FABIO SEITI AGUCHIKU

ESPECIFICAÇÃO E VERIFICAÇÃO FORMAL DE REQUISITOS PARA
SISTEMAS DE TRÁFEGO AÉREO

São Paulo
2018

FABIO SEITI AGUCHIKU

ESPECIFICAÇÃO E VERIFICAÇÃO FORMAL DE REQUISITOS PARA
SISTEMAS DE TRÁFEGO AÉREO

Dissertação apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do
título de Mestre em Ciências

São Paulo

2018

FABIO SEITI AGUCHIKU

ESPECIFICAÇÃO E VERIFICAÇÃO FORMAL DE REQUISITOS PARA SISTEMAS DE TRÁFEGO AÉREO

Dissertação apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do
título de Mestre em Ciências

Área de concentração:
Engenharia de Controle e Automação Mecânica

Orientador:
Newton Maruyama

São Paulo
2018

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 6 de setembro de 2018

Assinatura do autor:

Fabio Seiti Aguchiku

Assinatura do orientador:

Newton Maruyama

Catálogo-na-publicação

Aguchiku, Fabio Seiti

Especificação e verificação formal de requisitos para sistemas de tráfego aéreo / F. S. Aguchiku -- versão corr. -- São Paulo, 2018.
108 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Tráfego Aéreo (Automação; Gerenciamento) 2. Lógica Temporal
3. Especificação de Sistemas e Programas I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

AGRADECIMENTOS

Ao professor Newton Maruyama, pela paciência com que me explicava cada detalhe para a realização desta pesquisa e por me auxiliar a explorar as alternativas que eu encontrava.

A Fabio Kawaoka Takase, Eric Conrado de Souza e demais colegas da Atech pelo apoio técnico e por apresentarem o assunto que me motivou a iniciar este trabalho.

Ao meu amigo e colega Rafael Leme Costa, pelo esforço conjunto no desenvolvimento de nossas pesquisas.

Aos meus amigos e minhas amigas da área de inovação da Atech, pela convivência diária e por cobrir minhas ausências para que eu pudesse conduzir minha pesquisa.

A minha mãe e meu pai, Aurea e Sergio, minha irmã Patricia e meu irmão Douglas, pelo constante suporte e por sempre apoiar minhas decisões.

RESUMO

A evolução de sistemas de gerenciamento de tráfego aéreo é pesquisada para suportar o crescimento na demanda por transporte aéreo. Uma alternativa para essa evolução é o aumento no grau de automação. Os sistemas automatizados precisam ser tão seguros quanto os sistemas em operação atualmente. Com o uso de técnicas de especificação e verificação formal é possível avaliar os requisitos de sistemas. Neste trabalho, é proposto um ciclo de especificação formal, que consiste em um conjunto de diretrizes para aplicação de técnicas de métodos formais em requisitos escritos em linguagem natural. O resultado esperado da aplicação deste ciclo é um conjunto de requisitos escritos em linguagem natural verificados formalmente. O ciclo é composto pelas etapas: levantamento de requisitos do sistema e classificação em padrões de especificação; mapeamento dos requisitos para as linguagens de especificação formal LTL (*Linear Temporal Logic*) e CTL (*Computation Tree Logic*); verificação formal da especificação com o verificador NuSMV; ajustes na especificação baseada nos resultados da verificação; ajustes nos requisitos baseados nos ajustes na especificação. As diretrizes propostas são definidas com a análise da verificação formal do *Automated Airspace Concept* (AAC), padrões de especificação e diretrizes para uso do verificador NuSMV. Os resultados esperados são obtidos na aplicação do ciclo de especificação em dois estudos de caso. A principal contribuição do trabalho é o conjunto de diretrizes para elaboração de expressões escritas em linguagem de especificação formal baseadas em requisitos escritos em linguagem natural e que podem ser verificadas formalmente.

Palavras-chave: Sistemas de Gerenciamento de Tráfego Aéreo, Métodos Formais, Lógica Temporal, Sistemas Críticos.

ABSTRACT

Air traffic management systems evolution is being researched to support air transportation demand growth. An evolution alternative is system automation degree increase. Automated systems need to be as safe as current operating systems. It is possible to analyze system requirements with the application of formal specification and formal verification techniques. In this work, a specification cycle is proposed. The specification cycle is a set of guidelines to use formal method techniques on requirements written in natural language. The specification cycle application expected result is a set of formally verified requirements written in natural language. This cycle is comprised of the following stages: system requirements elicitation and specification pattern classification; requirements mapping to LTL (Linear Temporal Logic) and CTL (Computation Tree Logic) formal specification languages; specification formal verification using the NuSMV verifier; formal specification adjustment based on verification results; requirements adjustment based on formal specification adjustment. The proposed guidelines are defined with the Automated Airspace Concept (AAC) formal verification analysis, specification patterns and guidelines for the NuSMV formal verifier use. The expected results are accomplished in the specification cycle application on two study cases. The main contribution of this work is the set of guidelines applied to formulate formally verifiable expressions specified in formal specification languages based on system requirements written in natural language.

Keywords: Air Traffic Management Systems, Formal Methods, Temporal Logic, Safety-Critical Systems

LISTA DE ILUSTRAÇÕES

Figura 1	<i>Workflow</i> de aplicação de ciclo de especificação	16
Figura 2	Ciclo de Especificação	18
Figura 3	Resumo Esquemático da Coordenação entre Sistemas no AAC . . .	37
Figura 4	SCA – Visão de Topo	39
Figura 5	SCA – Visão de Perfil	39
Figura 6	Transição de valores da variável <code>aircraftPos</code>	75
Figura 7	Contra-exemplo do requisito A9	86

LISTA DE TABELAS

Tabela 1	Requisitos extraídos da descrição do conceito operacional do AAC (ZHAO; ROZIER, 2014)	45
Tabela 2	Requisitos extraídos de entrevistas com desenvolvedores do AAC (ZHAO; ROZIER, 2014)	45
Tabela 3	Requisitos extraídos das propriedades exigidas do conceito operacional do SATS (ABBOTT et al., 2004)	46
Tabela 4	Requisito A1	54
Tabela 5	Requisito A2 – Parte 1	56
Tabela 6	Requisito A2 – Parte 2	57
Tabela 7	Requisito A3	60
Tabela 8	Requisito A4	61
Tabela 9	Requisito A5	63
Tabela 10	Requisito A6a	64
Tabela 11	Requisito A6b	64
Tabela 12	Requisito A7	65
Tabela 13	Requisito A8	66
Tabela 14	Requisito A9	66
Tabela 15	Requisito A10	68
Tabela 16	Requisito A11	69
Tabela 17	Requisito B2	77
Tabela 18	Requisito B3	78
Tabela 19	Requisito B4	78
Tabela 20	Requisito B5a	79
Tabela 21	Requisito B5b	80
Tabela 22	Requisitos do AAC após aplicação do ciclo de especificação	92

LISTA DE ABREVIATURAS E SIGLAS

AAC	<i>Automated Airspace Concept</i>
ACC	<i>Area Control Center</i>
ADS-B	<i>Automatic Dependent Surveillance–Broadcast</i>
AMHS	<i>ATS Message Handling System</i>
AMM	<i>Automated Management Module</i>
APP	<i>Approaching Control Center</i>
ASAS	<i>Airborne Separation Assistance System</i>
ATC	<i>Air Traffic Control</i>
ATCo	<i>Air Traffic Controller</i>
ATFCM	<i>Air Traffic Flow and Capacity Management</i>
ATFM	<i>Air Traffic Flow Management</i>
ATM	<i>Air Traffic Management</i>
ATS	<i>Air Traffic Service</i>
CD&R	<i>Conflict Detection & Resolution</i>
CSP	<i>Communicating Sequential Processes</i>
CTL	<i>Computation Tree Logic</i>
FAF	<i>Final Approach Fix</i>
FDPS	<i>Flight Data Processing System</i>
FIR	<i>Flight Information Region</i>
FIXM	<i>Flight Information Exchange Model</i>
FO	<i>Flight Object</i>
FOS	<i>Flight Object Server</i>
FSM	<i>Finite State Machine</i>
IAF	<i>Initial Approach Fix</i>
ICAO	<i>International Civil Aviation Organization</i>
LN	<i>Linguagem Natural</i>
LTL	<i>Linear Temporal Logic</i>
MAHF	<i>Missed Approach Holding Fix</i>
MTCD	<i>Medium-Term Conflict Detector</i>
SATS	<i>Small Aircraft Transportation System</i>
SBVR	<i>Semantics of Business Vocabulary and Business Rules</i>
SCA	<i>Self Controlled Area</i>
SDPS	<i>Surveillance Data Processing System</i>

TCA	<i>Terminal Control Area</i>
TCAS	<i>Traffic Alert and Collision Avoidance System</i>
TMA	<i>Terminal Maneuvering Area</i>
TSAFE	<i>Tactical Separation Assurance Flight Environment</i>
VDM	<i>Vienna Development Method</i>
VSA	<i>Virtual Stationary Automata</i>

SUMÁRIO

1 INTRODUÇÃO	13
1.1 OBJETIVO	15
1.2 CONTRIBUIÇÕES	17
2 MÉTODOS FORMAIS	20
2.1 LÓGICA TEMPORAL	20
2.2 ESPECIFICAÇÃO FORMAL BASEADA EM ESTADOS	21
2.3 CONCORRÊNCIA	21
3 GERENCIAMENTO DE TRÁFEGO AÉREO	22
3.1 OPERAÇÃO DE VOO – CENÁRIO ATUAL	24
3.2 EVOLUÇÃO DOS SISTEMAS ATM	26
3.2.1 Sistemas relacionados à operação em rota	27
3.2.2 Sistemas relacionados à operação em área terminal	29
3.2.3 Sistemas relacionados à fase de planejamento	29
4 REVISÃO BIBLIOGRÁFICA	30
4.1 VERIFICAÇÃO FORMAL DE SISTEMAS ATM	30
4.2 MAPEAMENTO DE LINGUAGEM NATURAL PARA LINGUAGENS DE ESPECIFICAÇÃO FORMAL	32
5 CICLO DE ESPECIFICAÇÃO DE REQUISITOS	34
5.1 CONCEITO OPERACIONAL AAC – COORDENAÇÃO ENTRE SISTEMAS	36
5.1.1 Análise da Verificação Formal do AAC	37
5.2 CONCEITO OPERACIONAL SATS	37
5.2.1 Elementos Modelados	41
6 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA EM LINGUAGEM NATURAL E EM LINGUAGEM DE ESPECIFICAÇÃO FORMAL	43
6.1 ESPECIFICAÇÃO EM LINGUAGEM NATURAL	43
6.1.1 Estudo de Caso A – Coordenação AAC	44
6.1.2 Estudo de Caso B – SATS	46
6.2 ESPECIFICAÇÃO FORMAL	46
6.2.1 Modelagem do Sistema	47

6.2.2	Mapeamento de Linguagem Natural Controlada para LTL e CTL .	48
6.2.3	Estudo de Caso A – Coordenação AAC	50
6.2.4	Estudo de Caso B – SATS	69
6.2.5	Conclusões do Capítulo	82
7	VERIFICAÇÃO FORMAL	84
7.1	VERIFICAÇÃO COM NUSMV E RESULTADOS DE VERIFICAÇÃO	84
7.1.1	Estudo de Caso A – Coordenação AAC	86
7.1.2	Estudo de Caso B – SATS	87
7.2	AJUSTE – ESPECIFICAÇÃO FORMAL E MODELO	88
7.2.1	Estudo de Caso A – Coordenação AAC	88
7.2.2	Estudo de Caso B – SATS	90
7.3	AJUSTE – LINGUAGEM NATURAL	90
7.3.1	Estudo de Caso A – Coordenação AAC	90
7.3.2	Estudo de Caso B – SATS	91
7.3.3	Conclusões do Capítulo	93
8	RESULTADOS E DISCUSSÕES	95
8.1	ETAPA DE ESPECIFICAÇÃO EM LINGUAGEM NATURAL . . .	95
8.2	ETAPA DE ESPECIFICAÇÃO FORMAL	95
8.3	ETAPA DE VERIFICAÇÃO FORMAL E ANÁLISE DO RESULTADO DA VERIFICAÇÃO;.	97
8.4	ETAPA DE AJUSTE NA ESPECIFICAÇÃO FORMAL	98
8.5	ETAPA DE AJUSTE NA ESPECIFICAÇÃO EM LINGUAGEM NATURAL	99
8.6	APLICAÇÃO DO CICLO DE ESPECIFICAÇÃO NOS ESTUDOS DE CASO	99
9	CONCLUSÕES	101
9.1	TRABALHOS FUTUROS	102
	Referências	104

1 INTRODUÇÃO

Com a evolução dos recursos computacionais, há um aumento na quantidade de sistemas empregando componentes de *software* para a implementação de suas soluções. Espera-se que estes sistemas funcionem de maneira efetiva, eficiente, sem apresentar falhas e sejam desenvolvidos dentro das restrições de recursos do projeto (*e.g.* financeiros, humanos, prazos), especialmente tratando-se de sistemas críticos. Sistemas críticos são sistemas cuja falha resulta em perdas graves, como vidas humanas e financeiras. O uso de *software* está aumentando neste tipo de sistema (LARRUCEA; COMBELLES; FAVARO, 2013; BOWEN, 2000), o que reforça a necessidade de minimização de falhas, e consequente aumento na confiabilidade do sistema.

Dentro da classe de sistemas críticos, os sistemas de gerenciamento de tráfego aéreo (ATM – *Air Traffic Management*) são caracterizados por serem intensivos em *software*, portanto o aumento no poder computacional pode influenciar os responsáveis por sistemas ATM a aumentar a utilização de recursos computacionais nestes sistemas, resultando em um maior grau de automação, como no caso de Azzopardi (2015). É necessário garantir que os novos sistemas não apresentem falhas durante sua operação.

A detecção de erros de projeto em fases iniciais do desenvolvimento é uma alternativa para evitar falhas durante a operação do sistema. Além da confiabilidade, há a abordagem financeira de restringir erros de projeto a fases iniciais: estudos indicam que erros detectados em fases iniciais do desenvolvimento de *software* resultam em custos menores do que erros detectados em fases posteriores (MCCONNELL, 2004; STECKLEIN et al., 2004); por exemplo, um erro detectado durante a fase de desenvolvimento é corrigido com menor custo do que o mesmo erro, ou suas propagações, detectado durante a fase de testes. De maneira simplificada, o ciclo de desenvolvimento de um *software* pode ser separado nas fases de análise de requisitos, projeto, implementação, implantação e manutenção. Durante as fases de análise e projeto, modelos são utilizados para representar o sistema e durante a implementação, os modelos das fases anteriores são usados como base. O termo modelo é usado para referir-se a qualquer tipo de representação do sistema.

Existem diferentes técnicas para detecção de erros de projeto em fases iniciais do desenvolvimento de um sistema. Uma abordagem empregada em sistemas ATM são métodos formais de especificação e verificação de sistemas (ZHAO; ROZIER, 2014; ZAFAR, 2014; HANH; HUNG, 2007; CERONE; LINDSAY; CONNELLY, 2005). Estas técnicas utilizam representação matemática para especificação semântica única do sistema e verificação de propriedades do sistema.

Geralmente, os requisitos de um sistema são escritos em linguagem natural (LN).

Mesmo utilizando técnicas para estruturação de requisitos e utilização de recursos gráficos, esta forma de representação é informal. Fraser, Kumar e Vaishnavi (1991) afirmam existir uma lacuna entre linguagens de especificação formais e informais. Com o uso de especificação formal, é possível alcançar precisão e avaliar a completude da especificação, por meio de técnicas de verificação formal. Linguagens informais são ideais para a transmissão dos requisitos entre os envolvidos no projeto. Ambas possuem vantagens e desvantagens (FRASER; KUMAR; VAISHNAVI, 1991) e devem ser usadas de maneira complementar (FATWANTO, 2012). Por causa desta lacuna o uso de linguagens de especificação formal tem sido limitado. Dwyer, Avrunin e Corbett (1998) propõem uma abordagem para aumentar o uso de linguagens de especificação formal. Esta abordagem consiste em estabelecer padrões de especificação, uma técnica de solução recorrente que define padrões de propriedades de sistema e a representação de cada um destes padrões em linguagens de especificação formal. Autili et al. (2015) estendem esta abordagem apresentando ferramentas que podem preencher a lacuna entre linguagens formais e informais; especificações de propriedades de sistema em linguagem natural controlada são classificadas em padrões de especificação e um conjunto de padrões é proposto para linguagens que expressam propriedades em termos probabilísticos e de tempo real.

Neste trabalho, um método de especificação e verificação formal de requisitos de sistemas de tráfego aéreo é proposto. Este método corresponde a um ciclo de especificação. Este ciclo é composto por etapas e cada etapa possui um objetivo e um conjunto de diretrizes para alcançar o objetivo da etapa. O estabelecimento deste método e aplicação em estudos de caso está dentro do contexto de Dwyer, Avrunin e Corbett (1998) para o aumento do uso de técnicas de especificação e verificação formal. Deste modo, a principal contribuição deste trabalho é propor uma técnica para elaboração de expressões escritas em linguagem de especificação formal a partir de requisitos escritos em linguagem natural e verificação formal destas expressões com uma ferramenta de verificação formal.

A elaboração de expressões possui duas fases. Na primeira fase, os requisitos escritos em linguagem natural são classificados em um padrão de especificação e são reescritos em linguagem natural controlada de acordo com o padrão de especificação em que o requisito foi classificado. Na segunda fase, os requisitos em linguagem natural controlada são mapeados para expressões especificadas em linguagens de especificação formal; o mapeamento é definido pelo padrão de especificação. Este processo é feito manualmente.

Existem trabalhos que analisam requisitos de sistemas ATM com métodos formais (ZHAO; ROZIER, 2014; MUÑOZ; CARREÑO; DOWEK, 2006). O foco destes trabalhos é a análise do sistema e não no processo para efetuar a análise. Portanto, nestes trabalhos expressões em linguagem de especificação formal referentes a propriedades do sistema

analisado são propostas, porém não é apresentado como foram elaboradas. A análise dos resultados da verificação formal são específicos para os requisitos dos sistemas estudados, portanto é uma análise aplicável para o sistema estudado. Portanto devido ao foco dado por estes trabalhos, duas lacunas são observadas: uma lacuna no mapeamento de linguagem natural para linguagem de especificação formal e uma lacuna na análise dos resultados da verificação formal e como esta análise pode ser usada para o ajuste de requisitos do sistema. O método de especificação proposto neste trabalho pode ser usado para preencher estas lacunas.

As diretrizes de cada etapa do ciclo são propostas a partir de:

- análise do trabalho apresentado por Zhao e Rozier (2014);
- conceito e aplicação de padrões de especificação em requisitos escritos em linguagem natural controlada;
- diretrizes para uso da ferramenta de verificação formal NuSMV (CIMATTI et al., 2002; ARCAINI; GARGANTINI; RICCOBENE, 2017).

O resultado esperado da aplicação do ciclo de especificação em requisitos de um sistema é obter um conjunto de requisitos escritos em linguagem natural ajustados devido a um processo de verificação formal. O ciclo de especificação proposto é aplicado em dois sistemas como estudos de caso. O critério para avaliação do ciclo de especificação, e do resultado deste trabalho, é analisar se foi possível obter um conjunto de requisitos escritos em linguagem natural e verificados formalmente dos sistemas dos estudos de caso. Um dos sistemas é o mesmo estudado em Zhao e Rozier (2014), logo outro critério para avaliação do ciclo de especificação é a comparação do resultado de verificação formal dos dois trabalhos. Para efetuar esta comparação, o mesmo conjunto de requisitos e o mesmo modelo do sistema propostos em Zhao e Rozier (2014) serão usados neste trabalho.

