modelagem de equações de movimento. A partir da análise dessas equações em pontos de equilíbrio são propostas estratégias baseadas na teoria de controle que garantem a realização do voo. Já o **Subsistema de Navegação**, deve manter a aeronave em uma trajetória predefinida e para isso precisa identificar a sua localização no espaço a todo momento. Atualmente, a tecnologia de GPS é empregada para a solução de localização, porém também existem técnicas baseadas em sistemas inerciais para locais fechados, e até mesmo para apoio ao *Global Position System* (GPS) em locais abertos. Sendo assim, conhecida a localização da aeronave em tempo real, é possível criar algoritmos que relacionem a localização atual com a desejada, mapeada pela trajetória. Essa trajetória é geralmente definida em pontos de posicionamento em relação a um sistema de coordenadas que são associados a uma velocidade desejada, formando os chamados *waypoints*;

Figura 3 – Placa do piloto automático SLUGS.



Fonte: O autor.

- **Estação de Controle Terrestre** (exemplificada na Figura 4), também chamada de estação base, considerada o centro de controle operacional de todo o sistema, pois é a partir dela que se pode controlar desde o lançamento, passando pelo voo ou operação até chegar à recuperação do veículo. Ela é também responsável por receber e processar os dados dos sensores que compõem o *payload*, controlar o seu funcionamento (geralmente em tempo

Figura 4 – Exemplo de estação de controle terrestre.

Fonte: Obtida em (PASTOR; LOPEZ; ROYO, 2007).

real) e fornecer uma interface entre o sistema e o mundo exterior. Ela pode variar desde um *smartphone* até um super computador, que é projetado para acompanhar o desenvolvimento da missão e, em alguns momentos, operar o VANT e seu *payload* (PASTOR; LOPEZ; ROYO, 2007);

- ***Payload***, em geral são sensores e equipamentos adicionados á aeronave com a finalidade de realizar uma missão. O *payload* pode também englobar armamentos e outros dispositivos, excluindo-se aviônicos (eletrônicos da aviação), *links* de comunicação e combustível (PASTOR; LOPEZ; ROYO, 2007; TRINDADE *et al.*, 2012);

- ***Links* de Comunicação de Dados**, uma combinação de mecanismos de comunicação (rádio, satélites, dentre outros) que tem por objetivo garantir conexão contínua entre o VANT e a estação de controle terrestre (PASTOR; LOPEZ; ROYO, 2006);

- **Sistemas de suporte**, incluem sistemas de manutenção, lançamento, pouso e transporte da aeronave (AUSTIN, 2010).

Uma vez que o foco deste trabalho encontra-se no estudo de segurança, tanto *security* como *safety* em piloto automático de VANTs, o mesmo será detalhado na Seção 2.3.

2.2.2 Classificações dos VANTs

Os VANTs podem ser classificados de acordo com seu porte, alcance e capacidade para realização de missões. Existem diversas classificações, dentre elas cabe destacar as apresentadas: pelo DoD (DOD, 2007; DOD, 2009) dos Estados Unidos, pelo *Safety and Airspace Regulation Group* - Grupo de Segurança e Regulamentação do Espaço Aéreo da *Civil Aviation Authority* (CAA) (CAA, 2015) do Reino Unido, pela Agência Nacional de Aviação Civil (ANAC) (ANAC, 2015) e pela *European Aviation Safety Agency* (EASA) (EASA, 2015).

A classificação utilizada neste trabalho é a proposta por (AUSTIN, 2010), uma vez que muitos trabalhos publicados na área fazem uso desta classificação uma vez que ela inclui a missão como parâmetro para realizar a classificação. Com isto, faz-se necessário conhecê-la para entender se as propostas discutidas nos trabalhos se aplicam a aeronaves de pequeno, médio ou grande porte. Os termos descritos abaixo descrevem um grande número de sistemas compreendendo desde aeronaves de 35 metros até veículos de apenas 40 milímetros:

- ***High altitude long endurance*** **(HALE)** – Mais de 15.000 m de altitude e 24+ horas de autonomia. Esse tipo de aeronave é capaz de realizar tarefas de reconhecimento e vigilância de extremo longo alcance e estão cada vez sendo mais equipadas com armamentos. Eles geralmente são operados pelo pessoal das forças aéreas a partir de estações de controle fixas;

- ***Medium altitude long endurance*** **(MALE)** – 5000 - 15.000 m de altitude e 24 h de autonomia. Suas funções são semelhantes às dos sistemas de HALE, mas geralmente operam em áreas um pouco menores, ainda assim excedendo os 500 km e sendo controlados a partir de estações de controle fixas;

- ***Medium Range or Tactical*** **UAV (TUAV)** – Com uma faixa de operação na ordem entre 100 e 300 km. Essas aeronaves são menores e operadas por sistemas mais simples do que os usados em aeronaves do tipo HALE ou MALE e são também operadas pelas forças armadas e navais;

- ***Close-Range*** **UAV** – usados por batalhões do exército para operações militares/navais e para diversos fins civis. Eles geralmente operam em faixas de até cerca de 100 km e são, provavelmente, os mais usados nos cenários citados, incluindo diversas outras aplicações como reconhecimento, monitoramento, segurança aérea, vigilância, inspeção de linhas de transmissão, pulverização de lavouras, monitoramento de tráfego, entre outros;

- ***Mini*** **UAV (MUAV)** – Refere-se a VANTs menores que uma certa massa (ainda não definida), geralmente menor que 20 kg, mas não tão pequenos como um MAV. São aeronaves que permitem a realização de lançamento à mão e de operação a distâncias de até 30 km, aproximadamente. Estes também são usados por grupos de batalha móveis e para diversos fins civis;

- ***Miniature Air Vehicle* (MAV)** – O MAV foi inicialmente definido como um VANT de envergadura inferior a 150 mm. Esse conceito já foi um pouco modificado, mas o MAV é aplicado principalmente em operações em ambientes urbanos, especialmente em edifícios. Ele costuma fazer voos a menores velocidades, executando operações tais como pairar no ar e realizar pequenos e rápidos pousos em espaços reduzidos. Geralmente espera-se que estas aeronaves sejam lançadas à mão, o que exige que elas sejam mais leves e, por consequência, elas acabam sendo mais vulneráveis a turbulências atmosféricas;

- ***Nano Air Vehicles* (NAV)** – Estes são projetados para terem o tamanho de uma semente e utilizados para formação de enxames de modo a serem aplicados para missões de vigilância com uma faixa de alcance extremamente pequena.

2.2.3 Aplicações de VANTs

Como descrito anteriormente, existe um grande número de possibilidades para aplicação de VANTs na execução dos mais diversos tipos de missões, não só na área militar, mas também no âmbito civil. Alguns exemplos de aplicações são listadas por (BEKMEZCI; SAHINGOZ; TEMEL, 2013; AUSTIN, 2011; MERINO *et al.*, 2006):

- Aquisição de dados e imagens de alvos em áreas inacessíveis por meios terrestres;
- Localização de alvos;
- Rastreamento;
- Mapeamento de vias e construções;
- Cenários de desastres causados pelo homem ou pela natureza;
- Busca e resgate;
- Reforço na aplicação da lei;
- Vigilância e monitoramento de tráfego;
- Identificação de rodovias;
- Inspeção de linhas de transmissão de energia;
- Medições;
- Agricultura de precisão;
- Cinematografia.

Uma vez caracterizadas as missões, cabe salientar que, para que elas possam ser executadas de forma confiável, faz-se necessária não só a garantia da segurança física (*safety*), mas também da segurança computacional (*security*), independentemente do tipo de aeronave, seja ela de pequeno, médio ou grande porte, esse tipo de mecanismo torna-se indispensável quando se optar por uma aeronave mais segura e pela inserção da mesma em um espaço aéreo segregado ou não-segregado.

Desta forma, o foco desta dissertação de mestrado é desenvolver a integração de *safety* e *security* que possa atender as necessidades para VANTs dos mais diferentes tipos e tamanhos, tomando como foco inicial o piloto automático do mesmo.

2.3 Piloto Automático

Um piloto automático pode ser descrito como o conjunto de *software* e *hardware* que juntos são responsáveis por controlar determinado tipo de veículo sem o auxílio de um piloto humano. Este sistema pode ser mais simples, como um piloto automático de nivelamento de casa com um único eixo, ou extremamente complexo que controla completamente o voo de uma aeronave desde sua decolagem até seu pouso (BEARD, 2012).

Um piloto automático, de modo geral, é implementado por meio de quatro subsistemas: Sistema de Navegação, Sistema de Controle, Unidade Inercial e Unidade Barométrica. Cada um desses subsistemas pode ser executado em processadores ou microprocessadores gerais ou dedicados.

O Sistema de Navegação é responsável por gerenciar as rotas e os periféricos adicionados à aeronave. No sistema o controle de trajetória é uma instanciação da lógica proposta por (PARK; DEYST; HOW, 2004) e os dados dos sensores são filtrados muitas vezes por um filtro de médias, que explora o fato dos dados de velocidade fornecidos pelo GPS terem uma precisão maior do que os de posicionamento.

De modo geral, o Sistema de Navegação recebe como entradas:

- **Plano de voo**, que consiste em um programa escrito em uma linguagem bastante simples especificando a missão que se deseja que o VANT desempenhe. Trata-se de uma entrada que deve ser fornecida antes da decolagem.
- **Região autorizada**, que consiste na especificação de um polígono georreferenciado ao qual todo plano de voo válido deve estar inscrito. Trata-se de uma entrada que, idealmente, é fornecida no momento da venda do VANT e não pode ser atualizada livremente pelo usuário.
- **Dados de sensores**, produzidos pela Unidade Barométrica e pela Unidade Inercial, entre os quais se destacam a altitude, a posição geográfica, a velocidade aerodinâmica e a

velocidade em relação ao solo. Trata-se de uma série temporal, com a qual o sistema é alimentado continuamente durante o voo.

- **Sinais secundários diversos de controle e comunicação**, que servem, por exemplo, para habilitar ou não a operação do sistema durante o voo, já que é permitido a um operador em terra assumir o comando do VANT.

Como saída, o Sistema de Navegação produz de modo geral:

- Comandos básicos de navegação, compostos por apenas dois valores, ângulo de *roll* e altitude. Trata-se de valores de referência que o sistema atualiza continuamente e envia ao Sistema de Controle;
- Alarmes para acionamento de periféricos, como câmeras.

O Sistema de Controle possui quatro funções principais:

- Ativar o modo de voo adequado, que pode ser: **Modo direto:** os comandos de solo de pilotagem são simplesmente repassados aos atuadores. Dessa forma, o piloto em solo possui controle total sobre a aeronave; **Modo estabilizado:** o piloto em solo comanda a aeronave de forma simplificada, com seus comandos sendo modificados antes de serem enviados para os atuadores. Isso permite um controle em tempo real simplificado para pilotos inexperientes; e **Modo autônomo:** o Sistema de Controle executa os comandos básicos fornecidos pelo Sistema de Navegação. Esse modo permite a execução autônoma de um plano de voo preestabelecido em solo.
- Processar comandos de pilotagem e configuração, que vêm codificados em modulação por largura de pulso do inglês *Pulse Width Modulation* (PWM), da mesma forma como são enviados para os servomecanismos. Esses comandos precisam ser decodificados para que possam ser tratados pelos algoritmos de controle;
- Assegurar a condição de voo adequada da aeronave dentro de suas especificações (envelope de voo). Os algoritmos de controle devem ser eficientes e bem calibrados de modo a manter o voo estável da aeronave, mesmo em condições adversas;
- Executar comandos básicos fornecidos pelo Sistema de Navegação. No modo autônomo, esses comandos definem a trajetória a ser seguida pela aeronave para a correta execução da missão. Nos modos de voo estabilizado e autônomo, as funções de controle são executadas por meio de algoritmos de controle, chamados de controladores. Os controladores usados podem ser de diversos tipos, desde os mais simples como *Proportional Integral Derivative* (PID) (ÅSTRÖM; HÄGGLUND, 2006), até controles robustos e baseados em H infinito (SIMON, 2006), tanto para os ângulos de rolagem quanto para a velocidade.

As entradas do Sistema de Controle em geral são:

- **Comandos de pilotagem e configuração**, recebidos a partir de um transmissor na estação de base. Os comandos de pilotagem correspondem às posições dos atuadores da aeronave conforme comandado pelo piloto em solo. Os comandos de configuração são comandos diversos, como a seleção do modo de voo;
- **Dados de sensores**, informando a situação da aeronave;
- **Comandos básicos de navegação**, que correspondem a manobras básicas da aeronave, utilizadas para atingir determinado objetivo pelo Sistema de Navegação. Esses comandos são gerados e fornecidos em tempo real pelo Sistema de Navegação de acordo com o plano de voo especificado e a condição atual da aeronave.

As saídas do Sistema de Controle são simplesmente os comandos dos atuadores da aeronave, que são os servomecanismos das superfícies de controle aerodinâmicas e o controlador do motor elétrico.

A Unidade Inercial é o módulo responsável por determinar a situação espacial da aeronave em tempo real. Para isso são utilizados como entrada não apenas dados de sensores inerciais, mas de sistemas de posicionamento global por satélite por meio de um receptor GPS. Em geral também são utilizados filtros para obtenção de resultados mais precisos e em frequências maiores. Os filtros geralmente utilizados são os filtros de Kalman, sejam os filtros tradicionais como o Kalman *Extended* e *Unscented* (LEWIS, 1986), (CRASSIDIS; JUNKINS, 2011).

Os dados de entrada dessa unidade são geralmente:

- **A posição geográfica (latitude, longitude e altitude) da aeronave**, fornecidos por um receptor GPS;
- **A velocidade da aeronave em relação ao solo em três eixos (norte, leste e vertical)**, fornecidos por um receptor GPS;
- **A aceleração da aeronave em três eixos**, no sistema de coordenadas da aeronave, coletadas por meio de acelerômetros;
- **Velocidades angulares da aeronave em três eixos**, no sistema de coordenadas da aeronave, coletadas por meio de giroscópios.

Já os dados de saída da Unidade Inercial costumam ser:

- Posição geográfica (latitude, longitude e altitude) da aeronave, filtrados e com frequência maior do que a do receptor GPS;

- Velocidade da aeronave em relação ao solo em três eixos (norte, leste e vertical), filtrada e com frequência maior do que a do receptor GPS;
- Atitude, representada por ângulos de Euler.

A Unidade Barométrica é responsável por fornecer a altitude barométrica, a velocidade vertical e a velocidade aerodinâmica da aeronave, todas baseadas em dados de pressão do ar. As suas entradas são duas pressões que atuam nos sensores: a pressão estática que é a pressão atmosférica na altitude de voo da aeronave; e a pressão total que é a soma da pressão estática com a pressão de impacto, que é a pressão criada pelo movimento da aeronave para a frente.

A pressão estática é utilizada tanto para a medição da altitude quanto para a medição da velocidade vertical. Apesar de essa pressão variar de um dia para o outro, a diminuição da pressão estática com a altitude é geralmente contínua em um mesmo local e hora. Assim, uma relação pressão-altitude baseada nas condições atmosféricas usuais é adotada como padrão, denominada atmosfera padrão. As medições de pressão estática são então utilizadas para prover indicações da altitude-pressão e da sua taxa de variação. Para a medição da velocidade aerodinâmica, a pressão de impacto é obtida como uma pressão diferencial a partir das medidas de pressão total e pressão estática.

Atualmente existe uma ampla variedade de pilotos automáticos para VANTs. Entretanto, a grande maioria desse pilotos automáticos, sejam eles comerciais ou acadêmicos, são pouco adaptáveis (tanto para modificações quanto para ampliações), sendo a grande maioria proprietários e não abertos (LIZARRAGA; CURRY; ELKAIM, 2011). Com o objetivo de estudo, foi escolhido o piloto automático SLUGS para o desenvolvimento deste trabalho. Desde modo o mesmo será melhor detalhado a seguir.

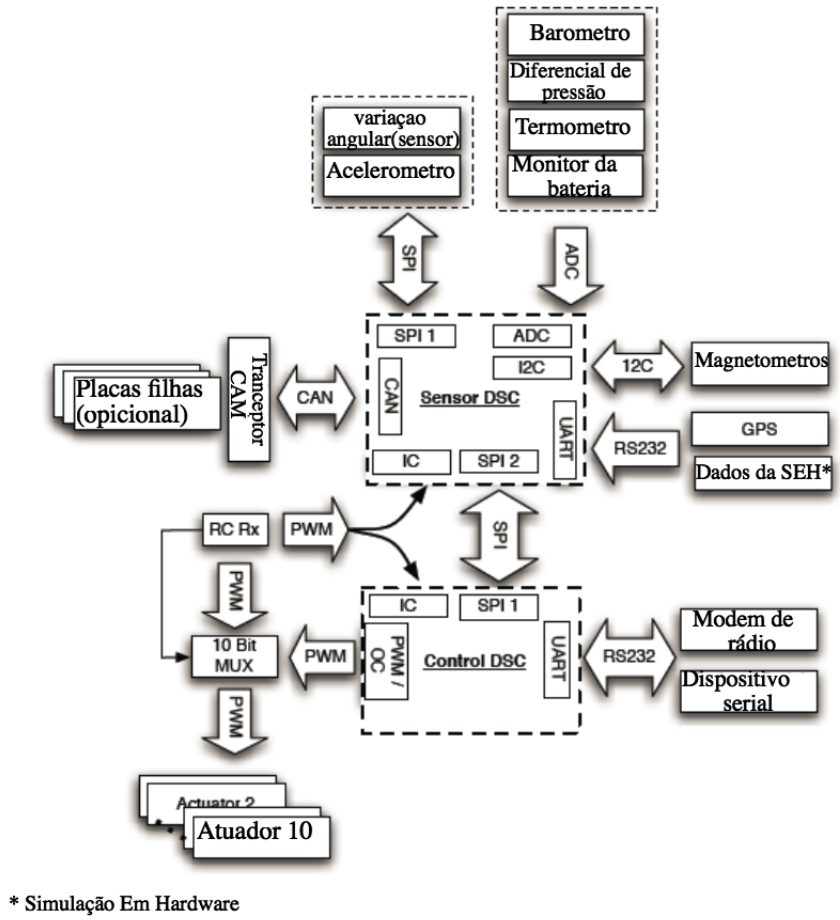
2.3.1 O Piloto Automático SLUGS

O *Autonomous Systems Lab* da *University of California Santa Cruz* desenvolveu o *Santa Cruz Low-cost UAV GNC System* (SLUGS) (LIZARRAGA; CURRY; ELKAIM, 2011), um piloto automático com um *hardware* que dá suporte a uma ampla gama de sensores, além de ser totalmente modificável por meio do sistema de *daughterboards* (“placas-filha”, ou seja, extensões da placa de circuitos principal do sistema, ou placa-mãe) que são conectadas às portas da rede *Controller Area Network* (CAN), uma rede de comunicação interna que possibilita que diversos microcontroladores e outros dispositivos da aeronave se comuniquem entre si sem a necessidade de um computador central.

O SLUGS conta com duas unidades de processamento, enquanto uma é utilizada para estimar sua posição e atitude a outra é responsável por sua navegação e controle. Suas unidades são definidas respectivamente por *Sensor DSC* e *ControlDSC*, com objetivos individuais muito bem definidos, de modo que para quem queira desenvolver uma das partes seja possível fazê-lo de forma isolada sem comprometer as funções da outra unidade. As duas unidades são

interconectadas por um barramento e são capazes de se comunicar por meio do protocolo *Serial Peripheral Interface* (SPI). Existem diversas portas/barramentos para troca de informação entre as unidades ou entre as *daughterboards*, ilustradas na Figura 5

Figura 5 – Distribuição dos componente de Hardware do SLUGS.



Fonte: Adaptada de: (FERNANDEZ, 2009).

O Sensor DSC tem como objetivo receber e processar dados sensoriais, transformando-os em unidades de engenharia que resultam posteriormente na posição e atitude da aeronave. Tal sistema possui os seguintes periféricos e módulos de conexões:

- Módulos SPI, responsáveis pelo recebimento de dados da unidade de medição inercial e pelo envio de dados ao Controle DSC;
- Módulos *Universal Asynchronous Receiver/Transmitter* (UART), utilizados para o gerenciamento do GPS e dos dados do teste de HIL;
- Módulos *Input Capture* (IC), responsáveis pela leitura dos comandos de PWM, recebidos do console na estação de base, e para leituras dos dados do motor;

- Sensores analógicos: sensores barométricos, responsáveis por computar a altitude e a velocidade do ar, o termômetro e o monitor de bateria.
- Conversores analógico-digital, utilizados para a leitura e conversão dos dados dos sensores analógicos;
- Barramentos CAN, que possibilitam que novas placas de expansão sejam conectadas à placa principal.

O sistema possui algoritmos para compensar erros causados pela variação de temperatura dos sensores e estimar posição e atitude da aeronave, além de efetuar a conversão dos dados dos sensores e do uso de filtros para atenuar ruídos. O algoritmo de compensação de temperatura utiliza um coeficiente linear extraído de uma curva de variação da leitura dos sinais para cada valor de temperatura, enquanto o algoritmo de estimação utiliza um filtro complementar para a posição horizontal e para a velocidade lidas pelo GPS e uma associação de valores lidos pelo GPS e pelo sensor barométrico para a altitude.

O Controle DSC, por sua vez, recebe comandos da estação base e dados do Sensor DSC sendo o responsável por manter a aeronave estável durante a execução de trajetórias.

O sistema possui os seguintes periféricos e módulos de conexões:

- Módulos SPI que são responsáveis pelo recebimento de dados do Sensor DSC;
- Módulos UART que são responsáveis por repassar a telemetria do voo para a estação de base por meio do transmissor de rádio;
- Módulos IC que são responsáveis pela leitura dos comandos de PWM, recebidos do console na estação de base; e
- Módulos PWM, pelos quais são enviados os sinais para controle dos servomecanismos das superfícies de controle.

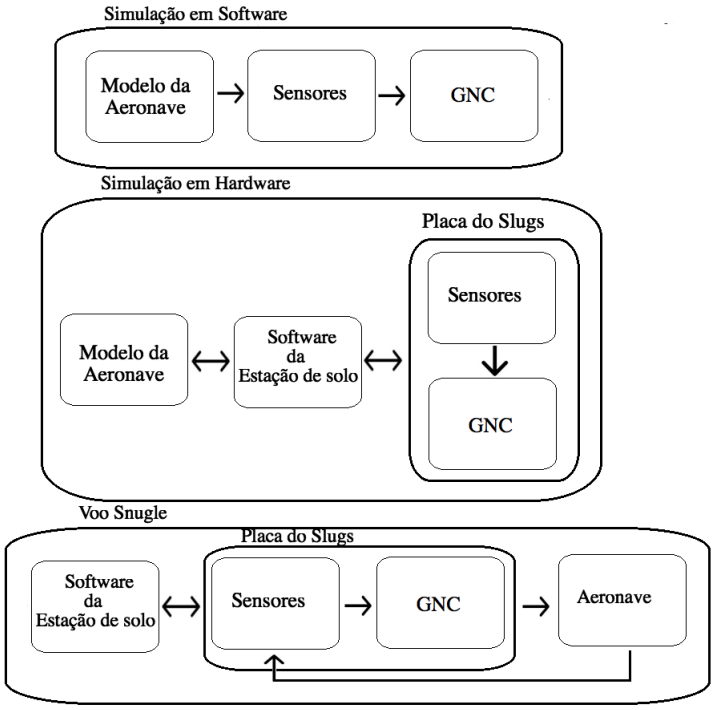
O piloto automático conta com um sistema de teste de três camada no contexto de navegação e controle, sendo:

- Simulação interna;
- *Hardware-in-the-loop* e;
- Voo real.

Estes modelos de testes são ilustrados de forma detalhada na Figura 6.

A simulação interna, baseada na técnica do *Inner/OuterLoop*, ou laços interno/externos, está implementada no código do Simulink® e possibilita que o usuário realize testes de voos

Figura 6 – Esquema do sistema de testes de três camadas do SLUGS.



Fonte: Adaptada de (TOEPKE, 2012).

sem necessidade do sistema físico. Uma vez que os testes na simulação interna são realizados e o sistema de software está validado, passa-se para a simulação *Hardware-in-the-Loop* (HIL), dependente do hardware.

Os simuladores HIL são ferramentas essenciais para o desenvolvimento de sistemas de controle de voo. De modo resumido, um simulador HIL gera dados sintéticos dos sensores da aeronave, como se eles estivessem em um voo real. Estes dados são analisados pelos processadores, que produzem comandos de controle que são mandados de volta ao simulador.

Por fim, uma vez que os testes com as simulações internas e as simulações HIL forem completados, os testes com voos reais podem ser realizados. Estes testes finais têm o intuito de confirmar que todos os componentes do sistema funcionam corretamente.

São três os modos de operação do SLUGS:

- ***Pilot Feedthrought***, no qual os comandos do piloto são recebidos via rádio e são repassados diretamente para o piloto automático;

- ***Manual Direct***, no qual a estação de base envia comandos com os valores de referência para o laço interno do piloto automático. Esse modo pode ser combinado com o modo ***Pilot Feedthrought***, permitindo que o piloto controle algumas superfícies de controle e o piloto automático controle outras;

- ***Autonomous-waypoint Navigation***, no qual a estação de base configura os *waypoints* que formam a trajetória e o algoritmo de navegação gera os valores de referência para o laço interno.

Como já mencionado, o algoritmo de controle e navegação do Controle DSC é implementado em uma arquitetura chamada *inner/outer loop*. Assim, o laço interno é composto por controladores PIDs que são responsáveis por estabilizar a aeronave, enquanto o laço externo é responsável por gerenciar a navegação baseada em *waypoints*, gerando os valores de referência que são passados ao laço interno.