A aplicação de uma iteração do ciclo de especificação pode ser representada pelo *workflow* apresentado na figura 1. Após a aplicação, o ciclo pode ser aplicado ao conjunto de requisitos resultantes da iteração anterior, ou um novo conjunto de requisitos pode ser considerado. A aplicação do ciclo nos estudos de caso alcançou o objetivo proposto.

1.1 Objetivo

O objetivo deste trabalho é estabelecer um método composto por técnicas de especificação e verificação formal para verificação de requisitos de sistemas de tráfego aéreo. Este método tem como entrada um conjunto de requisitos do sistema escritos em linguagem natural e

Figura 1 - Workflow de aplicação de ciclo de especificação

resulta em um conjunto de requisitos que passaram por um processo de verificação formal. O método propõe o uso de técnicas de especificação e verificação formal para avaliar se ajustes são necessários no conjunto de requisitos do sistema.

O resultado esperado desta pesquisa é o estabelecimento de um método que resulte em requisitos especificados em linguagem natural ajustados devido à verificação formal dos requisitos de um sistema. Um conjunto de requisitos é especificado formalmente, a especificação é verificada formalmente e ajustes baseados nos resultados da verificação são feitos nos requisitos do sistema.

1.2 Contribuições

A principal contribuição deste trabalho é o estabelecimento de diretrizes para elaboração de expressões escritas em linguagem de especificação formal a partir de requisitos escritos em linguagem natural que podem ser verificadas formalmente. Estas diretrizes correspondem às diretrizes de cada etapa do ciclo de especificação proposto. As etapas do ciclo são apresentadas a seguir.

1. Etapa de especificação em linguagem natural – Interpretação da especificação do sistema em linguagem natural, que envolve coleta de requisitos do sistema e sua classificação.
2. Etapa de especificação formal – Especificação formal dos requisitos do sistema baseada nos requisitos escritos em linguagem natural.
3. Etapa de verificação formal e análise do resultado da verificação – Verificação formal da especificação formal dos requisitos do sistema com o uso de uma ferramenta de verificação formal e análise do resultado da verificação formal. Cada requisito pode ser verificado individualmente. O requisito pode ser avaliado “verdadeiro” ou “falso”. O resultado “verdadeiro” expressa que o requisito é satisfeito pela especificação do sistema e o resultado “falso” expressa o contrário. O resultado desejado para todos requisitos é “verdadeiro”. Caso o resultado seja “verdadeiro”, deve-se avaliar se o resultado não é um falso positivo e se for “falso” deve-se avaliar se o requisito está errado, se a especificação do sistema está errada ou se ambos estão errados. O termo errado é empregado no sentido do requisito ou a especificação não ser coerente com o conceito do sistema.
4. Etapa de ajuste na especificação formal – Ajustes na especificação formal baseados na análise do resultado da verificação formal.

5. Etapa de ajuste na especificação em linguagem natural – Ajustes da especificação em linguagem natural baseados nos ajustes feitos na especificação formal.

Uma representação gráfica do ciclo é apresentada na Figura 2. Os números na figura representam a etapa do ciclo de especificação.

Figura 2 - Ciclo de Especificação

Os requisitos são especificados por meio de classificação em padrões de especificação. Outra contribuição deste trabalho corresponde à definição de padrões de especificação e elaboração de novos padrões, baseados em características de sistemas de tráfego aéreo.

As diretrizes de modelagem de sistemas propostas para o ciclo de especificação são outra contribuição do trabalho. A técnica de verificação formal escolhida utiliza proposições especificadas formalmente e um modelo. O objeto de estudo deste trabalho corresponde aos requisitos do sistema, que são mapeados para proposições. As diretrizes são usadas para o caso em que não existe um modelo do sistema.

As diretrizes são definidas e avaliadas por meio de estudos de caso. A apresentação da aplicação do ciclo de especificação em estudos de caso é uma contribuição deste trabalho, visto que o foco de outros trabalhos que aplicam métodos formais em sistemas ATM são os resultados da verificação e não o processo que antecede a verificação.

O objetivo desta pesquisa será alcançado com a definição de uma tática para cada uma das etapas do ciclo. O termo tática se refere a um conjunto de diretrizes para alcançar o objetivo da etapa. Para a definição destas diretrizes são conduzidos estudos que compreendem análise das características de sistemas de gerenciamento de tráfego aéreo para uso da linguagem de especificação formal mais adequada; mapeamento de requisitos em linguagem natural para proposições em linguagem de especificação formal por meio de padrões de especificação; e técnicas de verificação. As diretrizes propostas são baseadas

em trabalhos como Zhao e Rozier (2014)), Dowek, Munoz e Carreno (2004), em que um conjunto de requisitos do sistema é mapeado para linguagens de especificações formais.

Este ciclo pode ser usado para especificações num mesmo nível de abstração, *i.e.*, cada iteração do ciclo é aplicada em requisitos num mesmo nível de abstração, ou pode ser usado para o refinamento do modelo, com requisitos mais próximos do nível de implementação.

No capítulo 2, métodos formais considerados para esta pesquisa são apresentados e no capítulo 3 o cenário atual de sistemas ATM e evoluções propostas por indústria e academia são apresentados, para a descrição do panorama destes tipos de sistema e levantamento de suas características.

2 MÉTODOS FORMAIS

2.1 Lógica Temporal

A lógica temporal compreende símbolos, regras e sintaxes para representação e análise de proposições expressas com termos temporais. Geralmente são utilizadas em técnicas de verificação formal para especificação de propriedades de um sistema.

Inicialmente, dois tipos de lógica temporal foram propostos (EMERSON; HALPERN, 1986): lógica temporal linear (PNUELI, 1977), do inglês *linear-time temporal logic*, e lógica temporal de ramificação (BEN-ARI; PNUELI; MANNA, 1983), do inglês *branching-time temporal logic*. *Linear Temporal Logic* (LTL) é um exemplo de lógica temporal linear e *Computation Tree Logic* (CTL) (CLARKE; EMERSON; SISTLA, 1986) é um exemplo de lógica temporal de ramificação.

LTL é usada para especificação de proposições sobre um caminho de execução. Operadores são definidos para a especificação de fórmulas proposicionais sobre a execução de um sistema. Estas operações são:

- $\Box$ ou G (*Globally*) – a proposição deve ser verdadeira em todos caminhos de execução.
- $\Diamond$ ou F (*Future*) – a proposição será verdadeira em algum estado no futuro.
- $\bigcirc$ ou X (*neXt*) – a proposição será verdadeira no próximo passo do caminho de execução.
- U (*Until*) – uma expressão deve ser verdadeira até outra proposição ser verdadeira.

CTL possui a capacidade de especificar proposições sobre um caminho ou sobre todos caminhos de execução. A linguagem usa os mesmos operadores usados em LTL para a especificação de proposições sobre a execução de um caminho e usa os outros operadores para a especificação de proposições sobre a quantidade de caminhos em que a proposição deve ser satisfeita:

- A (*All*) – a proposição deve ser verdadeira em todos os caminhos de execução.
- E (*Eventually*) – a proposição deve ser verdadeira em ao menos um caminho de execução.

Expressões em LTL e CTL podem ser usadas para verificação de modelos. Ferramentas que utilizam a técnica de verificação de modelos (*model checking*) verificam se proposições são satisfeitas por um modelo. O verificador SPIN (HOLZMANN, 2003) modela o sistema

com a linguagem de modelagem Promela e o verificador NuSMV (CIMATTI et al., 2002) utiliza uma linguagem própria para a modelagem do sistema em uma máquina de estados finita (FSM – *Finite State Machine*). Ambas ferramentas possuem recursos para especificação de proposições em LTL e CTL e verificação de modelos.

2.2 Especificação Formal Baseada em Estados

Kaur, Gulati e Singh (2012) apresentam um estudo comparativo entre três linguagens de especificação formal baseada em estados. Estas linguagens especificam variáveis que compõem o estado do sistema e operações que podem alterar o valor das variáveis. Como o estado do sistema é definido pelo valor das variáveis estas operações alteram o estado do sistema. É possível especificar invariantes do sistema com estas linguagens; estas invariantes definem os estados válidos do sistema. Das linguagens comparadas, a notação Z (SPIVEY, 1989) é a mais abstrata, podendo especificar o maior número de sistemas. Um dos verificadores mais consolidados da linguagem é o Z/EVES (SAALTINK, 1997), um provador de teoremas (*Theorem Prover*). Esta técnica de verificação envolve processos menos automatizados e que exigem mais interação do usuário se comparado com técnicas de verificação de modelos.

2.3 Concorrência

Clarke, Emerson e Sistla (1986) propõem o uso de lógica temporal para verificação de concorrência em sistemas. Existem especificações formais concebidas especialmente para o tratamento de concorrência em sistemas. CSP (*Communicating Sequential Processes*) (HOARE, 1978) é uma teoria de concorrência representada por uma álgebra de processos que especifica processos concorrentes. A linguagem formal baseada em CSP especifica processos compostos por eventos e canais de comunicação. Processos concorrentes podem sincronizar em eventos e transferir dados por canais. A especificação pode ser usada para verificar se a ordem em que eventos são executados é a esperada ou se o sistema possui *deadlocks*, por exemplo. No geral, CSP realiza verificações com relação à corretude do sistema, porém não a seu desempenho.

Um dos verificadores mais consolidados é o FDR (GIBSON-ROBINSON et al., 2014), um verificador de refinamento (*refinement checker*) que utiliza a técnica de verificação de modelos. No contexto de CSP, o processo A refina o processo B se todo comportamento possível pelo processo A é permitido pelo processo B. O comportamento é definido pelo modelo de semântica denotacional empregado.

3 GERENCIAMENTO DE TRÁFEGO AÉREO

Este capítulo descreve e analisa o panorama de sistemas de tráfego aéreo e pesquisas para evoluções na área. Pontos desta análise serão usados para definir parte das diretrizes do ciclo de especificação.

O transporte aéreo global pode ser visto como um sistema naturalmente distribuído composto por espaço aéreo, aeronaves e condições meteorológicas. O gerenciamento do tráfego aéreo é alcançado com o uso de sistemas, regras e profissionais responsáveis pela garantia de organização, interoperabilidade, confiabilidade, eficiência e segurança do sistema de transporte aéreo (AZZOPARDI, 2015).

Os sistemas ATM podem ser classificados de acordo com o horizonte temporal de sua aplicação. Uma classe de sistemas é o Gerenciamento de Fluxo de Tráfego Aéreo (ATFM – *Air Traffic Flow Management*, ou ATFCM – *Air Traffic Flow and Capacity Management*). Esta classe de sistemas é responsável pelo planejamento pré-voo e opera num nível estratégico, que precede eventos no intervalo de 1 hora a meses. Os sistemas desta classe buscam a utilização mais eficiente do espaço aéreo, atendendo aos requisitos de segurança. A classe de sistemas de Controle de Tráfego Aéreo (ATC – *Air Traffic Control*) opera no nível tático, que precede eventos em 20 minutos. Estes sistemas são executados durante a operação do voo. Devido ao horizonte temporal, as ações desta classe de sistemas priorizam a segurança das operações aéreas; a eficiência da operação fica em segundo plano.

Atualmente, os sistemas ATM são centralizados em seres humanos, *i.e.*, o processo de tomada de decisão possui pessoas como elemento central (AZZOPARDI, 2015). Neste cenário, os sistemas ATFM e ATC são ferramentas de suporte a decisão. Em sistemas ATC, o elemento central no processo de tomada de decisão é o controlador de tráfego aéreo (ATCo – *Air Traffic Controller*). Diferentes autores concordam que a principal função do ATCo, e de serviços de tráfego aéreo em geral, é evitar o conflito de aeronaves (POPESCU, 2013; BAYEN, 2004; OLIVEIRA, 2007; PIZZO, 2008). Duas ou mais aeronaves estão em conflito se a distância (vertical e horizontal) entre elas é menor do que um valor mínimo estabelecido por regulamentos do espaço aéreo. Outro termo utilizado para se referir ao estado de conflito é perda de separação. O objetivo de sistemas ATM é evitar a perda de separação entre aeronaves, desde a fase de planejamento do voo até o estacionamento da aeronave no aeródromo de chegada.

Devido ao aumento no número de voos e a expectativa de crescimento na utilização do espaço aéreo (AZZOPARDI, 2015), pesquisadores de indústria e academia estão procurando soluções para o aumento da eficiência da utilização do espaço aéreo.

A abordagem da indústria, liderada pelas iniciativas NextGen (FEDERAL AVIATION ADMINISTRATION, 2016) dos EUA e SESAR (SESAR JOINT UNDERTAKING, 2016) da União Europeia, consiste na evolução de procedimentos e algoritmos, e na melhoria da interoperabilidade entre sistemas. Nesta abordagem, os sistemas continuam sendo ferramentas de suporte a decisão, mantendo operadores humanos no centro do processo de tomada de decisão (AZZOPARDI, 2015), embora existam programas com o objetivo de aumentar a automação de processos. Esta automação reduz a participação humana e é observada em iniciativas como os sequenciadores automáticos de decolagem (DMAN – *Departure Manager*) e chegada (AMAN – *Arrival Manager*) (AZZOPARDI, 2015). Padrões estão sendo estabelecidos para a melhoria da interoperabilidade entre sistemas (MAYER; FRÖHLICH, 2014) visando o aumento da difusão de dados. A indústria busca o aumento da eficiência na utilização do espaço aéreo melhorando a consciência situacional de operadores e diminuindo sua carga de trabalho. O ATCo continua sendo figura central na tomada de decisão, operando os sistemas em centros de controle.

No meio acadêmico, pesquisas com abordagens semelhantes são encontradas, porém existem trabalhos que repensam as operações e estruturas dos sistemas ATM. Estes trabalhos ressaltam que o número de aeronaves que um ATCo consegue controlar é limitado, portanto o ATCo é um gargalo do sistema. Para a resolução deste gargalo, as pesquisas buscam a automação dos sistemas de modo a minimizar ou remover a participação humana no processo de tomada de decisão. Nesta abordagem, o sistema ATM pode ser instalado em aeronaves, o que descentraliza o sistema e aumenta sua escalabilidade. Pesquisas deste tipo são caracterizadas por propostas de alto grau de automação e descentralização dos sistemas.

As pesquisas do meio acadêmico propõem sistemas que aplicam técnicas matemáticas – como por exemplo *Virtual Stationary Automata*, autômatos híbridos, campos de força – para melhoria dos sistemas. A indústria segue a mesma abordagem, mas define também procedimentos que devem ser tomados caso estes sistemas apresentem falhas durante a operação. Para lidar com estas falhas durante a operação, níveis de redundância são previstos para a operação do sistema. O sistema de tráfego aéreo sempre utilizou controladores humanos para alcançar o uso seguro do espaço aéreo. Logo, o último nível de redundância costuma ser os ATCos. Por isso, a indústria continua a desenvolver sistemas que mantêm o ATCo no centro do processo de tomada de decisão, mesmo que isto resulte num avanço mais lento com relação a aplicação de técnicas matemáticas, em comparação com a academia.

A seguir, um cenário geral de operação aérea de um voo comercial é apresentado, utilizada para discussão de prováveis evoluções de sistemas ATM.

3.1 Operação de Voo – Cenário Atual

Esta seção descreve como uma operação aérea é conduzida com os sistemas atuais, contextualizando a operação nos elementos do espaço aéreo. A descrição se baseia em Bayen (2004), que descreve as operações no espaço aéreo dos EUA, e em Pizzo (2008) e Oliveira (2007), que descrevem as operações no espaço aéreo brasileiro.

Antes do voo, o piloto ou a companhia aérea deve planejar o voo e preencher um plano de voo. Este plano deve possuir informações como identificação, tipo de aeronave, velocidade e altitude de cruzeiro, rota do voo, pontos e aerovias que serão usados durante a navegação da aeronave. A rota de um voo é composta pelo aeródromo de origem, os centros por onde irá passar e o aeródromo de destino (BAYEN, 2004). Neste instante, o voo está na fase de planejamento.

Antes da decolagem, o plano é emitido para os sistemas ATM. Com base no conjunto de planos de voo emitidos e em dados meteorológicos, o sistema ATFM verifica se não há problemas, como conflito de voos ou excesso em uma determinada região, em solo e em rota, propondo alterações necessárias.

Após a resolução dos problemas identificados, o plano de voo atualizado é emitido para o sistema ATC do centro de controle de tráfego aéreo do aeródromo de origem. Este centro realiza a extração da rota e envia uma mensagem para os sistemas ATC dos centros por onde o voo irá passar, com informações necessárias para o rastreamento da aeronave.

A partir do momento em que está pronto para deixar o portão no aeródromo de origem até o momento em que estaciona no portão do aeródromo de destino, o voo estará sob controle de tráfego aéreo. Este controle é feito por meio da comunicação via rádio entre piloto e ATCo.

No instante em que está no portão até a decolagem, o voo está sob controle da torre de controle. A ação do controlador corresponde a autorização do voo ao taxiamento, orientação do piloto durante o taxiamento e autorização de decolagem. Neste intervalo o voo está sob operação em solo.

Após a decolagem, a aeronave estará em espaço aéreo. Considerando aeroportos de médio a grande porte, a aeronave estará num volume do espaço aéreo de alta densidade conhecido como TMA (*Terminal Maneuvering Area*) ou TCA (*Terminal Control Area*). Nesta região, o controle da aeronave é feito pelo Centro de Aproximação (APP – *Approaching Control Center*) cuja função é o direcionamento das aeronaves entre a decolagem e a entrada em rota; e organização das aeronaves para o processo de aterrissagem. Os controladores dessa região devem tomar ações para prevenção de conflito de aeronaves e orientação durante o processo de subida e descida. Nestas condições o voo está sob operação em área terminal.

Após a entrada em rota, o voo está sob controle de um ACC (*Area Control Center*). As autoridades aéreas costumam segmentar o espaço aéreo em regiões conhecidas como FIR (*Flight Information Region*): cada ACC é responsável por uma FIR. As FIR são segmentados em setores aéreos. Cada setor possui ao menos um controlador, que é o responsável pelo controle dos voos naquele setor. Em regras estabelecidas pela ICAO (*International Civil Aviation Organization*), as aeronaves devem manter uma separação mínima na horizontal e na vertical. A função do controlador é manter esta separação. Outra função é a realização do processo de coordenação e transferência do controle do voo para setores e centros adjacentes. Nestas condições, o voo está em rota.

Enquanto em rota, o voo é transferido entre controladores de setores de centros ACC, até chegar ao TMA do aeródromo de destino, quando seu controle passa novamente a um APP, que deve encaminhá-lo para o processo de aterrissagem. Nesta condição o voo está sob operação em área terminal.

Após o APP, o controle da aeronave é transferido para a torre de controle, que autoriza a aterrissagem da aeronave, entrada no pátio e estacionamento da aeronave. Neste intervalo o voo está sob operação em solo.

Resumidamente, um voo é controlado inicialmente por uma torre de controle. Este controle é transferido para um APP após a decolagem e a entrada em um TMA. O APP transfere o controle a um ACC. O ACC transfere o controle do voo quando este deixa a FIR de sua responsabilidade. Ao chegar ao TMA do aeródromo de destino, o controle é transferido para o APP responsável. O APP transfere o controle para a torre de controle quando a aeronave se aproxima do aeródromo de destino. A torre de controle orienta a aeronave durante o processo de aterrissagem até o estacionamento da aeronave. Existem aeródromos de pequeno porte que não possuem radar, torre de controle ou não estão inseridos em um TMA. Nestes casos, a operação do voo segue outras regras definidas por serviços de tráfego aéreo.