A estrutura do laço interno é dividida em canal lateral, responsável pelo controle do ângulo de *pitch* e da velocidade, e canal longitudinal, responsável pelo controle dos ângulos de *roll* e *yaw*. Os controladores possuem limites de saturação para os servomecanismos, podem ter seus parâmetros ajustados individualmente durante o voo e enviam os sinais para as superfícies de controle - ailerons, profundores e leme - e para o motor.

O laço interno também mantém a velocidade constante e controla a altitude, durante o voo, por meio de dois controladores PIDs em cascata, o primeiro que comanda o *pitch*, e o segundo que comando o profundor para manter o valor do *pitch* determinado pelo primeiro controlador.

O laço externo utiliza a técnica de navegação baseada em *waypoints*, calculando-se a diferença entre o ponto atual e o próximo ponto. O ângulo entre a aeronave e a diferença entre os pontos é usado para comandar a aceleração lateral da aeronave, o que corresponde ao controle do ângulo de *roll*. Quanto menor for essa diferença de distâncias, mais precisa é a trajetória, porém aumenta-se o número de oscilações ao redor dos pontos desejados.

O código do SLUGS foi totalmente desenvolvido em Simulink® o que permite uma prototipagem do controle possibilitando um melhor entendimento do sistema. A utilização da capacidade de geração automática de códigos dessa plataforma possibilita que se altere desde poucos detalhes na arquitetura do controle do sistema até a completa reestruturação desta arquitetura sem que seja necessário modificar diretamente o código fonte do sistema.

O piloto automático SLUGS é interessante por utilizar Desenvolvimento Dirigido a Modelos (detalhado na Seção 2.4), além de ter sido desenvolvido com a ferramenta Simulink® e ser gratuito. Esses quesitos, associado ao fato de haver um conhecimento prévio do mesmo pelo grupo de pesquisa em que este projeto está inserido, permitiu a sua escolhido como plataforma alvo.

2.4 Engenharia Dirigida a Modelos

2.4.1 Abordagens Orientadas a Modelos

Abordagens dirigidas a modelos apresentam novas maneiras de gerenciar a complexidade do desenvolvimento do software e de possibilitar a criação de novos casos de testes, de automatizar procedimentos de teste e de incentivar a reutilização, permitindo com isso reduzir a quantidade de erros e acelerar a produção de software (SOUSA *et al.*, 2011).

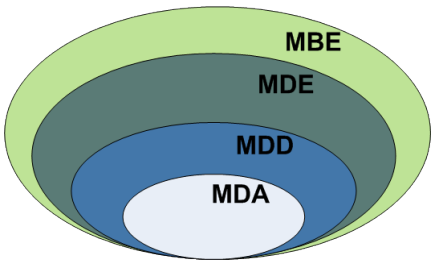
Essa abordagem permite a redução de custo e tempo de desenvolvimento de software, possibilitando uma melhoria da qualidade dos produtos e retorno rápido do investimento efetuado, pois permite a inclusão de benefícios de tecnologias emergentes nos sistemas existentes.

Na abordagem orientada a modelos, os modelos se tornam artefatos para serem mantidos junto com o código, desde que o benefício obtido a partir da produção dos modelos seja consideravelmente maior e, que o esforço necessário para mantê-los de acordo com o código seja consideravelmente menor do que a prática atual.

Engenharia Dirigida a Modelos (*Model Driven Engineering* (MDE)) constitui aquela em que são utilizados modelos na construção do sistema, na comunicação das decisões de projeto e na geração dos artefatos de projeto (MILICEV, 2009). A MDE é um superconjunto do Desenvolvimento Dirigido a Modelos (do inglês *Model Driven Development* (MDD)), pelo fato de englobar outras tarefas dirigidas por modelos do processo completo da Engenharia de Software. Já a Arquitetura Dirigida a Modelos (*Model Driven Architecture* (MDA)) constitui uma visão particular da OMG[3], sendo um *framework* conceitual que realiza a abordagem MDD, a qual, por sua vez, estabelece um conjunto de modelos para apoiar a construção de software para que os mesmos sejam os principais artefatos do processo de desenvolvimento. O Teste Dirigido a Modelo (*Model Driven Test* (MDT)) visa agregar a qualidade aos produtos de software que são desenvolvidos seguindo estas abordagens.

A hierarquia da abordagem orientada a modelos é apresentada de modo sumarizado na Figura 7.

Figura 7 – Hierarquia da Abordagem Orientada a Modelos.



Fonte: Extraída de (CABOT, 2015).

[3] http://www.omg.org/mda/executive_overview.htm

2.4.2 Desenvolvimento Orientado a Modelos

Um dos novos paradigmas voltados para o desenvolvimento de software é o MDD, cuja finalidade principal é a introdução de modelos que apresentam certo rigor durante todo o processo de desenvolvimento do software, permitindo assim a geração automática de código a partir desses modelos.

Segundo (SELIC, 2003) MDD pode ser definido como uma especialização da Engenharia Dirigida a Modelos esta, por sua vez, é descrita por (SCHMIDT, 2006) como uma abordagem própria para utilização em sistemas de diferentes complexidades pois consegue expressar os conceitos do domínio em que é empregada de forma eficaz.

O termo modelo pode variar de acordo com a área que é utilizado, na computação há um consenso proposto por Rothenberg em (ROTHENBERG *et al.*, 1989) que diz:

"modelagem, no sentido mais amplo, é o uso econômico de algo no lugar de outra coisa para algum propósito cognitivo. Ela permite usar algo que é mais simples, mais seguro ou mais barato do que a realidade para algum propósito. Um modelo representa a realidade para um determinado propósito; o modelo é uma abstração da realidade, no sentido de que ele não pode representar todos os aspectos da realidade. Desta forma, é possível lidar com o mundo de uma forma simplificada, evitando a complexidade, o perigo e a irreversibilidade da realidade."

A principal diferença entre o processo tradicional e o MDD são os artefatos gerados ao longo das fases do MDD, uma vez que esses são modelos passíveis de transformação especificados por meio de Linguagens Específica de Domínio (*Domain-Specific Language* (DSL)), permitindo altos índices de automação entre as fases.

A ideia principal é permitir que o engenheiro de software não interaja manualmente com todo o código, mas sim que o mesmo se concentre em modelos de mais alto nível, ficando assim protegido das complexidades necessárias para a implementação nas mais diferentes plataformas. Com isso busca-se evitar problemas em reutilização de conhecimento, na produtividade, na manutenção e documentação quando são realizadas alterações, na validação e otimização, na portabilidade e interoperabilidade (BHANOT *et al.*, 2005), (MERNIK; HEERING; SLOANE, 2005), (KLEPPE *et al.*, 2003), (DEURSEN; KLINT *et al.*, 1998), que são vistas como as principais vantagens do uso de MDD. Ademais, tem-se a possibilidade de expressar modelos usando termos e conceitos que são ligados ao domínio do problema e não ligados às tecnologias empregadas na implementação do mesmo, isto faz com que seja mais fácil de se especificar e de se compreender os modelos (SELIC, 2003).

Por outro lado, tem-se também as desvantagens na utilização de MDD, como a rigidez imposta pela falta de alcance do código pelo desenvolvedor, a complexidade obtida pelos códigos que são gerados automaticamente, o desempenho que muitas vezes mesmo com otimizações fica aquém do desejado, a curva de aprendizado e o alto investimento inicial (THOMAS, 2004), (AMBLER, 2003).

Os principais elementos do MDD são:

- ferramentas de modelagem;
- ferramenta para definir transformações;
- o modelo;
- as transformações;
- os mecanismos para executar as transformações;
- o código-fonte e;
- outros modelos.

Os modelos descrevem os conceitos do domínio, e são utilizadas ferramentas de modelagem para isso. Esses modelos, gerados pelas ferramentas, servem de entrada para transformações que irão gerar outros modelos ou códigos-fonte. As transformações são definidas por meio da construções de regras de mapeamento de modelo para modelo, ou de modelo para código. Por fim, há um mecanismo que aplique as transformações, permitindo que essas sejam não só executadas mas também rastreadas, de modo a saber a origem de cada elemento gerado, seja para o novo modelo ou para o código-fonte.

Vários software embarcados, incluem-se aqui os pilotos automáticos de VANTs, têm sido desenvolvido fazendo uso de abordagens orientadas a modelo, principalmente por meio do paradigma MDD. O piloto automático do SLUGs é um deles, e será foco de estudo nesta dissertação de mestrado.

2.5 Considerações Finais

Este capítulo abordou conceitos básicos de VANTs, sua composição, incluindo e detalhando o piloto automático, a parte responsável pela estabilização e pelo voo autônomo desses veículos. É possível observar que sendo um dos elementos chave dos VANTs, o software do piloto automático pode apresentar vulnerabilidades de segurança computacional, e essas vulnerabilidades podem afetar a segurança física do mesmo.

Neste sentido, o piloto automático SLUGs, desenvolvido com base no paradigma de desenvolvimento orientado a modelos foi escolhido como plataforma alvo dos testes de segurança computacional e física a serem realizados neste trabalho. O SLUGS foi também detalhado, sendo justificada sua escolha.

Por fim o capítulo apresentou os conceitos relacionados ao desenvolvimento orientado a modelos, deixando clara quais são as vantagens e desvantagens de seu uso.

Uma vez apresentados os conceitos e elementos básicos que compõe este projeto, o próximo capítulo apresenta o ciclo de vida de engenharia de segurança física e computacional.

CAPÍTULO

3

ENGENHARIA DE SEGURANÇA

3.1 Considerações Iniciais

Neste capítulo são apresentados os conceitos sobre segurança física e computacional, incluindo terminologias de segurança, conceitos de ataques e vulnerabilidades, bem como análise de risco, e técnicas necessárias para o correto entendimento deste trabalho.

3.2 Terminologias

Embora os principais termos de segurança física e computacional utilizados nesta dissertação possam mudar dependendo do domínio, padrão e região geográfica, a fim de evitar ambiguidades, a definição desses termos é destacada nesta seção.

A **segurança física** no contexto da avaliação de risco é definida como **liberdade do risco inaceitável** (PAPADOPOULOS; MCDERMID, 1999b). A segurança física pode ser confundida com confiabilidade. No entanto, eles têm significados diferentes. **Confiabilidade** refere-se à *probabilidade de que um equipamento ou componente desempenhe sua função pretendida satisfatoriamente por um tempo prescrito e sob condições ambientais estipuladas* (LEVESON; GOETSCH, 1995). Portanto, a confiabilidade está relacionada a todas as falhas potenciais, enquanto a segurança física está associada apenas a **falhas perigosas** (LEVESON, 1986). Um **perigo** (***hazard***) é definido como *a fonte potencial de dano causado pelo comportamento defeituoso do item* (PALIN *et al.*, 2011; ISO, 2011). Um item é definido como *um sistema ou uma matriz de sistemas que implementam uma função* (PALIN *et al.*, 2011; ISO, 2011). Considerando que a segurança física deve ser considerada no contexto de um sistema particular em um determinado ambiente operacional, **um sistema** é definido como *uma combinação, com limites definidos, de elementos que são usados juntos em um ambiente operacional definido para executar uma determinada tarefa ou atingir um propósito específico. Os elementos podem incluir pessoas, procedimentos, materiais, ferramentas, equipamentos, instalações, serviços e/ou*

software, conforme apropriado (MOD, 2007).

No contexto de análise de risco e processos de avaliação de risco da engenharia de sistemas para sistemas críticos de segurança, um **risco de segurança** é definido como a *combinação da probabilidade de dano e a gravidade desse dano* (MOD, 2007). **Danos** podem ser definidos como *morte, danos físicos, danos à saúde das pessoas ou danos à propriedade ou ao meio ambiente* (MOD, 2007). **Uma falha** (*failure*) é *a incapacidade de um sistema ou componente do sistema para executar uma função necessária dentro dos limites especificados. Uma falha pode ser produzida quando uma falha é encontrada* (JACKLIN, 2012). **Uma falha** (*fault*) é uma *manifestação de um erro, se ocorrer, pode causar uma falha (failure)* (JACKLIN, 2012).

Os requisitos de segurança podem ser alocados para mitigar os efeitos de uma falha. O **requisito de segurança** é definido como as medidas de redução de risco associadas a um determinado risco ou falha de componente (MOD, 2007). Existem três tipos de requisitos de segurança. Os **requisitos de segurança do sistema** são alocados para os perigos no nível do sistema. As decisões de projeto e implementação, por exemplo, funções do sistema, destinadas a eliminar ou minimizar os efeitos de falhas na segurança do sistema são chamadas de **requisitos de segurança funcional** (PALIN *et al.*, 2011) ou **requisitos de segurança derivados** (MOD, 2007). Finalmente, o **requisito de integridade de segurança** especifica a confiabilidade (risco), por exemplo, em termos de probabilidade e severidade, associada a um *requisito de falha funcional*, *perigo* ou *segurança funcional*. Na gestão de segurança física, **garantia** é definida como *ações planejadas e sistemáticas necessárias para fornecer confiança e evidências adequadas de que um produto ou processo atende aos requisitos de segurança* para obter a certificação de segurança (JACKLIN, 2012). A partir da análise das definições existentes para **evidência**, nesta dissertação, a evidência é considerada como a *informação que serve como fundamento e ponto de partida dos argumentos (de segurança), com base nos quais o grau de verdade das alegações em argumentos pode ser estabelecido , desafiado e contextualizado* (SUN, 2012).

A **segurança computacional**, por sua vez, trata da **prevenção** e da **detecção** de ações não autorizadas a um sistema computacional (GOLLMANN, 2010). Sendo assim, a necessidade de proteção deve ser definida a partir das possíveis **ameaças**, *qualquer coisa que possa afetar ou atingir o funcionamento, operação, disponibilidade, integridade da rede ou sistema* e riscos que um sistema pode sofrer.

Por **proteção** entende-se *a necessidade de tomar medidas para se resguardar contra um objeto*, seja ele um crime, uma sabotagem ou um ataque. **Ataque** consiste por sua vez em *técnica específica usada para explorar uma vulnerabilidade* (REBECCA, 2000), ou seja, explorar as **fraquezas** inerente de um elemento do sistema ou mais especificamente uma **brecha**, um ponto fraco ou falha. Um ataque pode ser entendido também como uma atividade maliciosa que procura violar a política de segurança (DEBAR; DACIER; WESPI, 1999).

Quando um ataque é bem sucedido diz-se que ocorreu uma **intrusão** (VACCA, 2012), ou seja, ocorreu uma violação intencional da política de segurança de um sistema (REBECCA,

2000). **Política de segurança** corresponde a uma especificação que define *o que é considerado valioso e um conjunto de passos que devem ser seguidos para salvaguardar os bens ou valores* (GARFINKEL; SPAFFORD; SCHWARTZ, 2003).

Com intuito de *coletar informações e encaminhá-las para análise*, objetivando saber se ocorreu ou não um ataque, ou se há possibilidade de ocorrência, é feito o **monitoramento** (REBECCA, 2000). Os dados então são analisados e a interpretação, combinação e análise de informações sobre um alvo de ataque para propósitos de detecção e resposta (AMOROSO, 1999) é determinado **correlação**.

Em caso de haver um *relato de ocorrência de uma violação da política de segurança* (AMOROSO, 1999), tem-se então um **alarme**. Quando um alarme é disparado e *não existe uma real violação da política de segurança* tem-se um **falso-positivo** (PROCTOR, 2000), e em caso contrário onde *não há um disparo de alarme mas há uma violação da política de segurança* (PROCTOR, 2000), tem-se um **falso positivo**.

De modo análogo há também o **verdadeiro-positivo**, que compreende a situação onde *um alarme é disparado devido a uma violação da política de segurança* (PROCTOR, 2000) e o **verdadeiro-negativo**, que compreende a situação onde *um alarme deixa de ser disparado quando não existe uma violação da política de segurança* (PROCTOR, 2000).

Foram apresentados nesta seção os termos mais relevantes para este trabalho e seus significados, Vale ressaltar que esses termos não representam uma terminologia exaustiva utilizada na área de segurança, tanto física quanto computacional, entretanto, as informações aqui descritas facilitam o entendimento de novos conceitos e discussões encontrados a partir desta seção. Alguns conceitos serão reforçados e mais detalhados nas seções que seguem.

3.3 Segurança Física - *Safety*

O termo segurança física, ou simplesmente segurança na língua portuguesa, pode assumir diferentes significados dependendo do contexto em que está inserido, conforme previamente mencionado. Pode-se definir a segurança física como o fato e evitar perigos para o ambiente físico devido à operação de um dispositivo sob condição de falha, seja esta normal ou única. Essa definição pode ser ampliada considerando-se também o alcance dos riscos elencados, incluindo o funcionamento defeituoso da unidade de computação, efeitos térmicos, problemas de biocompatibilidade, falhas de software, riscos mecânicos e elétricos. No contexto de avaliação de risco, que é o contexto em que o termo se insere neste projeto, *safety* pode ser definido como "*freedom from unacceptable risk*" (PAPADOPOULOS, 2015), ou seja inexistência de risco inaceitável. A segurança física é essencial e conta com o gerenciamento de *saúde* dos componentes do sistemas e da aviônica.

O conceito de segurança física está associado a falhas perigosas, conhecidas no inglês

como *hazardous failures* (LEVESON, 1986). Os perigos (do inglês *Hazards*) no ambiente físico podem ocorrer devido a condições anormais nas propriedades do veículo. Portanto, eles podem ser matematicamente caracterizados como restrições sobre os valores das propriedades dos elementos desse veículo. Faz-se possível então relacionar propriedades computacionais às propriedades dos elementos do veículo por meio do mapeamento de interação. Entretanto, a natureza dinâmica do ambiente físico não pode ser deixada de lado, tem que ser levada em consideração.

Assim, para garantir a segurança física faz-se necessário caracterizar com precisão a dinâmica espaço-temporal do ambiente físico e seu acoplamento com as unidades de computação bem como com cada elemento que compõe o veículo. As variações nas propriedades (*safety* e *reliability* aud podem variar de acordo com a configuração do sistema e características de seu ambiente operacional) podem, portanto, causar violações de restrições nas propriedades dos elementos. Assegurar a segurança do ambiente físico devido à operação das unidades de computação devem essencialmente considerar a caracterização do mapeamento das propriedades dos elementos e as computacionais.

Na Figura 8 são ilustrados os passos sugeridos pelo *standard IEC 61508*(SCHOITSCH, 2005), que de forma geral podem ser divididos em 3 grandes blocos, no bloco 1 tem-se a integração da **Identificação do Perigo**, no bloco 2 **Avaliação de Risco** e no bloco 3 **Definição de Requisitos de Segurança Física**. Tais etapas são detalhadas nas seções seguintes.

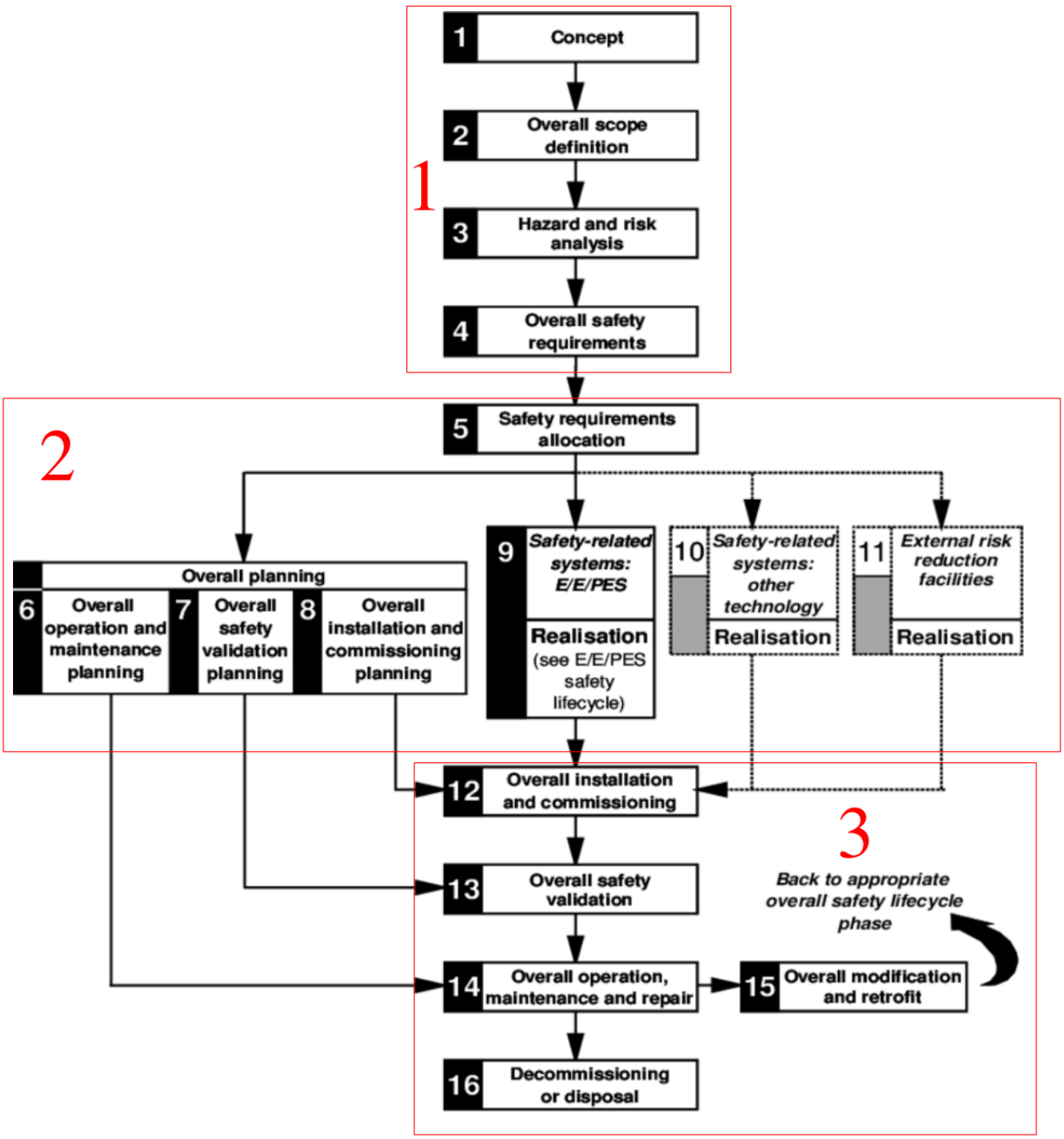
3.3.1 Identificação de perigo

A identificação de perigo (*hazard identification*) pode ser executada após a especificação dos requisitos do sistema e o entendimento do sistema e do ambiente-alvo de operação. O sistema deve ser especificado em termos de ambiente físico e operacional, limites do sistema, funções e dependências inter-funcionais. A identificação de perigo estabelece perigos e sequências de eventos que podem causar falhas no sistema (MOD, 2007).

Os perigos identificados são normalmente armazenados em um registro, que é atualizado por todo o ciclo de vida de desenvolvimento do sistema. A identificação de risco pode ser realizada por meio de *checklists*, análise "what-if", avaliação funcional de falha (*Functional Failure Assessment*) (Society of Automotive Engineers (SAE), 1994), ou estudos de risco e operabilidade (*HAZard and OPerability studies* (HAZOP)) (KLETZ, 1999).

O HAZOP é um método recomendado para identificar *hazards* e problemas de forma a prevenir possíveis problemas durante a operação. A técnica HAZOP permite que analistas de segurança (*safety*) façam suposições sobre como o sistema poderá falhar de forma sistemática. Normalmente este tipo de analise é feito por equipes, e cada membro da equipe é estimulado a verificar as anotações de outros membros. São selecionadas as *guide-words* que foram utilizadas pela equipe, para que exista uma padronização das anotações, assim como o que elas indicariam

Figura 8 – Adaptação de modelo da EIC 61508.



Fonte: Adaptada de IEC 61508(SCHOITSCH, 2005).

para cada parâmetro. Essas palavras e seus impactos no componente podem variar de equipe para equipe, sendo assim tabelas são providas para garantir os real impacto das *guide-words*. Um exemplo desse tipo de tabela é ilustrado na Tabela 1.

A identificação de perigo pode ser executada interativamente de acordo com o andamento do projeto do sistema, sendo assim, o nível de entendimento dos perigos evolui paralelamente à evolução da arquitetura do sistema e seu desenvolvimento, sendo este último baseado nas informações obtidas a partir da operação do sistema (HABLI; KELLY; PAIGE, 2009).

Tabela 1 – Tabela de guide-words e parâmetros genéricos.

Guide Words\ Parâmetro	**More**	**Less**	**None**	**Part of**
Falha de Componente	-	-	Failure	-
Tempo de execução	Muito demorado / Atrasado(*Late*)	Muito rápido / Adiantado (*Early*)	Não executado	Execução de atividades além das esperadas
Pressão	Alta Pressão	Baixa Pressão	Vácuo	-
Temperatura	Alta Temperatura	Baixa Temperatura	-	-

Fonte: Adaptada de (KLETZ, 2001)

3.3.2 Avaliação de risco

A avaliação de risco (*risk assessment*) é realizada após a identificação de perigos com o objetivo de estimar, com base em critérios probabilísticos tais como probabilidade e severidade, o risco representado por estes perigos.

Em muitos países, como Reino Unido e Estados Unidos, é necessário estabelecer um critério de tolerabilidade de riscos bem como uma demonstração de que riscos estabelecidos por perigos são tão baixos quanto razoavelmente praticáveis ("*As Low As Reasonably Practicable* (ALARP)). Isso significa que para um risco ser considerado ALARP, o custo de medidas adicionais de redução de riscos deve ser grosseiramente desproporcional ao benefício obtido pela redução de risco (Health and Safety Executive, 2016). Determinar se um risco é considerado ALARP requer, na maioria dos casos, a adoção de técnicas formais de tomada de decisões. De acordo com o princípio ALARP, um risco pode ser classificado como:

- **Intolerável**: o risco não satisfaz o princípio ALARP e portanto deverá ser reduzido;
- **Tolerável**: o risco satisfaz o princípio ALARP;
- **Amplamente aceitável**: o risco é aceitável contanto que o sistema opere de acordo com boas práticas, possua métodos para a verificação formal e para geração de evidências, as quais demonstram que um risco associado a um determinado perigo é aceitável.

3.3.3 Definição de Requisitos de Segurança Física

A definição de requisitos de segurança física é realizada a partir da análise dos resultados fornecidos pela identificação de perigos e pela avaliação de risco. Dessa forma, as medidas de redução de risco são definidas para cada perigo identificado no sistema na forma de Requisitos de Segurança Funcional (do inglês: *Functional Safety Requirement*) ou níveis de integridade de segurança (do inglês Safety Integrity Levels (SILs)). Os requisitos de integridade de segurança

são definidos de acordo com os critérios quantitativos/probabilísticos estabelecidos no padrão de segurança alvo (MOD, 2007).

A maioria dos padrões de segurança física adota tais critérios para estabelecer os requisitos de integridade (ISO, 2011; ARP4754A, 2010). Nestes padrões, quanto maior é o risco de uma ameaça à segurança física (perigo), mais rigorosos serão os requisitos de integridade. Esses padrões também estabelecem diferentes critérios quantitativos para especificar os requisitos de integridade de segurança. Por exemplo, um nível de integridade ou nível de *safety* pode ser definido pela relação entre tempo médio para falha (do inglês - *Mean-time to Failure*), probabilidade de operação sem falhas ou indisponibilidade.

3.4 Ferramentas e técnicas de Análise de Segurança Física

Como já mencionado anteriormente, as atividades de Identificação de Perigos, Avaliação de Riscos e Definição de Requisitos de Segurança são o ponto de partida do ciclo de *Engenharia de Segurança de um sistema crítico*. Há na literatura um conjunto de técnicas de apoio às atividades de engenharia de segurança. Essas técnicas são classificadas em **tradicionais** e **dirigidas a modelos**, como descrito nas seções subsequentes.

3.4.1 Técnicas Tradicionais

As técnicas de análise de *safety* existentes são comumente categorizadas como indutivas e dedutivas (ARP4754A, 2010). As técnicas de análise indutiva são conhecidas por funcionarem atuando *de baixo para cima*, uma vez que a análise começa a partir de eventos de falha em componentes e identifica seus potenciais efeitos no sistema como um todo, identificando os perigos nos quais os eventos de falha podem contribuir direta ou indiretamente. Exemplos de técnicas indutivas citam-se *Functional Failure Analysis* (FFA) (ENGINEERS, 1996), *Failure Mode and Effect Analysis* (FMEA) (MILITARY, 1949) e *Event Tree Analysis* (ERICSON, 2005). Ao contrário, por ser uma técnica *top-down*, a análise dedutiva parte da obtenção das informações *de cima para baixo*, ou seja, a partir de uma falha em nível de sistema (um perigo, por exemplo) e procura eventos de falha que possivelmente são causas para o evento superior. Tem-se como exemplo desta técnica *Fault Tree Analysis* (ERICSON, 1999), (National Aeronautics and Space Administration (NASA), 2002).

3.4.1.1 Functional Failure Analysis - FFA

A análise funcional de falhas visa examinar como as funções do sistema podem impactar ou contribuir para a segurança do sistema (ENGINEERS, 1996). A FFA identifica as condições de falha associadas às funções do sistema e seus efeitos na segurança geral. As condições de

falha podem ser identificadas com base nos seguintes desvios: função não fornecida, função fornecida quando não requerida ou função fornecida incorretamente. Posteriormente, os efeitos de cada condição de falha são identificados pela definição de como as condições de falha afetam o comportamento pretendido do sistema, seu ambiente e usuários. Os efeitos fornecem suposições sobre o ambiente do sistema, por exemplo, temperatura externa e fase de voo nos sistemas aeroespaciais. A FFA classifica as condições de falha com base na gravidade de seus efeitos, onde as condições de falha que levam à morte ou lesões são classificadas como “Catastróficas” ou “Perigosas”. Por outro lado, os efeitos de condição de falha menos graves são normalmente classificados como “Principais” ou “Secundários”. Os critérios para classificar os efeitos das condições de falha variam de acordo com o padrão de segurança alvo. Finalmente, os requisitos de integridade de segurança são alocados para efeitos de condição de falha de acordo com sua gravidade.

Considerando o ciclo de vida de segurança, como definido nos padrões ISO 26262, ARP 4754A e IEC 61508(SCHOITSCH, 2005), e análise de segurança de composição, na identificação de perigos, a FFA identifica combinações entre condições de falha funcional, perigos nomeados, que levam à ocorrência de nível de falhas de sistema. Os riscos apresentados por esses perigos são avaliados com base em critérios probabilísticos, por exemplo, probabilidade e gravidade, e os requisitos de integridade de segurança são alocados para cada perigo identificado. Posteriormente, na análise, a FFA identifica condições de falha associadas a cada função do sistema que, direta ou indiretamente, contribuem para a ocorrência de perigos (WILKINSON; KELLY, 1998a)

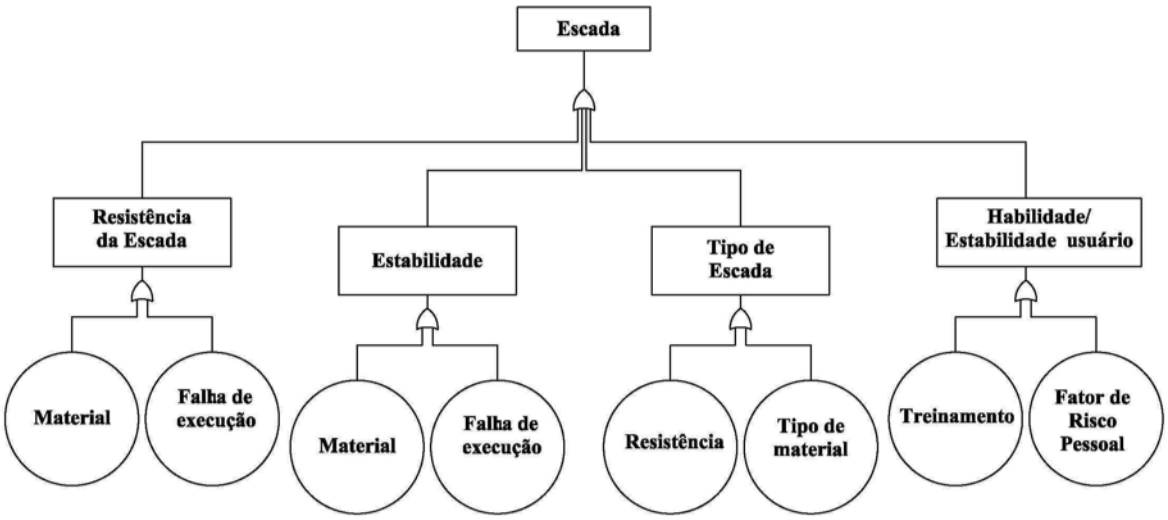
3.4.1.2 Fault Tree Analysis - FTA

A análise de árvore de falhas *Fault Tree Analysis* (FTA) (ERICSON, 1999), (National Aeronautics and Space Administration (NASA), 2002) é um método de análise de segurança bem estabelecido e amplamente utilizado na engenharia de sistemas críticos nos domínios aeronáutico, automotivo, de trem e nuclear.

Este método serve para auxiliar analistas de segurança a identificar potenciais falhas de sistema de modo antecipado, ainda no projeto dos mesmos, de modo a usar tais informações para prevenção de falhas. A FTA é uma técnica dedutiva de análise de eventos de alto nível, identifica as falhas de sistema, deduzindo então suas causas.

Neste método, a análise é realizada baseada no projeto preliminar da arquitetura do sistema de modo a identificar falhas em componentes, chamados de eventos básicos, que levam a ocorrência de falhas de sistema (perigos). Árvores de falhas são representações gráficas de combinações lógicas de falhas de componentes que consistem em eventos de alto nível conectados a um ou mais eventos básicos por meio de portas lógicas como **AND**, **OR** ou **NOT** como exemplificado na figura 9. A FTA pode ser executada qualitativamente, por meio de análise lógica, ou quantitativamente, por meio de análises probabilísticas.

Figura 9 – FTA genérica.



Fonte: Adaptada de (National Aeronautics and Space Administration (NASA), 2002).

3.4.1.3 *Failure Mode and Effect Analysis - FMEA*

A FMEA, ao contrário da FTA é uma técnica indutiva em que a análise começa a partir de modos de falha conhecidos, inferindo os efeitos dessas falhas na segurança geral do sistema (PAPADOPOULOS *et al.*, 2011; PUMFREY, 1999). Assim, o FMEA difere do FFA, onde a análise é baseada em modos hipotéticos de falha. Um efeito em um FMEA corresponde a um evento de topo de uma árvore de falhas. Os efeitos podem ser avaliados com base em critérios como gravidade, probabilidade e detectabilidade, que podem ser combinados para estimar o risco geral. Os resultados de uma FMEA são geralmente apresentados em uma forma tabular. Independentemente do domínio, as tabelas de FMEA incluem basicamente a identificação de (PUMFREY, 1999):

- O componente ou subsistema em consideração;
- Os modos de falha conhecidos do componente/subsistema; e
- Os efeitos de cada modo de falha.

Se *Failure Modes, Effects and Criticality Analysis* (FMECA) (MILITARY, 1949), uma variação do FMEA, que classifica o risco apresentado por cada modo de falha com base em suas contribuições para os perigos no nível do sistema de acordo com critérios como gravidade/probabilidade/classificação de risco. Uma vez realizada, colunas para informações críticas devem ser incluídas (National Aeronautics and Space Administration (NASA), 2002).

Apesar das semelhanças na estrutura e no processo de análise de técnicas indutivas de FMEA/FMECA e FFA, o FFA é uma técnica preditiva comumente usada no início do ciclo de vida de segurança, para identificar ameaças potenciais à segurança do sistema e alocar requisitos

de segurança, e FMEA é usado tardiamente no ciclo de vida de segurança para demonstrar que o sistema atendeu aos requisitos de segurança alocados (PUMFREY, 1999). As FTAs e o FMEA/FMECA fornecem informações valiosas sobre o comportamento de falha do sistema e podem ser usados complementares entre si. As FTAs fornecem as informações causais para as falhas no nível do sistema, e a FMEA/FMECA fornece uma visão do impacto das falhas dos componentes na segurança geral do sistema. No entanto, devido a FTA e FMEA serem métodos manuais, sua aplicação em sistemas complexos é propensa a erros e consome tempo. Como o projeto arquitetural do sistema afeta a segurança geral, mudanças no projeto do mesmo exigem uma revisão dispendiosa de FTAs e artefatos de FMEA (PAPADOPOULOS *et al.*, 2011). Isso ocorre porque a FTA e a FMEA não produzem modelos reutilizáveis. Além disso, a realização de FTA e FMEA exige o conhecimento de todo o sistema, o que não é feito facilmente no contexto de sistemas complexos que combinam tecnologias tradicionais com controladores baseados em computador (LISAGOR; MCDERMID; PUMFREY, 2006).

As deficiências das técnicas tradicionais de FTA e FMEA foram reconhecidas não apenas por pesquisadores, mas também por fabricantes de aeronaves com a revisão do documento SAE ARP 4761 (ENGINEERS, 1996) que inclui uma seção específica dedicada à avaliação automatizada de segurança baseada em modelos. Portanto, várias ferramentas e técnicas foram desenvolvidas para automatizar o processo de análise de segurança do sistema. Essas técnicas são discutidas posteriormente, já que constituem técnicas mais modernas e normalmente dirigidas a modelos.

3.4.2 Técnicas Dirigidas a Modelos

As ferramentas e técnicas modernas, chamadas também de técnicas dirigidas a modelos, para apoiar atividades de Engenharia de Segurança (do inglês - *Safety Engineering*) mudaram, permitindo a integração dessas atividades ao processo de projeto do sistema e a síntese automática de artefatos como FTA e FMEA a partir de um modelo de projeto de sistema com informações de comportamento de falha (DELANGE; FEILER, 2014); (PAPADOPOULOS; MCDERMID, 1999a); (JOSHI *et al.*, 2005). Essas ferramentas permitem a reutilização de informações de segurança em um processo de projeto iterativo. Essas técnicas e ferramentas de análise de segurança podem ser classificadas como composicionais ou extensões de técnicas de verificação formal (LISAGOR; MCDERMID; PUMFREY, 2006).

3.4.2.1 Técnicas Composicionais

As técnicas composicionais ou baseadas em componentes, fornecem linguagens formais e semi-formais para apoiar a especificação, composição e análise do comportamento de falha do sistema com base em informações de segurança sobre os componentes do sistema. Com base em um idioma específico, um modelo de lógica de falha do sistema é criado caracterizando o comportamento de falha de cada componente individual do sistema a partir da análise de

como falhas internas e falhas de entrada podem causar uma falhas nas portas de saída de um componente. O modelo de lógica de falha (do inglês - *Failure Logic Analysis*) conecta os modos de falha de saída de um componente com os modos de falha de entrada de outro componente, criando fluxos de modos de falha.

Uma vez que essas técnicas composicionais são baseadas na abordagem de modelagem de erros *Failure Logic Analysis*, elas permitem que analistas de segurança especifiquem o comportamento de falha do sistema de forma incremental à medida que o projeto progride da arquitetura do sistema para um nível detalhado. Esse processo facilita a adoção de uma abordagem de "divisão e conquista"para avaliação de segurança, onde a avaliação de um sistema complexo é dividida em tarefas mais gerenciáveis de caracterização do comportamento de falha de componentes individuais. As técnicas composicionais permitem a reutilização de informações de segurança superando as limitações relacionadas à necessidade de revisar artefatos como FTAs e FMEAs quando o projeto do sistema é alterado. A modelagem de erros baseadas nas notações UML SysML (KÄSSMEYER; SCHULZE; SCHURIUS, 2015); (DAVID; IDASIAK; KRATZ, 2010), AADL Error Annex (DELANGE; FEILER, 2014) e *Hierarchically Performed Hazard Origin and Propagation Studies* (HiP-HOPS) (PAPADOPOULOS; MCDERMID, 1999a) são exemplos de técnicas de análise de segurança composicional. Em geral essas técnicas não são totalmente automatizadas, pois exige a entrada de dados de maneira manual.

3.4.2.1.1 Failure Propagation and Transformation Notation (FPTN) e Extensão

Failure Propagation and Transformation Notation (FPTN) e sua extensão *Failure Propagation and Transformation Calculus* (FPTC) foram as primeiras técnicas composicionais de apoio à análise de segurança (do inglês – *safety analysis*).

FPTN fornece uma notação para representar o comportamento de falha (do inglês – *failure behavior*) do sistema baseada no conceito de módulo de componente (do inglês – *component module*). FPTN permite que engenheiros de segurança descrevam a propagação de falhas de componentes por toda a arquitetura do sistema. Nesta notação, os componentes são conectados por meio de links entre portas de saída de um componente a portas de entrada de outro componente. Isso permite a propagação de falhas de um componente a outro.

A notação FPTC foi criada para superar as limitações de FPTN relacionadas à construção de um modelo de erros separado do design do sistema. FPTC conecta o modelo de erros ao modelo arquitetural do sistema, permitindo a identificação e o gerenciamento de dependências entre os modelos. A notação FPTC define os seguintes tipos de falha (do inglês – *failure types*): *omission*, *commission*, *value*, *early* e *late*, que podem ocorrer em componentes do sistema. Uma falha do tipo *omission* (omissão) significa que um dado de entrada ou de saída de um componente não foi recebido. Uma falha do tipo *commission* (comissão) indica que um dado foi recebido quando não era esperado receber esse dado. Falhas do tipo *value* (valor) podem ocorrer quando o valor de dado recebido está abaixo ou acima dos limites estabelecidos. Finalmente, um dado pode

chegar a um componente antes (*early*) ou depois (*late*) do esperado. Esses tipos de falhas são usados para caracterizar como componentes, as suas entradas e as suas saídas podem falhar. Essas falhas são especificadas como anotações nos componentes do modelo do sistema. AADL Error Annex(DELANGE; FEILER, 2014) e HiP-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies)(PAPADOPOULOS; MCDERMID, 1999a) são técnicas composicionais modernas construídas com base nos conceitos de FPTN e FPTC.

3.4.2.2 Técnicas Baseadas em Simulação de Modelos

A segunda categoria é a baseada em extensões das técnicas existentes, também denominadas abordagens de injeção de falhas. Essas técnicas envolvem o uso de linguagens de modelagem formal e técnicas de verificação de modelo para deduzir automaticamente condições anormais (modos de falha) que podem levar o sistema a um estado inseguro (LISAGOR; MCDERMID; PUMFREY, 2006). Essas técnicas simulam o funcionamento normal do projeto e, em seguida, injetam falhas para determinar seus efeitos sobre a segurança do sistema (PAPADOPOULOS; MCDERMID, 1999a).

As abordagens baseadas em injeção de falhas não requerem a especificação de lógica de falha de componente para descrever a propagação de falhas, pois todas as informações necessárias são extraídas do modelo, com base em uma biblioteca de tipos de falha (do inglês - *failure types*) pré-definidos. AltaRica 3.0 (BATTEUX *et al.*, 2013), *Safety Analysis Modeling Language* (SAML) (GUDEMANN; ORTMEIER, 2010), e *Formal Safety Analysis Platform*-NuSMV (BOZZANO; VILLAFIORITA, 2007) são exemplos de técnicas baseadas em injeção de falhas.

Técnicas de engenharia de segurança baseadas em componentes e em extensões de abordagens de verificação formal podem ser aplicadas de forma complementar e em diferentes estágio do ciclo de desenvolvimento do sistema. Dessa forma, as técnicas baseadas em componentes são aplicáveis a partir da concepção do sistema, e extensões de técnicas de verificação formal como AltaRica 3.0, que são aplicáveis em fases posteriores do projeto.

Nesta trabalho, é considerado o uso de HiP-HOPS para apoiar a análise de propriedade de *safety* e *security* de sistemas autônomos do domínio de VANTs. Sendo assim, a Seção 3.5 é dedicada a um detalhamento desta ferramenta.

3.5 HiP-HOPS

A ferramenta HiP-HOPS (PAPADOPOULOS; MCDERMID, 1999a) é um *software* de análise de dependência e otimização de sistemas críticos complexos desenvolvida pelo *Dependable Systems Research Group* da *University of Hull* que trabalha como um *plug-in* para Simulink®.

HiP-HOPS é também considerado um método e apoio ferramental que foi construído com base nas técnicas FPTN(WALLACE, 2005) e FPTC e que permite a modelagem de erros (do inglês – *failure modeling*) em modelos de sistema desenvolvidos em MATLAB/Simulink®[1], SimulationX[2] e ferramentas de modelagem UML baseadas na plataforma Eclipse como EAST-ADL(CUENOT *et al.*, 2010).

HiP-HOPS requer como entrada um conjunto de dados de falha de componentes (*component failure data*), que descreve como falhas nas portas de saída dos componentes são causadas pela ocorrência de combinações entre falhas internas e nas portas de entrada dos componentes. HiP-HOPS sintetiza essas informações em árvores de falha (do inglês – *fault trees*) e FMEA, refletindo a propagação das falhas pela arquitetura do sistema. HiP-HOPS também permite a especificação de dados probabilísticos, como taxas de falhas e de reparo (do inglês - *failure and repair rates*), no modelo de erros do sistema. HiP-HOPS armazena dados probabilísticos e as informações sobre falhas de componentes em bibliotecas, possibilitando o reúso dessas informações em outros modelos.

A lógica de falhas relaciona-se a componentes de *design* de forma intuitiva e clara. A relação entre o projeto do sistema e a modularidade inerente aos modelos de lógica de falha permitem a reutilização da lógica de falha de componentes sempre que os componentes são reutilizados. Portanto, quando o *design* do sistema é alterado, a identificação do efeito dessas mudanças no modelo de lógica de falha é simplificada e, o esforço para a sua adaptação é proporcional ao tamanho da mudança.

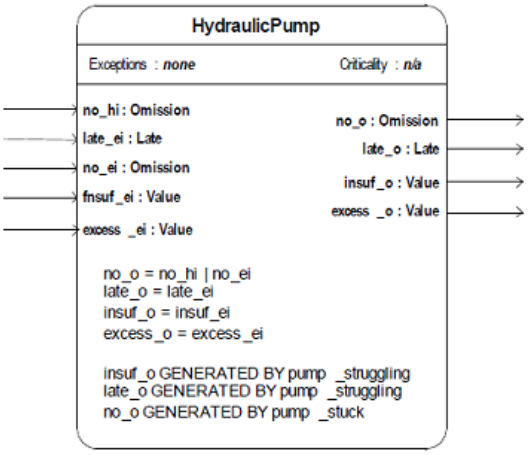
São anotados no modelo todos os *Basic Events* , *Output Deviations* e *Hazards*, estes serão detalhados respectivamente nas sessões 3.5.1, 3.5.2 e 3.5.3. O HiP-HOPS utiliza as anotações feitas no modelo como entrada em um algoritmo *top-down* para análise de dependência que gera automaticamente a FTA (sessão 3.4.1) e a Análise de Modo de Falha e Análise de Efeito ou *Failure Models and Effects Analyses* (FMEAs).

Na Figura 10 é ilustrado o modelo de erros que caracteriza as potenciais falhas que podem ocorrer no componente de uma bomba hidráulica (*hydraulic pump*) utilizando as técnicas HiP-HOPS e FPTN. O componente "bomba"(*pump*) exerce um nível constante de pressão hidráulica em sua porta de saída (o). A bomba possui duas portas de entrada: uma hidráulica (“hi”) e outra elétrica (“ei”). A porta “hi” alimenta o componente bomba com um fluído hidráulico não pressurizado e “ei” alimenta o motor com energia elétrica. Em relação ao modelo de erros do componente bomba, uma omissão de fluído hidráulico ou de energia elétrica causa uma omissão de pressão. Neste exemplo, assume-se que a bomba não é sensível a quaisquer outras falhas em “hi” e “ei”. Ao mesmo tempo, assume-se que todas as falhas de corrente elétrica podem resultar em falhas similares nas saídas do componente bomba. Por exemplo, um baixo fluxo de energia elétrica pode causar um baixo fluxo de pressão. Além disso, pode ocorrer uma falha interna

[1] <http://www.mathworks.com/products/simulink/
[2] <http://simulationx.com

Figura 10 – Modelos de erros do componente bomba hidráulica em HiP-HOPS (esquerda) e FPTN (direita).

Internal Failure	Failure Rate	Repair Rate
pump_stuck	5e-5	1e-4
pump_struggling	1.5e-5	1e-4
Output Deviation	**Description**	**Causes**
Omission-o	No pressure (i.e. omission) on pump's output. Caused by omission of fluid on the hydraulic input ("hi") or omission on the electrical input ("ei") or by the internal failure of the pump ("stuck").	Omission-hi OR Omission-ei OR pump_stuck
Value-o	Insufficient pressure on pump's output (value FM). Caused by insufficient voltage on electrical input or by the internal failure of the pump ("struggling").	Value-ei OR pump_struggling
Value-o	Excessive pressure on pump's output (value FM). Caused by excessive voltage on electrical input.	Value-ei
Late-o	Late reaching required pressure on start Caused by late provision of electrical current or by internal failure of a pump ("struggling").	Late-ei OR pump_struggling



Fonte: Obtido de (LISAGOR; MCDERMID; PUMFREY, 2006).

no componente bomba, tornando-o não funcional (*broken*) ou menos responsivo a estímulos elétricos (*struggling*), devido a um aumento de impedância nas bobinas. As falhas internas do componente bomba podem causar a omissão de pressão (*pressure*) ou pressão insuficiente. Um componente bomba que está no estado *struggling* durante a inicialização pode fornecer maior pressão hidráulica do que requerido.

Técnicas de análise de segurança composicionais como HiP-HOPS são eficazes na coleta de informações de segurança e confiabilidade de uma arquitetura de sistema. Segurança e confiabilidade são fatores importantes a serem considerados para tomar decisões de projeto. No entanto, a avaliação manual de múltiplas escolhas no espaço de projeto em contraste a objetivos de otimização múltiplos e frequentemente contraditórias, como segurança, confiabilidade e custo, é demorado e pode restringir os candidatos a serem investigados. Por outro lado, as técnicas de otimização de projeto suportam a análise de arquiteturas com maior grau de variabilidade, com centenas e milhares de soluções candidatas no espaço de projeto, para obter soluções quase ótimas que atendam a objetivos múltiplos e contraditórios.

O HiP-HOPS fornece extensões, estruturadas em algoritmos genéticos baseados em *Tabu Search*(COIT; SMITH, 1996), NSGA-II(DEB *et al.*, 2002) e *Penalty-Based*(KULTUREL-KONAK; SMITH; COIT, 2003), para suportar a alocação automática e ótima de requisitos de integridade de segurança. Esses algoritmos dão suporte aos analistas de segurança no processo de tomada de decisão no início do ciclo de vida do desenvolvimento buscando alcançar uma solução de projeto que atenda aos requisitos de segurança e minimize o custo.