Alguns dos sistemas que suportam as operações de controle são:

- **FDPS (*Flight Data Processing System*)**: Sistema de suporte a decisão. Cada centro possui um sistema FDP com armazenamento local e distribuído entre todas as posições de controle. O sistema realiza a extração de rota dos planos de voo que recebe. Este sistema registra quais voos estão sob controle naquela posição, realizando a coordenação e transferência automatizada do controle de voos entre setores e centros adjacentes. O sistema apresenta a posição dos voos que estão sob controle na posição, por meio de dados de radar e extrapolação de dados. Recebe os dados de tempo real provenientes do SDPS (*Surveillance Data Processing System*) e troca mensagens de coordenação com outros sistemas provenientes do AMHS (*ATS Message Handling*

System) (ATS – *Air Traffic Service*). O SDPS recebe dados de diferentes radares e gera uma síntese, *i.e.*, coleta informações de vigilância de fontes diferentes referentes a um único voo e disponibiliza uma informação única.

- ***Conflict Detector***: Ferramenta de suporte que detecta conflitos num período de tempo determinado. Por exemplo, o MTCD (*Medium-Term Conflict Detector*) detecta conflitos a médio prazo (cerca de 20 a 30 minutos). Os dados de entrada são o plano de voo e as detecções em tempo real. Com base nestes dados, o sistema apresenta voos que entrarão em conflito. Pode ser uma ferramenta complementar ao FDPS.

3.2 Evolução dos Sistemas ATM

Para atender ao crescimento da demanda por voos, é necessário aumentar a eficiência na utilização do espaço aéreo mantendo, ou aumentando, os níveis de segurança atuais. Existem ao menos três tipos de abordagem para a evolução dos sistemas atuais: melhoria de procedimentos e algoritmos; aumento do nível de interoperabilidade e difusão de dados entre os sistemas; aumento do grau de automação dos sistemas.

A indústria está promovendo o conceito de *Flight Object* (FO) e *Flight Object Server* (FOS) para o aumento da interoperabilidade entre sistemas, principalmente entre os FDPS. As mensagens seriam transmitidas usando um formato padrão correspondente ao *Flight Object*, que possui todas informações sobre um voo, e seriam centralizadas no FOS, para que todos envolvidos numa operação aérea tenham acesso à mesma informação e possam fornecer dados ao servidor. A especificação do NextGen para o formato da mensagem é o FIXM (*Flight Information Exchange Model*) que especifica o comportamento do FO por meio de máquinas de estado, definindo quais atualizações podem ser feitas dependendo do estado do FO. As máquinas de estado dos FO deve ser sincronizada entre FOS e clientes. No trabalho de Kharchenko et al. (2015) o conceito e as consequências da implementação do FOS são estudados, enquanto Mayer e Fröhlich (2014) apresentam uma proposta de implementação.

Uma proposta consolidada no meio acadêmico e na indústria é a utilização de um novo sistema de vigilância baseado em satélite. A tecnologia usada é o ADS-B (*Automatic Dependent Surveillance–Broadcast*). A aeronave adquire informações de vigilância, como posição e velocidade, via satélite e realiza o *broadcast* desta informação automaticamente. A informação é adquirida por meio de radares secundários, que recebem o sinal de *broadcast*. Nesta solução, uma aeronave fornece informações para outras aeronaves, diferente dos radares primários utilizados atualmente. No trabalho de Strohmeier et al. (2014), o ADS-B

é analisado e são levantados problemas que devem ser resolvidos para a utilização da tecnologia. Estes problemas estão relacionados a segurança (*safety* e *security*) do meio de comunicação. Strohmeier, Lenders e Martinovic (2015) apresentam diferentes tipos de interferência que são testados para avaliação da segurança do ADS-B.

3.2.1 Sistemas relacionados à operação em rota

Nesta seção são apresentados trabalhos com foco na operação em rota. Também são apresentados trabalhos cujo foco não se restringe apenas à operação em rota, e podem ser aplicados para operações em áreas terminais.

O conceito de *Free Flight* foi proposto para o aumento da eficiência na utilização do espaço aéreo. Neste conceito, as aeronaves têm a possibilidade de voar livremente pelo espaço aéreo sem estar restritas a uma aerovia. Pesquisas estão sendo conduzidas para a viabilização deste conceito. Uma linha de pesquisa endereça o problema de Detecção e Resolução de Conflitos (CD&R – *Conflict Detection & Resolution*) num espaço aéreo *Free Flight*.

No campo de CD&R, diferentes pesquisas foram desenvolvidas visando a automação do processo, removendo esta responsabilidade do ATCo. Lalish e Morgansen (2012) classificam estes algoritmos seguindo diferentes critérios. Um dos critérios é relativo ao modelo matemático adotado. Um dos métodos emprega manobras pré-definidas, utilizando autômatos híbridos para modelar a dinâmica das aeronaves dependendo do tipo de manobra que ela está realizando. Outro método utiliza otimização para definição de trajetórias ótimas para aeronaves, modelando uma função de custo. Um terceiro método emprega campos de força para separação de aeronaves. Outro critério de classificação é o grau de centralização do algoritmo. Em algoritmos centralizados, o algoritmo utiliza dados de todas aeronaves e o processamento é feito num nó central. A abordagem descentralizada utiliza dados de aeronaves localizadas dentro de um determinado alcance. Cada aeronave executa seu algoritmo para prevenção de conflitos. A abordagem distribuída é mista; utiliza dados de um grupo de aeronaves e cada elemento executa seu próprio algoritmo de controle. Lalish e Morgansen (2012) propõem um algoritmo distribuído, em duas fases: na primeira uma manobra para resolução de conflito é projetada usando técnicas de otimização e na segunda fase, técnicas de campo de força são empregadas de maneira pró-ativa para manter o espaço aéreo livre de conflitos.

Na pesquisa feita por Durand e Barnier (2015), um algoritmo de resolução de conflitos descentralizado autônomo é testado. Este algoritmo seria processado nas aeronaves que evitariam conflitos autonomamente. É observado que este algoritmo não consegue

encontrar soluções ao introduzir restrições de velocidade das aeronaves. Este algoritmo é comparado ao algoritmo evolucionário centralizado, que utiliza informação centralizada para coordenação das manobras de voo. O algoritmo centralizado consegue encontrar soluções sob condições que o algoritmo descentralizado não consegue, sugerindo que o ATM precisa de uma coordenação centralizada para prevenção de conflitos, visto que há restrições de velocidade no espaço aéreo.

Uma proposta de sistema que mantém um grau de centralização é o *Automated Airspace Concept* (AAC) (ERZBERGER, 2004; ERZBERGER; HEERE, 2010; ERZBERGER; LAUDERDALE; CHU, 2012). Este conceito utiliza três sistemas para detecção e resolução de conflitos: o sistema *AutoResolver*, o sistema TSAFE (*Tactical Separation Assurance Flight Environment*) e o TCAS (*Traffic Alert and Collision Avoidance System*). O *AutoResolver* e o TSAFE são sistemas desenvolvidos para o AAC, enquanto o TCAS é um sistema embarcado em aeronaves que detecta colisão iminente e emite avisos com manobras para as aeronaves envolvidas na previsão de colisão. Dependendo do intervalo de tempo em que o conflito foi detectado, um dos sistemas é responsável pela resolução do conflito. Este conceito é candidato a adoção pelo NextGen (ZHAO; ROZIER, 2014) e seria integrado ao sistema ATC dos centros de controle dos EUA. Este conceito será explorado nos capítulos 4 e 5.

Visando a redução das responsabilidades do ATCo, Bayen (2004) modelou as manobras de aeronaves para prevenção de conflitos com um autômato híbrido e modelou o processo de tomada de decisão de um controlador aéreo com uma função de custo para determinar qual seria a manobra escolhida pelo controlador. Estes modelos foram simulados e comparados com dados reais para validação dos modelos.

O trabalho de Brown (2006) aplica o conceito de *Virtual Stationary Automata* (VSA) em ATC. Este conceito consegue emular uma estrutura centralizada e fixa com elementos móveis conectados em uma rede. Com esta técnica, é possível executar algoritmos centralizados sem necessidade de uma instalação central. O algoritmo centralizado é emulado em elementos descentralizados do sistema. O espaço aéreo é modelado por meio de uma estrutura do tipo grafo. Neste modelo, as aeronaves percorrem os eixos do grafo que representam trechos do espaço aéreo, e recebem autorização para passar pelos nós que ligam os eixos. O VSA executa um algoritmo de controle para garantir que não há mais de uma aeronave em cada trecho do espaço aéreo e que no máximo uma aeronave está autorizada a realizar a transição de um trecho para outro. O conceito de VSA é usado para que um conjunto de aeronaves localizados numa região geográfica emulem o centro que executa o algoritmo de controle. Caso não haja aeronaves naquela região, o centro não é emulado.

3.2.2 Sistemas relacionados à operação em área terminal

Existem trabalhos específicos para as áreas terminais de controle (TMA). O trabalho de Oliveira e Cugnasca (2002) utiliza um autômato híbrido para modelagem do comportamento de aeronaves entrando numa área terminal. O espaço aéreo é modelado como uma sequência de pontos de referência. Cada ponto de referência possui uma janela de transferência; a aeronave deve estar dentro desta janela ao passar pelo ponto de referência. Em trabalho posterior, Oliveira (2007) utilizou Rede de Petri estocástica e dinamicamente colorida para avaliação do risco de introduzir o procedimento ASAS (*Airborne Separation Assistance System*) em áreas terminais de alta densidade de voo.

Os TMA correspondem a áreas terminais de controle responsáveis pelo controle de aeronaves em áreas em volta de aeroportos de grande porte. Para aeródromos de pequeno porte, Abbott et al. (2004) apresentam o conceito de *Small Aircraft Transportation System* (SATS). O objetivo do SATS é o aumento do volume de operações aéreas de aviação geral em aeródromos de pequeno porte que não possuem radares e torre de controle. Para alcançar este objetivo, regras e procedimentos são estabelecidas para a operação nestes aeródromos. As regras e procedimentos devem ser seguidas em área auto-controlada (SCA – *Self Controlled Area*) que corresponde a superfície em volta destes aeródromos. O piloto da aeronave é o responsável pela separação entre sua aeronave e outras e um sistema automatizado centralizado é responsável pela autorização de entrada de aeronaves na SCA. Esse sistema é denominado de *Airport Management Module* (AMM). Muñoz, Carreño e Dowek (2006) apresentam um estudo para a verificação formal do conceito operacional do SATS, analisando perspectivas diferentes do conceito com diferentes técnicas. Este sistema será discutido nos capítulos 4 e 5.

3.2.3 Sistemas relacionados à fase de planejamento

Em Popescu (2013), são analisados modelos para o projeto da geometria de rotas aéreas. Neste trabalho, o autor afirma que o projeto do espaço aéreo tem como objetivo a capacidade do espaço aéreo; segurança e carga de trabalho do controlador são restrições ao projeto. O autor propõe modelos para a introdução de segurança e carga de trabalho aos objetivos do projeto do espaço aéreo. O estudo modela e simula o controle de custo de uma operação aérea para análise da carga de trabalho de um controlador e analisa quais geometrias de rotas aéreas apresentam segurança inerente, *i.e.*, emitem menos avisos TCAS.

4 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta trabalhos utilizados para a definição das táticas e diretrizes de cada etapa do ciclo de especificação. Na seção 4.1 são discutidos trabalhos que utilizam especificações formais para a especificação de sistemas de tráfego aéreo e verificam formalmente a especificação. O capítulo 2 introduziu linguagens de especificação formal discutidas neste texto e o capítulo 3 apresentou pesquisas feitas em sistemas ATM. Esta seção apresenta trabalhos com objetivos mais próximos ao objetivo desta pesquisa, apresentado na seção 1.1. Na seção (4.2) são apresentados trabalhos que definem técnicas para representação de requisitos de sistemas em linguagens de especificação formal.

4.1 Verificação Formal de Sistemas ATM

O trabalho desenvolvido por Zhao e Rozier (2014) realiza a verificação formal do sistema descrito em Erzberger, Lauderdale e Chu (2012), o *Automated Airspace Concept*. O AAC utiliza 3 sistemas diferentes para assegurar a separação entre aeronaves no espaço aéreo: o sistema *AutoResolver*, o sistema TSAFE e o TCAS. O sistema *AutoResolver* é responsável pela detecção de conflitos a médio prazo (horizonte temporal correspondente ao período entre 2 e 20 minutos) e pela proposição de manobras para resolução de conflitos. Caso o controlador concorde com a proposta, a solução é transmitida para a aeronave. O sistema TSAFE detecta conflitos a curto prazo, cujo horizonte temporal padrão é inferior a 2 minutos, e possui 2 modos de operação. Caso a previsão corresponda a um período acima de um limiar pré-definido, 1 minuto por padrão, o sistema apresenta uma solução para o controlador, que deve aprová-la antes do envio às aeronaves, num processo semelhante ao do sistema *AutoResolver*. Caso a previsão esteja abaixo do limiar, o controle da aeronave é transferido para o sistema TSAFE que define uma resolução e a envia para a aeronave, sem necessidade de autorização do controlador. Este é um processo automatizado que visa a segurança do espaço aéreo e reduz a carga de trabalho do controlador. O TCAS é um sistema embarcado em aeronaves para detecção de colisão iminente e define uma solução coordenada para as aeronaves envolvidas.

A verificação de Zhao e Rozier (2014) é feita com relação à coordenação destes 3 sistemas. O sistema é modelado em uma máquina de estados finita, requisitos são especificados como proposições em LTL e CTL e verifica-se formalmente se as proposições são satisfeitas pelo modelo com o verificador NuSMV.

Yousaf, Zafar e Khan (2010) apresentam a especificação formal de procedimentos de decolagem utilizando VDM++, uma extensão com orientação a objetos e concorrência

do VDM (*Vienna Development Method*). Seu objetivo é o início de estudos para o desenvolvimento de sistemas automatizados para procedimentos de decolagem em aeroportos. Em um trabalho semelhante (YOUSAF et al., 2012), o procedimento de aterrissagem é especificado por meio de VDM++.

Utilizando a mesma abordagem, Zafar (2014) especifica o procedimento de decolagem em aeroportos utilizando a notação Z. Visando a automação do processo de decolagem, a superfície do aeroporto é modelada por meio de uma estrutura do tipo grafo. A expectativa dos autores é de que essa pesquisa seja a base para a automação de sistemas de controle de torre em trabalhos futuros. A especificação é verificada formalmente com a ferramenta Z/EVES.

O trabalho de Cerone, Lindsay e Connelly (2005) explora as falhas que podem ocorrer devido à presença humana no processo de tomada de decisão em sistemas de tráfego aéreo. Os autores modelam o processo de tomada de decisão de um ATCo e denominam este modelo de *Operator Choice Model* (OCM). O modelo é desenvolvido utilizando a linguagem CSP e são definidas propriedades em lógica temporal linear para verificação do modelo. Através desta verificação, os autores avaliam quais os tipos de erros que ocorrem no OCM e relacionam estes erros com falhas do processo de tomada de decisão de um ATCo.

No trabalho de Hanh e Hung (2007), o estudo do OCM é expandido para avaliar propriedades de segurança em sistemas ATC. O cenário utilizado é a detecção e resolução de um conflito. Este estudo modela o OCM como um modelo de autômato temporizado probabilístico (*Probabilistic Timed Automata*), define propriedades com CTL Probabilística de Tempo Real (*Probabilistic Real Time Computation Tree Logic*) e verifica as propriedades especificadas para o modelo com a ferramenta de verificação de modelos PRISM.

Muñoz, Carreño e Dowek (2006) verificam formalmente o sistema SATS com o provador de teoremas PVS. O sistema foi verificado por meio da análise de seus estados, que são definidos pelos estados de seus componentes: o sistema AMM (*Automated Management Module*) e a SCA (*Self Controlled Area*). A SCA é dividida em zonas e seu estado é definido pela ocupação destas zonas, *i.e.*, quantas aeronaves estão em cada zona da SCA. O sistema AMM autoriza as aeronaves a entrar na SCA. Quando uma aeronave é autorizada, o sistema AMM fornece uma zona de entrada e um número de sequenciamento para a aeronave. O estado do sistema AMM é definido pela próxima zona de entrada da SCA e pelo próximo número de sequenciamento. As regras e procedimentos do SATS são modelados nas transições de estado do AMM e da SCA.

Por meio da análise dos trabalhos discutidos anteriormente, é possível determinar uma abordagem para verificação formal em sistemas ATM. Esta abordagem utiliza a técnica de verificação de modelos, portanto verifica se proposições especificadas em linguagem

de especificação formal são satisfeitas por um modelo. As proposições são baseadas em requisitos do sistema e o modelo é baseado na especificação do sistema. A diferença entre os termos requisito de sistema e especificação do sistema para o resto deste texto será a seguinte: os requisitos de sistema descrevem o que o sistema deve fazer, e a especificação de sistema descreve como o sistema deve ser organizado e/ou implementado para fazer o que deve fazer. Por exemplo, requisitos que definem como o sistema deve se comportar serão denominados como especificação de sistema. Os requisitos de sistema são as propriedades que o sistema deve possuir, independente de seu comportamento.

Nos capítulos 6 e 7, os trabalhos de Zhao e Rozier (2014) e Muñoz, Carreño e Dowek (2006) serão estudados para definição das diretrizes das táticas para cada etapa do ciclo representado pela figura 2.

4.2 Mapeamento de Linguagem Natural para Linguagens de Especificação Formal

O conceito de padrões de especificação de requisitos (*requirement specification patterns*) é introduzido em Dwyer, Avrunin e Corbett (1998). Deste ponto do texto em diante, o termo será utilizado como padrão de especificação. O padrão de especificação é definido por uma propriedade recorrente de sistemas e define uma expressão em linguagem de especificação formal para esta propriedade. Por exemplo, é definido o padrão de Ausência (*Absence Pattern*), que descreve a ausência de certos eventos e/ou estados com um determinado escopo. O trabalho apresenta como especificar cada padrão nas linguagens de especificação formal LTL e CTL para cada um dos escopos apresentados: Globalmente; Antes de um evento R; Após um evento Q; Entre um evento Q e um evento R; Após um evento Q Até um evento R. Os autores do artigo mantêm um *site*¹ que reúne padrões de especificação associados aos escopos.

Autili et al. (2015) expandem a ideia de Dwyer, Avrunin e Corbett (1998) introduzindo novos padrões de especificação e associando expressões em linguagem natural controlada com os padrões de especificação. Os padrões introduzidos em Dwyer, Avrunin e Corbett (1998) são classificados como qualitativos, e estes padrões são expandidos com o uso de versões probabilísticas e de tempo real. A introdução destas novas versões possibilita a utilização de outras linguagens de especificação formal, como PCTL (CTL probabilístico). A principal ideia em associar os padrões a expressões em linguagem natural controlada é facilitar a classificação de requisitos em um determinado padrão de especificação. No

¹<http://patterns.projects.cs.ksu.edu/>, consultado em 01/06/2018

presente trabalho, apenas os padrões qualitativos serão usados.

No trabalho de Arcaini, Gargantini e Riccobene (2017) é apresentado o NuSeen, um *framework* que auxilia na utilização do verificador formal NuSMV durante o processo de modelagem e verificação. O trabalho apresenta ideias para a modelagem do sistema – como organizar o modelo, quais estruturas da linguagem de modelagem do NuSMV utilizar e como utilizar – e discute meta-propriedades do modelo. Um modelo deve possuir estas meta-propriedades para garantir consistência, completude e minimalidade. As meta-propriedades são especificadas em LTL e CTL e verifica-se se estas proposições são satisfeitas pelo modelo.