Os algoritmos de otimização do HiP-HOPS também auxiliam os projetistas a cumprir os padrões de segurança, automatizando o processo de decomposição de requisitos de integridade de

segurança alocados ao nível do sistema em todos os modos de falhas contribuintes na arquitetura do sistema e, avaliando possíveis alocações, determinando melhores soluções de alocação com base em heurística de custo (PARKER; WALKER; PAPADOPOULOS, 2013).

Os algoritmos de otimização da *Tabu Search* do HiP-HOPS suportam a decomposição automática dos requisitos de integridade de segurança nos sistemas aeroespaciais (SOROKOS *et al.*, 2015), explorando o espaço da solução de forma mais eficiente, fornecendo soluções de alocação quase otimizadas dentro de intervalos de tempo aceitáveis em comparação com algoritmos baseados em penalidades (PARKER; WALKER; PAPADOPOULOS, 2013) e NSGA-II (PAPADOPOULOS *et al.*, 2011) (AZEVEDO *et al.*, 2013). A alocação ótima de requisitos de integridade de segurança contribui para a obtenção de certificação, dando suporte à decomposição de requisitos de integridade de segurança de acordo com os padrões de segurança da SAE ARP 4754A (ARP4754A, 2010).

3.5.1 Basic Events

Os *Basic Events* são definidos nas entradas de cada componente e informam quais os possíveis tipos de falhas aquela entrada pode sofrer.

As falhas são divididas em quatro grupos sendo:

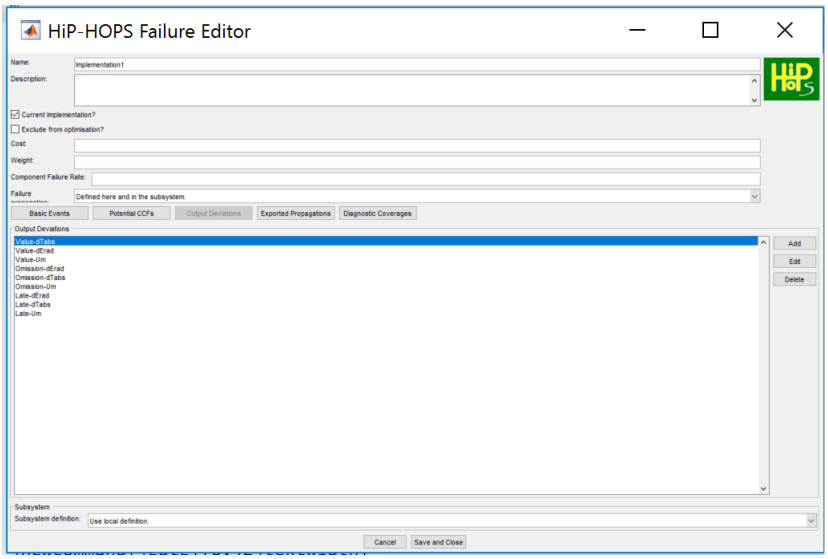
- *Late* (atrasado): O valor da entrada chega atrasado, por exemplo, *LFailure1*, que indica atraso na entrada número 1 do componente;
- *Early* (adiantado): O valor da entrada chega adiantado, por exemplo, *EFailure1*, que indica adiantamento na entrada número 1 do componente;
- *Value* (falha): O valor da entrada chega incorreto, por exemplo, *VFailure1*, que indica que o valor está incorreto na entrada número 1 do componente;
- *Omission* (omissão): O valor da entrada não chega, por exemplo, *OFailure1*, que indica que não chegou nenhum valor para a entrada número 1 do componente;
- *Comission* (comissão): O valor da entrada chega em um momento que não deveria, por exemplo, *CFailure1*, que indica que chegou algum valor inesperado para a entrada número 1 do componente.

3.5.2 Output Deviations

Os *output deviations* fazem parte do modelo de falha arquitetônica, que por sua vez, representa hipóteses sobre desvios de saída e entrada de componentes (*input* e *output deviations*) que podem levar a falhas de nível de sistema nos mais diversos cenários (AERONAUTICS; (NASA), 2002).

O nome de um *output deviations* é composto por duas partes: o tipo de falha (descritos na sessão 3.5.1) e a porta, do componente, onde a falha ocorre. As partes são separadas por um hífen (ilustrado na Figura 11). Por exemplo 'Value-dErad' (HIP-HOPS MANUAL, 2013).

Figura 11 – Exemplo *Output Deviation*.



Fonte: O autor.

Cada *Output Deviations* é composto por um ou mais *Basic Events* ligados por expressões lógicas como OR, AND, NOT, XOR.

3.5.3 Hazards

Os *Hazards* são falhas causadas por um ou mais erro/falha em algum componente do sistema. Uma vez que os erros/falhas são definidos nos *output deviations*, consequentemente os *hazars* são definidos por um ou mais *Output Deviations* ligados por expressões lógicas como OR, AND, NOT, XOR entre outras como ilustrado na Figura 12. Esse tipo de notação garante que todas as partes da falha sejam definidas de forma correta e que não fique um *Hazard* definido de forma incompleta ou inconsistente.

3.5.4 Safety Case

Um *Safety Case* pode ser definido como um conjunto lógico e hierárquico de documentos que descreve o risco com base nos perigos apresentados pelo sistema, incluindo potenciais falhas e acidentes, e as medidas razoavelmente praticáveis que devem ser implementadas para mitigar possíveis danos. O documento leva em conta outros sistemas similares já conhecidos e orienta quais processos devem ser realizados para que os riscos sejam controlados.(REGULATION, 2014). Esse documento tem como foco apresentar as seguintes informações sobre determinado projeto:

- Identificar os perigos e riscos;

- Descrever como os riscos são controlados; e
- Descrever o sistema de gerenciamento de segurança no local para garantir que os controles sejam efetivamente e consistentemente aplicados.

Uma outra definição de *Safety Case* é dada por (HIGH; KELLY; MCDERMID, 1997) como sendo "Um argumento claro, abrangente e defensável, apoiado por um conjunto de evidências, o que demonstra que um sistema é aceitável de forma segura para operar em um contexto particular".

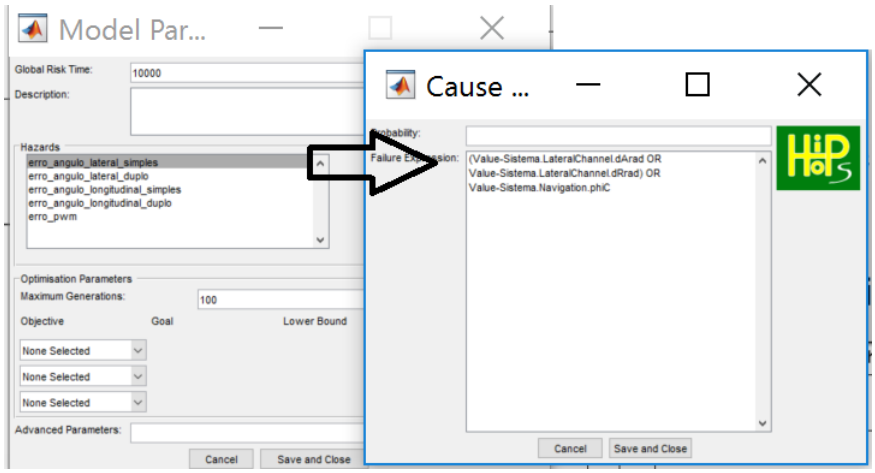
3.5.4.1 Ciclo de vida de um Safety Case

É recomendado que os *Safety Cases* sejam implementados de forma incremental, e em paralelo com o sistema (KELLY, 2004), assim o *Safety Case* conterá mais informações sobre o desenvolvimento, arquitetura e sobre o próprio ciclo de vida do sistema (WILKINSON; KELLY, 1998b).

Durante seu ciclo de vida é possível distinguir algumas fases (MOD, 2007) sendo:

- ***Safety Case* preliminar**: justifica como serão abordados os requisitos de *safety* e integridade definidos no plano de segurança. Ele apresenta uma *Safety Case* preliminar mostrando as principais metas de segurança;
- ***Safety Case* provisório ou interino**: fornece de forma provisória evidências que justificam as especificações de *safety* e integridade do projeto;
- ***Safety Case* operacional**: fornece de forma completa as evidências que justificam as especificações de *safety* e integridade do projeto.

Figura 12 – Exemplo de *Hazard*.



Fonte: O autor.

As técnicas de análise de segurança estão evoluindo para lidar com a complexidade dos sistemas críticos de segurança modernos, por exemplo por meio do uso de abordagens funcionais para avaliação de segurança. Essas técnicas tem se mostrado promissoras e tem permitido que *Safety Cases* sejam obtidos e avaliações mais efetivas sejam realizadas. Entretanto, a segurança computacional, pode afetar requisitos de segurança física. Além disso, as semelhanças entre *safety* e *security* são frequentemente obscurecidas pelo uso de diferentes conceitos e terminologias.

Na verdade, há uma variação considerável na terminologia dentro e entre as diferentes comunidades. Assim, para alcançar uma compreensão compartilhada dos conceitos-chave dentro de cada domínio, é necessário estabelecer uma língua franca ou, ao menos mecanismos que permitam averiguar seus impactos e suas interações, permitindo assim sua integração, foco deste trabalho.

3.6 *Segurança Computacional - Security*

A vida cotidiana está cada vez mais permeada de computadores embarcados, estejam eles em carros, telefones celulares, equipamentos de vídeo para leitores de MP3, máquinas de lavar louça e até mesmo em torradeiras.

A segurança computacional tem sido objeto de pesquisas intensivas no contexto de sistemas de computação e comunicação de propósito geral. No entanto, a segurança é muitas vezes mal interpretada por projetistas de sistemas embarcados como sendo a adição de recursos, como algoritmos criptográficos específicos e protocolos de segurança, ao sistema.

3.6.1 Conceituação

A segurança de computadores consiste em proteger informações pessoais ou confidenciais e/ou recursos computacionais de indivíduos ou organizações que poderiam deliberadamente destruir ou utilizar tais informações para fins maliciosos (STAPKO, 2008). Quando se fala em *security* pode-se ressaltar quatro principais atributos necessários e indispensáveis para que ela seja garantida (KUROSE; ROSS, 2006); (TANENBAUM; COMPUTADORES, 2003):

- **Confidencialidade:** Os dados/informações somente devem ser acessados por quem tenha direto de acessá-los;
- **Integridade:** Deve-se garantir que dados/informações não devem sofrer quaisquer tipo de alterações;
- **Disponibilidade:** Os dados/informações devem estar disponíveis sempre que necessário;
- **Autenticação:** deve-se garantir que quem esteja tentando acessar os dados/informações seja quem realmente diz ser.

Uma das perguntas mais frequentes é: *Qual a diferença quando se fala em segurança para dispositivos embarcados?*

As conexões da Internet expõem aplicativos para intrusões e ataques maliciosos. Infelizmente, as técnicas de segurança desenvolvidas para a informática empresarial e para *desktops* podem não satisfazer os requisitos de aplicativos embarcados, tais como (KOOPMAN, 2004):

- **Sensibilidade ao custo:** Os sistemas embarcados são muitas vezes altamente sensíveis aos custos - mesmo cinco centavos podem fazer uma grande diferença ao construir milhões de unidades por ano. Por esse motivo, a maioria das unidades de processamento fabricadas em todo o mundo, apesar de terem melhorado e muito sua capacidade de processamento, ainda apresentam uma quantidade de memória limitada, principalmente quando o assunto é segurança. Muitos processadores, por exemplo, não podem armazenar uma grande chave criptográfica. Isso pode tornar as melhores práticas do mundo empresarial muito caras para serem práticas em aplicativos embarcados. Cortar elementos relacionados à segurança para reduzir os custos de hardware pode dar ao investidor uma vantagem de mercado para produtos sensíveis ao preço, principalmente quando ainda não há uma medida quantitativa e qualitativa de segurança antes da implantação do sistema.

- **Interatividade:** Atualmente a maioria dos sistemas embarcados interagem com o mundo real (haja vista o advento da Internet das Coisas do inglês *Internet of Things* (IoT) (ATZORI; IERA; MORABITO, 2010). Uma violação de segurança, portanto, pode resultar em efeitos colaterais físicos, incluindo danos à propriedade, ferimentos pessoais e até mesmo a morte (ligação direta com a *safety*). A recuperação de transações financeiras pode reparar algumas violações de segurança da empresa, mas a reversão de um acidente de carro ou de um VANT não é possível. Ao contrário da computação empresarial orientada a transações, os sistemas embarcados geralmente executam cálculos periódicos para executar *loops* de controle com prazos estimados em tempo real. Sendo assim, quando um atraso de apenas uma fração de segundo pode causar uma perda de estabilidade de controle, de modo que os sistemas tornam-se vulneráveis aos ataques projetados para perturbar o tempo do sistema. Os sistemas embarcados geralmente não possuem um administrador de sistema real. Diversas outras questões podem ser ainda levantadas de modo que, não havendo um administrador, a vulnerabilidade do sistema torna-se muito maior quando comparado a um sistema empresarial.

- **Limitações de energia:** Sistemas embarcados muitas vezes têm restrições de energia significativas, e muitos deles são alimentados por bateria. Alguns sistemas embarcados podem receber uma carga de bateria nova diariamente, mas outros devem durar meses ou anos com uma única bateria. Ao tentar drenar a bateria, um invasor pode causar falhas do sistema, mesmo quando a invasão no sistema é impossível. Esta vulnerabilidade é crítica, por exemplo, em dispositivos com bateria que usam comunicação sem fio com energia

elétrica, uma vez que o consumo de energia é alto e pode ser ampliado quando se deseja efetuar um ataque.

- **Ambiente de desenvolvimento:** Grande parte dos sistemas são desenvolvidos por equipes reduzidas. No caso dos sistemas embarcados esse número é ainda maior, sendo que muitas vezes o desenvolvimento é executado por uma única pessoa. Uma vez que, nem mesmo para sistemas empresariais existe a presença de um especialista de segurança, certamente que para o desenvolvimento de sistemas embarcados essa realidade é ainda pior. No entanto, mesmo os programas aparentemente triviais podem precisar fornecer algum nível de segurança. Até que a prática de desenvolvimento padrão inclua análises rigorosas de segurança, os desenvolvedores podem ignorar as soluções já disponíveis.

A segurança para esses sistemas embarcados é uma questão ainda em aberto e pode revelar-se um problema de longo prazo mais difícil do que a segurança hoje para computação de *desktops* e corporativa.

Esses problemas em segurança nos sistemas embarcados não são novos. Na realidade, é uma nova dimensão que os projetistas devem considerar ao longo do processo de projeto, juntamente com outras métricas, como custo, desempenho e potência computacional. No entanto, à medida que mais sistemas embarcados são conectados à Internet, os potenciais danos de tais vulnerabilidades aumentam dramaticamente.

As tendências recentes no projeto de plataforma de sistemas embarcados críticos aumentam os riscos de que o uso indevido (acidental ou intencional) da informação do sistema possa ocorrer. Novas plataformas, e a heterogeneidade das mesmas, aumentaram a interconectividade de equipamentos tanto dentro do sistema como no ambiente em que eles se encontram.

Em geral, sistemas embarcados, mais especificamente os veículos aéreos não tripulados (detalhados na seção 2), foco deste trabalho, são constituídos de uma grande variedade de itens de software e hardware: altamente críticos que controlam a aeronave e de itens de baixa criticidade que informam e em alguns casos permitem levar entretenimento os passageiros (quando considerados veículos de transporte de pessoas e de grande porte). Consequentemente, esses veículos podem ser alvo de problemas de segurança computacional que poderiam ter um impacto na segurança da aeronave.

Sendo assim, os desafios exclusivos dos sistemas embarcados exigem novas abordagens de segurança que atuem todos os aspectos do projeto desse tipo de sistema, desde a arquitetura até a implementação. O processamento de segurança, que se refere aos cálculos que devem ser realizados em um sistema com a finalidade de segurança computacional, pode facilmente sobrecarregar as capacidades computacionais dos processadores nos sistemas embarcados. Esse desafio, que pode ser chamado de "intervalo de processamento de segurança", é agravado por aumentos nas quantidades de dados manipulados e nas taxas de dados que precisam ser alcançadas.

Um outro problema tão sério quanto é a "falha da bateria"em sistemas embarcados alimentados por estas, que é causada pela disparidade entre requisitos de energia que aumentam rapidamente para uma operação segura e melhorias lentas no que tange a tecnologia da bateria. O desafio final é o "hiato de garantia", que se relaciona com a diferença entre as medidas de segurança funcionais (por exemplo, serviços de segurança, protocolos e seus algoritmos criptográficos constituintes) e implementações seguras reais.

Os atacantes podem lançar ataques únicos aos sistemas de controle (ou seja, ataques que não são possíveis em sistemas de Tecnologia de Informação (TI) tradicionais).

A pesquisa em segurança tem se concentrado tradicionalmente na proteção da informação. Os pesquisadores não consideraram como os ataques afetam os algoritmos de estimativa e controle - e, finalmente, como os ataques afetam o mundo físico. A principal distinção de sistemas de embarcados em relação a outros sistemas empresariais é a interação do sistema com o mundo físico. Em geral, a segurança computacional desenvolveu tecnologias maduras e princípios de projetos (autenticação, controle de acesso, integridade da mensagem, separação de privilégios, entre outros) que podem ajudar a prevenir e reagir aos ataques contra sistemas embarcados.

Uma direção importante para futuras pesquisas é identificar os riscos e as consequências de um ataque bem-sucedido, bem como identificar e categorizar novos tipos de ataques e suas consequências. Mas a separação dos ambientes de teste e produção nem sempre é possível. Alguns objetos de teste são difíceis de serem isolados dos ambientes em que estão inseridos ou os objetivos do teste proíbem o isolamento efetivo do objeto de teste.

3.7 *Security Hazop*

De forma análoga ao HAZOP tradicional já explicado em 3.3.1, o *security* HAZOP visa a manutenção dos principais pilares da *security* : Confidencialidade, Integridade, Disponibilidade e Autenticação já detalhados em 3.6.1. Porém, enquanto as *guide-words* do HAZOP comum tendem a focar em falhas de sistema as do *Security* HAZOP enfatizam as vulnerabilidades do sistema nas interações com humanos e a informações incorretas passadas para o mesmo.

A construção se baseia em *guide-words* e atributos. As *guide-words* são palavras chave pré-determinadas e especificas para a área em questão. Para determinar quais seriam essas palavras chaves foram levadas em conta as utilizadas não somente por (WINTHER; JOHNSEN; GRAN, 2001) mas também as encontradas na dissertação de (MARCONATO,) que, por sua vez especifica, com base em (HUSSEIN; ZULKERNINE, 2006), não apenas as causas e efeitos de alguns ataques mas também os autores e as vitimas dos mesmo. Para cada item analisado deve-se elencar algumas questões sobre como ele pode ocorrer e quais seriam suas prováveis consequências.

O processo de criação do *Security* HAZOP se inicia com a seleção de cenários, onde são elencados os principais cenários e são selecionados os mais críticos. Concluído este processo são selecionadas as *guide-words* relacionadas aos ataques presentes nos cenários selecionados, e então é feita pelo especialista a anotação, tanto de *component fault modelling* quanto de *system fault modelling*, do modelo. Feito este processo para todos os cenários selecionados, o modelo anotado é utilizado como entrada para o processo de síntese de FTA e FMEA.

Diferente do processo de *safety* convencional, no caso do *security*, *hazards* podem ser causados por entidades externas ao sistema, como um atacante mau intencionado ou um comando recebido de forma incorreta.

3.7.1 Identificação dos ataques

Esta etapa se assemelha muito à etapa conhecida como pré-ataque, onde o especialista reúne o máximo de informações possível sobre o sistema alvo. Isso incluiria a determinação das superfícies de ataque do sistema alvo e a documentação de todas as informações disponíveis. Nesta etapa são elencados os possíveis tipos de ataques e suas prováveis consequências, ou seja, quais funcionalidades do sistema seriam afetadas por ele.

3.7.2 Avaliação de risco

Com as informações coletadas durante a processo de identificação de ataques é possível identificar quais são os ataques ao qual o sistema estaria mais vulnerável e também as consequências desses ataques à segurança do sistema. Isso permite a realização de uma seleção de prioridade, que define a qual ou quais ataques o sistema deve ser testado para que suas consequências sejam confirmadas e mitigadas de forma a não influenciar na usabilidade e funções do sistema.

3.7.3 Alocação de Requisitos de Security

Com base na avaliação de risco, são realizados os testes para que sejam confirmadas ou não as consequências dos ataques.

É possível inferir qual/quais partes do sistemas carecem de melhorias em sua *security*, e em quais destas o impacto da correção de determinada falha seria mais crucial para o sistema (falhas que provoquem danos colaterais menores devem ter menor prioridade quando comparadas a falhas que causem dados catastróficos para o sistema, mesmo que sejam mais fácies de serem exploradas).

3.8 Trabalhos Relacionados

Foi realizada uma revisão sistemática buscando encontrar trabalhos que tenham características muito similares à proposta de mestrado apresentada nesta dissertação. Entretanto, como nenhum trabalho exatamente igual a esta proposta foi encontrado, esta revisão sistemática apresenta artigos publicados que têm contribuições importantes para cada um dos tópicos de pesquisa deste mestrado. Esta seção apresenta uma síntese dos resultados obtidos a partir dos trabalhos selecionados na última fase da revisão sistemática cujo protocolo é apresentado no apêndice A.

Durante a leitura dos artigos foi observada a possibilidade de geração automática dos *safety cases* como em (DENNEY; PAI; HABLI, 2012), onde é detalhado o processo de geração de *safety cases* por meio de verificação formal, ou seja, utilizando palavras chaves e requisitos previamente definidos na documentação do projeto, seguindo um algorítimo que possui três principais fases: primeiro a *hazard identification and risk analysis* (*preliminary hazard analysis* (PHA)) onde é feita uma análise prévia do sistema e da documentação; durante a segunda fase cada subcomponente é analisado de forma mais individual e; durante a terceira fase de *subsystem hazard analysis* (SSHA) e, por fim ocorre a a geração da FMEA e FTA.

Muitas vezes quanto são descritos os argumentos de *safety* que são passados para as FTAs e utilizados no *safety cases* os mesmo são descritos de forma quantitativa, o que permite mensurar seus efeitos dentro de um sistema. Entretanto, (DENNEY; PAI; HABLI, 2011), afirma a necessidade de se ter uma descrição não só quantitativa mas também qualitativa para que se possa compreender o verdadeiro impacto deste argumento dentro do sistema, ou seja, não basta apenas enumerá-los mas faz-se necessário também entender e mensurar a diferença de impacto que cada um deles provoca na *safety* do projeto (como um todo).

Uma das possíveis soluções para mitigação dos ricos e diminuição da ocorrência de *hazards* é a criação de estruturas de redundância dentro do projeto, tanto em software quanto em hardware, mas seu uso acarreta em custos de espaço físico, maior custo monetário e peso. Tais quesitos são de suma importância para VANTs a fim de criar redundância de forma otimizada, reduzindo seu impacto no projeto ao mesmo tempo que gera um ganho de segurança (NYSTROM *et al.*, 2006). É possível observar que esse trabalho também demonstra como o uso de FTAs para escolha de quais componentes acarretará maior ganho de *safety* caso sejam criadas redundâncias.

Por outro lado, (CHANDHRASEKARAN; CHOI, 2010) utilizou de simulação em hardware (*Hardware In the Loop* - HIL) avaliação de *safety* e *security* em VANT, para isto, durante a simulação injetou falhas, valores aleatórios, em sensores e atuadores com diferentes taxas e avaliou como o software e hardware se comportaram além do ruído causado pelas falhas em cada um dos sensores.

3.9 Considerações Finais

Neste capítulo foram apresentados os conceitos relacionados a segurança, tanto *security* quanto *safety*. Também foram apresentadas ferramentas e métodos de apoio à analise de *safety* e *security*. O HiP-HOPS foi apresentado de forma detalhada, já que é a ferramenta utilizada neste estudo. Uma vez apresentados os conceitos e a necessidade de integração entre *safety* e *security*, o próximo capítulo apresenta a UNiVErSE, uma abordagem de apoio à análise de *safety* e *security* em sistemas críticas.

CAPÍTULO

4

UNIVERSE - UMA ABORDAGEM DE APOIO À ANÁLISE DE *SAFETY*/*SECURITY* EM SISTEMAS CRÍTICOS

4.1 Considerações iniciais

Com base na revisão da literatura e na identificação do estado da arte, este capítulo apresenta os fatores que contribuíram para a concepção da proposta de análise de *safety* integrada a análise de *security* em veículos aéreos não tripulados tendo como estudo de caso o piloto automático de um VANT.

4.2 UNiVErSE

O *UNmanned Vehicle safEty Security improvEr* (UNiVErSE) é uma abordagem que tem por objetivo o aumento da segurança física de veículos não tripulados que toma por base algumas etapas de verificação para posteriormente auxiliar na redução de riscos ou vulnerabilidades computacionais.

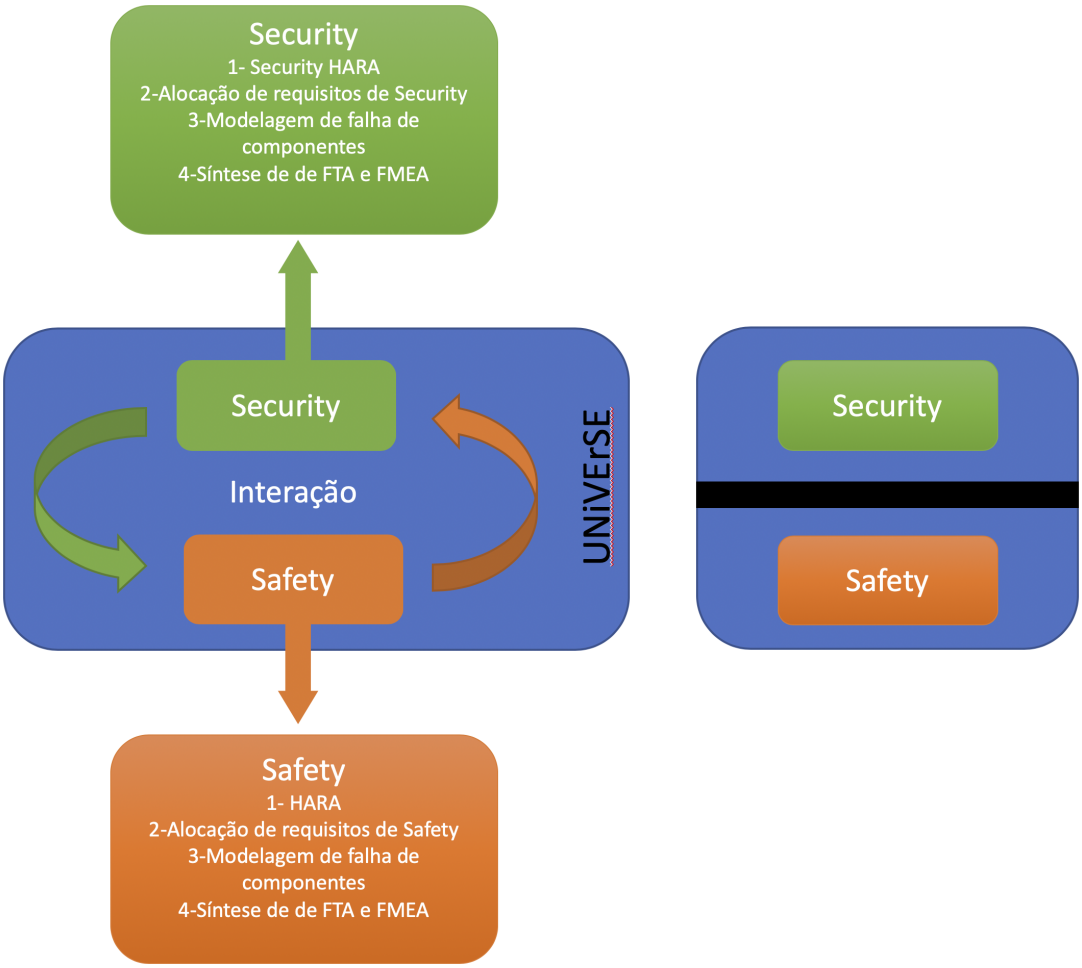
Para que os objetivos sejam atingidos a UNiVErSE analisa de forma integrada os quesitos de *safety* e *security* fazendo com que lacunas antes existentes entre os dois universos sejam investigadas, analisadas e então protegidas.

O UNiVErSE prove um apoio na verificação e validação de segurança (*safety security*) para veículos não tripulados, permitindo a exploração de métodos, técnicas e ferramentas que lidem com questões de segurança, respeitando requisitos pré-estabelecidos. As atividades de investigação são desenvolvidas visando a conciliação entre os requisitos de *safety* e *security*. Na Figura 13 é ilustrada a diferença entre como o UNiVErSE analisa *safety*/*security* (a esquerda) e como *safety* e *security* são normalmente analisadas (a direita).

Os passos da abordagem, ilustrado na Figura 14, deixam claro que a analise deve ser feita de maneira paralela, começando pela seleção dos cenários e terminando com a síntese da FTA e FMEA, que abrangem tando *safety* quanto *security*.

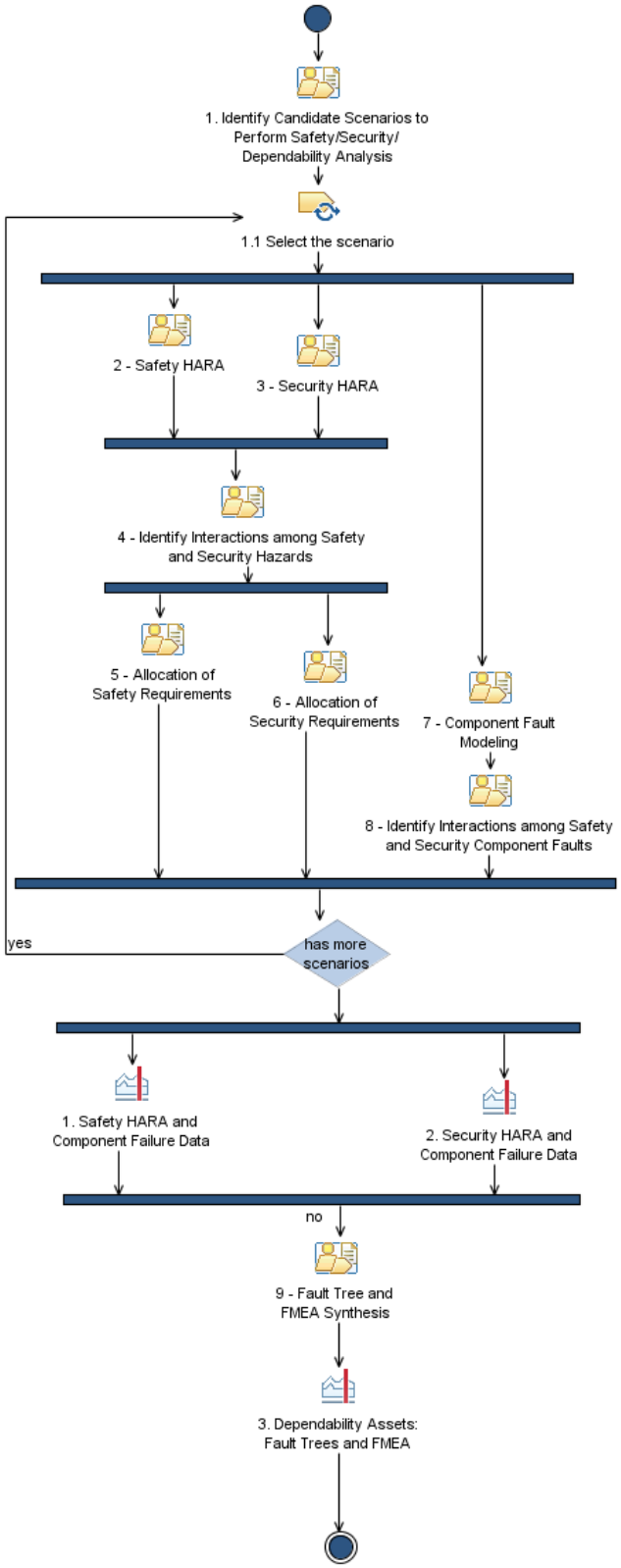
Como resultado, a UNiVErSE permite a obtenção de uma análise não só da *security* mas também da *safety* do sistema em análise (como primeiro caso de estudo será feita uma análise no piloto automático de um VANT). Na Figura

Figura 13 – Universo de atuação da plataforma para que os objetivos sejam atingidos - UNiVErSE.



A Figura 13 (a esquerda) expõe de maneira clara a interação entre *safety* e *security* da abordagem. É possível verificar que existe a interação e isto é feito pois uma vez que as falhas de *security* podem ser exploradas por *hackers*, violando requisitos de *safety*, a implementação de um mecanismo de *security* pode impactar nas características de *safety*, a existência de uma plataforma que integre esses dois requisitos considerados essenciais para sistemas embarcados críticos, o UNiVErSE corrobora no processo de diminuição dessa lacuna ainda existente na literatura.

Figura 14 – Diagrama SPEN 2.0 UNiVErSE.



4.3 Identificar cenários para análise de *safety*/*security*

Para identificar os cenários que serão analisados deve-se recolher o maior número de informações a cerca dos usos para o qual o sistema crítico será alocado e qual atividade o sistema empregará, por exemplo: saber que determinado sistema crítico atuará em um ambiente de temperatura e umidade controlada tornam os *hazards* ligados a estes quesitos menos prováveis e, provavelmente menos severos. Conhecer as regulamentações a qual o ambiente em que o sistema será utilizado também é imprescindível pois, dependendo delas, existem casos que obrigatoriamente devem ser cobertos independentes da aplicação do seu sistema.

Com estas informações, aliadas ao conhecimento de um especialista na área, são selecionados o/os principal/is cenário/os de acordo com a severidade e probabilidade de ocorrência do *hazard*

4.4 *Safety/Security Analysis*

Nesta seção, é apresentada a *DEPendable-Autonomous System Engineering* (DEPendable-ASE), uma abordagem baseada em modelos para suportar a análise de confiabilidade no desenvolvimento seguro de sistemas críticos autônomos. O DEPendable-ASE foi desenvolvido com base no DEPendable-SPLE (OLIVEIRA A. L., 2018), uma abordagem orientada que fornece suporte a análise de confiabilidade em engenharia de linha de produto de software crítica para *safety*.

De modo análogo ao DEPendable-SPLE, a abordagem DEPendable-ASE enfatiza a importância de considerar o impacto do contexto nas propriedades de confiabilidade, já que mudanças no contexto podem contribuir para elevar diferentes riscos, com diferentes causas e, diferentes mecanismos de mitigação devem ser definido para eliminar ou minimizar os efeitos de risco na segurança geral. Além disso, diferentes falhas nos componentes podem ser levantadas e contribuem para a ocorrência de perigos em diferentes contextos.

Na Figura 14 é ilustrada a estrutura da abordagem proposta em um diagrama de atividade *SPEM*2.0.

No primeiro passo são definidos os cenários com base nas interações entre as escolhas de projeto e os contextos em que sistema critico atua. Em seguida é selecionado o escopo de atuação do sistema para que sejam selecionados os senários mais adequados, com base no HARA destes cenários, são executadas a alocação dos requesitos de *safety* e a modelagem de falha de componente. Os passos da *Dependability analysis* devem ser executados de forma iterativa e incremental em todos os cenários selecionados.

Feita esta análise os dados são utilizados como entrada para um software que utiliza técnicas de analise composicional (PAPADOPOULOS *et al.*, 2011) (DELANGE; FEILER, 2014) (MAZZINI S., 2016), neste caso HiP-HOPs (PAPADOPOULOS *et al.*, 2011) (ja explorado na

seção 3.5) em conjunto com MATLAB/Simulink para gerar de forma automática a FTA e a FMEA para cada *hazard* identificado. As FTAs fornecem suporte para a identificação de propagação de falhas a nível de sistema, já as FMEAS, nos permitem identificar como determinado componente pode, ou não, contribuir de forma direta ou indireta em cada um dos *hazards*.

Neste estudo as definições de HARA foram incrementadas de forma a também identificarem termos e ambientes relacionados a *security* (SHARA), visto que a abordagem proposta tem como objetivo uma análise que englobe ambos os quesitos de forma integrada.

Os passos ilustrados na Figura 14 são detalhados a seguir:

1. Identificação e seleção de cenários: São identificados quais são os possíveis cenários que o sistema será exposto. São selecionados os cenários ao qual o sistema estará mais exposto, ou, ao qual o especialista julga mais relevante/necessário.

2. HARA: o HARA pode ser dividir em três grandes etapas: Identificação de risco, Avaliação de risco e Alocação de requisitos de *safety*. Durante a execução destas etapas, são obtidas informações para a anotação de um modelo que será utilizado para elaboração da tabela HARA.

3. SHARA: São seguidas as mesmas etapas do HARA, porém são os dados colhidos dos sistema com relação a *security* e , por consequência, as anotações do modelo remetem às falhas de *security*.

4. Identificação de interação de falhas *safety/security*: Assim como os dois passos anteriores, são executados sobre o mesmo modelo. Neste caso, são acumuladas anotações de *safety* e de *security* o que possibilita analisar a interação entre ambas.

5. Alocação de requisitos de *safety*: Com os *hazards* identificados e avaliados, com o uso de alguns padrões, estes são classificados para cada um dos cenários analisados.

6. Alocação de requisitos de *Security*: Com os *hazards* identificados e avaliados, com o uso de alguns padrões, estes são classificados para cada um dos cenários analisados, de modo análogo aos de *safety*.

7. Modelagem de falha de componente: Cumpridas as etapas anteriores os dados são utilizados para popular uma tabela onde cada falha é associada a um componente especifico e de como ele é afetado pelos demais componentes. Tais falhas são demonstradas a partir de expressões logicas.

8. Identificação de interação de falhas de componentes(*safety*/*security*): Com base nos passos anteriores o especialista consegue identificar em quais locais existem interações entre as falhas de componentes (*safety*/*security*).

9. Síntese de FTA e FMEA: Feitas todas as anotações sobre o modelo tanto a FTA quanto a FMEA são geradas automaticamente (como descrito na seção 3.5).

4.4.1 Hazard Analysis and Risk Assessment (HARA) e SecurityHARA (SHARA)

O *Hazard Analysis and Risk Assessment* ou HARA é executado no processo de Identificação de Perigo (seção 3.3.1), avaliação de risco (seção 3.3.2) e alocação de requisitos de *safety* (seção 3.3.3). HARA é considerado uma atividade preliminar no processo de avaliação de um sistema (ARP4754A, 2010).

O HARA utiliza como entrada os cenários previamente selecionados com base nos contextos de uso do sistema analisado (tarefa 1.1 da Figura 14) e o modelo de arquitetura do sistema. Durante o HARA as combinações entre falhas de componente e falhas de sistema são chamadas de *hazards*. Tais *hazards* são especificados utilizando expressões lógicas e potenciais falhas na arquitetura dos componentes do sistema. Estas falhas comumente estão associadas a termos tais como : *Value*, *Omission*, *Comission*, *Late* e *Early* no caso de *safety* e para *security*: ATK_DOS , GPSSPOOF e logical_Problems, tais *guide-words* foram criadas com base em 3.7 porém incluem *guide-words* relacionadas à *security*. Assim interações entre a arquitetura interna dos componentes e o contexto de uso do sistema são analisados para identificar como possíveis erros podem surgir. Ao final deste processo é possível obter uma lista de *hazards* associados a um determinado contexto.

O resultado destes processos são comumente exposto em forma de tabelas, um exemplo disto é a Tabela 2, nela são expostos 2 *hazards* distintos (*Hazard*-1 e *Hazard*-2) que ocorrem em diferentes contextos, assim como suas respectivas causas expostas em forma de expressões e também sua classificação quanto à severidade e ao DAL. De modo análogo os resultados são apresentados na Tabela 3 em que

Tabela 2 – Tabela HARA genérica.

Usage Ctx	Hazard	Hazard Causes	Severity	DAL
Ctx-1	Hazard-1	Value-C1 OR Late-C2	Major	B
Ctx-2	Hazard-2	(Late-C2 OR Value-C1) AND Comission-C1	Hzdous	C

Fonte: O autor.

4.4.2 Allocation of System Safety/Security Requirements

Utilizando os resultados obtidos com a HARA, junto aos requisitos de *safety* são alocados os requisitos necessários para eliminar ou mitigar os *hazards* identificados. Os Safety Integrity Levels (SILs) são definidos como as medidas de redução de risco associadas a uma falha ou

Tabela 3 – Tabela de SHARA genérica

Exepression	thread	Causes	Consequense
Troca de valor da Variável	Alteração do valor da Variável	execução de *exploit*	O valor da variável ficará errado
Obtenção de informações	Obtenção de informações do banco de dados	*SQL injection*	Quebra da confidencialidade

Fonte: O autor.

risco de um componente especifico (MOD, 2007). Os requisitos de *safety* são divididos em três grupos :

- *System safety requirements:* São alocados com objetivo de eliminar ou mitigar os *hazards* em nível de sistema.
- *system functions/derived safety requirements* (MOD, 2007): funções do sistema, destinadas a eliminar ou minimizar os efeitos de falhas na segurança do sistema (ISO, 2011).
- *Safety integrity requirement specifies the reliability*: ações planejadas e sistemáticas necessárias para fornecer confiança e evidências adequadas de que um produto ou processo atende aos requisitos de segurança.

Tais SILs são alocados para cada *hazard* identificado de acordo com a classificação de risco obtida durante a fase de avaliação de risco (já detalhada na seção 3.3.2). Além disso, os SILs alocados para as falhas de sistema podem ser decompostos e esta decomposição pode ser realizada com suporte automatizado de algoritmos genéticos e meta-heurísticos de decomposição SIL (PAPADOPOULOS *et al.*, 2011). Por fim, a alocação de requisitos de segurança funcional visa identificar as funções do sistema que podem eliminar ou minimizar o impacto de um risco ou um *hazard* em nível do componente. Um exemplo de requisito de segurança funcional é a utilização de redundância. Toda vez que um ou maus requisitos de segurança funcional são implantados no sistema é necessária refazer a análise para que seja avaliado o impacto da modificação sobre o sistema.

Como resultado deste processo é obtido um conjunto de requisitos de segurança funcional específico do contexto e SILs a serem alocados para mitigar os efeitos de risco. Já quando se trata de *security* como não existem padrões pré-estabelecidos para a classificação de vulnerabilidades (de *security*) em sistema embarcados, fica a cargo do especialista classificar as vulnerabilidades e os ataques selecionados para o cenário em questão.

4.4.3 Component Fault Modeling

Nesta etapa, o comportamento de falha associado a cada componente de arquitetura é especificado, são realizadas as anotações do que pode falhar em um determinado componente e

como ele é afetado por falhas em outros locais do sistema. Esta informação é denominada falha logica de componente. Comumente nas técnicas de análise de decomposição o comportamento das falhas do componente são especificados com anotações feitas no modelo do sistema. Essas anotações incluem expressões que mostram como os *output deviations* (já descritos na seção 3.5.2) podem ser impactados por *basic events* (já descritos na seção 3.5.1), e são denominadas expressões de falha. Os *output deviations* podem incluir a *Omission* inesperada de uma saída, *Comission* não intencional de uma saída, *Value* de saída incorretos, ou *Late / Early* saída que está sendo enviada muito cedo ou atrasada (PAPADOPOULOS *et al.*, 2011).

Tabela 4 – Tabela de Modelagem de falha de componente genérica

Component	Expression	Output
Comp1	(VFailure1 OR Value-Val1) and Late-Val2	Out2
Comp2	Value-Val1 And Value-Val2	Out6

Fonte: O autor.

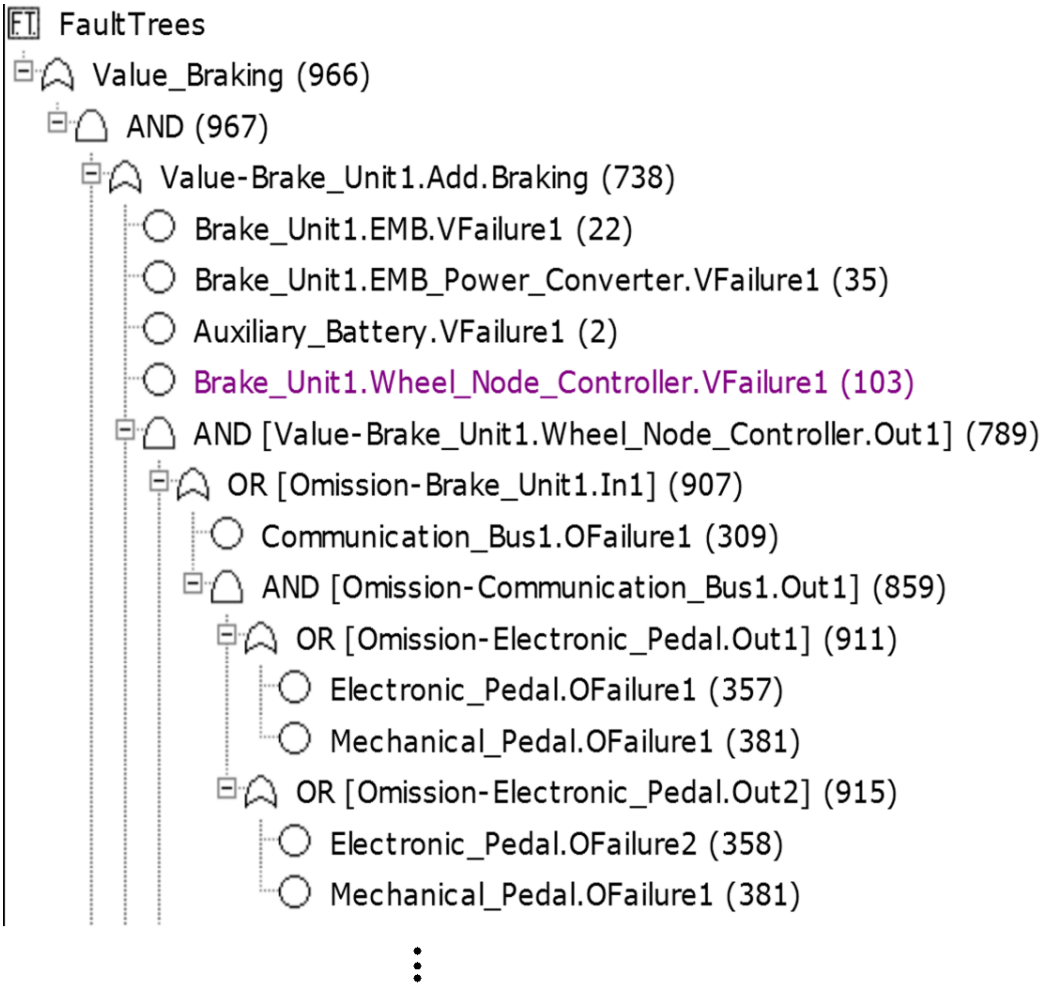
Esta etapa se inicia com a seleção de um componente, seguido pela identificação de potenciais *output deviations* que podem contribuir para a ocorrência de cada *hazard* identificado e com a inferência (com base em seus conhecimentos prévios da taxonomia do ataque) de qual será o efeito provocado pelo ataque e qual/quais componentes serão afetados, por exemplo: um ataque que visa sobrecarregar a rede de determinado equipamento, terá os componentes da placa de rede afetados. Em seguida, as causas potenciais de cada um dos *output deviations* identificado deve ser especificadas pela análise de combinações potenciais entre falhas internas de componentes e *basic events* que podem levar à ocorrência de cada *output deviation*. A análise continua enquanto existem componentes a serem analisados. Ao final do processo, um conjunto de dados de falha de componente mostrando como os componentes podem contribuir para a ocorrência de riscos do sistema em cada cenário de destino é entregue. Assim, o modelo do sistema é aprimorado com informações de confiabilidade.

4.5 Fault Trees and FMEA Synthesis

O modelo do sistema anotado com informações sobre as possíveis falhas de sistema e suas causas, feitas nos passos anteriores da abordagem, são os artefatos de entrada para a síntese de árvores de falhas (FTAs) e de FMEA (do inglês - *Failure Modes and Effects Analysis*), que são artefatos exigidos por padrões de segurança tais como o ARP 4754A, para que seja obtida a certificação de *safety* de um sistema crítico.

Nesta fase, o modelo do sistema com as informações sobre falhas de sistema e suas causas é utilizado por técnicas composicionais de análise de segurança, por exemplo, HiP-HOPS, para gerar automaticamente as FTAs e a tabela FMEA (que mostra o impacto de falhas de componentes na ocorrência de falhas de sistema). Exemplos de FTA e FMEAs para um sistema automotivo de frenagem híbrida geradas automaticamente pela ferramenta HiP-HOPS

Figura 15 – Exemplo de FTA



Fonte: O autor.

Figura 16 – Exemplo de FMEA

Component: Auxiliary_Battery			
Failure Mode	System Effect	Severity	Single Point of Failure
VFailure1 (2)	Value_Braking	0	true
Component: Electronic_Pedal			
Failure Mode	System Effect	Severity	Single Point of Failure
VFailure1 (359)	Value_Braking	0	true
Component: Mechanical_Pedal			
Failure Mode	System Effect	Severity	Single Point of Failure
OFailure1 (381)	No_Braking_Front	0	true
VFailure1 (382)	Value_Braking	0	true
Component: Powertrain_Battery			
Failure Mode	System Effect	Severity	Single Point of Failure
VFailure1 (388)	Value_Braking	0	true

Fonte: O autor.

são ilustrados respectivamente nas Figuras 15 e 16. As FTAs e FMEA podem variar dependendo das escolhas de projeto e do contexto no qual o sistema está inserido.

Tanto as FTAs quanto a tabela FMEA são utilizadas para demonstrar que o sistema atende

aos requisitos de segurança e proteção. Análise e avaliação de risco do sistema, informações de falha do componente, árvores de falhas e FMEA podem ser artefatos de entrada para se produzir um *Assurance case* (KELLY, 2004), que pode ser gerado automaticamente a partir das informações contidas nesses artefatos com o suporte de técnicas baseadas em modelos (HAWKINS R., 2015).

4.6 Considerações Finais

Neste capítulo foi apresentada a UNiVErSE, uma abordagem dirigida a modelos de apoio à análise das propriedades de *safety* (proteção) e de *security* (segurança) de sistemas de veículos aéreos autônomos. UNiVErSE apoia engenheiros de segurança a realizar atividades de engenharia de **dependabilidade** (segurança, proteção, disponibilidade e confiabilidade) considerando o impacto de variações em **decisões de projeto** e no **contexto de uso** do sistema. A abordagem UNiVErSE pode contribuir para aumentar a precisão na análise das propriedades de segurança e proteção de sistemas de veículos aéreos autônomos e outras classes de sistemas críticos. Esta abordagem fornece diretrizes sistemáticas para realizar atividades de engenharia de segurança com o apoio de técnicas dirigidas a modelos.