Um conjunto dos padrões de especificação apresentado em Dwyer, Avrunin e Corbett (1998) e Autili et al. (2015) será utilizado para especificar os requisitos em linguagem natural controlada no ciclo de especificação e para o mapeamento dos requisitos em linguagem natural controlada para linguagem de especificação formal. A versão empregada dos padrões é a qualitativa, e as linguagens de especificação formal usadas serão LTL e CTL. Outros padrões de especificação são propostos para trabalhar com os requisitos de sistemas dos estudos de caso no presente trabalho. O conjunto de padrões de especificação é apresentado no capítulo 8.

O processo de modelagem apresentado em Arcaini, Gargantini e Riccobene (2017) será usado como base para a definição de diretrizes para a modelagem do sistema no ciclo de especificação. O modelo gerado deverá possuir algumas das meta-propriedades apresentadas.

No capítulo 6 o conjunto de padrões de especificação, meta-propriedades e diretrizes de modelagem serão apresentados.

5 CICLO DE ESPECIFICAÇÃO DE REQUISITOS

Por meio da análise de pesquisas desenvolvidas para a evolução dos sistemas de tráfego aéreo, pode-se esperar que as novas soluções utilizarão sistemas com nível de automação maior. Tarefas realizadas por operadores humanos serão conduzidas por processos automatizados. Um desafio desta abordagem é garantir que os processos automatizados são, no mínimo, tão seguros quanto os processos tradicionais (ERZBERGER; LAUDERDALE; CHU, 2012). Métodos formais podem ser usados para a verificação das novas soluções, aumentando a confiabilidade dos sistemas por meio da redução da quantidade de erros de projeto em fases iniciais do desenvolvimento do sistema.

Com o estudo realizado de operações aéreas, percebe-se que sistemas de tráfego aéreo são reativos, definindo a resposta a estímulos do ambiente em que estão inseridos. Máquinas de estados são formas adequadas para representação deste tipo de sistema. Os requisitos de sistema especificam eventos que devem ocorrer em determinadas condições, portanto linguagens que expressem proposições sobre caminhos de execução de um sistema podem ser usadas para especificação de requisitos do sistema.

No trabalho de Zhao e Rozier (2014), a ferramenta de verificação formal NuSMV é utilizada para a verificação formal de um sistema ATC. O sistema é modelado por uma FSM e proposições são especificadas em LTL e CTL. O NuSMV é usado para verificar se as proposições são propriedades do modelo. Este conjunto de formalismos – FSM e lógica temporal – é adequado para sistemas ATC e será utilizado nesta pesquisa.

A técnica de verificação formal utilizada é a técnica de verificação de modelos. O modelo do sistema será representado por uma máquina de estados finita e as proposições serão especificadas em LTL e CTL. As proposições são baseadas em requisitos de sistema e o modelo é baseado na especificação de sistema. Verifica-se se estas proposições são satisfeitas pelo modelo, ou seja, se as proposições correspondem a propriedades do modelo, utilizando a mesma abordagem de Zhao e Rozier (2014).

Após a aplicação do ciclo de especificação, espera-se que a especificação em linguagem natural seja mais consistente com o conceito do sistema, ou seja, os requisitos do sistema possuam menos erros. Um modelo do sistema representado em FSM é necessário ao utilizar o NuSMV. Em sistemas intensivos em *software*, um modelo representado por uma FSM está mais próxima do nível de implementação do que uma especificação em linguagem natural. Após a aplicação das etapas do ciclo de especificação, o modelo pode ser usado como base para implementação do sistema, ao invés de parte da especificação do sistema em linguagem natural.

Para estabelecer o método de especificação formal, é necessária a definição de uma tática

para cada uma das etapas do ciclo de especificação apresentadas na seção 1.1:

1. Etapa de especificação em linguagem natural;
2. Etapa de especificação formal;
3. Etapa de verificação formal e análise do resultado da verificação;
4. Etapa de ajuste na especificação formal;
5. Etapa de ajuste na especificação em linguagem natural.

As diretrizes das táticas serão elaborada com dois estudos de caso: a coordenação entre os sistemas do AAC e o conceito operacional para pouso do SATS. Conceito operacional corresponde a uma descrição do sistema, do ponto de vista do usuário, que apresenta as capacidades de um sistema e como estas capacidades são empregadas para alcançar um determinado fim.

Um conjunto de diretrizes é proposto com base na análise do trabalho desenvolvido por Zhao e Rozier (2014). O conceito operacional do AAC é verificado formalmente com a utilização do verificador NuSMV. O foco da verificação é a coordenação entre os três sistemas utilizados para prevenção de conflito e colisão entre aeronaves: o sistema *AutoResolver*, o sistema TSAFE e o TCAS. Padrões de especificação (AUTILI et al., 2015) serão aplicados para avaliar se podem ser inseridos ao conjunto de diretrizes. O foco de Zhao e Rozier (2014) é a análise da coordenação entre os sistemas, portanto seu trabalho não detalha como as expressões em linguagem de especificação formal são elaboradas a partir de um conjunto de requisitos. No presente trabalho, o mesmo conjunto de requisitos será usado para a elaboração de expressões em linguagem de especificação formal com o uso de padrões de especificação. O mesmo modelo proposto em Zhao e Rozier (2014) será usado para a etapa de verificação formal, logo se a aplicação do ciclo de especificação alcançar o mesmo resultado que os alcançados em Zhao e Rozier (2014), a proposta de especificação por padrões será avaliada positivamente.

Este conjunto será então aplicado ao segundo estudo de caso para a avaliação do conjunto de diretrizes.

Nas seções seguintes, os conceitos operacionais dos sistemas dos estudos de caso serão descritos assim como os trabalhos que serão usados para definição de diretrizes das etapas do ciclo de especificação. Estas descrições correspondem a especificação dos sistemas e são usadas para a modelagem do sistema.

5.1 Conceito Operacional AAC – Coordenação Entre Sistemas

O AAC utiliza 3 sistemas diferentes para a prevenção de colisão e conflito entre aeronaves no espaço aéreo.

Os três sistemas utilizados no AAC são o sistema *AutoResolver*, o sistema TSAFE e o TCAS. Os sistemas *AutoResolver* e o TSAFE foram projetados durante o desenvolvimento do AAC, enquanto o TCAS é um sistema desenvolvido separadamente, e que deve funcionar de maneira independente ao AAC. O *AutoResolver* e o TSAFE respondem a previsões de conflito entre aeronaves. O TCAS visa a prevenção de colisão iminente. Os sistemas possuem diferentes graus de automação.

O *AutoResolver* utiliza posição e rota planejada de aeronaves para a previsão de conflitos no médio prazo. O período de médio prazo não é fixo, em Erzberger e Heere (2010) o valor apresentado como padrão corresponde ao intervalo entre 2 e 20 minutos. O *AutoResolver* é um sistema de auxílio ao controlador. Quando um conflito é detectado, o sistema exibe um alerta ao controlador, que decide se vai tomar alguma ação para a resolução do conflito. Em Erzberger, Lauderdale e Chu (2012), algoritmos para detecção e resolução de conflitos para o *AutoResolver* são apresentados.

O sistema TSAFE utiliza posição, velocidade e atitude de aeronaves para a previsão de conflitos no curto prazo, cujo valor padrão corresponde a um intervalo inferior a 2 minutos. O TSAFE possui 2 modos de operação: um com menor grau de automação e outro com maior grau de automação. O de menor grau de automação é usado caso o conflito seja previsto para um instante maior do que um limiar (com valor padrão de 1 minuto). Neste modo, o sistema alerta o controlador, que pode propor uma solução numa maneira semelhante a do *AutoResolver* ou transferir o controle das aeronaves em conflito para o TSAFE, que calcula uma resolução para o conflito e envia as ações necessárias para as aeronaves. O modo de operação de maior grau de automação é acionado caso o conflito seja previsto para um horizonte abaixo do limiar. Neste modo, o TSAFE envia um alerta ao controlador e o controle das aeronaves é transferido para o TSAFE para a resolução do conflito. Em Erzberger e Heere (2010) são propostos algoritmos para a resolução de conflitos do TSAFE. Os autores denominam a capacidade automatizada de detecção e apresentação de conflitos de TSAFE *Alert* e a capacidade de geração de resoluções e autorização para resolução de conflitos independente do controlador de TSAFE *Resolution*.

O TCAS utiliza os mesmos dados do TSAFE e é utilizado para a prevenção de colisão iminente entre aeronaves. O horizonte temporal com que TCAS opera é inferior a 30 segundos. A solução é decidida colaborativamente entre os TCAS das aeronaves. As manobras para resolução da colisão são alteração de nível para recuperação da separação

vertical, e alteração de curso para recuperação da separação horizontal. A figura 3 apresenta um resumo com as responsabilidades de cada sistema.

Caso a aeronave receba comandos de mais de um sistema, o piloto deve realizar as instruções do comando de maior prioridade. O comando proveniente do TCAS é o mais prioritário, seguido pelo comando proveniente do TSAFE, e por último, o comando proveniente do *AutoResolver*/controlador.

Figura 3 - Resumo Esquemático da Coordenação entre Sistemas no AAC

5.1.1 Análise da Verificação Formal do AAC

Zhao e Rozier (2014) levantam requisitos do conceito operacional do AAC a partir de Erzberger e Heere (2010) e por meio de entrevistas com desenvolvedores do sistema. Estes requisitos são usados para especificar conjunto de proposições em LTL e CTL. O documento do conceito operacional do AAC é usado para a modelagem do sistema. Os autores utilizam o NuSMV para verificar se as proposições correspondem a propriedades do modelo e ajustes nos requisitos e no conceito operacional do AAC são propostos caso a proposição não seja uma propriedade do modelo.

A máquina de estados finita codificada no *script* para NuSMV de Zhao e Rozier (2014) foi usada para o processo de verificação formal do estudo de caso e alterado para avaliar as diretrizes propostas para as etapas do ciclo de especificação. O conjunto de proposições foi especificado utilizando padrões de especificação.

5.2 Conceito Operacional SATS

No trabalho de Abbott et al. (2004) o conceito operacional do SATS é apresentado. Este relatório define um conjunto de regras para as operações de pouso e decolagem de aeronaves

em aeródromos sem a infra-estrutura completa necessária para um serviço de controle de tráfego aéreo. O SATS define dois elementos para a operação segura e eficiente destes aeródromos: a SCA (*Self Controlled Area*) que corresponde a área em volta do aeródromo em que as operações do SATS ocorrem; e o AMM (*Airport Management Module*), um sistema automatizado instalado em solo cuja função é a autorização de entrada na SCA e define o sequenciamento de pousos de aeronaves. A separação de aeronaves na SCA é responsabilidade dos pilotos.

A SCA é dividida em regiões denominados de fixos e possui um desenho com formato em T. Nas extremidade do topo do T se encontram os fixos de aproximação inicial (IAF – *Initial Approach Fix*). Estes são os pontos em que as aeronaves entram na SCA. As aeronaves devem entrar no fixo e descer em incrementos de 1000 pés até alcançar a altitude de aproximação inicial definida em 1000 pés. No relatório do conceito operacional, é definida a altitude nominal de 3000 pés para a SCA, logo existem 4 regiões iniciais nesta configuração:

- IAF esquerda a 3000 pés;
- IAF esquerda a 2000 pés;
- IAF direita a 3000 pés;
- IAF direita a 2000 pés;

Na extremidade inferior do T está localizado o fixo de aproximação final (FAF – *Final Approach Fix*) que as aeronaves utilizam para efetuar o pouso. Durante todo procedimento de pouso uma ordem de arremetida (*missed approach*) pode ser emitida. Neste caso, a aeronave inicia o procedimento de arremetida. No procedimento de arremetida, a aeronave se dirige ao fixo de arremetida (MAHF – *Missed Approach Holding Fix*) de menor altitude possível. O MAHF é atribuído à aeronave pelo AMM na emissão de autorização de entrada da aeronave. Na proposta inicial da geometria da SCA os MAHFs são geograficamente iguais aos IAFs. A figura 4 apresenta uma visão de topo da SCA, enquanto a figura 5 apresenta uma visão de perfil.

O AMM emite autorização de entrada na SCA a aeronaves que emitem solicitação de entrada se as condições para autorização forem satisfeitas. Nesta autorização, são definidas a ordem de sequenciamento (a aeronave sabe qual aeronave está a frente dela na fila de pouso), o IAF, o tipo de entrada (explicados a seguir) e o MAHF.

Os procedimentos possíveis de um voo chegando à SCA são apresentados a seguir.

Troca de Informação Preliminar em Espaço Aéreo ATC – O AMM envia em *broadcast* quantas aeronaves estão em fila para pouso ou em espera para entrada na SCA.

Figura 4 - SCA – Visão de Topo

Figura 5 - SCA – Visão de Perfil

As aeronaves próximas à SCA informam o IAF planejado. O AMM então avalia se a aeronave pode entrar na área ou se vai ficar em espera. O AMM define um número de sequência da aeronave, que indica sua posição na fila de pouso. O AMM também informa qual o número de sequência da aeronave a sua frente, que será denominada de aeronave-líder. As aeronaves emitem em *broadcast* seus números de sequência, logo o piloto sabe qual a aeronave-líder. Ao receber esta autorização, a aeronave vai em direção ao fixo atribuído, na altitude de controle ATC acima do IAF (sem descer ao nível superior da SCA).

Transição Vertical do Espaço Aéreo ATC para a SCA – O AMM aguarda uma janela para emitir a autorização para a entrada da aeronave na SCA. Caso as condições sejam satisfeitas, o AMM emite uma notificação de entrada vertical e uma notificação indicando qual é a aeronave-líder. As condições para a emissão da notificação de entrada vertical são:

- existem menos de duas aeronaves atribuídas ao IAF planejado, como IAF ou MAHF;
- não existem outras entradas no mesmo fixo;
- não existe aeronave atribuída àquele fixo como MAHF em procedimento de aproximação da pista;
- a entrada não irá resultar na atribuição de mais de 2 aeronaves ao mesmo fixo, como IAF ou MAHF.

Entrada Vertical – Após receber a notificação de entrada vertical, a aeronave aguarda a região imediatamente inferior estar disponível. Isto ocorre quando a aeronave-líder deixou a região imediatamente inferior (partiu para a próxima região inferior ou iniciou a aproximação). Nesta fase, o sistema AMM emite o MAHF.

Transição e Entrada Lateral do Espaço Aéreo ATC para a SCA – Em cenários de menor movimentação, o AMM pode emitir uma notificação de entrada lateral, em que a aeronave não espera acima da SCA. A aeronave se dirige ao IAF atribuído de menor altitude. O AMM atribui as mesmas informações da entrada vertical: IAF, MAHF e a identificação da aeronave-líder. Para a emissão de uma notificação de entrada lateral, nenhuma das condições seguintes pode ser verdadeira:

- uma aeronave está no IAF ou está atribuída ao IAF;
- a aeronave está na região de decolagens (ao lado oposto dos IAFs, com relação à pista) da SCA;

- a aeronave precisa atravessar mais de uma região de pouso para chegar ao IAF;
- a entrada resulta na atribuição de mais de 2 aeronaves ao mesmo fixo, como IAF ou MAHF.

Descida – conforme os níveis inferiores do IAF estiverem disponíveis, a aeronave desce ao nível disponível. Isto ocorre quando a aeronave-líder desceu ao nível inferior ou iniciou o procedimento de aproximação.

Iniciando Aproximação – ao chegar ao nível mais inferior do IAF, a aeronave estará em posição para iniciar a aproximação. A aproximação consiste no alinhamento com a pista e na descida para o nível da pista. A condição de início de aproximação é garantir a separação com relação a aeronave à sua frente. Baseada no número de sequência, o piloto deve avaliar se a aeronave-líder está a uma distância mínima.

Aproximação – Durante a aproximação, o piloto deve garantir a separação para a aeronave-líder.

Pouso – O piloto deve seguir o procedimento de pouso em condições visuais.

Arremetida – Caso seja necessário realizar um procedimento de arremetida, a aeronave deve se dirigir ao nível mais inferior disponível da MAHF atribuída a ela. Ao chegar nesta região, o AMM atribui um novo número de sequência, aeronave-líder e MAHF. A aeronave então realiza o procedimento de descida vertical no IAF descrito anteriormente.

5.2.1 Elementos Modelados

Em Dowek, Munoz e Carreno (2004), um modelo do SATS é apresentado para verificação formal do SATS com um provador de teoremas. Este modelo apresenta algumas simplificações que serão usadas para o modelo do SATS. Outras simplificações serão introduzidas observando o modelo em Dowek, Munoz e Carreno (2004) visando a redução do espaço de estados do modelo – utilização de menos variáveis. As hipóteses adotadas serão:

- A SCA será composta pelas seguintes regiões: IAF esquerda a 3000 pés, IAF esquerda a 2000 pés, IAF direita a 3000 pés, IAF direita a 2000 pés, aproximação intermediária, aproximação final e pista.
- O procedimento de arremetida só será realizado se a aeronave estiver na aproximação final.
- O procedimento de entrada lateral não será modelado.
- Os pilotos não solicitam entrada em um IAF específico. O AMM atribui à aeronave o IAF disponível.

- O tempo da transição de uma região para a região seguinte imediata é igual para todas regiões.
- Apenas o procedimento de pouso será modelado.
- As ações de ATC não serão consideradas no modelo.
- Os pilotos conhecem e seguem os procedimentos de maneira exata.

Estas hipóteses serão utilizadas para modelagem do SATS no capítulo 6.

6 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA EM LINGUAGEM NATURAL E EM LINGUAGEM DE ESPECIFICAÇÃO FORMAL

Este capítulo é dividido em duas seções. A seção 6.1 apresenta a coleta de requisitos de sistema e sua classificação em um padrão de especificação. Na seção 6.2 os requisitos coletados em linguagem natural são mapeados para proposições em LTL e CTL e a especificação do sistema é usada para modelagem do sistema.

6.1 Especificação em Linguagem Natural

A especificação em linguagem natural é a mais utilizada para a difusão do conceito do sistema entre os envolvidos em seu desenvolvimento, sendo geralmente a forma de descrição original do sistema. Portanto, é necessário extrair deste tipo de descrição os requisitos que serão mapeados para especificação formal e serão usados para modelagem do sistema.

Os requisitos que serão mapeados para linguagens de especificação formal devem descrever propriedades do sistema que definem o que o sistema deve fazer ou o que deve ser capaz de fazer sem especificar como o sistema deve fazer. Estes requisitos devem ser classificados em um padrão de especificação.

Analisando os requisitos da coordenação entre sistemas do AAC e os requisitos do SATS foi possível definir um conjunto de padrões de especificação. Os padrões e sua forma de escrita em linguagem natural controlada são apresentados a seguir.

- Ausência - Global – como definido em Autili et al. (2015).
 - Nunca ocorre que $\{P\}$ seja verdadeiro.
- Ausência - Após $\{Q\}$ até $\{R\}$ – como definido em Autili et al. (2015).
 - Após $\{Q\}$ até $\{R\}$, não ocorre que $\{P\}$ seja verdadeiro.
- Resposta - Global – como definido em Autili et al. (2015).
 - Se $\{P\}$ ocorreu então em resposta $\{S\}$ ocorre eventualmente.
- Precedência - Após $\{Q\}$ Até $\{R\}$ – como definido em Autili et al. (2015).
 - Após $\{Q\}$ Até $\{R\}$, se $\{P\}$ ocorreu, então $\{S\}$ deve ter ocorrido.
- Resposta Imediata - Global – variação do padrão de especificação de resposta, em que uma variável proposicional $\{S\}$ deve ser verdadeira no estado seguinte após uma variável $\{P\}$ se tornar verdadeira.