O uso da abordagem UNiVErSE contribui para reduzir a complexidade e o esforço na realização de atividades de engenharia de segurança para a certificação de sistemas de veículos aéreos autônomos. A abordagem UNiVErsE também pode contribuir para reduzir o número de erros durante a realização de atividades de engenharia de segurança de sistemas críticos, por meio da síntese automática de artefatos como árvores de falhas e tabelas FMEA, fornecido por técnicas dirigidas a modelos, requeridos para a certificação sistemas de veículos aéreos autônomos.

Embora a ferramenta HiP-HOPS, integrada ao MATLAB/Simulink, ter sido utilizada nesta dissertação para apoiar as atividades de análise das propriedades de *safety* e de *security* do piloto automático SLUGS, a abordagem UNiVErSE é aplicável independente de técnica e da ferramenta de análise de segurança baseada em componentes. No Capítulo 5 é descrita a aplicação da abordagem **UNiVErSE** para apoiar a análise das propriedades de *safety* e de *security* do sistema de piloto automático do veículo aéreo autônomo SLUGS utilizando a ferramenta HiP-HOPS.

CAPÍTULO

5

ESTUDO DE CASO - *SLUGS AUTOPILOT*

5.1 Considerações Iniciais

Neste capítulo é apresentada a aplicação da abordagem UNiVErSE para apoiar a análise das propriedades de *safety* e de *security* do piloto automático SLUGs com o apoio da ferramenta HiP-HOPS.

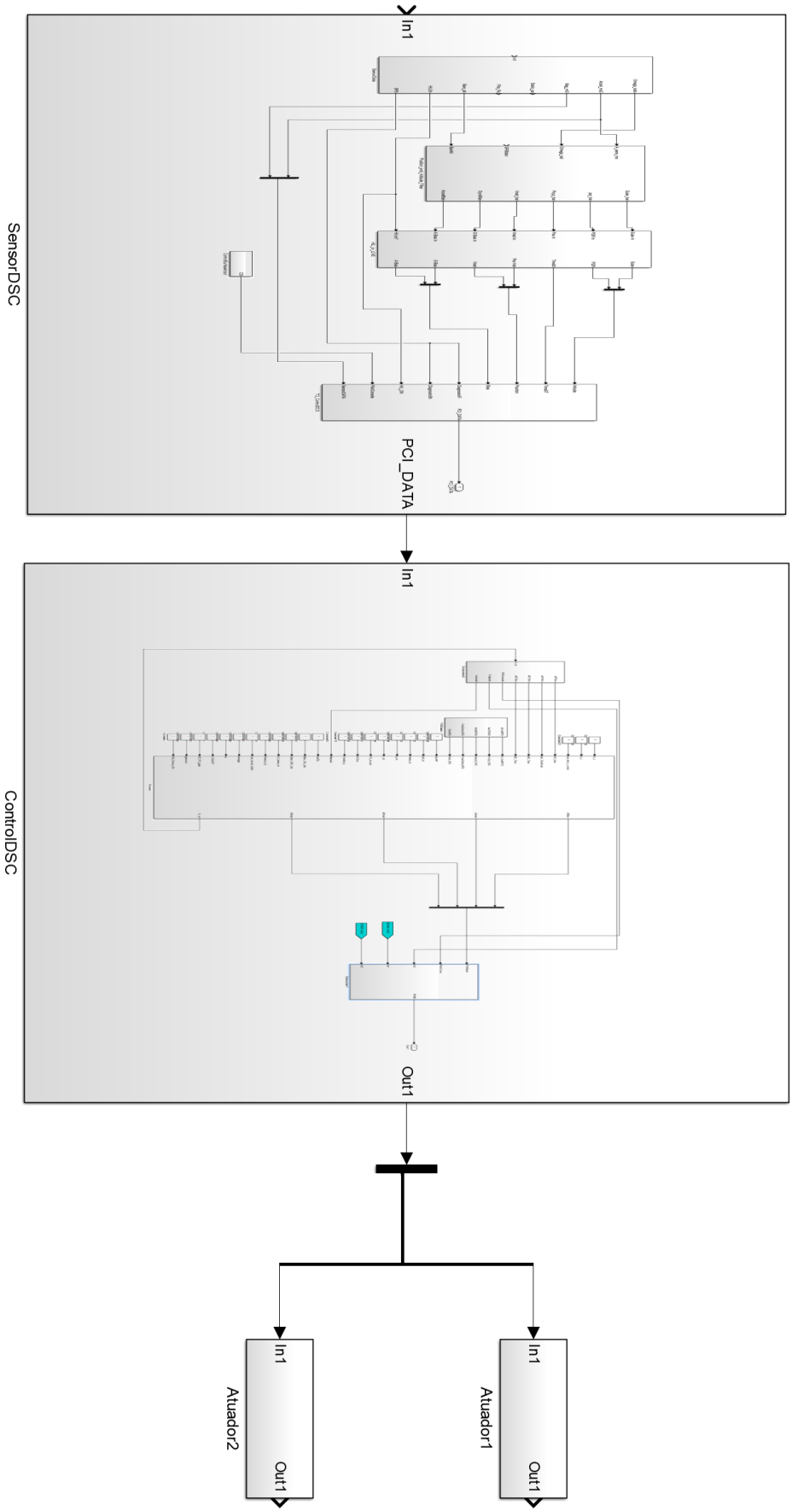
5.2 Projeto do Piloto Automático SLUGS

O sistema de piloto automático de aeronave não tripulada SLUGS foi escolhido para a validação da abordagem UNiVErSE. Na Figura 17 é ilustrado o principal diagrama de blocos do SLUGS que foi implementado em *Simulink* (OMG, 2015). O subsistema *CONTROL DSC* (*Digital Signal Controllers*) é composto por cinco componentes:

- *Navigation*: responsável por ler os dados dos *waypoints* captados pelos sensores de navegação e produzir as coordenadas de voo que são enviadas para os subsistemas *Longitudinal Channel* e *Lateral Channel*.
- *Longitudinal Channel* é responsável por controlar os aspectos longitudinais do avião.
- *Lateral Channel* é responsável pelos aspectos laterais.
- *ComputePSIDot* é responsável por processar algumas informações que são repassadas para *L1OutputFeedbackController*.

Os passos definidos ma abordagem UNiVErSE (4) foram seguidos durante a análise das propriedades de *safety* e de *security* do piloto automático e são descritos a seguir.

Figura 17 – Diagrama de blocos do SLUGS.



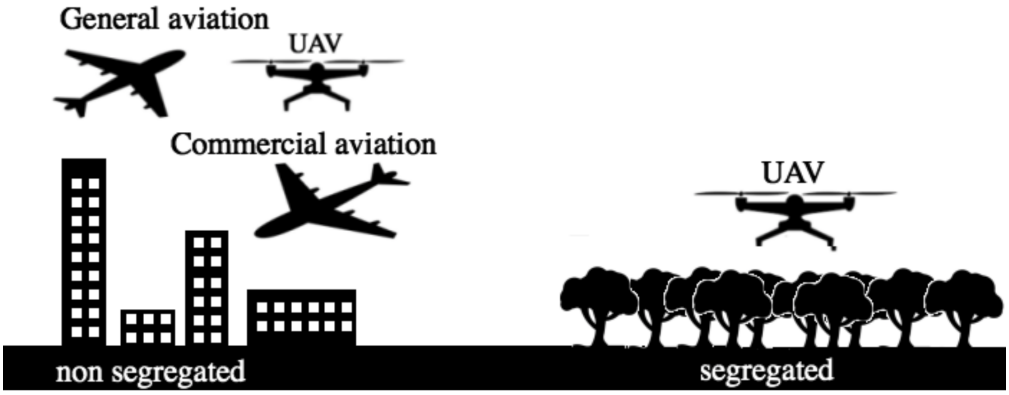
Fonte: O autor

5.2.1 Identificando cenários candidatos para análise de safety/security do SLUGS

Diferentes cenários dão suporte a diferentes análises de *safety*/*security* e necessitam de diferentes conhecimentos sobre o ambiente e a aplicação do VANT nesse domínio. A partir de configurações e contextos de utilização do SLUGS os seguintes cenários foram considerados durante as atividades de análise de perigos (do inglês - *hazard analysis*) e modelagem de erros de componentes: o SLUGS operando em espaço aéreo segregado/controlado (SLUGS/*Controlled*) e operando em espaço aéreo não-segregado/não controlado (SLUGS/*Uncontrolled*).

O espaço aéreo não controlado/não segregado refere-se à operação de uma aeronave não tripulada fora de um espaço aéreo controlado ou segregado, que abrange as dimensões do espaço aéreo alocadas para uso exclusivo de usuários específicos (ICAO, 2011). O Espaço aéreo controlado/segregado se refere ao não existe a separação de espaço para diferentes tipos de aeronaves (tripuladas ou não), ou seja, aeronaves de diversos tipos, tamanhos e configurações voam sem distinção. Já no espaço aéreo segregado existe a separação, como por exemplo em um aeroporto onde existe uma série de normas que limitam os tipos de aeronave que podem e quando podem sobrevoar aquele ambiente. A descrição desses ambientes de operação são ilustrados na Figura 18.

Figura 18 – Tipos de espaço aéreo.



Fonte: O autor.

Os cenários descritos anteriormente oferecem suporte para que analistas de segurança extraiam conhecimento de domínio necessário para a análise de confiabilidade do sistema. As escolhas de *design* relevantes e os contextos de uso dos *stakeholders* foram usados como critérios para avaliar os cenários candidatos para a execução da análise de perigos e outras atividades de engenharia de *safety* e de *security* do piloto automático SLUGS.

5.2.2 Análise de Perigos e Avaliação de Riscos de Safety

Durante o processo de análise de perigos e avaliação de riscos de *safety* e de *security* foi considerado o padrão de segurança SAE ARP 4754A (EUROCAE, 2010). A partir da

arquitetura apresentada no diagrama de bloco do SLUGS foram identificados os seguintes *hazards* ilustrados na Tabela 5: valor incorreto do ângulo longitudinal (*value longitudinal angle*), valor incorreto do canal lateral (*value lateral channel*), and valor incorreto de sinal PWM (*Pulse-Wide Modulation*). Cada *hazard* identificado foi classificado em um determinado nível DAL (*Development Assurance Level*) de acordo com a sua *severidade* e *probabilidade de ocorrência*.

Na Tabela 5 são apresentados três *hazards* e suas respectivas classificações de risco com o SLUGS operando em Espaço Aéreo Segregado (EAS) e Não Segregado (EANS) (veja subseção 5.2.1). Valores incorretos nas portas de saída *DErad* e *dTabs* do componente *Longitudinal Channel* levam a ocorrência do *hazard value longitudinal angle* (Tabela 5).

O *hazard value lateral angle* em decorrência de valores incorretos das portas de saída *dArad* e *dRad* do componente *Lateral Channel*. Finalmente, um *valor incorreto* na porta de saída *pwmSign* do componente *PWMGen ou* se o valor da porta de saída pwmSign chegar após (*late*) o esperado pode levar a ocorrência de um *value PWM signals*.

Tabela 5 – Resultados da análise de perigos e avaliação de riscos de *safety* do piloto automático SLUGS.

Contexto de Uso	Hazard	Causas do Hazard	Severidade	DAL
EAS	Value longitudinal angle	Value-LongCh.dErad AND Value-LongCh.dTabs	*Hazardous*	B
EAS	Value lateral channel	Value-LateralCh.dArad AND Value-LateralCh.dRad	*Hazardous*	B
EAS	Value PWM signals	Value-PWMGen.pwmSign OR Late-PWMGen.pwmSign	*Hazardous*	B
EANS	**Value longitudinal angle**	**Value-LongCh.dErad AND Value-LongCh.dTabs**	***Major***	**C**
EANS	Value lateral channel	Value-LateralCh.dArad AND Value-LateralCh.dRad	*Hazardous*	B
EANS	**Value PWM signals**	**Value-PWMGen.pwmSign OR Late-PWMGen.pwmSign**	***Major***	**C**

Fonte: O autor.

Como ilustrado na Tabela 5, a severidade e a probabilidade de ocorrência de um *hazard* e, consequentemente o seu DAL, podem variar de acordo com o contexto. Isto ocorre, por exemplo em *Value double longitudinal angle*, que pode ocorrer devido a um erro no valor dos ângulos (longitudinais) que são repassados para a aeronave realizar determinada manobra. Considerando o espaço aéreo segregado, a severidade deste *hazard* é mais grave, ou seja *Hazardous*. Já em um espaço aéreo não segregado, a severidade deste *hazard* é *Major*. Tal variação também ocorre com o *hazard Value PWM Signals*, que indica que um valor incorreto foi transmitido pelo componente *PWM* aos atuadores da aeronave.

5.2.3 Análise de Perigos e Avaliação de Riscos de Security

Como não há na literatura uma padronização de *guide-words* para a realização da análise de perigos de *security* como existe para a análise de *safety*, nesta dissertação foram utilizados termos e guide-words de security definidos na abordagem *Security HAzard and OPerability*

studies (Security HAZOP) (WINTHER; JOHNSEN; GRAN, 2001) descrita na Seção 3.7. Da mesma forma que a análise das propriedades de *safety* do sistema, esta atividade foi realizada com base no modelo arquitetural do piloto automático SLUGS especificado em MATLAB/Simulink e considerando a operação do SLUGS em espaço aéreo segregado e não segregado.

Com base no conhecimento de domínio de especialistas e do autor desta dissertação, foram feitas suposições sobre as potenciais ameaças de *security* (segurança) as quais o piloto automático SLUGS estaria suscetível nos contextos de operação considerados nesta análise. Na Tabela 6 é ilustrada a lista de ameaças (*hazards*) de *security* relacionadas ao SLUGS operando no espaço aéreo segregado e o espaço aéreo não segregado, bem como as causas de tais ameaças, a severidade e o nível DAL relacionado a cada *hazard* de *security*. Cada linha da tabela contém a descrição do objetivo e a descrição (nome do *hazard* de *security*) do ataque, as suas causas e consequências. A severidade do ataque e o nível DAL de cada ataque para a segurança do sistema foram omitidos para facilitar o entendimento do leitor.

Tabela 6 – Resultados da análise de perigos de *security* para o piloto automático SLUGS

Objetivo do ataque	**Hazard**	**Causa (s)**	**Consequência(s)**
EAS - Realizar consulta ao Banco de dados de forma não autorizada	Injeção de uma consulta	Injetar uma consulta SQL em um campo que apresenta vulnerabilidades OR alterar uma consulta legitima	Quebra do sigilo dos dados do banco AND vazamento de informações.
EANS - Interrupção da comunicação do VANT com a GS	Sobrecarga da rede de comunicação	Inúmeras requisições a GS OR req. ao VANT tornando a comunicação inviável	Perda de comunicação entre GS e VANT
EAS - Alteração da coordenada de GPS do VANT	Sobreposição de sinal GPS	Replicação de sinal falso de GPS AND sobreposição do verdadeiro sinal de GPS	Perda da real posição do VANT podendo levar a perda do VANT

Fonte: O autor.

O primeiro e o terceiro perigo de *security* podem ocorrer durante a operação do piloto automático SLUGS em espaço aéreo segregado e o segundo perigo, ou seja, a *interrupção da comunicação do VANT com a Ground Station (estação de solo)*, pode ocorrer apenas durante a operação do SLUGS em espaço aéreo não segregado.

O mesmo padrão para o HARA foi utilizado para a descrição dos *hazards*, a Tabela 7 ilustra parte da definição de dois *security hazard*. É possível observar que em sua definição existem tanto requisitos diretamente ligados a *security* quanto elementos de *safety*.

Tabela 7 – Tabela de definição de SHARA

Hazard	Hazard-Causes
Ataque de GPS_SPOOF	(GPS_Spoof-SensorDSC.SensorData.SensorSuite.READ_SENSOR.GPS_SPOOF OR GPS_Spoof-SensorDSC.SensorData.SensorSuite.READ_SENSOR.VFailure1) AND GPSSensor.GPS_SPPOF ....
Ataque de DoS/DDoS	(GSSignal.CFailure1 AND GSSignal.ATKDOS) OR (GPSSensor.CFailure1 OR GPSSensor.LFailure1 AND SensorDSC.SensorData.SensorSuite.READ_SENSOR.LFailure1) ...

Fonte; O autor.

5.2.4 Allocation of Safety/Security Requirements

A variação inerente à classificação dos riscos relacionados a cada perigo identificado e pelos modos de falha de componentes em diferentes contextos de uso (veja Tabela 5 coluna "*severity*"), pode ser propagada para a alocação de requisitos funcionais de segurança (do inglês - *functional safety requirements*) e DALs. Isso implica que medidas de mitigação destinadas a eliminar ou minimizar os efeitos de um perigo na segurança do sistema podem variar de acordo com o contexto de uso do sistema. Assim, diferentes escolhas de projeto foram sugeridas para mitigar efeitos de risco e falhas de componentes de acordo com o contexto. Por exemplo, uma arquitetura de sistema de controle SLUGS redundante pode ser adotada para mitigar os efeitos de risco no contexto de uso de espaço aéreo segregado/controlado.

A variação na avaliação de risco do SLUGS impacta diretamente na alocação de SILs, neste caso, os Development Assurance Levels (DALs) para mitigar os efeitos de cada perigo identificado ou falhas de componentes na segurança do sistema. O perigo *value double longitudinal angle* tem uma severidade catastrófica com probabilidade de ocorrência de 10^{e-9} por hora de operação em um contexto de espaço aéreo controlado. Dessa forma, **DAL A** foi alocado para mitigar os efeitos da ocorrência de tal perigo. Por outro lado, **DAL B** é suficiente para mitigar os efeitos deste perigo quando o SLUGS é direcionado apenas para operar em um espaço aéreo não controlado. É importante ressaltar que a variação nos DALs atribuídos a riscos relacionado a *hazards* ou falhas de componentes pode afetar o processo de desenvolvimento do sistema.

Padrões de segurança orientadas a processo, por exemplo, DO-178C e SAE ARP 4754A, definem um conjunto de objetivos de segurança que devem ser abordados de acordo com o nível DAL que varia do menos crítico **DAL E** ao mais crítico **DAL A**.

Os objetivos de segurança definem um conjunto de atividades a serem executadas e artefatos a serem produzidos. Atender a níveis DALs mais críticos exige os mais rigorosos objetivos de segurança, atividades de engenharia de sistemas e artefatos de software, aumentando os custos de desenvolvimento. Alocar DALs menos rigorosos a componentes menos críticos e DALs rigorosos apenas a um subconjunto de componentes altamente críticos pode contribuir para reduzir os custos de desenvolvimento do sistema.

5.2.5 Safety/Security Component Fault Modelling

Durante a fase de análise de falha de componente, 60 falhas foram anotadas em 17 elementos do SLUGS. Na Tabela 8 é ilustrada uma expressão de falha associada a um *ouput deviation* no componente *LongitudinalChannel*.

No exemplo da Tabela 8 um *ouput deviation* do tipo *value* na porta de saída *dErad* pode ocorrer devido a uma falha interna no componente ou a um *basic event* do tipo *value* nas portas de entrada do componente *LongitudinalChannel*.

Tabela 8 – SLUGS *Component Fault Modelling*.

Component	Output Deviation	Failure Exp.
LongitudinalChannel	Value-dErad	VFailure1 OR (Value-uc OR Value-manual OR Value-dynp...)
	Value-dTabs	VFailure1 OR (Value-uc OR Value-manual OR Value-dynp...)

Fonte: O autor.

5.2.6 Síntese de Árvores de Falhas e FMEA de Safety e de Security

Nesta etapa, o modelo implementado em *Matlab/Simulink* com todas as anotações foram usados como entrada para o HiP-HOPS (PAPADOPOULOS *et al.*, 2011) para sintetizar automaticamente tanto as FTAs de cada perigo de *safety* e de *security* para cada *hazard* quanto a Tabela FMEA que relaciona as falhas de componentes com *hazards*. Na Figura 19 é ilustrado um trecho da FTA gerada para o *hazard Value double longitudinal angle* ilustrado na Tabela 8.

A ocorrência de *output deviations* nos componentes *LongitudinalChannel.dErad* e *LongitudinalChannel.dTabs* são as principais responsáveis para que ocorra o *hazard*

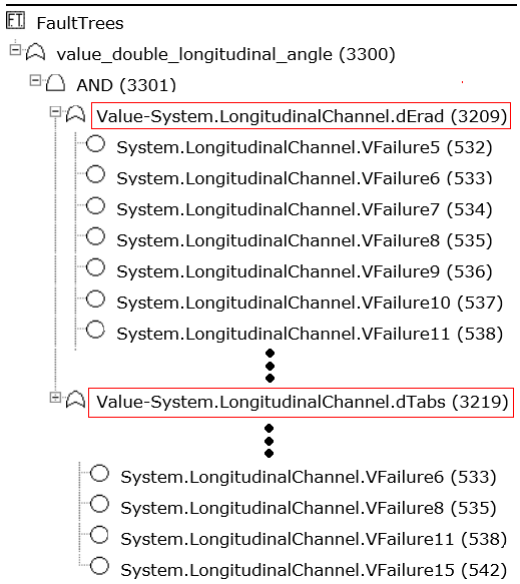
5.2.7 Análise de FTAs e geração de hipóteses

Com base na análise das FTAs notou-se que tanto a FTA de erro longitudinal duplo e a FTA de erro lateral duplo possuem entradas que estão em ambos os lados do **E** lógico. Na Figura 20 é possível verificar que a ocorrências das seguintes falhas 6"*h_m*", 8 "*theta_m*"e 11 "*Manual*" e, ou a ocorrência das falhas "*R_m*" e 2 "*Manual*") podem levar a ocorrência dos dois *hazards* representados pelos *top events* de cada FTA da Figura 20.

5.3 Discussões: Relacionamento entre *Safety/Security Guide-words*

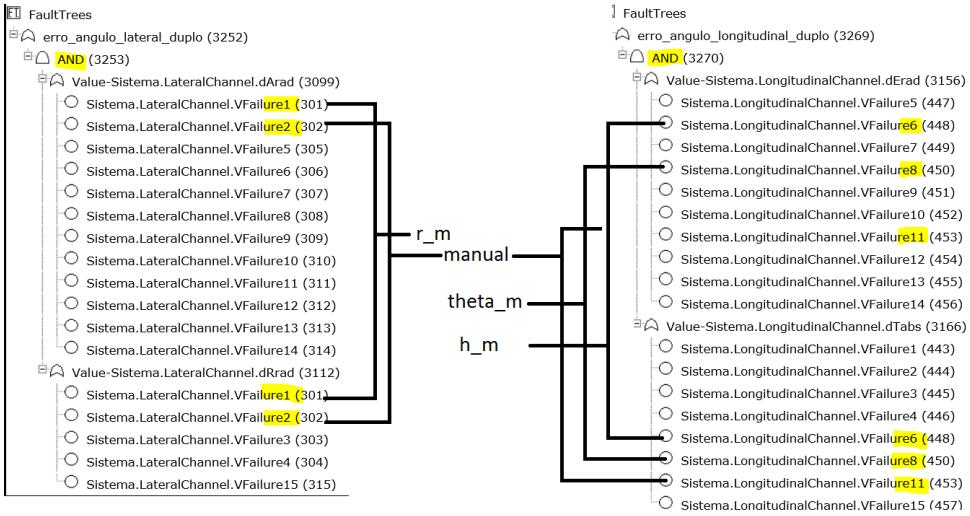
Durante este estudo foram identificadas correlações entre as falhas de *security* com as falhas do hardware (apresentadas na Tabela 9), e em alguns ataques que ferem apenas a

Figura 19 – Trecho da FTA do *hazard Value double longitudinal angle*.



Fonte: O autor.

Figura 20 – Comparação entre FTAs.



Fonte: O autor.

confidencialidade (como no ataque de *Man in The Middle - MITM*) é possível que não seja gerado nenhum efeito colateral no hardware ligado diretamente ao ataque. Isto ocorre porque este tipo de ataque pode não visar afetar diretamente o funcionamento do alvo mas apenas capturar as informações que transitam entre a *GroundStation* e o VANT.

5.3.1 Interação entre Falhas de Safety/Security em Nível de Sistema

Feitas as análises de ambos os pontos (*safety* e *security*) fica evidente que existem pontos mais suscetíveis a cada uma das abordagens, o que deixa ainda mais claro que, a análise destas propriedades de maneira isolada deixa muitos pontos expostos ou não verificados.

A partir da análise das FTAs relacionadas à *safety* 21 fica claro que existem elemen-

Tabela 9 – Interação *Safety/security*.

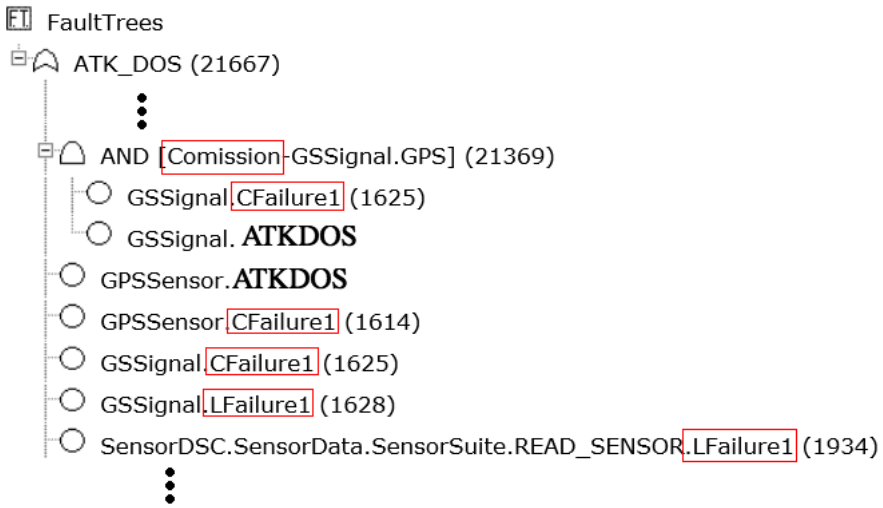
Ataque	Atributo afetado	Falha provocada
GPSSPOOF	Integridade	*Value* / *Commission*
DOS_ATK	Disponibilidade	Late / *Omission*
MIM	Confidencialidade	*Late* / *Value* [1]

Fonte: O autor.

tos relacionados diretamente a falhas no hardware (*safety*) mas também existem elementos ligados diretamente a ataques (GPSSPOOF) feitos ao sistema (*secutiry*) e que suas principais consequências se concentram no *Control DSC*, responsável pelos atuadores da aeronave.

Também foi observado que falhas de *safety* são capazes de gerar ambientes mais propensos a ataques de *security*, mesmo que não possam por si só gera-los. Esta constatação fica evidente quado se observa a FTA do ataque de DoS (Figura 21, no qual é possível observar que o ataque tem como resultado a geração de erros do tipo *Late* e *Omission* no hardware. Logo, quanto maior a ocorrência destes dois tipos de erro, mais promissor será o ataque. Portanto, em um sistema onde há problemas/vulnerabilidades de *safety* que por si só são capazes de gerar essas determinadas falhas, quando em conjunto com o ataque de *security*, sua taxa de erro será maior do que em um ambiente que não possui falhas de *safety*, ficando claro que um problema de *safety* pode influenciar na ocorrência de ameaças de *security*.

Figura 21 – Fração da FTA de DoS.



Fonte: O autor.

É válido ressaltar que apenas a ocorrência de erros de *Late* e *Comission* em determinados componentes não qualifica um ataque de DoS, que necessariamente é causado por uma fonte externa ao sistema.

Isto demonstra que se *safety* for analisada de forma isolada, não seriam cobertas diversas possibilidades de falhas que necessitam, parcialmente ou totalmente, que falhas de *security* também ocorram. A Tabela 10 ilustra essa interação.

Tabela 10 – Interação *Safety/Security*.

Hazard	*Safety*	*Security*	Componente
Ataque de DoS	*Late* e *Omission*	Disponibilidade	SensorDSC.Sensor data.SensorSuite. READ_SENSOR. SensorGPS ControlDSC.DataSource. Sensor2Control
Ataque de GPSSPOOF	Value	Integridade	SensorDSC.Sensor data.SensorSuite READ_SENSOR.SensorGPS ControlDSC.DataSource. Sensor2Control Atuador1

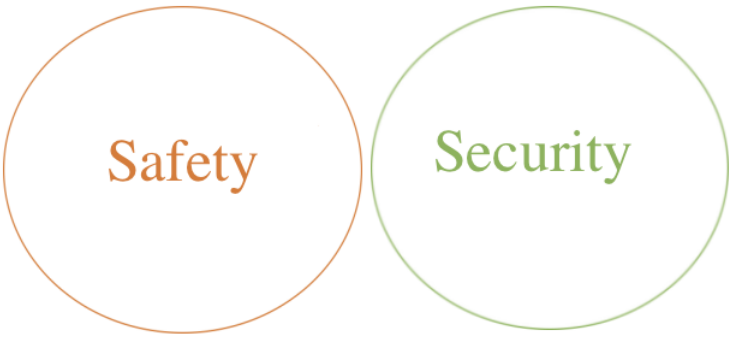
Fonte: O autor.

5.3.2 Interação entre Falhas de Safety/Security em Nível de Componente

Durante o estudo também foram analisados pontos em que interações entre os *falhas de componentes* relacionadas a *safety* e *security*, ou seja falhas que impactam em outro e vice-versa. Porém, falhas de *security* em componentes podem causar impactos de forma direta a *safety* por exemplo: um ataque de DoS/DDoS causará falhas de *LATE* e de *OMISSION* nos componentes ligados à recepção e interpretação dos dados. Já problemas de *safety* não geraram por si só um problema de *security*, porém podem deixar o sistema mais propício a um ataque de *security*, por exemplo: se os componentes responsáveis pela leitura do sinal de GPS por algum defeito ou falha de *safety*, gerarem mais erros de *VALUE*, quando um atacante tentar realizar um ataque de GPSSPOOF (cujo o impacto é gerar erros do tipo *VALUE*) além dos erros provocados pelo ataque tem-se também o erro intrínseco ao componente que está sendo atacando, tendo-se um ataque muito mais efetivo.

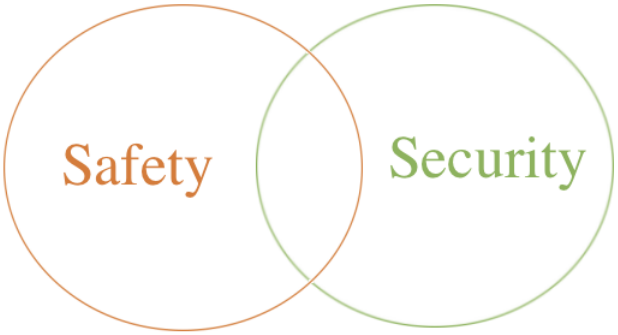
Na Tabela 10 são ilustradas as situações citadas anteriormente e qual a falha de *safety*/*security* é provocada, além de qual componente seria afetado. Para a ocorrência destes *hazards* devem ocorrer eventos tanto de *safety* quanto de *security*, o que demonstra que o estudo de segurança de sistemas embarcados criticos devem contemplar os dois requisitos. Na Figura 22 é ilustrado como são tratadas as análises de *safety* e *security* atualmente, de forma isoladas onde uma não interfere na outra. Na Figura 23 é ilustrado como esta interação se apresentou durante o estudo, com pontos que só existem entre os dois conjuntos e que não seriam abordados caso não seja considerada a interação entre os dois lados.

Figura 22 – Como *safety* e *security* são vistos atualmente.



Fonte: O autor.

Figura 23 – Como *safety* e *security* se apresentaram durante o estudo.



Fonte: O autor.

5.4 Considerações Finais

Neste capítulo foi apresentada a aplicação da abordagem UNiVErSE proposta nesta dissertação para analisar as propriedades de *safety* e de *security* do piloto automático SLUGS. Como resultado, foi possível encontrar alguns componentes chave no sistema capazes de por si só gerarem uma falha em nível de sistema. Também foram identificados os componentes que são afetados/sobrecarregados durante os principais ataques que se transformam em vulnerabilidades de *security*, de modo que foram também identificadas interações entre aspectos de *safety* e *security* de sistemas de veículos aéreos autônomos.

CAPÍTULO

6

CONCLUSÕES

Esta dissertação apresenta uma nova abordagem de análise de segurança (computacional e física) em sistemas embarcados, denominada UNiVErSE, que tem como diferencial uma visão mais abrangente que as comumente usadas. Nesta abordagem são considerados e avaliados aspectos de *safety* e *security*, que são tratados de forma separada e individualizada e que passam a ser tratados de forma integrada. A UNiVErSe busca a integração de *safety* e *security*.

A relação entre *safety* e *security* que antes era nebulosa, após a conclusão dos estudos apresentados nesta dissertação de mestrado se torna clara. Durante o estudo de caso apresentado, foi possível gerar artefatos (FTAs e FEMEAs) que demonstraram a existência de uma forte interação entre ambas, e que pode ser evidenciado por meio da FTA de DoS ilustrada na Figura 21.

A forte relação entre *security* e *safety* foi apresentada e destacada. Certamente estudos mais aprofundados, objetivando o desenvolvimento de ferramentas e abordagens que possam a partir de agora trabalhar de forma integrada as duas visões é algo que se almeja, já que permitirá uma maior robustez dos sistemas embarcados críticos.

6.1 Dificuldades Encontradas

Dentre as principais dificuldades encontradas a obtenção de conhecimento, mesmo que superficial, de conceitos e funcionamento do domínio aeronáutico se mostrou razoavelmente complexo porém a existência de vasto material disponível se mostrou suficiente.

Mesmo com o Laboratório de sistemas embarcados críticos (LSEC) já possuindo exemplares do Hardware e cabos necessários para o funcionamento do SLUGS, a compilação e *upload* do piloto automático para a placa foi bem trabalhoso, a documentação para este processo é bem superficial e com alguns equipamentos que o laboratório não dispõe, logo foram feitas algumas adaptações para que fosse possível sua execução.

Durante a fase de análise de *security* foi pensado se uma simulação feita sobre a placa, fora de uma aeronave, em uma ambiente fixo (o VANT não estaria realmente percorrendo o trajeto) e com um canal de comunicação (que é o principal alvo dos ataques de *security*) diferente (no caso cabeado ligado a porta serial do computador) do ambiente real (*link* de rádio), levaria a informações obtidas apenas de forma teórica. Como os ataques testados, e seus possíveis impactos, já são muito bem estudados e conhecidos foi possível inferir em quais pontos do sistema e do hardware eles afetariam e quais seriam seus possíveis impactos.

Outra dificuldade foi o processo de adaptação dos modelos já existentes do SLUGS para uma versão que se adequasse a algumas limitações inerentes ao software escolhido (HiP-HOPS) para a geração da árvore de falhas. Este processo gerou uma demanda de tempo razoável pois não existe na literatura uma metodologia para sua realização tão pouco manuais.

Desse modo, diversos foram os aprendizados técnicos e científicos obtidos a partir da execução do presente trabalho.

6.2 Contribuições

Dentre as contribuições deste trabalho é possível citar:

1. Avanço no estado da arte no estudo da integração de *safety* e *security* em sistemas embarcados.
2. Criação de uma metodologia robusta e completa para análise de *safety* e *security* de forma integrada.
3. Melhor entendimento da interação entre *safety* e *security*.

6.3 Limitações

Durante a execução da pesquisa não foi possível realizar os ataques de *Security* em um ambiente real, com uma aeronave voando sobre o controle do SLUGS, pois, apesar do laboratório dispor de Drones e das placas do SLUGS, a realização do experimentos de forma real levaria mais tempo do que o disponível para a realização do trabalho. Esse trabalho deverá ser realizado em uma próxima etapa por um aluno do iniciação científica.

6.4 Produção Intelectual

- M.L., Franco; K.R.L.J.C., Branco; R.T.V, Braga; A.L, Oliveira; C., Dezan; J.P. Diguet. Model-Based Dependability Analysis of Unmanned Aerial Vehicles - A Case Study. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and

Networks Workshops (DSNW), 2018, Luxembourg. 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2018. p. 263-270.

6.5 Trabalhos Futuros

Como trabalhos futuros foram elencadas as seguintes possibilidades:

- Realização dos ataques de *security* em ambiente HIL e em voo para comparar os resultados obtidos.
- Realização da metodologia criada em outro piloto automático, como por exemplo, paparazi.
- Comparação dos resultados obtidos com a aplicação da metodologia em diferente pilotos automáticos, gerando assim um conjunto de árvores de falhas que forneçam informações que possam ser utilizadas para gerar uma taxonomia ou um conjunto de ações/políticas que possam ser implementadas de forma automática.
- Realizar testes com outros ataques e comparar os resultados.

REFERÊNCIAS

167, R. F. S. *Software considerations in Airborne Systems and equipment certification*. [S.l.]: RTCA, Incorporated, 1992. Citado na página 25.

AERONAUTICS, N.; (NASA), S. A. **Fault Tree Analysis with Aerospace Applications**. [S.l.], 2002. Citado na página 63.

AMBLER, S. W. Agile model driven development is good enough. **IEEE Software**, IEEE, v. 20, n. 5, p. 71–73, 2003. Citado na página 45.

AMOROSO, E. **Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response. Intruion**. [S.l.]: Net Books,, 1999. Citado na página 51.

ANAC. **Requisitos Gerais para Veículos Aéreos não Tripulados e Aeromodelos**. [S.l.], 2015. Citado na página 34.

ARP4754A, S. Guidelines for development of civil aircraft and systems. **SAE International**, 2010. Citado nas páginas 55, 63 e 78.

ÅSTRÖM, K. J.; HÄGGLUND, T. **Advanced PID control**. [S.l.]: ISA-The Instrumentation, Systems and Automation Society, 2006. Citado na página 37.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. Citado na página 67.

AUSTIN, R. **Unmanned aircraft systems : UAVS design, development and deployment**. Chichester: Wiley, 2010. 332 p. Citado nas páginas 31, 33 e 34.

____. **Unmanned Aircraft Systems: UAVS Design, Development and Deployment**. [S.l.]: Wiley, 2011. ISBN 978-0-470-05819-0. Citado na página 35.

AZEVEDO, L. S.; PARKER, D.; WALKER, M.; PAPADOPOULOS, Y.; ARAUJO, R. E. Automatic decomposition of safety integrity levels: optimization by tabu search. In: **SAFECOMP 2013-Workshop CARS (2nd Workshop on Critical Automotive applications: Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security**. [S.l.: s.n.], 2013. p. NA. Citado na página 63.

BATTEUX, M.; PROSVIRNOVA, T.; RAUZY, A.; KLOUL, L. The altarica 3.0 project for model-based safety assessment. In: IEEE. **Industrial Informatics (INDIN), 2013 11th IEEE International Conference on**. [S.l.], 2013. p. 741–746. Citado na página 60.

BEARD, T. W. M. R. W. **SMALL UNMANNED AIRCRAFT Theory and Practic**. [S.l.]: Princeton University Press, 2012. Citado na página 36.

BEKMEZCI, İ.; SAHINGOZ, O. K.; TEMEL, Ş. Flying ad-hoc networks (FANETs): A survey. **Ad Hoc Networks**, Elsevier BV, v. 11, n. 3, p. 1254–1270, may 2013. Citado na página 35.

BHANOT, V.; PANISCOTTI, D.; ROMAN, A.; TRASK, B. Using domain-specific modeling to develop software defined radio components and applications. In: SN. **The 5th OOPSLA Workshop on Domain-Specific Modeling, San Diego USA**. [S.l.], 2005. p. 33–42. Citado na página 45.

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. **Advanced Engineering Informatics**, Elsevier, v. 21, n. 2, p. 133–151, 2007. Citado na página 110.

BLOOMFIELD, R.; LALA, J. Safety-critical systems: The next generation. **IEEE Security & Privacy**, IEEE, v. 11, n. 4, p. 11–13, 2013. Citado na página 24.

BOZZANO, M.; VILLAFIORITA, A. The fsap/nusmv-sa safety analysis platform. **International Journal on Software Tools for Technology Transfer (STTT)**, Springer, v. 9, n. 1, p. 5–24, 2007. Citado na página 60.

BUDGEN, D.; KITCHENHAM, B.; CHARTERS, S. M.; TURNER, M.; BRERETON, P.; LINKMAN, S. G. Preliminary results of a study of the completeness and clarity of structured abstracts. In: **EASE**. [S.l.: s.n.], 2007. Citado na página 109.

CAA. **Unmanned aircraft system operations in UK airspace – Guidance**. [S.l.], 2015. Citado na página 34.

CABOT, J. Clarifying concepts: Mbe vs mde vs mdd vs mda. **Post at MOdeling LAnguages, http://modeling-languages. com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/. 22http://www. mrc-productivity. com**, v. 86, 2015. Citado na página 44.

CHANDHRASEKARAN, V. K.; CHOI, E. Fault tolerance system for uav using hardware in the loop simulation. In: IEEE. **New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on**. [S.l.], 2010. p. 293–300. Citado na página 71.

COIT, D. W.; SMITH, A. E. Penalty guided genetic search for reliability design optimization. **Computers & industrial engineering**, Elsevier, v. 30, n. 4, p. 895–904, 1996. Citado na página 62.

CRASSIDIS, J. L.; JUNKINS, J. L. **Optimal Estimation of Dynamic Systems, Second Edition**. 2nd. ed. [S.l.]: Chapman & Hall/CRC, 2011. ISBN 1439839859, 9781439839850. Citado na página 38.

CUENOT, P.; FREY, P.; JOHANSSON, R.; LÖNN, H.; PAPADOPOULOS, Y.; REISER, M.-O.; SANDBERG, A.; SERVAT, D.; KOLAGARI, R. T.; TÖRNGREN, M. *et al.* 11 the east-adl architecture description language for automotive embedded software. In: **Model-based engineering of embedded real-time systems**. [S.l.]: Springer, 2010. p. 297–307. Citado na página 61.

DAVID, P.; IDASIAK, V.; KRATZ, F. Reliability study of complex physical systems using sysml. **Reliability Engineering & System Safety**, Elsevier, v. 95, n. 4, p. 431–450, 2010. Citado na página 59.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE transactions on evolutionary computation**, IEEE, v. 6, n. 2, p. 182–197, 2002. Citado na página 62.

DEBAR, H.; DACIER, M.; WESPI, A. Towards a taxonomy of intrusion-detection systems. **Computer Networks**, Elsevier, v. 31, n. 8, p. 805–822, 1999. Citado na página 50.

DELANGE, J.; FEILER, P. Architecture fault modeling with the aadl error-model annex. In: IEEE. **Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on**. [S.l.], 2014. p. 361–368. Citado nas páginas 26, 58, 59, 60 e 76.

DENNEY, E.; PAI, G.; HABLI, I. Towards measurement of confidence in safety cases. In: IEEE. **Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on**. [S.l.], 2011. p. 380–383. Citado na página 71.

____. Perspectives on software safety case development for unmanned aircraft. In: IEEE. **Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on**. [S.l.], 2012. p. 1–8. Citado na página 71.

DEPENDABLE SYSTEMS RESEARCH GROUP. **HiP-HOPS Automated Fault Tree, FMEA and Optimisation Tool User Manual**. Hull, 2013. 43 p. Disponível em: <http://www.hip-hops.eu/images/Manual/HiP-HOPS_Manual.pdf>. Citado na página 64.

DEURSEN, A. V.; KLINT, P. *et al.* Little languages: little maintenance? **Journal of software maintenance**, v. 10, n. 2, p. 75–92, 1998. Citado na página 45.

DO, R. **254, Design Assurance Guidance for Airborne Electronic Hardware, Washington, DC, RTCA**. [S.l.]: Inc, 2000. Citado na página 25.

____. **326 Airworthiness Security Process Specification**. [S.l.]: December, 2010. Citado na página 25.

DOD. **Unmanned Systems Roadmap 2007-2032**. Washington,DC, 2007. 187 p. Citado na página 34.

____. **Unmanned Systems Integrated Roadmap FY2009-2034**. Washington,DC, 2009. 210 p. Citado na página 34.

DUDEK, M. J. G. **Computational Principles of Mobile Robotics**. [S.l.]: Cambridge University Press, 2010. Citado na página 29.

EASA. **Introduction of a regulatory framework for the operation of unmanned aircraft**. [S.l.], 2015. Citado na página 34.

ENGINEERS, S. of A. [S.l.]: SAE, 400 Commonwealth Drive Warrendale PA United States, 1996. Citado nas páginas 55 e 58.

ERICSON, C. A. Fault tree analysis. In: **System Safety Conference, Orlando, Florida**. [S.l.: s.n.], 1999. p. 1–9. Citado nas páginas 55 e 56.

____. Event tree analysis. **Hazard Analysis Techniques for System Safety**, Wiley Online Library, p. 223–234, 2005. Citado na página 55.

EUROCAE. **RP4754A - guidelines for development of civil aircraft and systems**. [S.l.]: Technical Report, EUROCAE, 2010. Citado na página 85.

FALKENHAINER, B.; FORBUS, K. D. Cmpositional modeling: finding the right model for the job. **Artificial intelligence**, Elsevier, v. 51, n. 1-3, p. 95–143, 1991. Citado na página 27.

FAUGHNAN, M. S.; HOURICAN, B. J.; MACDONALD, G. C.; SRIVASTAVA, M.; WRIGHT, J.-P. A.; HAIMES, Y. Y.; ANDRIJCIC, E.; GUO, Z.; WHITE, J. C. Risk analysis of unmanned aerial vehicle hijacking and methods of its detection. In: IEEE. **Systems and Information Engineering Design Symposium (SIEDS), 2013 IEEE**. [S.l.], 2013. p. 145–150. Citado na página 24.

FERNANDEZ, M. I. L. **Design, implementation and flight verification of a versatile and rapidly reconfigurable UAV GNC research platform**. [S.l.]: University of California, Santa Cruz, 2009. Citado na página 40.

GARFINKEL, S.; SPAFFORD, G.; SCHWARTZ, A. **Practical UNIX and Internet security**. [S.l.]: "O'Reilly Media, Inc.", 2003. Citado na página 51.

GOLLMANN, D. Computer security. **Wiley Interdisciplinary Reviews: Computational Statistics**, Wiley Online Library, v. 2, n. 5, p. 544–554, 2010. Citado na página 50.

GUDEMANN, M.; ORTMEIER, F. A framework for qualitative and quantitative formal model-based safety analysis. In: IEEE. **High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on**. [S.l.], 2010. p. 132–141. Citado na página 60.

HABLI, I.; KELLY, T.; PAIGE, R. Functional hazard assessment in product-lines–a model-based approach. In: **MDPLE'2009 1st International Workshop on Model-Driven Product Line Engineering CTIT PROCEEDINGS**. [S.l.: s.n.], 2009. p. 26. Citado na página 53.

HAWKINS R., H. I. K. D. P. R. K. T. Weaving an assurance case from design: a model-based approach. In: IEEE. **Proc. of the 16th HASE, Daytona Beach**. [S.l.], 2015. p. 110–117. Citado na página 82.

Health and Safety Executive. **Risk management: ALARP at a glance**. 2016. Disponível em: <http://www.hse.gov.uk/risk/theory/alarpglance.htm>. Citado na página 54.

HIGH, K. M.; KELLY, T.; MCDERMID, J. Safety case construction and reuse using patterns. In: CITESEER. **16th International Conference on Computer Safety and Reliability (SAFECOMP'97**. [S.l.], 1997. Citado na página 65.

HUSSEIN, M.; ZULKERNINE, M. Umlintr: a uml profile for specifying intrusions. In: IEEE. **Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on**. [S.l.], 2006. p. 8–pp. Citado na página 69.

ICAO. **Circular 328 AN/190: Unmanned Aircraft Systems Operations (UAS). International Civil Aviation Organization (ICAO), Montreal, Canada**. 2011. Citado na página 85.

ISO. **ISO 26262: road vehicles functional safety**. [S.l.]: Technical Report, ISO, 2011. Citado nas páginas 49, 55 e 79.

JACKLIN, S. Certification of safety-critical software under do-178c and do-278a. In: **Infotech@ Aerospace 2012**. [S.l.: s.n.], 2012. p. 2473. Citado na página 50.

JAVAID, A. Y.; SUN, W.; DEVABHAKTUNI, V. K.; ALAM, M. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In: IEEE. **Homeland Security (HST), 2012 IEEE Conference on Technologies for**. [S.l.], 2012. p. 585–590. Citado na página 24.

JIMNEZ, M.; PALOMERA, R.; COUVERTIER, I. **Introduction to Embedded Systems: Using Microcontrollers and the MSP430**. [S.l.]: Springer Publishing Company, Incorporated, 2013. ISBN 1461431425, 9781461431428. Citado na página 23.

JOSHI, A.; MILLER, S. P.; WHALEN, M.; HEIMDAHL, M. P. A proposal for model-based safety analysis. In: IEEE. **Digital Avionics Systems Conference, 2005. DASC 2005. The 24th**. [S.l.], 2005. v. 2, p. 13–pp. Citado na página 58.

KASHIKAR, M.; NIMBHORKAR, S. Designing acknowledgement based manet using public key cryptography. In: IEEE. **Computer Science & Education (ICCSE), 2013 8th International Conference on**. [S.l.], 2013. p. 228–233. Citado na página 24.

KÄSSMEYER, M.; SCHULZE, M.; SCHURIUS, M. A process to support a systematic change impact analysis of variability and safety in automotive functions. In: ACM. **Proceedings of the 19th International Conference on Software Product Line**. [S.l.], 2015. p. 235–244. Citado na página 59.

KELLY, T. **A systematic approach to safety case management**. [S.l.], 2004. Citado nas páginas 65 e 82.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004. Citado na página 110.

KLEPPE, A. G.; WARMER, J.; BAST, W.; EXPLAINED, M. **The model driven architecture: practice and promise**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2003. Citado na página 45.

KLETZ, T. A. **HAZOP and HAZAN: identifying and assessing process industry hazards**. [S.l.]: IChemE, 1999. Citado na página 52.

____. **HAZOP and HAZAN: identifying and assessing process industry hazards**. [S.l.]: IChemE, 2001. Citado na página 54.

KOOPMAN, P. Embedded system security. **Computer**, IEEE, v. 37, n. 7, p. 95–97, 2004. Citado na página 67.

KOOPMAN, P.; WAGNER, M. Autonomous vehicle safety: An interdisciplinary challenge. **IEEE Intelligent Transportation Systems Magazine**, IEEE, v. 9, n. 1, p. 90–96, 2017. Citado na página 24.

KULTUREL-KONAK, S.; SMITH, A. E.; COIT, D. W. Efficiently solving the redundancy allocation problem using tabu search. **IIE transactions**, Taylor & Francis, v. 35, n. 6, p. 515–526, 2003. Citado na página 62.

KUROSE, J. F.; ROSS, K. W. Redes de computadores e a internet. **São Paulo: Person**, p. 28, 2006. Citado na página 66.

LEVESON, N.; GOETSCH, S. Safeware: System safety and computers. **Medical Physics-New York-Institute of Physics**, [New York, etc., Published for the American Association of Physicists in ..., v. 23, n. 10, p. 1821, 1995. Citado na página 49.