- Se $\{P\}$ ocorreu então em resposta $\{S\}$ ocorre imediatamente.
- Resposta Imediata a Transição - Global – variação do padrão de especificação de resposta, em que uma variável proposicional $\{S\}$ deve ser verdadeiro após uma transição em específico.
 - Se $\{P\}$ é verdade e $\{R\}$ ocorre, então imediatamente $\{S\}$ ocorre como resposta.
- Múltiplas Respostas - Global – variação do padrão de especificação de resposta, em que uma das variáveis proposicionais $\{S1\}, \{S2\} \dots \{Sn\}$ deve ser verdadeiro se a variável $\{P\}$ for verdadeira.
 - Se $\{P\}$ ocorreu, então em resposta $\{S1\};$ ou $\{S2\}; \dots ;$ ou $\{Sn\}$ eventualmente ocorre.

6.1.1 Estudo de Caso A – Coordenação AAC

Zhao e Rozier (2014) levantam requisitos do AAC a partir do conceito operacional do TSAFE apresentado em Erzberger e Heere (2010) e da expectativa dos desenvolvedores do sistema. Percebe-se então que existem ao menos dois tipos de fontes para a coleta de requisitos de sistema: documentos e pessoas envolvidas.

O conjunto de requisitos foi levantado após avaliar o que se pretende verificar. No caso, pretende-se verificar a coordenação entre os sistemas que compõem o AAC, *i.e.*, assumindo que os sistemas operem sem falhas em operação isolada, verifica-se se em conjunto o comportamento do sistema AAC é o esperado.

Os requisitos destacados por Zhao e Rozier (2014) referentes ao conceito operacional do AAC estão listados na tabela 1 (com tradução livre da língua inglesa).

Os requisitos da tabela 2 foram levantados por meio de entrevistas com os desenvolvedores do sistema.

O requisito A1 é classificado como padrão de Resposta Imediata a Transição - Global.

O requisito A2 é classificado como padrão de Múltiplas Respostas - Global.

Os requisitos A3, A5 e A11 são classificados como padrão de Resposta Imediata - Global.

Os requisitos A4, A7 e A8 são classificados como padrão de Resposta - Global.

Os requisitos A6, A9 e A10 são classificados como padrão de Precedência - Após $\{Q\}$ Até $\{R\}$.

Tabela 1 - Requisitos extraídos da descrição do conceito operacional do AAC (ZHAO; ROZIER, 2014)

No	Requisito
A1	Enquanto o TSAFE não emite um alerta de conflito com previsão abaixo do limiar, o controlador é responsável pela formulação de resoluções e pela resolução de conflitos.
A2	Após o TSAFE emitir um alerta, enquanto o tempo para perda de separação for acima do limiar, o controlador deve escolher uma dentre estas três opções: (1) inibir o TSAFE <i>Resolution</i> de enviar uma resolução para o conflito; (2) autorizar o TSAFE <i>Resolution</i> a enviar uma resolução para a aeronave; (3) não agir no momento para o determinado conflito.
A3	Caso o controlador autorize o TSAFE <i>Resolution</i> a enviar uma resolução, o envio é feito imediatamente.
A4	Caso a estimativa para perda de separação caia para abaixo do limiar sem que o controlador tenha tomado alguma ação, a responsabilidade pela resolução do conflito é transferida automaticamente para o TSAFE <i>Resolution</i> .
A5	Após o TSAFE enviar uma resolução para a aeronave, uma mensagem no monitor do controlador indica quando o conflito foi resolvido e o controle da aeronave é transferido para o controlador, que fica responsável pela orientação da aeronave para a rota original.
A6	O piloto deve ser treinado para responder ao TSAFE e entender as condições que disparam um alerta TSAFE e TCAS.

Tabela 2 - Requisitos extraídos de entrevistas com desenvolvedores do AAC (ZHAO; ROZIER, 2014)

No	Requisito
A7	Todos alertas detectados pelo TSAFE serão resolvidos eventualmente.
A8	Caso o TSAFE assuma o controle de uma aeronave, eventualmente o controle será transferido para o controlador.
A9	Caso o controlador transfira o controle de uma aeronave para o TSAFE, esta aeronave não pode executar comandos do controlador.
A10	As resoluções do TSAFE mais antigas serão sempre executadas antes de resoluções mais recentes.
A11	Se um comando do TSAFE for enviado a uma aeronave, o controlador deve transferir o controle desta aeronave para o TSAFE.

6.1.2 Estudo de Caso B – SATS

O conjunto de requisitos do SATS foi definido a partir das propriedades exigidas do sistema, retiradas da especificação do sistema (ABBOTT et al., 2004). A tabela 3 apresenta os requisitos (com tradução livre da língua inglesa).

Tabela 3 - Requisitos extraídos das propriedades exigidas do conceito operacional do SATS (ABBOTT et al., 2004)

No	Requisito
B1	Não são permitidas mais do que 4 aeronaves realizando o procedimento de pouso na SCA.
B2	Não são permitidas entradas simultâneas em um IAF.
B3	Após entrar na SCA, as aeronaves devem iniciar o procedimento de descida apenas quando a região imediatamente inferior estiver disponível.
B4	Os MAHF devem ser atribuídos de maneira alternada para aeronaves em sequência.
B5	Após arremetida as aeronaves devem iniciar procedimento de subida até o nível mais inferior disponível do MAHF atribuído.
B6	Entradas verticais são permitidas apenas para as altitudes mais baixas de voos IFR sobre a SCA.
B7	As aeronaves em procedimento na SCA devem possuir uma velocidade de subida de pelo menos 300 milhas por hora.

Os requisitos B1 e B4 são classificados como padrão de Ausência - Global.

Os requisitos B2 e B3 são classificados como padrão de Ausência - Após {Q} até {R}.

O requisito B5 é classificado como padrão de Resposta - Global.

Os requisitos B6 e B7 não serão verificados pois estão fora do escopo da modelagem. O requisito B6 especifica ações ATC, e o requisito B7 especifica o desempenho mínimo da aeronave para realizar a transição entre regiões da SCA.

6.2 Especificação Formal

Um modelo do sistema é definido numa representação de máquina de estados finita e verifica-se se propriedades representadas em LTL e CTL são válidas para este modelo. Para a escolha destes formalismos, foi necessário considerar o sistema modelado e a finalidade do modelo. Nos sistemas dos estudos de caso e sistemas ATM em geral, como discutido no capítulo 3, é especificado um conjunto de regras que definem como os agentes no sistema

(aeronaves, operadores) devem reagir a determinados eventos. A finalidade do modelo é sua verificação, logo a ferramenta de verificação deve ser considerada para a adoção da forma de representação. No ciclo de especificação a ferramenta de verificação usada é o NuSMV.

A especificação formal envolve:

- modelagem do sistema representada em máquina de estados finita, com base na especificação de sistema;
- mapeamento dos requisitos de sistema em linguagem natural para proposições em LTL e CTL.

A modelagem deve ser feita antes da especificação das proposições em LTL e CTL, pois as proposições devem ser especificadas com os termos usados no modelo.

6.2.1 Modelagem do Sistema

Com base na análise do modelo usado em Zhao e Rozier (2014), das diretrizes apresentadas em Arcaini, Gargantini e Riccobene (2017) e no manual do NuSMV, as seguintes diretrizes para modelagem foram estabelecidas.

O NuSMV possui uma estrutura denominada MODULE que permite organizar entidades do sistema em módulos. Com esta construção, entidades do sistema são modeladas como elementos do modelo. Cada módulo deve representar entidades do sistema que possuem comportamento distintos. Por exemplo, no caso de uma aeronave A precisar seguir uma determinada sequência de movimentos para decolagem e uma aeronave B seguir uma sequência diferente, cada aeronave deve ser representada por módulos diferentes. Caso a sequência seja a mesma, ou com poucas variações, as aeronaves podem ser representadas pelo mesmo módulo. Os módulos possuem parâmetros que são usados para acessar variáveis de outros módulos. Estes parâmetros devem ser usados para representar informações que entidades do sistema possuem de outras entidades modeladas.

A definição de cada módulo é segmentada em seções. As seções que devem ser usadas são VAR, DEFINE, ASSIGN e INVAR.

A seção VAR define as variáveis do módulo, que representam o estado dos elementos modelados por este módulo. As variáveis são usadas para observar o estado do elemento, ou seja, as proposições em LTL e CTL devem ser especificadas usando estas variáveis para a verificação do modelo. Cada variável tem um tipo; os tipos podem ser primitivos do NuSMV ou outros módulos do modelo. A escolha do tipo deve refletir os valores que a variável pode assumir.

A seção `DEFINE` possui expressões para melhorar a legibilidade e reuso de trechos do código. Deve ser usada para representação de combinação de valores de determinadas variáveis, por meio de expressões com estas variáveis. O tipo destas expressões não é predeterminado.

A seção `ASSIGN` é utilizada para a definição das transições de estado. Esta definição pode ser feita com o uso dos operadores `init` e `next`. Os operadores possuem como parâmetro a variável cujo valor será definido. O operador `init` define o valor inicial da variável e o operador `next` define o valor da variável no próximo passo de execução. Os valores podem ser atribuídos por meio de expressões ou atribuição direta. No caso do operador `init`, a atribuição direta é mais utilizada; no caso do operador `next`, as operações mais utilizadas são “if-then-else” e “case”. A operação `case` pode representar o gatilho para uma transição e o valor que a variável deve assumir no passo de execução seguinte. A ordem em que as transições são apresentadas interfere na atribuição: a primeira condição verdadeira será a acionada. A seção de `ASSIGN` também pode ser usada de maneira semelhante à seção `DEFINE`. A diferença é de que as variáveis usadas nesta seção precisam ser definidas na seção `VAR`, portanto são tipificadas.

A seção `INVAR` é utilizada para complementar a transição de estados. É possível criar transições não-determinísticas. A seção `INVAR` deve ser utilizada para representar estados impossíveis do sistema. A utilização desta seção pode resultar em um *deadlock* na execução do modelo.

Existe um módulo específico denominado `main`, que define todos elementos do modelo e a relação entre eles. Os elementos do modelo são definidos como variáveis do módulo `main` e as relações são definidas por meio da referência a variáveis de elementos como parâmetros de outros elementos.

Para modelar o sistema é utilizada parte da especificação de sistema que define como o sistema deve se comportar para atender aos requisitos do sistema.

6.2.2 Mapeamento de Linguagem Natural Controlada para LTL e CTL

Os requisitos levantados em linguagem natural devem ser mapeados para expressões em LTL e CTL. As expressões são especificadas a partir do mapeamento correspondente ao padrão de especificação em que o requisito foi classificado.

A seguir, o mapeamento de cada um dos padrões de especificação é apresentado.

- Ausência - Global
 - Nunca ocorre que $\{P\}$ seja verdadeiro.

- $\Box(\neg\{P\})$
- Ausência - Após $\{Q\}$ até $\{R\}$
 - Após $\{Q\}$ até $\{R\}$, não ocorre que $\{P\}$ seja verdadeiro.
 - $\Box(\{Q\} \wedge \neg\{R\} \implies (\neg\{P\}U\{R\}))$
- Resposta - Global
 - Se $\{P\}$ ocorreu então em resposta $\{S\}$ ocorre eventualmente.
 - $\Box(\{P\} \implies \Diamond\{S\})$
- Precedência - Após $\{Q\}$ Até $\{R\}$
 - Após $\{Q\}$ Até $\{R\}$, se $\{P\}$ ocorreu, então $\{S\}$ deve ter ocorrido.
 - $\Box(\{Q\} \wedge \neg\{R\} \implies (\neg\{P\}U(\{S\} \vee \{R\})))$
- Resposta Imediata - Global
 - Se $\{P\}$ ocorreu então em resposta $\{S\}$ ocorre imediatamente.
 - $\Box(\{P\} \implies \bigcirc\{S\})$
- Resposta Imediata a Transição - Global
 - Se $\{P\}$ é verdade e $\{R\}$ ocorre, então imediatamente $\{S\}$ ocorre como resposta.
 - $\Box(\{P\} \wedge \bigcirc\{R\} \implies \bigcirc(\{R\} \wedge (\bigcirc\{S\})))$
- Múltiplas Respostas - Global
 - Se $\{P\}$ ocorreu, então em resposta $\{S_1\}$; ou $\{S_2\}$; ... ; ou $\{S_n\}$ eventualmente ocorre.
 - $AG(\{P\} \implies EF\{S_x\})$
 - $\Box(\{P\} \implies \Diamond(\{S_1\} \vee \{S_2\} \vee \dots \vee \{S_n\}))$
 - Neste caso, haverá n proposições em CTL e uma proposição em LTL.

Para utilizar estes padrões, é necessário identificar as variáveis proposicionais representados pela letras P, Q, R e S e especificá-las em termos do modelo que será verificado.

6.2.3 Estudo de Caso A – Coordenação AAC

O sistema AAC é modelado em Zhao e Rozier (2014) baseado na especificação do conceito operacional e nos requisitos levantados. Este modelo será descrito nesta seção. As representações formais em LTL e CTL foram especificadas com base nos requisitos. O *script* contendo o modelo e as proposições em LTL e CTL pode ser obtido *online*². Este modelo foi alterado no presente trabalho para aplicar as diretrizes de modelagem do ciclo de especificação. Por exemplo, no modelo original a seção INIT é usada para definir os valores iniciais de algumas variáveis. Nas diretrizes propostas do ciclo de especificação, o operador `init()` deve ser usado na seção ASSIGN para definir os valores iniciais das variáveis, logo esta foi a construção usada na modelagem neste trabalho.

Para a verificação da coordenação entre os sistemas usados no AAC, o modelo utiliza 3 aeronaves. A modelagem parte do pressuposto que os sistemas não falham individualmente; não há falha de comunicação entre os sistemas, controladores e aeronaves; os controladores sempre fornecem soluções efetivas (hipótese de controlador competente); o piloto segue sempre o comando de maior prioridade (hipótese de piloto competente).

O modelo é composto por 4 módulos: **Aircraft**, **Controller**, **TSAFE** e **Environment**. O módulo **Aircraft** representa as aeronaves sob controle do AAC. O módulo **Controller** representa os controladores humanos, que recebe alertas de conflitos e envia comandos para o módulo **Aircraft**. O termo “recebe” indica que elementos deste módulo acessam o valor de uma variável de outro elemento do modelo e o termo “envia” indica que o módulo possui uma variável acessada por outros elementos do modelo. O módulo **TSAFE** representa o sistema TSAFE, que recebe alertas de conflitos, recebe comandos do módulo **Controller** e envia comandos para o módulo **Aircraft**. O módulo **Environment** simula conflitos e colisões, enviando alertas de conflitos para os módulos **Controller** e **TSAFE**. Para simulação de colisões, o módulo **Environment** gera o evento de comando do sistema TCAS. Este evento simula o envio de comando TCAS para o módulo **Aircraft**.

Os conflitos de médio-prazo detectados pelo sistema *AutoResolver* são modelados pelo módulo **Environment** que possui variáveis booleanas denominadas `ar_conflict_XY`. O XY representa as aeronaves que estão em conflito, logo, o módulo possui as variáveis `ar_conflict_12`, `ar_conflict_13` e `ar_conflict_23`. Estas variáveis são lidas no módulo **Controller**, que pode propor uma solução e enviá-la a aeronave ou pode não tomar ação alguma. O envio de solução é representada pela variável `CTR_commandX`, em que X representa a aeronave para a qual o comando é enviado. Esta variável é booleana,

²<https://ti.arc.nasa.gov/m/profile/kyrozier/AAC/AAC.html>, consultado em 30/05/2018

indicando se há ou não comando para a aeronave. O código apresentado na listagem 6.1 foi retirado de Zhao e Rozier (2014) e apresenta a definição do módulo **Environment**.

```

1 MODULE Environment (
2   CTR_command_done1, CTR_command_done2, CTR_command_done3
3   , TCAS_command1, TCAS_command2, TCAS_command3
4   , TCAS_command_done1, TCAS_command_done2, TCAS_command_done3
5   , TSAFE_command_done1, TSAFE_command_done2, TSAFE_command_done3
6 )
7
8 VAR
9   ar_conflict_12 : boolean;
10  ar_conflict_23 : boolean;
11  ar_conflict_13 : boolean;
12
13  tsafe_alert_12 : {Non, AT, BT};
14  tsafe_alert_23 : {Non, AT, BT};
15  tsafe_alert_13 : {Non, AT, BT};

```

Listagem 6.1: Módulo **Environment**

O módulo **Environment** gera os alertas de médio-prazo e gera os comandos TCAS com os parâmetros **TCAS_commandX**. Os parâmetros presentes nas linhas 2-5 são usados na atribuição dos valores das variáveis. Os conflitos de curto-prazo detectados pelo **TSAFE** são simulados pelas variáveis denominadas **tsafe_conflict_XY**. Estas variáveis podem assumir 3 valores enumerados: **Non**, **BT** e **AT**. O valor **Non** representa o caso em que não há conflito. O valor **BT** representa o cenário em que o conflito é esperado para um período abaixo do limiar. O valor **AT** representa o cenário em que o conflito é esperado para um período acima do limiar ³. As variáveis são enviadas para os módulos **Controller** e **TSAFE**. Caso o valor da variável seja **AT**, as variáveis do módulo **TSAFE** não são atribuídas para resolução do conflito. Nestas condições, o módulo **Controller** pode: enviar uma solução para o elemento de módulo **Aircraft**, representado pela variável **CTR_commandX**; autorizar o módulo **TSAFE** ao envio de uma resolução para a aeronave, representado pela variável **tsafe_sndcmd** do módulo **Controller** e acessada pelo módulo **TSAFE**; ou pode não realizar ação para a resolução do conflito. Isto é especificado por meio de transições do FSM do módulo **Controller** e reflete o requisito 2 da tabela 1. Caso o alerta possua valor **BT**, o controle da aeronave é transferido para o módulo **TSAFE** que envia um comando ao elemento do módulo **Aircraft** com a variável **tsafe_commandX**.

O envio de resolução do módulo **Controller** é representado pela variável booleana denominada **CTR_command1**. O valor da variável é verdadeiro quando o controlador passa

³No *script* de NuSMV, **BT** é uma sigla para “*Below Threshold*”, ou seja, para o caso em que o conflito é previsto para um intervalo abaixo do limiar e **AT** é uma sigla para “*Above Threshold*”, para intervalos acima do limiar. No artigo de Zhao e Rozier (2014), as siglas representam, respectivamente, “*Before Threshold*” e “*After Threshold*”, possuindo então significados inversos ao *script*. A notação adotada neste trabalho será coerente com a notação do *script*.

um comando para a aeronave. Seu valor é determinado como apresentado na listagem 6.2.

```

1  ASSIGN
2  init(CTR_command1) := FALSE;
3  next(CTR_command1) := case
4  !CTR_control_1 : FALSE;
5  CTR_res1 & CTR_window : TRUE;
6  CTR_command1 : FALSE;
7  TRUE: CTR_command1;
8  esac;

```

Listagem 6.2: Transição variável CTR_command1

A função `init(CTR_command1)` define o valor inicial da variável. A função `next(CTR_command1)` define o valor que a variável assume a cada transição. Este valor depende de diferentes condições. Caso o valor da variável `CTR_control_1` seja falso, o que representa o cenário em que o controlador não possui controle sobre a aeronave 1, o controlador não pode passar um comando para aeronave e seu valor é falso. Caso o valor da variável `CTR_command1` seja verdadeira – o comando acabou de ser enviado – o controlador não pode passar outro comando. Caso as variáveis `CTR_res1` e `CTR_window` sejam verdadeiras, o comando é enviado. A variável `CTR_window` indica o cenário em que um comando pode ser enviado, ou seja, quando há algum conflito e o controlador escolhe entre propor uma solução ou não. Isto é feito com atribuição não-determinística, como pode ser observado no trecho de código apresentado na listagem 6.3.

```

1  init(CTR_window) := FALSE;
2  next(CTR_window) := case
3  CTR_window : FALSE;
4  ar_conflict_12 | (tsafe_alert_12!=Non) |
5  ar_conflict_13 | (tsafe_alert_13!=Non) |
6  ar_conflict_23 |
7  (tsafe_alert_23!=Non) : {TRUE, FALSE};
8  TRUE: CTR_window;
9  esac;

```

Listagem 6.3: Transição variável CTR_Window

O conjunto `{TRUE, FALSE}` representa o conjunto de valores que a variável pode assumir. Sempre que houver alerta de conflito detectados pelos sistemas *AutoResolver* ou *TSAFE*, existe a possibilidade do controlador propor uma solução. O valor da variável `CTR_res1` é atribuído por meio de invariantes, como apresentado na listagem 6.4.

```

1  INVAR
2  (CTR_res1 -> (ar_conflict_12 | ar_conflict_13 |
3  tsafe_alert_12 != Non | tsafe_alert_13 != Non))
4  &
5  (tsafe_sndcmd -> (!CTR_res1 & !CTR_res2 & !CTR_res3));

```

Listagem 6.4: Invariante CTR_res1

A primeira invariante indica que o valor de `CTR_res1` é verdadeiro quando existe um conflito envolvendo a aeronave 1. Porém caso a variável `tsafe_sndcmd` seja verdadeira, `CTR_res1` é falso. A variável `tsafe_sndcmd` indica a transferência de controle para o TSAFE. Logo, quando esta transferência é feita, o controlador não pode propor uma solução.

As condições de controle da aeronave são representadas pelas variáveis booleanas `CTR_controlX` e `TSAFE_controlX`. As condições de controle seguem os requisitos das tabelas 1 e 2. Como os requisitos determinam as condições em que o controle deve ser transferido ou não, utilizam-se valores booleanos para representação do controle da aeronave.

O módulo `Aircraft` recebe os comandos dos módulos `Controller`, `TSAFE` e `Environment` e possui uma variável booleana para cada comando. O valor da variável indica se o comando foi realizado. Por exemplo, a variável `TCAS_cmd_done` indica se o comando do TCAS foi realizado. A ordem em que os comandos são realizados caso haja mais de um disponível segue a ordem de maior prioridade: TCAS, TSAFE e controlador, como especificado em Erzberger e Heere (2010). O módulo `Aircraft` é apresentado na listagem 6.5.

O parâmetro `CTR_ac_in` representa o envio de um comando proveniente do `Controller` e o `TSAFE_ac_in` representa o comando do TSAFE. A variável `TCAS_command` é controlada pelo módulo de `Environment`, que é o responsável por simular o comando TCAS.

```

1 MODULE Aircraft (CTR_ac_in, TSAFE_ac_in
2 )
3
4 VAR
5   CTR_command : boolean;
6   CTR_command_done : boolean;
7   TSAFE_command : boolean;
8   TSAFE_command_done : boolean;
9   TCAS_command : boolean;
10  TCAS_command_done : boolean;
```

Listagem 6.5: Módulo Aircraft

Os requisitos de propriedades do AAC são mapeados em LTL e CTL de acordo com o mapeamento do padrão de especificação em que o requisito foi classificado. A elaboração das especificações em LTL e CTL são apresentadas a seguir.

- Requisito A1 – Enquanto o TSAFE não emite um alerta de conflito com previsão abaixo do limiar, o controlador é responsável pela formulação de resoluções e resolução de conflitos. (Resposta Imediata a Transição)
- Requisito A1 em Linguagem Controlada – Se o controlador possui o controle de 2 aeronaves e o TSAFE não emite conflito abaixo do limiar para estas aeronaves, então imediatamente o controlador possui o controle das aeronaves em resposta.

A tabela 4 apresenta as variáveis proposicionais do requisito em linguagem natural controlada de maneira mais detalhada, visando a escrita da expressão LTL correspondente.

Tabela 4 - Requisito A1

Variável proposicional	Variável proposicional detalhada
o controlador possui o controle de 2 aeronaves (P)	o controlador tem o controle da aeronave 1 ($P1$) e o controlador tem o controle da aeronave 2 ($P2$) e não há conflito entre as aeronaves ($P3$)
o TSAFE não emite conflito abaixo do limiar para estas aeronaves (R)	o TSAFE não emite conflito abaixo do limiar para estas aeronaves (R)
o controlador possui o controle das aeronaves (S)	o controlador possui o controle das aeronaves (S)

As proposições em termos do modelo podem ser escritas como:

$$\begin{aligned}
 P &: P1 \wedge P2 \wedge P3 \\
 P1 &: \text{controller.CTR_control_1} \\
 P2 &: \text{controller.CTR_control_2} \\
 P3 &: \text{env.tsafe_alert_13} = \text{Non} \wedge \text{env.tsafe_alert_23} = \text{Non} \wedge \\
 &\quad \text{env.tsafe_alert_12} = \text{Non} \\
 R &: \text{env.tsafe_alert_12} \neq \text{BT} \\
 S &: \text{controller.CTR_control_1} \wedge \text{controller.CTR_control_2}
 \end{aligned}$$

A expressão em LTL para o requisito A1 é apresentada a seguir.

$$\begin{aligned}
 &\Box(\text{controller.CTR_control_1} \wedge \text{controller.CTR_control_2} \wedge \\
 &\text{env.tsafe_alert_13} = \text{Non} \wedge \text{env.tsafe_alert_23} = \text{Non} \wedge \\
 &\text{env.tsafe_alert_12} = \text{Non} \wedge \\
 &\bigcirc(\text{env.tsafe_alert_12} \neq \text{BT}) \\
 &\implies \bigcirc(\text{env.tsafe_alert_12} \neq \text{BT} \wedge \\
 &\bigcirc(\text{controller.CTR_control_1} \wedge \text{controller.CTR_control_2}))
 \end{aligned} \tag{1}$$

- Requisito A2 – Após o TSAFE emitir um alerta, enquanto o tempo para perda de separação for acima do limiar, o controlador deve escolher uma dentre estas três

opções: (1) inibir o TSAFE *Resolution* de enviar uma resolução para o conflito; (2) autorizar o TSAFE *Resolution* a enviar uma resolução para a aeronave; (3) não agir no momento para o determinado conflito. (Múltiplas Respostas)

- Requisito A2 em Linguagem Controlada – Se o TSAFE emite um alerta perda de separação entre as aeronaves 1 e 2 acima do limiar e o controlador é o responsável por resolver o conflito, então em resposta o controlador não autoriza o TSAFE a enviar uma resolução enquanto possuir o controle das aeronaves; ou o controlador autoriza o TSAFE a enviar uma resolução; ou não autoriza o TSAFE a enviar resolução e não envia comandos até outro alerta ser emitido.

As tabelas 5 e 6 apresentam as proposições do requisito em linguagem natural controlada com mais detalhes para o mapeamento para expressões em CTL e LTL.

As proposições em termos do modelo podem ser escritas como:

$$\begin{aligned}
 P &: P1 \wedge P2 \\
 P1 &: P11 \wedge P12 \wedge P13 \wedge P14 \wedge P15 \wedge P16 \\
 P11 &: \text{env.tsafe_alert_12} = \text{AT} \\
 P12 &: \text{env.tsafe_alert_13} = \text{Non} \\
 P13 &: \text{env.tsafe_alert_23} = \text{Non} \\
 P14 &: \neg \text{env.ar_conflict_12} \\
 P15 &: \neg \text{env.ar_conflict_13} \\
 P16 &: \neg \text{env.ar_conflict_23}
 \end{aligned}$$

Tabela 5 - Requisito A2 – Parte 1

Variável proposicional	Variável proposicional detalhada
o TSAFE emite um alerta perda de separação entre as aeronaves 1 e 2 acima do limiar ($P1$)	há um alerta de conflito do TSAFE acima do limiar entre as aeronaves 1 e 2 ($P11$) e não há alerta de conflito do TSAFE entre as aeronaves 1 e 3 ($P12$) e não há alerta de conflito do TSAFE entre as aeronaves 2 e 3 ($P13$) e não há alerta de conflito do AutoResolver entre as aeronaves 1 e 2 ($P14$) e não há alerta de conflito do AutoResolver entre as aeronaves 1 e 3 ($P15$) e não há alerta de conflito do AutoResolver entre as aeronaves 2 e 3 ($P16$)
o controlador é o responsável por resolver o conflito($P2$)	o controlador possui o controle de uma aeronave ao menos ($P21$) e o controlador não autoriza o TSAFE a enviar uma resolução ($P22$) e a aeronave 1 não executou o comando enviado pelo TSAFE ($P23$) e a aeronave 2 não executou o comando enviado pelo TSAFE ($P24$) e o controlador não envia comandos para as aeronaves 1 e 2 ($P25$)

$$P2 : P21 \wedge P22 \wedge P23 \wedge P24 \wedge P25$$

$$P21 : \text{controller.CTR_control}$$

$$P22 : \neg \text{controller.tsafe_sndcmd}$$

$$P23 : \neg \text{ac1.TSAFE_command_done}$$

$$P24 : \neg \text{ac2.TSAFE_command_done}$$

$$P25 : \neg (\text{controller.CTR_command1} \vee \text{controller.CTR_command2})$$

Tabela 6 - Requisito A2 – Parte 2

Variável proposicional	Variável proposicional detalhada
o controlador não autoriza o TSAFE a enviar uma resolução enquanto possuir o controle das aeronaves (S_1)	o controlador não autoriza o TSAFE a enviar uma resolução enquanto possuir o controle das aeronaves (S_1)
o controlador autoriza o TSAFE a enviar uma resolução (S_2)	o controlador autoriza o TSAFE a enviar uma resolução (S_2)
não autoriza o TSAFE a enviar resolução (S_{31})	o controlador não autoriza o TSAFE a enviar resolução (S_{31})
não envia comandos (S_{32})	o controlador não envia comandos para as aeronaves (S_{32})
outro alerta ser emitido (S_{33})	outro alerta ser emitido (S_{33})

$$S_1 : \neg \text{controller.tsafe_sndcmd } U$$

$$(\text{controller.CTR_command1} \vee \text{controller.CTR_command2})$$

$$S_2 : \neg (\text{controller.CTR_command1} \vee \text{controller.CTR_command2}) \ U$$

$$\text{controller.tsafe_sndcmd}$$

$$S_3 : S_{31} \wedge S_{32} U S_{33}$$

$$S_{31} : \neg \text{controller.tsafe_sndcmd}$$

$$S_{32} : \neg \text{controller.CTR_command1} \vee \text{controller.CTR_command2}$$

$$S_{33} : \text{env.tsafe_alert_12} \neq AT \vee \text{env.tsafe_alert_13} \neq \text{Non} \vee$$

$$\text{env.tsafe_alert_23} \neq \text{Non} \vee \text{env.ar_conflict_12} \vee$$

$$\text{env.ar_conflict_13} \vee \text{env.ar_conflict_23}$$

$$\begin{aligned}
& AG((env.tsafe_alert_{12} = AT \wedge env.tsafe_alert_{13} = Non \wedge \\
& env.tsafe_alert_{23} = Non \wedge \\
& \neg env.ar_conflict_{12} \wedge \neg env.ar_conflict_{13} \wedge \neg env.ar_conflict_{23} \wedge \\
& controller.CTR_control \wedge \neg controller.tsafe_sndcmd \wedge \\
& \neg ac1.TSAFE_command_done \wedge \neg ac2.TSAFE_command_done \wedge \\
& \neg(controller.CTR_command1 \vee controller.CTR_command2) \\
& \implies E(\neg controller.tsafe_sndcmd \ U \\
& (controller.CTR_command1 \vee controller.CTR_command2))))
\end{aligned} \tag{2}$$

$$\begin{aligned}
& AG((env.tsafe_alert_{12} = AT \wedge env.tsafe_alert_{13} = Non \wedge \\
& env.tsafe_alert_{23} = Non \wedge \\
& \neg env.ar_conflict_{12} \wedge \neg env.ar_conflict_{13} \wedge \neg env.ar_conflict_{23} \wedge \\
& controller.CTR_control \wedge \neg controller.tsafe_sndcmd \wedge \\
& \neg ac1.TSAFE_command_done \wedge \neg ac2.TSAFE_command_done \wedge \\
& \neg(controller.CTR_command1 \vee controller.CTR_command2) \\
& \implies (E\neg(controller.CTR_command1 \vee controller.CTR_command2) \ U \\
& controller.tsafe_sndcmd))
\end{aligned} \tag{3}$$

$$\begin{aligned}
& AG((env.tsafe_alert_{12} = AT \wedge env.tsafe_alert_{13} = Non \wedge \\
& env.tsafe_alert_{23} = Non \wedge \\
& \neg env.ar_conflict_{12} \wedge \neg env.ar_conflict_{13} \wedge \neg env.ar_conflict_{23} \wedge \\
& controller.CTR_control \wedge \neg controller.tsafe_sndcmd \wedge \\
& \neg ac1.TSAFE_command_done \wedge \neg ac2.TSAFE_command_done \wedge \\
& \neg(controller.CTR_command1 \vee controller.CTR_command2) \\
& \implies (E(\neg controller.tsafe_sndcmd \wedge \\
& \neg(controller.CTR_command1 \vee controller.CTR_command2)) \ U \\
& (env.tsafe_alert_{12} \neq AT \vee env.tsafe_alert_{13} \neq Non \vee \\
& env.tsafe_alert_{23} \neq Non \vee env.ar_conflict_{12} \vee \\
& env.ar_conflict_{13} \vee env.ar_conflict_{23}))))
\end{aligned} \tag{4}$$

$$\begin{aligned}
& \Box((\text{env.tsafe_alert}_{12} = AT \wedge \text{env.tsafe_alert}_{13} = \text{Non} \wedge \\
& \text{env.tsafe_alert}_{23} = \text{Non} \wedge \\
& \neg \text{env.ar_conflict}_{12} \wedge \neg \text{env.ar_conflict}_{13} \wedge \neg \text{env.ar_conflict}_{23} \wedge \\
& \text{controller.CTR_control} \wedge \neg \text{controller.tsafe_sndcmd} \wedge \\
& \neg \text{ac1.TSAFE_command_done} \wedge \neg \text{ac2.TSAFE_command_done} \wedge \\
& \neg(\text{controller.CTR_command1} \vee \text{controller.CTR_command2}) \\
& \implies (\neg \text{controller.tsafe_sndcmd } U \\
& (\text{controller.CTR_command1} \vee \text{controller.CTR_command2})) \vee \\
& (\neg(\text{controller.CTR_command1} \vee \text{controller.CTR_command2}) \vee \\
& \text{controller.tsafe_sndcmd}) \vee \\
& (\neg(\text{controller.tsafe_sndcmd} \vee \\
& \text{controller.CTR_command1} \vee \text{controller.CTR_command2 } U \\
& (\text{env.tsafe_alert}_{12} \neq AT \vee \text{env.tsafe_alert}_{13} \neq \text{Non} \vee \\
& \text{env.tsafe_alert}_{23} \neq \text{Non} \vee \text{env.ar_conflict}_{12} \vee \\
& \text{env.ar_conflict}_{13} \vee \text{env.ar_conflict}_{23}))))))
\end{aligned} \tag{5}$$

O requisito A2 possui 3 expressões em CTL para representar cada uma das alternativas possíveis e uma expressão em LTL para verificar que estas são as únicas alternativas possíveis. O lado esquerdo das proposições em CTL é igual, o que indica que todas alternativas possuem a mesma condição.

- Requisito A3 – Caso o controlador autorize o TSAFE *Resolution* a enviar uma resolução, o envio é feito imediatamente. (Resposta Imediata)
- Requisito A3 em Linguagem Controlada – Se o controlador autoriza o TSAFE *Resolution* a enviar uma solução então em resposta o TSAFE envia o comando imediatamente.

A tabela 7 apresenta as variáveis proposicionais do requisito A3 em linguagem natural controlada e a seguir as proposições em termos do modelo são apresentadas.

Tabela 7 - Requisito A3

Variável proposicional	Variável proposicional detalhada
o controlador autoriza o TSAFE <i>Resolution</i> a enviar uma solução (P)	há conflito acima do limiar entre as aeronaves 1 e 2 ($P1$) e não há conflito abaixo do limiar entre as aeronaves 1 e 3 ($P2$) e não há conflito abaixo do limiar entre as aeronaves 2 e 3 ($P3$) e o controlador possui o controle da aeronave ($P4$) e o controlador autoriza o envio de comando pelo TSAFE ($P5$) e o TSAFE não está ocupado ($P6$).
o TSAFE envia o comando (S)	o TSAFE envia comando para a aeronave 1 ($S1$) ou o TSAFE envia comando para a aeronave 2

$$P : P1 \wedge P2 \wedge P3 \wedge P4 \wedge P5 \wedge P6$$

$$P1 : \text{env.tsafe_alert_12} = \text{AT}$$

$$P2 : \text{env.tsafe_alert_13} \neq \text{BT}$$

$$P3 : \text{env.tsafe_alert_23} \neq \text{BT}$$

$$P4 : \text{controller.CTR_control}$$

$$P5 : \text{controller.tsafe_sndcmd}$$

$$P6 : \text{tsafe.TSAFE_window}$$

$$S : S1 \vee S2$$

$$S1 : \text{tsafe.TSAFE_command1}$$

$$S2 : \text{tsafe.TSAFE_command2}$$

A expressão em LTL para o requisito A3 é apresentada a seguir.

$$\begin{aligned} & \Box((\text{env.tsafe_alert_12} = \text{AT} \wedge \text{env.tsafe_alert_13} \neq \text{BT} \wedge \\ & \text{env.tsafe_alert_23} \neq \text{BT} \wedge \text{controller.CTR_control} \wedge \\ & \text{controller.tsafe_sndcmd} \wedge \text{tsafe.TSAFE_window}) \\ & \implies \bigcirc(\text{tsafe.TSAFE_command1} \vee \text{tsafe.TSAFE_command2})) \end{aligned} \quad (6)$$

- Requisito A4 – Caso a estimativa para perda de separação caia para menos de 1 minuto (abaixo do limiar) sem que o controlador tenha tomado alguma ação, a responsabilidade pela resolução do conflito é transferida automaticamente para o TSAFE *Resolution*. (Resposta)
- Requisito A4 em Linguagem Controlada – Se a estimativa para perda de separação for menor do que 1 minuto e o controlador não tomar nenhuma ação então em resposta a resolução do conflito é responsabilidade do TSAFE *Resolution* eventualmente.

A tabela 8 apresenta as variáveis proposicionais do requisito A4 em linguagem natural controlada e a seguir as proposições em termos do modelo são apresentadas.