LEVESON, N. G. Software safety: Why, what, and how. **ACM Computing Surveys (CSUR)**, ACM, v. 18, n. 2, p. 125–163, 1986. Citado nas páginas 49 e 52.

____. Safety as a system property. **Communications of the ACM**, ACM, v. 38, n. 11, p. 146, 1995. Citado na página 26.

LEWIS, F. **Optimal estimation: with an introduction to stochastic control theory**. [S.l.]: Wiley, 1986. (A Wiley-interscience publication). ISBN 9780471837411. Citado na página 38.

LISAGOR, O.; MCDERMID, J.; PUMFREY, D. Towards a practicable process for automated safety analysis. In: **24th International system safety conference**. [S.l.: s.n.], 2006. v. 596, p. 607. Citado nas páginas 58, 60 e 62.

LIZARRAGA, M.; CURRY, R.; ELKAIM, G. Reprogrammable uav autopilotsystem: System hardware and software. **Circuit Cellar Magazine**, v. 249, p. 24–35, April 2011. Citado na página 39.

MARCONATO, E. A. **Modelo de arquitetura em camadas para interconexão de sistemas em SANT**. Tese (Doutorado) — Universidade de São Paulo. Citado na página 69.

MATARIC, M. J. **The Robotics Primer**. [S.l.]: MIT Press, 2007. Citado na página 29.

MAZZINI S., F. J. P. S. B. L. Hess: an open source methodology and toolset for the development of critical systems. In: SPRINGER. **LNCS PJoin Proceedings of EduSymp**. [S.l.], 2016. p. 59–66. Citado nas páginas 26 e 76.

MERINO, L.; CABALLERO, F.; DIOS, J. M. de; FERRUZ, J.; OLLERO, A. A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires. **Journal of Field Robotics**, Wiley-Blackwell, v. 23, n. 3-4, p. 165–184, 2006. Citado na página 35.

MERNIK, M.; HEERING, J.; SLOANE, A. M. When and how to develop domain-specific languages. **ACM computing surveys (CSUR)**, ACM, v. 37, n. 4, p. 316–344, 2005. Citado na página 45.

MICA, J.; COSTELLO, J. **Unmanned aircraft systems: Federal actions needed to ensure safety and expand their potential uses within the national airspace system**. [S.l.]: Report, 2008. Citado na página 23.

MILICEV, D. **Model-driven development with executable UML**. [S.l.]: John Wiley & Sons, 2009. Citado na página 44.

MILITARY, U. Procedure for performing a failure mode effect and criticality analysis. **United States military procedure MIL-P-1629**, 1949. Citado nas páginas 55 e 57.

MOD. **DEF-STAN 00-56 Issue 4 Part 1: Safety management requirements for defense systems**. [S.l.], 2007. Citado nas páginas 50, 52, 55, 65 e 79.

National Aeronautics and Space Administration (NASA). **Fault Tree Analysis with Aerospace Applications**. Washington, D.C., 2002. Citado nas páginas 55, 56 e 57.

NONAMI, K.; KENDOUL, F.; SUZUKI, S.; WANG, W.; NAKAZAWA, D. **Autonomous Flying Robots Unmanned Aerial Vehicles and Micro Aerial Vehicles**. [S.l.]: Springe, 2010. 329 p. Citado na página 29.

NYSTROM, B.; AUSTRIN, L.; ANKARBACK, N.; NILSSON, E. Fault tree analysis of an aircraft electric power supply system to electrical actuators. In: IEEE. **Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on**. [S.l.], 2006. p. 1–7. Citado na página 71.

OLIVEIRA A. L., B. R. T. B. M. P. C. P. Y. H. I. K. T. Variability management in safety-critical software product line engineering. In: SPRINGER. **LNCS Proc. of the 17th International Conference on Software Reuse(ICSR)**. [S.l.], 2018. Citado na página 76.

OMG. **Systems modelling language (SysML)**. [S.l.]: OMG, version 1.4, 2015. Citado na página 83.

PALIN, R.; WARD, D.; HABLI, I.; RIVETT, R. Iso 26262 safety cases: Compliance and assurance. IET, 2011. Citado nas páginas 49 e 50.

PAPADOPOULOS, Y. A synthesis of logic and biology in the design of dependable systems. **ScienceDirect**, v. 48, p. 01–08, março 2015. Citado na página 51.

PAPADOPOULOS, Y.; MCDERMID, J. Hierarchically performed hazard origin and propagation studies. **Computer safety, reliability and security**, Springer, p. 688–688, 1999. Citado nas páginas 58, 59 e 60.

PAPADOPOULOS, Y.; MCDERMID, J. A. The potential for a generic approach to certification of safety critical systems in the transportation sector. **Reliability engineering & system safety**, Elsevier, v. 63, n. 1, p. 47–66, 1999. Citado na página 49.

PAPADOPOULOS, Y.; WALKER, M.; PARKER, D.; RÜDE, E.; HAMANN, R.; UHLIG, A.; GRÄTZ, U.; LIEN, R. Engineering failure analysis and design optimisation with hip-hops. **Engineering Failure Analysis**, Elsevier, v. 18, n. 2, p. 590–608, 2011. Citado nas páginas 26, 57, 58, 63, 76, 79, 80 e 89.

PARK, S.; DEYST, J.; HOW, J. P. A new nonlinear guidance logic for trajectory tracking. In: **AIAA guidance, navigation, and control conference and exhibit**. [S.l.: s.n.], 2004. p. 16–19. Citado na página 36.

PARKER, D.; WALKER, M.; PAPADOPOULOS, Y. Model-based functional safety analysis and architecture optimisation. **Embedded Computing Systems: Applications, Optimization, and Advanced Design: Applications, Optimization, and Advanced Design**, IGI Global, p. 79–92, 2013. Citado na página 63.

PASTOR, E.; LOPEZ, J.; ROYO, P. An embedded architecture for mission control of unmanned aerial vehicles. In: 9TH EUROMICRO CONFERENCE ON DIGITAL SYSTEM DESIGN ARCHITECTURES, METHODS AND TOOLS. Dubrovnik: IEEE Computer Society, 2006. p. 554–560. Citado na página 33.

____. Uav payload and mission control hardware / software architecture. **IEEE A&E Systems Magazine**, v. 22, p. 3–8, Junho 2007. Citado na página 33.

PROCTOR, P. E. **Practical intrusion detection handbook**. [S.l.]: Prentice Hall PTR, 2000. Citado na página 51.

PUMFREY, D. J. **The principled design of computer system safety analyses.** Tese (Doutorado) — University of York, 1999. Citado nas páginas 57 e 58.

REBECCA, B. **Intrusion detection**. [S.l.]: Macmillan Technical Publishing, 2000. Citado nas páginas 50 e 51.

REGULATION, O. for N. **Safety Assessment Principles**. [S.l.], 2014. 226 p. Disponível em: <http://www.onr.org.uk/saps/saps2014.pdf>. Citado na página 64.

ROTHENBERG, J.; WIDMAN, L. E.; LOPARO, K. A.; NIELSEN, N. R. **The nature of modeling**. [S.l.]: Rand, 1989. v. 3027. Citado na página 45.

SAMPIGETHAYA, K.; POOVENDRAN, R. Aviation cyber–physical systems: Foundations for future aircraft and air transport. **Proceedings of the IEEE**, IEEE, v. 101, n. 8, p. 1834–1855, 2013. Citado na página 25.

SCHMIDT, D. C. Model-driven engineering. **COMPUTER-IEEE COMPUTER SOCIETY-**, IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS, v. 39, n. 2, p. 25, 2006. Citado na página 45.

SCHOITSCH, E. Design for safety and security of complex embedded systems: A unified approach. In: **Cyberspace Security and Defense: Research Issues**. [S.l.]: Springer, 2005. p. 161–174. Citado nas páginas 52, 53 e 56.

SELIC, B. The pragmatics of model-driven development. **IEEE software**, IEEE, v. 20, n. 5, p. 19–25, 2003. Citado na página 45.

SIMON, D. **Optimal state estimation: Kalman, H infinity, and nonlinear approaches**. [S.l.]: John Wiley & Sons, 2006. Citado na página 37.

Society of Automotive Engineers (SAE). **ARP 4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment**. [S.l.]: SAE 400 Commonwealth Drive, 1994. Citado na página 52.

SOROKOS, I.; PAPADOPOULOS, Y.; AZEVEDO, L.; PARKER, D.; WALKER, M. Automating allocation of development assurance levels: an extension to hip-hops. **IFAC-PapersOnLine**, Elsevier, v. 48, n. 7, p. 9–14, 2015. Citado na página 63.

SOUSA, H.; LOPES, D.; ABDELOUAHAB, Z.; CLARO, D. B.; HAMMOUDI, S. An approach for model driven testing: framework, metamodels and tools. **International Journal of Computer Systems Science & Engineering**, CRL Publishing Ltd., v. 26, n. 4, p. 307–318, 2011. Citado na página 44.

STAPKO, T. Practical embedded security. **Building Secure Resource-Constrained Systems, Burlington Oxford**, 2008. Citado na página 66.

SUN, L. **Establishing confidence in safety assessment evidence**. Tese (Doutorado) — University of York, 2012. Citado na página 50.

TANENBAUM, A. S.; COMPUTADORES, R. de. 4ª edição. **Rio de Janeiro: Editora Campus**, 2003. Citado na página 66.

THOMAS, D. Mda: Revenge of the modelers or uml utopia? **IEEE software**, IEEE, v. 21, n. 3, p. 15–17, 2004. Citado na página 45.

TOEPKE, S. L. **Documentation, deployment and extension of a versatile and rapidly econfigurable UAV GNC research platform**. Tese (Doutorado) — University of California, Santa Cruz, 2012. Citado na página 42.

TRINDADE, O.; BARBOSA, L.; NERIS, L. O.; JORGE, L. A. C. A mission planner and navigation system for the arara project. In: ICAS - 23RD INTERNATIONAL CONGRESS OF AERONAUTICAL SCIENCES. Toronto: Optimage Ltd, 2002. p. 682.1 – 682.8. Citado na página 31.

TRINDADE, O.; BRANCO, K. R. L. J. C.; NERIS, L. O.; CHAVIER, L. F. C. Robôs aéreos. In: ROBÓTICA MÓVEL. [S.l.]: LCT, 2012. p. 461–478. Citado na página 33.

TRINDADE, O.; NERIS, L. D. O.; BARBOSA, L. C. P.; BRANCO, K. R. L. J. C. A layered approach to design autopilots. In: IEEE. **Industrial Technology (ICIT), 2010 IEEE International Conference on**. [S.l.], 2010. p. 1415–1420. Citado nas páginas 29 e 31.

VACCA, J. R. **Computer and information security handbook**. [S.l.]: Newnes, 2012. Citado na página 50.

WALLACE, M. Modular architectural representation and analysis of fault propagation and transformation. **Electronic Notes in Theoretical Computer Science**, Elsevier, v. 141, n. 3, p. 53–71, 2005. Citado na página 61.

WILKINSON, P.; KELLY, T. Functional hazard analysis for highly integrated aerospace systems. IET, 1998. Citado na página 56.

____. Functional hazard analysis for highly integrated aerospace systems. IET, 1998. Citado na página 65.

WINTHER, R.; JOHNSEN, O.-A.; GRAN, B. A. Security assessments of safety critical systems using hazops. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2001. p. 14–24. Citado nas páginas 69 e 87.

APÊNDICE

A

DADOS REVISÃO SISTEMÁTICA

A revisão sistemática é uma forma de identificar, avaliar e interpretar as pesquisas relevantes de uma área em torno de uma questão científica. Por meio dessa técnica é possível resumir possíveis evidências de uma tecnologia, identificar lacunas para sugerir áreas que devem ser investigadas, esclarecer o estado da arte para o posicionamento de novas pesquisas e examinar resultados empíricos que comprovem ou refutem uma determinada hipótese teórica, ou ainda que auxiliem na criação de novas hipóteses. Sendo assim, um protocolo predeterminado é necessário para reduzir a possibilidade do investigador ser tendencioso (BUDGEN *et al.*, 2007).

A.1 Planejamento

A primeira etapa da revisão sistemática é o planejamento do protocolo da revisão, ou seja, define-se quais são os objetivos, as questões , as fontes de pesquisa e os critérios de seleção que a revisão pretende abranger.

Tem-se como objetivos desta revisão sistemática os seguintes itens:

- Identificar trabalhos que analisem *safety* em sistemas de aeronaves não tripuladas usando *safety cases*;
- Identificar trabalhos que analisem *security* em sistemas de aeronaves não tripuladas usando *safety cases*;
- Identificar trabalhos que analisem *safety* em sistemas de aeronaves não tripuladas usando testes de penetração;
- Identificar trabalhos que analisem *security* em sistemas de aeronaves não tripuladas usando testes de penetração;

- Identificar trabalhos que analisem *safety* em sistemas de aeronaves não tripuladas usando *fault tree Analysis*.

Durante o planejamento, a atividade mais importante é a formulação das questões, pois as mesmas estabelecem o foco de interesse da pesquisa (KITCHENHAM, 2004). Com a resposta para tais questões é possível realizar uma análise mais especifica cujo intuito seja obter uma conclusão sobre o foco da pesquisa. Seguindo este principio foi elaborada a seguinte questão:

- Questão: Quais trabalhos analisam *satefy* ou *security* em sistemas de aeronaves não tripulados usando *safety cases* ou árvore de falhas?

Essa questão levou em consideração as seguintes especificidades (BIOLCHINI *et al.*, 2007):

- **População:** Serão utilizados artigos que direcionem esforços em aumentar a segurança, seja física ou computacional, em veículos aéreos não tripulados;
- **Intervenção:** Será observado se os artigos envolvem aspectos de segurança física e computacional no projeto e desenvolvimento de veículos aéreos não tripulados;
- **Comparação:** Serão avaliadas e analisadas as técnicas utilizadas para aumento da *safety* ou *security* em veículos aéreos não tripulados;
- **Resultados:** principais técnicas utilizadas para aumento da *safety* e *security* em veículos aéreos não tripulados e como são utilizadas.

definem-se

As fontes, idiomas e palavras-chave para a busca permitem obter os trabalhos relevantes para a área.

Para definir as fontes de busca da revisão sistemática foram observados alguns critérios sobre as mesmas, tais como: cobertura, conteúdo atualizado e disponibilidade. Sobre a Cobertura, só foram consideradas fontes de busca que retornem um número considerável de trabalhos. Mesmo retornando um grande número de trabalhos só foram considerados os que retornam trabalhos recentes sobre o tema (conteúdo atualizado). E por último, foram utilizadas apenas fontes que permitam o acesso de forma íntegra (disponibilidade). Com base nos critérios foram escolhidas as seguintes bases de busca eletrônica:

- IEEEXplore Digital Library(IEEE);
- ACM Digital Lybrary;
- Springer.

- Scopus

Os idiomas selecionados para a revisão sistemática foram o Inglês e o Português. O inglês pelo fato de ser a língua aceita internacionalmente para a redação de trabalhos científicos. O Português foi escolhido para que sejam contemplados trabalhos de autoria de pesquisadores brasileiros.

Foram ainda escolhidas as seguintes palavras-chave e seus respectivos sinônimos, de modo a contemplar os trabalhos que respondam à questão da revisão:

- UAV, unmanned aerial vehicle, UAS, unmanned aircraft, system;
- fault tree, FTA, safety case, Assurance Case, Safety Analysis, Hazard Analysis;
- safety, security, penetration test, pentest.

Conforme o protocolo da revisão sistemática, no planejamento faz-se ainda necessária a definição de critérios de seleção e os procedimentos de análise dos estudos. Esses critérios podem ser de inclusão ou de exclusão, garantindo a credibilidade da revisão e mantendo o foco da seleção.

Tendo como base os objetivos de pesquisa cinco critérios de inclusão de estudos foram definidos para aceitar às questões de pesquisa:

- **CI1 -** Estudos que analisem *safety* em sistemas de aeronaves não tripuladas usando *safety cases*;
- **CI2 -** Estudos que analisem *security* em sistemas de aeronaves não tripuladas usando *safety cases*;
- **CI3 -** Estudos que analisem *safety* em sistemas de aeronaves não tripuladas usando testes de penetração;
- **CI4 -** Estudos que analisem *security* em sistemas de aeronaves não tripuladas usando testes de penetração;
- **CI5 -** Estudos que analisem *safety* em sistemas de aeronaves não tripuladas usando *fault tree analysis* (FTA);

De modo análogo, tendo como base os objetivos da pesquisa, os seguintes critérios de exclusão de trabalhos foram definidos:

- **CE1 -** Estudos em que não seja possível o acesso na íntegra do artigo;

- **CE2 -** Estudos que estejam em idiomas diferentes do inglês ou do português;
- **CE3 -** Estudos que não sejam completos (resumos);
- **CE4 -** Estudos que não atendam aos critérios de inclusão.

A.2 Execução

A execução da revisão sistemática é dividida em quarto etapas, sendo a primeira a criação das *strings* de busca, efetuada com base nas palavras-chave. A segunda etapa corresponde a seleção preliminar dos artigos por meio dos títulos e resumos, podendo-se identificar estudos duplicados e efetuar a seleção com base nos critérios de inclusão e exclusão. A terceira etapa consiste na seleção final com a leitura completa dos artigos não excluídos e da extração inicial das primeiras informações por meio da elaboração de resumos. Por fim, a quarta e última etapa consiste na extração dos dados que abrangem as principais técnicas utilizadas para aumento da *safety* e *security* em veículos aéreos não tripulados e como são utilizadas.

Com base nas palavras-chave e, respeitando-se as especificações de cada base de dados por meio da utilização de operadores "AND"entre termos diferentes e "OR"para agrupar sinônimos, foram elaboradas as seguintes *strings* de busca:

- ***String* Original** (UAVs OR unmanned Aerial Vehicles) AND (Safety Analysis OR Hazard Analysis) AND (Model-based OR Safety Analysis) AND (Safety Case OR Assurance Case);
- **IEEE*Xplore Digital Library*:** ((UAV OR "unmanned aerial vehicle"OR UAS OR "unmanned aircraft system") AND ("fault tree"OR "FTA"OR "safety case"OR "Assurance Case"OR "Safety Analysis"OR "Hazard Analysis") AND ("safety"OR "security"OR "penetration test"OR "pentest"));
- **ACM *Digital Library*:** ((acmdlTitle:(+unmanned +aerial +vehicle) OR acmdlTitle:(+unmanned +aircraft +system) OR acmdlTitle:(UAV UAS)) AND (acmdlTitle:(+fault +tree) OR acmdlTitle:(+safety +case) OR acmdlTitle:(+Assurance +Case) OR acmdlTitle:(+Safety +Analysis) OR acmdlTitle:(+Hazard +Analysis) OR acmdlTitle:(FTA)) AND (acmdlTitle:(+penetration +test) OR acmdlTitle:(safety security pentest))) OR ((recordAbstract:(+unmanned +aerial +vehicle) OR recordAbstract:(+unmanned +aircraft +system) OR recordAbstract:(UAV UAS)) AND (recordAbstract:(+fault +tree) OR recordAbstract:(+safety +case) OR recordAbstract:(+Assurance +Case) OR recordAbstract:(+Safety +Analysis) OR recordAbstract:(+Hazard +Analysis) OR recordAbstract:(FTA)) AND (recordAbstract:(+penetration +test) OR recordAbstract:(safety security pentest)));
- ***Springer Link*:** ((uav OR "unmanned aerial vehicle"OR uas OR "unmanned aircraft system") AND ("fault tree "OR "FTA"OR "safety case"OR "Assurance Case"OR "Safety

Analysis"OR "Hazard Analysis") AND ("safety"OR "security"OR "penetration test"OR "pentest"));

- ***Scopus*:** TITLE-ABS-KEY (((uav OR "unmanned aerial vehicle"OR uas OR "unmanned aircraft system") AND ("fault tree"OR "FTA"OR "safety case"OR "Assurance Case"OR "Safety Analysis"OR "Hazard Analysis") AND ("safety"OR "security"OR "penetration test"OR "pentest"))) AND (LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (LANGUAGE , "English")).

Realizadas as buscas nos mecanismos foram obtidos um total de 118 artigos distribuídos nas base de dados conforme ilustrado na Figura 24).

A base de dados que apresentou maior número de artigos relacionado ao tema foi a *Scopus*, seguida pela *IEEEXplorer* e da *Association for Computing Machinery* (ACM).

A primeira seleção foi realizada com a leitura dos títulos e resumos dos artigos, permitindo assim que fosse realizada a identificação dos critérios de inclusão e exclusão. Os artigos aceitos foram os que obedeceram pelo menos um dos critérios de inclusão, enquanto os rejeitados foram os que possuíram pela menos um critério de exclusão. O número de artigos aceitos e rejeitos nessa fase estão ilustrados na Figura 25.

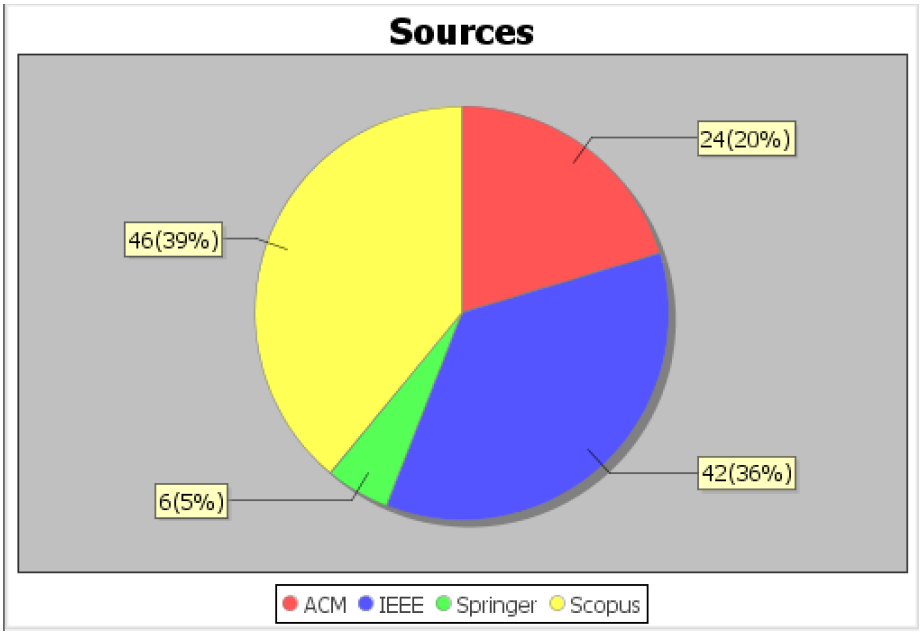


Figura 24 – Quantidade de artigos encontrados nas bases de dados para a primeira seleção.

A segunda seleção foi realizada após a leitura dos artigos completos entre os aceitos na primeira seleção, e foi realizada com base nos mesmos critérios utilizados para a primeira seleção. Durante a leitura completa dos artigos foi produzido um resumo como resultado.

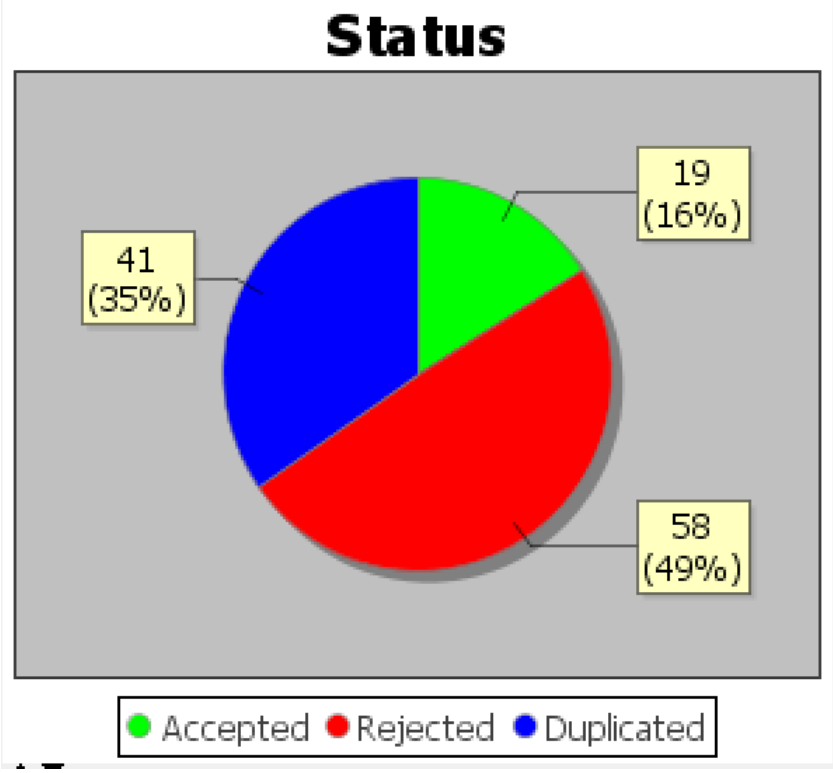


Figura 25 – Quantidade de artigos aceitos e rejeitados durante a primeira seleção aceitação dos artigos

Dentre os artigos obtidos foi constatado que nenhum deles trata de *safety* ou *security* em VANTs utilizando *pentest*, fazendo necessária a realização da palavra chave "*pentest*"de forma isolada para se obter o atual estado da arte.

Na última etapa da revisão sistemática foi produzida uma síntese discursiva para relatar os modelos, arquiteturas e modelos de arquiteturas encontrados, que foi apresentada em detalhes na Seção 3.8 do Capítulo 3.


ICMC
USP
SÃO CARLOS