Tabela 8 - Requisito A4

Variável proposicional	Variável proposicional detalhada
a estimativa para perda de separação for menor do que 1 minuto ($P1$)	há conflito abaixo do limiar entre as aeronaves 1 e 2 ($P11$) e não há conflito abaixo do limiar entre as aeronaves 1 e 3 ($P12$) e não há conflito abaixo do limiar entre as aeronaves 2 e 3 ($P13$).
o controlador não tomar nenhuma ação ($P2$)	o controlador possui controle das aeronaves ($P21$) e não há comandos para a aeronave 1 ($P22$) e não há comandos para a aeronave 2 ($P23$).
a resolução do conflito é responsabilidade do TSAFE <i>Resolution</i> (S)	o TSAFE envia comando para a aeronave 1 ($S1$) ou o TSAFE envia comando para a aeronave 2

$$\begin{aligned}
P &: P1 \wedge P2 \\
P1 &: P11 \wedge P12 \wedge P13 \\
P2 &: P21 \wedge P22 \wedge P23 \\
P11 &: \text{env.tsafe_alert_12} = \text{BT} \\
P12 &: \text{env.tsafe_alert_13} \neq \text{BT} \\
P13 &: \text{env.tsafe_alert_23} \neq \text{BT} \\
P21 &: \text{controller.CTR_control} \\
P22 &: \neg \text{ac1.TSAFE_command} \\
P23 &: \neg \text{ac2.TSAFE_command} \\
S &: S1 \vee S2 \\
S1 &: \text{tsafe.TSAFE_command1} \\
S2 &: \text{tsafe.TSAFE_command2}
\end{aligned}$$

$$\begin{aligned}
&\Box((\text{env.tsafe_alert_12} = \text{BT} \wedge \text{env.tsafe_alert_13} \neq \text{BT} \wedge \\
&\text{env.tsafe_alert_23} \neq \text{BT} \wedge \text{controller.CTR_control} \wedge \\
&\neg \text{ac1.TSAFE_command} \wedge \neg \text{ac2.TSAFE_command}) \\
&\implies \Diamond(\text{tsafe.TSAFE_command1} \vee \text{tsafe.TSAFE_command2}))
\end{aligned} \tag{7}$$

- Requisito A5 – Após o TSAFE enviar uma resolução para a aeronave, uma mensagem no monitor do controlador indica quando o conflito foi resolvido e o controle da aeronave é transferido para o controlador, que fica responsável pela orientação da aeronave para a rota original. (Resposta Imediata)
- Requisito A5 em Linguagem Controlada – Se o comando enviado pelo TSAFE for realizado e o TSAFE possui o controle da aeronave então em resposta o TSAFE não possui o controle da aeronave imediatamente.

A tabela 9 apresenta as variáveis proposicionais do requisito A5 em linguagem natural controlada.

Tabela 9 - Requisito A5

Variável proposicional	Variável proposicional detalhada
o comando enviado pelo TSAFE for realizado ($P1$)	a aeronave 1 realizou o comando enviado pelo TSAFE ($P11$) e a aeronave 1 não possui comandos do TSAFE ($P12$).
o TSAFE possui o controle da aeronave ($P2$)	o TSAFE possui o controle da aeronave ($P2$).
o TSAFE não possui o controle da aeronave 1 (S)	o TSAFE não possui o controle da aeronave 1 (S)

$$P : P1 \wedge P2$$

$$P1 : P11 \wedge P12$$

$$P11 : ac1.TSAFE_command_done$$

$$P12 : \neg ac1.TSAFE_command$$

$$P2 : tsafe.TSAFE_control1$$

$$S : \neg tsafe.TSAFE_control1$$

$$\begin{aligned} & \Box(ac1.TSAFE_command_done \wedge \neg ac1.TSAFE_command \wedge tsafe.TSAFE_control1 \\ & \implies \bigcirc \neg tsafe.TSAFE_control1) \end{aligned} \quad (8)$$

- Requisito A6 – O piloto deve ser treinado para responder ao TSAFE e entender as condições que disparam um alerta TSAFE e TCAS. (Precedência - Após {Q} Até {R})
- Requisito A6a em Linguagem Controlada – Após o comando TCAS ser enviado até o comando TCAS ser realizado, se um comando TSAFE foi realizado então o comando TCAS deve ter sido realizado.

A tabela 10 apresenta as variáveis proposicionais do requisito A6a em linguagem natural controlada.

Tabela 10 - Requisito A6a

Q	o comando TCAS ser enviado
R	o comando TCAS ser realizado
P	um comando TSAFE foi realizado
S	o comando TCAS deve ter sido realizado

$Q : ac1.TCAS_command$

$R : ac1.TCAS_command_done$

$P : ac1.TSAFE_command_done$

$S : ac1.TCAS_command_done$

$$\begin{aligned}
 & \Box(ac1.TCAS_command \wedge \neg ac1.TCAS_command_done \\
 & \implies (\neg ac1.TSAFE_command_done \ U \\
 & (ac1.TCAS_command_done \vee ac1.TSAFE_command_done)))
 \end{aligned}
 \tag{9}$$

- Requisito A6b em Linguagem Controlada – Após o comando TSAFE ser enviado até o comando TSAFE ser realizado, se um comando enviado pelo controlador foi realizado então o comando TSAFE deve ter sido realizado.

A tabela 11 apresenta as variáveis proposicionais do requisito A6b em linguagem natural controlada.

Tabela 11 - Requisito A6b

Q	o comando TSAFE ser enviado
R	o comando TSAFE ser realizado
P	um comando enviado pelo controlador foi realizado
S	o comando TSAFE deve ter sido realizado

$$\begin{aligned}
Q &: \text{ac1.TSAFE_command} \\
R &: \text{ac1.TSAFE_command_done} \\
P &: \text{ac1.CTR_command_done} \\
S &: \text{ac1.TSAFE_command_done}
\end{aligned}$$

$$\begin{aligned}
&\Box(\text{ac1.TSAFE_command} \wedge \neg(\text{ac1.TSAFE_command_done}) \\
&\implies (\neg \text{ac1.CTR_command_done} \ U \\
&(\text{ac1.TSAFE_command_done} \vee \text{ac1.TSAFE_command_done})))
\end{aligned} \tag{10}$$

O requisito A6 possui duas expressões em LTL, uma para especificação das condições do TSAFE e outra para as condições do TCAS.

- Requisito A7 – Todos alertas detectados pelo TSAFE serão resolvidos eventualmente. (Resposta)
- Requisito A7 em Linguagem Controlada – Se o TSAFE emite um alerta entre duas aeronaves então em resposta este conflito é resolvido eventualmente.

A tabela 12 destaca os termos para o mapeamento de linguagem natural controlada para LTL.

Tabela 12 - Requisito A7

P	o TSAFE emite um alerta entre duas aeronaves
S	este conflito é resolvido

$$\begin{aligned}
P &: \text{env.tsafe_alert_12} \neq \text{Non} \\
S &: \text{env.tsafe_alert_12} = \text{Non}
\end{aligned}$$

$$\Box(\text{env.tsafe_alert_12} \neq \text{Non} \implies \Diamond \text{env.tsafe_alert_12} = \text{Non}) \tag{11}$$

- Requisito A8 – Caso o TSAFE assuma o controle de uma aeronave, eventualmente o controle será transferido para o controlador. (Resposta)

- Requisito A8 em Linguagem Controlada – Se o TSAFE possui o controle de uma aeronave então em resposta o TSAFE não possuirá o controle da aeronave eventualmente.

A tabela 13 destaca os termos para o mapeamento do requisito A8 de linguagem natural controlada para LTL.

Tabela 13 - Requisito A8

P	o TSAFE possui o controle de uma aeronave
S	o TSAFE não possuirá o controle da aeronave

$$P : \text{tsafe.TSAFE_control1}$$

$$S : \neg \text{tsafe.TSAFE_control1}$$

$$\Box(\text{tsafe.TSAFE_control1} \implies \Diamond \neg \text{tsafe.TSAFE_control1}) \quad (12)$$

- Requisito A9 – Caso o controlador transfira o controle de uma aeronave para o TSAFE, esta aeronave não pode executar comandos do controlador. (Precedência - Após {Q} Até {R})
- Requisito A9 em Linguagem Controlada – Após o controlador não possuir o controle sobre uma aeronave e o comando do controlador não tiver sido realizado até o controlador possuir o controle da aeronave, se o comando do controlador foi realizado então o comando do TSAFE deve ter sido realizado.

A tabela 14 destaca os termos para o mapeamento do requisito A9 de linguagem natural controlada para LTL.

Tabela 14 - Requisito A9

$Q1$	o controlador não possuir o controle sobre uma aeronave
$Q2$	o comando do controlador não tiver sido realizado
R	o controlador possuir o controle da aeronave
P	o comando do controlador foi realizado
S	o comando do TSAFE deve ter sido realizado

$$\begin{aligned}
Q &: Q1 \wedge Q2 \\
Q1 &: \neg \text{controller.CTR_control_1} \\
Q2 &: \neg \text{ac1.CTR_command_done} \\
R &: \text{controller.CTR_control_1} \\
P &: \text{ac1.CTR_command_done} \\
S &: \text{ac1.TSAFE_command_done}
\end{aligned}$$

$$\begin{aligned}
&\Box(((\neg \text{controller.CTR_control_1} \wedge \neg \text{ac1.CTR_command_done}) \wedge \\
&\neg \text{controller.CTR_control_1}) \\
&\implies (\neg \text{ac1.CTR_command_done} \mathcal{U}(\text{ac1.TSAFE_command_done} \vee \\
&\text{controller.CTR_control_1})))
\end{aligned} \tag{13}$$

- Requisito A10 – As resoluções do TSAFE mais antigas serão sempre executadas antes de resoluções mais recentes. (Precedência - Após {Q} Até {R})
- Requisito A10 em Linguagem Controlada – Após o TSAFE enviar um comando para a aeronave 1 e não houver comando TSAFE para a aeronave 2 até a aeronave 1 realizar o comando enviado pelo TSAFE, se a aeronave 2 realiza um comando enviado pelo TSAFE então a aeronave 1 deve ter realizado o comando enviado pelo TSAFE.

A tabela 15 apresenta as variáveis proposicionais do requisito A10 em linguagem natural controlada.

Tabela 15 - Requisito A10

Variável proposicional	Variável proposicional detalhada
o TSAFE enviar um comando para a aeronave 1 ($Q1$)	o TSAFE enviar um comando para a aeronave 1 ($Q1$)
não houver comando TSAFE para a aeronave 2 ($Q2$)	não houver comando TSAFE para a aeronave 2 ($Q21$) e o comando TSAFE não ter sido realizado ($Q22$)
a aeronave 1 realizar o comando enviado pelo TSAFE (R)	a aeronave 1 realizar o comando enviado pelo TSAFE (R)
a aeronave 2 realiza um comando enviado pelo TSAFE (P)	a aeronave 2 realiza um comando enviado pelo TSAFE (P)
a aeronave 1 deve ter realizado o comando enviado pelo TSAFE (S)	a aeronave 1 deve ter realizado o comando enviado pelo TSAFE (S)

$$Q : Q1 \wedge Q2$$

$$Q1 : ac1.TSAFE_command$$

$$Q2 : Q21 \wedge Q22$$

$$Q21 : \neg ac2.TSAFE_command$$

$$Q22 : \neg ac2.TSAFE_command_done$$

$$R : ac1.TSAFE_command_done$$

$$P : \neg(ac2.TSAFE_command_done$$

$$S : ac1.TSAFE_command_done$$

$$\begin{aligned}
& \Box(((ac1.TSAFE_command \wedge \neg ac2.TSAFE_command \wedge \\
& \neg ac2.TSAFE_command_done) \wedge \neg(ac1.TSAFE_command_done)) \\
& \implies (\neg(ac2.TSAFE_command_done) \ U \\
& ac1.TSAFE_command_done \vee ac1.TSAFE_command_done))
\end{aligned} \tag{14}$$

- Requisito A11 – Se um comando do TSAFE for enviado a uma aeronave, o controlador deve transferir o controle desta aeronave para o TSAFE. (Resposta Imediata)

- Requisito A11 em Linguagem Controlada – Se o TSAFE possui um comando para uma aeronave e o controlador possui controle sobre essa aeronave então em resposta o controlador não possuirá o controle sobre a aeronave imediatamente.

A tabela 16 destaca os termos para o mapeamento do requisito A11 de linguagem natural controlada para LTL.

Tabela 16 - Requisito A11

$P1$	o TSAFE possui um comando para uma aeronave
$P2$	o controlador possui controle sobre essa aeronave
S	o controlador não possuirá o controle sobre a aeronave

$$P : P1 \wedge P2$$

$$P1 : \text{tsafe.TSAFE_command1}$$

$$P2 : \text{controller.CTR_control_1}$$

$$S : \neg \text{controller.CTR_control_1}$$

$$\begin{aligned} & \Box(\text{tsafe.TSAFE_command1} \wedge \text{controller.CTR_control_1}) \\ & \implies \bigcirc \neg \text{controller.CTR_control_1} \end{aligned} \tag{15}$$

6.2.4 Estudo de Caso B – SATS

O modelo do SATS é composto pelos módulos **AMM**, **SCA**, **Aircraft** e **Environment**. No modelo completo, 5 aeronaves são modeladas devido ao requisito B1 que especifica que no máximo 4 aeronaves são autorizadas na SCA ao mesmo tempo.

O módulo **AMM** modela as lógicas para autorização de entrada vertical na SCA. O módulo tem acesso aos módulos de **Aircraft** para acessar informações sobre qual aeronave está solicitando autorização de entrada e os números de sequência das aeronaves.

A listagem 6.6 apresenta a definição do módulo e suas variáveis.

```

1 MODULE AMM (ac1, ac2, ac3, ac4, ac5)
2
3 VAR
4   landingNumber : 0..9;
5   scaSide : {LEFT, RIGHT};
6
7   landingNumber1 : 0..10;
```

```

8  scaSideAc1 : {LEFT, RIGHT, INDEF};
9  authorizesEntry1 : boolean;
10 authAc1 : boolean;
11
12 landingNumber2 : 0..10;
13 scaSideAc2 : {LEFT, RIGHT, INDEF};
14 authorizesEntry2 : boolean;
15 authAc2 : boolean;
16
17 landingNumber3 : 0..10;
18 scaSideAc3 : {LEFT, RIGHT, INDEF};
19 authorizesEntry3 : boolean;
20 authAc3 : boolean;
21
22 landingNumber4 : 0..10;
23 scaSideAc4 : {LEFT, RIGHT, INDEF};
24 authorizesEntry4 : boolean;
25 authAc4 : boolean;
26
27 landingNumber5 : 0..10;
28 scaSideAc5 : {LEFT, RIGHT, INDEF};
29 authorizesEntry5 : boolean;
30 authAc5 : boolean;
31
32 lock : boolean;

```

Listagem 6.6: Variáveis do Módulo AMM

As variáveis `landingNumber` e `scaSide` armazenam o último valor de sequência de pouso e o último valor de IAF atribuídos. As demais variáveis são acessadas pelas aeronaves para atribuição dos números de sequência de pouso, IAF/MAHF e autorização de entrada na SCA. A listagem 6.7 apresenta um trecho do código que define as transições das variáveis relacionadas aos valores da aeronave 1.

A seção **DEFINE** possui variáveis que indicam se a entrada em um determinado lado da SCA está disponível. A seção **ASSIGN** apresenta as transições. Quando as condições para entrada na SCA são satisfeitas, o módulo **AMM** atribui os valores para as variáveis de número de sequência e IAF e indica que a aeronave está autorizada a entrar. Existem trechos semelhantes para cada uma das aeronaves.

```

1
2 DEFINE
3   hasAvailableIAF := scaSide=LEFT ? hasAvailableIAFLeft :
   hasAvailableIAFRight;
4   hasAvailableIAFLeft := count(ac1.scaSide=LEFT, ac2.scaSide=LEFT,
5     ac3.scaSide=LEFT, ac4.scaSide=LEFT, ac5.scaSide=LEFT)<2;
6   hasAvailableIAFRight := count(ac1.scaSide=RIGHT, ac2.scaSide=RIGHT,
7     ac3.scaSide=RIGHT, ac4.scaSide=RIGHT, ac5.scaSide=RIGHT)<2;
8   issueLandingNumber1 := landingNumber1=0 & ac1.aircraftPos=NEARBY &
   hasAvailableIAF;
9
10 ASSIGN
11   init(landingNumber1) := 0;
12   next(landingNumber1) :=
13     case
14       issueLandingNumber1 : landingNumber+1;
15       TRUE : landingNumber1;
16     esac;
17   init(scaSideAc1) := INDEF;

```

```

18     next(scaSideAc1) :=
19         case
20             issueLandingNumber1 : scaSide;
21             TRUE : scaSideAc1;
22         esac;
23     authorizesEntry1 := landingNumber1!=0;
24     init(authAc1) := FALSE;
25     next(authAc1) :=
26         case
27             authAc1 : FALSE;
28             issueLandingNumber1 : TRUE;
29             TRUE : authAc1;
30         esac;

```

Listagem 6.7: Atribuição de valores

O módulo SCA possui as regiões descritas em 5.2. Cada região é modelada por um número inteiro, que corresponde à quantidade de aeronaves na região em um determinado instante. Espera-se que no máximo uma aeronave esteja em uma determinada região, portanto os valores para a ocupação são 0, 1 e 2. O módulo possui acesso às posições das aeronaves. Estas variáveis são usadas pelas aeronaves para as modelar transições entre as regiões da SCA. A listagem 6.8 apresenta este módulo.

A variável `numAircrafts` representa o número de aeronaves na SCA. Seu valor foi declarado entre 0 e 5, pois existem 5 aeronaves no modelo. As demais variáveis indicam as regiões da SCA. Para definir seu valor, utiliza-se a função `count` da linguagem do NuSMV, que recebe um conjunto de expressões booleanas e retorna a quantidade de expressões verdadeiras. A estrutura de “if-then-else” é usada para restringir os valores ao intervalo entre 0 e 2.

```

1  MODULE SCA(ac1Pos, ac2Pos, ac3Pos, ac4Pos, ac5Pos)
2      VAR
3          numAircrafts : 0..5;
4          l2000 : 0..2;
5          r2000 : 0..2;
6          l3000 : 0..2;
7          r3000 : 0..2;
8          inf : 0..2;
9          faf : 0..2;
10         rwy : 0..2;
11
12     DEFINE
13         countAircrafts := countL2000 + countL3000 + countR2000 +
14             countR3000 + countInf + countFaf + countRwy;
15         countL2000 := count(ac1Pos=L2000, ac2Pos=L2000,
16             ac3Pos=L2000, ac4Pos=L2000, ac5Pos=L2000);
17         countL3000 := count(ac1Pos=L3000, ac2Pos=L3000,
18             ac3Pos=L3000, ac4Pos=L3000, ac5Pos=L3000);
19         countR2000 := count(ac1Pos=R2000, ac2Pos=R2000,
20             ac3Pos=R2000, ac4Pos=R2000, ac5Pos=R2000);
21         countR3000 := count(ac1Pos=R3000, ac2Pos=R3000,
22             ac3Pos=R3000, ac4Pos=R3000, ac5Pos=R3000);
23         countInf := count(ac1Pos=InF, ac2Pos=InF, ac3Pos=InF,
24             ac4Pos=InF, ac5Pos=InF);
25         countFaf := count(ac1Pos=FAF, ac2Pos=FAF, ac3Pos=FAF,
26             ac4Pos=FAF, ac5Pos=FAF);
27         countRwy := count(ac1Pos=LAND, ac2Pos=LAND, ac3Pos=LAND,
28             ac4Pos=LAND, ac5Pos=LAND);

```



```

29
30  ASSIGN
31    numAircrafts := countAircrafts<=5 ? countAircrafts : 5;
32    l2000 := countL2000<=2 ? countL2000 : 2;
33    l3000 := countL3000<=2 ? countL3000 : 2;
34    r2000 := countR2000<=2 ? countR2000 : 2;
35    r3000 := countR3000<=2 ? countR3000 : 2;
36    inf := countInf<=2 ? countInf : 2;
37    faf := countFaf<=2 ? countFaf : 2;
38    rwy := countRwy<=2 ? countRwy : 2;

```

Listagem 6.8: Módulo AMM

O módulo **Environment** é usado para a simulação de chegada de aeronaves na região próxima ao SCA e a necessidade de procedimento de arremetida. As variáveis deste módulo são acessadas pelas aeronaves para indicar que a aeronave está numa região em que pode solicitar autorização de entrada à SCA, representada pela variável **aircraftNear**, e que deve realizar um procedimento de arremetida, representada pela variável **aircraftMA**. A listagem 6.9 apresenta o código deste módulo.

```

1  MODULE Environment ( )
2    VAR
3      aircraftNear : 0..5;
4      aircraft1Near : boolean;
5      aircraft2Near : boolean;
6      aircraft3Near : boolean;
7      aircraft4Near : boolean;
8      aircraft5Near : boolean;
9      aircraft1MA : boolean;
10     aircraft2MA : boolean;
11     aircraft3MA : boolean;
12     aircraft4MA : boolean;
13     aircraft5MA : boolean;
14
15  ASSIGN
16     init(aircraftNear) := 0;
17     next(aircraftNear) := 0..5;
18
19     init(aircraft1Near) := FALSE;
20     next(aircraft1Near) :=
21       case
22         aircraftNear=1 : TRUE;
23         TRUE : FALSE;
24       esac;
25
26     init(aircraft1MA) := FALSE;
27     next(aircraft1MA) := {TRUE, FALSE};

```

Listagem 6.9: Módulo Environment

Cada aeronave possui variáveis correspondentes para aproximação e arremetida. A variável **aircraftNear** em conjunto com as variáveis **aircraft1Near**, **aircraft2Near**, **aircraft3Near**, **aircraft4Near**, **aircraft5Near** estabelecem que apenas uma aeronave se aproxima por vez. Esta hipótese foi levantada pois não faz parte do escopo da verificação estudar como o AMM resolve a situação em que mais de uma aeronave solicita autorização de entrada ao mesmo tempo. De maneira não-determinística, uma aeronave é escolhida

para receber a *flag* de que está próxima a SCA. O valor da variável de arremetida também é atribuída não-deterministicamente. Para simplificação do modelo, estas variáveis podem assumir valor verdadeiro em diferentes cenários, sendo modeladas no módulo **Aircraft** as condições em que as variáveis devem ser usadas ou podem ser ignoradas.

```

1  MODULE Aircraft (auth, seqNbr, assignedSide, near, missedApproach, sca,
    leadingAcPos)
2  VAR
3  landingSequence : -1..10;
4  scaSide : {LEFT, RIGHT, INDEF};
5  aircraftPos : {OUT, NEARBY, AUTHORIZED, L2000, L3000, R2000, R3000,
6    InF, FAF, MA, MAHF_L2000, MAHF_L3000, MAHF_R2000, MAHF_R3000, LAND};

```

Listagem 6.10: Módulo Aircraft – Variáveis

O módulo **Aircraft** representa as aeronaves no modelo. Os elementos modelados por este módulo têm acesso a informação da ocupação do SCA, das notificações do AMM e a posição da aeronave-líder. A informação da ocupação do SATS não é algo previsto no conceito operacional. Para a modelagem do sistema, é necessário supor a hipótese de que os pilotos possuem informação da ocupação de todas regiões da SCA. Esta hipótese deverá ser considerada nas etapas de ajuste do ciclo de especificação. As regras do SATS para evolução de um voo dentro da SCA estão modeladas nas transições dos valores das variáveis deste módulo. As variáveis do módulo são apresentadas na listagem 6.10.

As variáveis do módulo correspondem às notificações enviadas pelo AMM e a posição da aeronave na SCA. Como sugerido por Muñoz, Carreño e Doweck (2006), o valor de IAF e MAHF serão os mesmos, sendo representados pela variável *scaSide*. As posições da aeronave se referem ou à região em que estão ou ao procedimento que estão fazendo. Por exemplo, se a aeronave está na região do FAF (fixo de aproximação final) e precisa realizar o procedimento de arremetida, sua posição assume o valor de MA, que representa o procedimento de arremetida, e posteriormente MAHF_L2000, MAHF_L3000, MAHF_R2000 ou MAHF_R3000. Estes valores não representam a região em que a aeronave está, mas o procedimento que está sendo realizado; MAHF_L2000 representa o procedimento de arremetida e subida ao IAF esquerdo a 2000 pés. Os valores possíveis são:

- OUT: a aeronave não se encontra próxima à SCA.
- NEARBY: a aeronave está em posição de solicitar autorização de entrada.
- AUTHORIZED: o AMM emitiu notificação de entrada para a aeronave.
- L3000: a aeronave está no IAF esquerdo a 3000 pés.
- R3000: a aeronave está no IAF direito a 3000 pés.

- L2000: a aeronave está no IAF esquerdo a 2000 pés.
- R2000: a aeronave está no IAF direito a 2000 pés.
- InF: a aeronave está em direção ao fixo de aproximação final.
- FAF: a aeronave está realizando o procedimento de aproximação final.
- LAND: a aeronave pousou.
- MA: a aeronave irá realizar o procedimento de arremetida.
- MAHF_L3000: a aeronave está realizando o procedimento de arremetida, em direção ao IAF esquerdo a 3000 pés.
- MAHF_R3000: a aeronave está realizando o procedimento de arremetida, em direção ao IAF direito a 3000 pés.
- MAHF_L2000: a aeronave está realizando o procedimento de arremetida, em direção ao IAF esquerdo a 2000 pés.
- MAHF_R2000: a aeronave está realizando o procedimento de arremetida, em direção ao IAF direito a 2000 pés.

A listagem 6.11 apresenta o trecho de código que define as transições de valores de `aircraftPos`.

A variável `resetCondition` na seção `DEFINE` indica que a aeronave pousou e deve ter seus valores reiniciados para representar uma nova aeronave chegando à `SCA`. As transições de `aircraftPos` dependem da posição da aeronave, a disponibilidade da região seguinte, dos valores atribuídos pelo `AMM` e da posição da aeronave-líder. A figura 6 apresenta os valores que `aircraftPos` pode assumir e as transições possíveis.

```

1 DEFINE
2   resetCondition := aircraftPos=LAND;
3
4 ASSIGN
5   init(aircraftPos) := OUT;
6   next(aircraftPos) :=
7     case
8       resetCondition : OUT;
9       aircraftPos=OUT & near : NEARBY;
10      aircraftPos=NEARBY & auth : AUTHORIZED;
11      aircraftPos=AUTHORIZED & scaSide=LEFT & sca.l3000=0 : L3000;
12      aircraftPos=AUTHORIZED & scaSide=RIGHT & sca.r3000=0 : R3000;
13      aircraftPos=L3000 & sca.l2000=0 : L2000;
14      aircraftPos=R3000 & sca.r2000=0 : R2000;
15      (aircraftPos=L2000 | aircraftPos=R2000) & sca.inf=0 &
16        (leadingAcPos=FAF | leadingAcPos=LAND) : InF;
17      (aircraftPos=InF | aircraftPos=FAF) & missedApproach : MA;
18      aircraftPos=InF & sca.faf=0 : FAF;
```

```

19     aircraftPos=FAF & sca.rwy=0 : LAND;
20     aircraftPos=MA & scaSide=LEFT & sca.l2000=0 : MAHF_L2000;
21     aircraftPos=MA & scaSide=LEFT & sca.l2000!=0 & sca.l3000=0 :
MAHF_L3000;
22     aircraftPos=MA & scaSide=RIGHT & sca.r2000=0 : MAHF_R2000;
23     aircraftPos=MA & scaSide=RIGHT & sca.r2000!=0 & sca.r3000=0 :
MAHF_R3000;
24     aircraftPos=MAHF_L2000 : L2000;
25     aircraftPos=MAHF_R2000 : R2000;
26     aircraftPos=MAHF_L3000 : L3000;
27     aircraftPos=MAHF_R3000 : R3000;
28     TRUE: aircraftPos;
29     esac;

```

Listagem 6.11: Transição aircraftPos

Figura 6 - Transição de valores da variável aircraftPos

O módulo `main` define a relação entre os elementos definidos pelos módulos apresentados, como apresentado na listagem 6.12.

A lógica para definição da posição da aeronave-líder está codificada neste módulo por motivos de organização, visto que este módulo possui acesso a todos elementos. O trecho de código apresentado na listagem 6.13 contém esta lógica, apresentando a definição da posição da aeronave-líder para as aeronaves 1 e 2.

```

1  MODULE main
2
3  VAR
4      ac1 : Aircraft (amm.authAc1, amm.landingNumber1, amm.scaSideAc1,
5                      env.aircraft1Near, env.aircraft1MA, sca, ac1LeadingPos);
6      ac2 : Aircraft (amm.authAc2, amm.landingNumber2, amm.scaSideAc2,
7                      env.aircraft2Near, env.aircraft2MA, sca, ac2LeadingPos);
8      ac3 : Aircraft (amm.authAc3, amm.landingNumber3, amm.scaSideAc3,

```

```

9     env.aircraft3Near, env.aircraft3MA, sca, ac3LeadingPos);
10    ac4 : Aircraft (amm.authAc4, amm.landingNumber4, amm.scaSideAc4,
11     env.aircraft4Near, env.aircraft4MA, sca, ac4LeadingPos);
12    ac5 : Aircraft (amm.authAc5, amm.landingNumber5, amm.scaSideAc5,
13     env.aircraft5Near, env.aircraft5MA, sca, ac5LeadingPos);
14    env : Environment ();
15    amm : AMM (ac1, ac2, ac3, ac4, ac5);
16    sca : SCA (ac1.aircraftPos, ac2.aircraftPos, ac3.aircraftPos,
17     ac4.aircraftPos, ac5.aircraftPos);

```

Listagem 6.12: Módulo main

```

1  DEFINE
2    ac1LeadingPos :=
3      case
4        ac1.landingSequence=-1 : LAND;
5        (ac1.landingSequence - 1)=ac2.landingSequence : ac2.aircraftPos;
6        (ac1.landingSequence - 1)=ac3.landingSequence : ac3.aircraftPos;
7        (ac1.landingSequence - 1)=ac4.landingSequence : ac4.aircraftPos;
8        (ac1.landingSequence - 1)=ac5.landingSequence : ac5.aircraftPos;
9        TRUE : LAND;
10     esac;
11    ac2LeadingPos :=
12      case
13        ac2.landingSequence=-1 : LAND;
14        (ac2.landingSequence - 1)=ac1.landingSequence : ac1.aircraftPos;
15        (ac2.landingSequence - 1)=ac3.landingSequence : ac3.aircraftPos;
16        (ac2.landingSequence - 1)=ac4.landingSequence : ac4.aircraftPos;
17        (ac2.landingSequence - 1)=ac5.landingSequence : ac5.aircraftPos;
18        TRUE : LAND;
19     esac;

```

Listagem 6.13: Posição da aeronave-líder

Usando os valores de sequência de pouso, é analisado qual aeronave possui o número de sequência de pouso anterior, por exemplo, se a aeronave 1 possui número de sequência 5, é analisado qual aeronave possui o número de sequência 4. Caso não seja encontrada, assume-se que a aeronave não possui líder, e no modelo isto é representado pelo estado em que a posição da aeronave-líder é LAND.

Os requisitos de propriedades do SATS são mapeados de acordo com o padrão de especificação em que foram classificados.

- Requisito B1 – Não são permitidas mais do que 4 aeronaves realizando o procedimento de pouso na SCA. (Ausência – Global)
- Requisito B1 em Linguagem Controlada – Nunca ocorre que o número de aeronaves dentro da SCA seja maior do que 4.

$P : \text{sca.numAircrafts} > 4$

$$\Box(\neg(\text{sca.numAircrafts} > 4)) \quad (16)$$

- Requisito B2 – Não são permitidas entradas simultâneas em um IAF. (Ausência - Após {Q} Até {R})
- Requisito B2 em Linguagem Controlada – Após a autorização de entrada de uma aeronave até a aeronave entrar na SCA, não ocorre de outra aeronave ter a entrada autorizada.

A tabela 17 destaca os termos para o mapeamento do requisito B2 de linguagem natural controlada para LTL.

Tabela 17 - Requisito B2

Q	a autorização de entrada de uma aeronave
R	a aeronave entrar na SCA
P	outra aeronave ter a entrada autorizada

$$Q : \text{amm.authAc1}$$

$$R : \neg \text{amm.authAc1}$$

$$P : \text{amm.authAc2}$$

$$\Box(\text{amm.authAc1} \implies (\neg \text{amm.authAc2} \, U \, \neg \text{amm.authAc1})) \quad (17)$$

- Requisito B3 – Após entrar na SCA, as aeronaves devem iniciar o procedimento de descida apenas quando a região imediatamente inferior estiver disponível. (Ausência - Após {Q} Até {R})
- Requisito B3 em Linguagem Controlada – Após a aeronave chegar ao fixo a 3000 pés até o fixo de 2000 pés estar desocupado, não ocorre da aeronave chegar ao fixo de 2000 pés.

A tabela 18 destaca os termos para o mapeamento do requisito B3 de linguagem natural controlada para LTL.

Tabela 18 - Requisito B3

Q	a aeronave chegar ao fixo a 3000 pés
R	o fixo de 2000 pés estar desocupado
P	a aeronave chegar ao fixo de 2000 pés

$$Q : \text{ac1.aircraftPos} = \text{L3000}$$

$$R : \text{sca.l2000} = 0$$

$$P : \text{ac1.aircraftPos} = \text{L2000}$$

$$\begin{aligned} & \Box(\text{ac1.aircraftPos} = \text{L3000} \wedge \neg(\text{sca.l2000} = 0)) \\ & \implies (\neg \text{ac1.aircraftPos} = \text{L2000} \text{ U } \text{sca.l2000} = 0) \end{aligned} \quad (18)$$

- Requisito B4 – Os MAHF devem ser atribuídos de maneira alternada para aeronaves em sequência. (Ausência – Global)
- Requisito B4 em Linguagem Controlada – Nunca ocorre de 2 aeronaves com números de sequência de pouso consecutivos possuírem o mesmo MAHF atribuído.

A tabela 19 apresenta as variáveis proposicionais do requisito em linguagem natural controlada de maneira mais detalhada para o mapeamento da escrita da expressão LTL correspondente.

Tabela 19 - Requisito B4

Variável proposicional	Variável proposicional detalhada
2 aeronaves com números de sequência de pouso consecutivos possuírem o mesmo MAHF atribuído (P)	Os números de sequência de pouso da aeronave 1 e a da aeronave 2 são consecutivos ($P1$) e A aeronave 1 possui número de sequência de pouso ($P2$) e a aeronave 2 possui número de sequência de pouso ($P3$) e o MAHF da aeronave 1 é diferente do MAHF da aeronave 2 ($P4$).

As proposições em termos do modelo podem ser escritas como:

$$\begin{aligned}
 P &: P1 \wedge P2 \wedge P3 \wedge P4 \\
 P1 &: \neg(|ac1.landingSequence - ac2.landingSequence| = 1) \\
 P2 &: ac1.landingSequence \neq -1 \\
 P3 &: ac2.landingSequence \neq -1 \\
 P4 &: ac1.scaSide \neq ac2.scaSide
 \end{aligned}$$

$$\begin{aligned}
 &\Box(\neg((|ac1.landingSequence - ac2.landingSequence| = 1) \wedge \\
 &ac1.landingSequence \neq -1 \wedge ac2.landingSequence \neq -1 \wedge \\
 &(ac1.scaSide \neq ac2.scaSide))) \quad (19)
 \end{aligned}$$

- Requisito B5 – Após arremetida as aeronaves devem iniciar procedimento de subida até o nível mais inferior disponível do MAHF atribuído. (Resposta)
- Requisito B5a em Linguagem Controlada – Se a aeronave realiza o procedimento de arremetida e o fixo a 2000 pés está disponível então em resposta a aeronave vai para o fixo a 2000 pés eventualmente.

A tabela 20 apresenta as variáveis proposicionais do requisito B5a em linguagem natural controlada e a seguir as proposições em termos do modelo são apresentadas.

Tabela 20 - Requisito B5a

Variável proposicional	Variável proposicional detalhada
a aeronave realiza o procedimento de arremetida ($P1$)	a aeronave 1 está iniciando o procedimento de arremetida ($P11$) e o MAHF da aeronave 1 é o lado esquerdo ($P12$)
fixo a 2000 pés está disponível ($P2$)	fixo a 2000 pés está disponível ($P2$)
a aeronave vai para o fixo a 2000 pés (S)	a aeronave vai para o fixo a 2000 pés

$$\begin{aligned}
P &: P1 \wedge P2 \\
P1 &: P11 \wedge P12 \\
P11 &: \text{ac1.aircraftPos} = \text{MA} \\
P12 &: \text{ac1.scaSide} = \text{LEFT} \\
P2 &: \text{sca.l2000} = 0 \\
S &: \text{ac1.aircraftPos} = \text{L2000}
\end{aligned}$$

A expressão em LTL para o requisito B5a é apresentada a seguir.

$$\begin{aligned}
&\Box(\text{ac1.aircraftPos} = \text{MA} \wedge \text{ac1.scaSide} = \text{LEFT} \wedge \text{sca.l2000} = 0 \\
&\implies \Diamond(\text{ac1.aircraftPos} = \text{L2000}))
\end{aligned} \tag{20}$$

- Requisito B5b em Linguagem Controlada – Se a aeronave realiza o procedimento de arremetida e o fixo a 2000 pés não está disponível então em resposta a aeronave vai para o fixo a 3000 pés eventualmente.

A tabela 21 apresenta as variáveis proposicionais do requisito B5b em linguagem natural controlada.

Tabela 21 - Requisito B5b

Variável proposicional	Variável proposicional detalhada
a aeronave realiza o procedimento de arremetida ($P1$)	a aeronave 1 está iniciando o procedimento de arremetida ($P11$) e o MAHF da aeronave 1 é o lado esquerdo ($P12$)
fixo a 2000 pés não está disponível ($P2$)	fixo a 2000 pés não está disponível ($P2$)
a aeronave vai para o fixo a 3000 pés (S)	a aeronave vai para o fixo a 3000 pés

$$\begin{aligned}
P &: P1 \wedge P2 \\
P1 &: P11 \wedge P12 \\
P11 &: \text{ac1.aircraftPos} = \text{MA} \\
P12 &: \text{ac1.scaSide} = \text{LEFT} \\
P2 &: \text{sca.l2000} \neq 0 \\
S &: \text{ac1.aircraftPos} = \text{L3000}
\end{aligned}$$

A expressão em LTL para o requisito B5b é apresentada a seguir.

$$\begin{aligned}
&\Box(\text{ac1.aircraftPos} = \text{MA} \wedge \text{ac1.scaSide} = \text{LEFT} \wedge \text{sca.l2000} \neq 0 \\
&\implies \Diamond(\text{ac1.aircraftPos} = \text{L3000}))
\end{aligned} \tag{21}$$

O requisito B5 é mapeado para duas proposições: a primeira especifica o caso em que o nível mais inferior disponível é o de 2000 pés e a segunda especifica que o nível disponível é o de 3000 pés.

O modelo completo não pôde ser verificado pelo verificador NuSMV devido a sua complexidade, ocorrendo um problema recorrente em verificadores do tipo verificação de modelos: o problema da explosão de estados. Devido a este problema, simplificações no modelo foram feitas e são apresentadas a seguir.

Para verificar as especificações dos requisitos B2, B3 B4 e B5 as seguintes alterações no modelo foram realizadas. Os valores das aeronaves não são reiniciados, logo não há a possibilidade de verificar como o modelo se comporta em um cenário em que um número indefinido de aeronaves se aproxima da SCA. A quantidade máxima de aeronaves que se aproximam da SCA por simulação corresponde ao número de aeronaves modeladas. O número de aeronaves modeladas foi reduzido para 3. Para a verificação dos requisitos B2, B3, B4 e B5 são necessárias 3 aeronaves para simular cenários em que mais de um lado de entrada da SCA é utilizado e mais de um nível dos IAF estão ocupados. Com estas simplificações, o intervalo para a variável correspondente ao número de sequência de pouso passou do intervalo de 0 a 10 para o intervalo de 0 a 4.

Para a verificação da especificação do requisito B1 o número de aeronaves modeladas foi reduzido para três e os IAF e MAHF de 3000 pés foram removidos do modelo. Com isso, o requisito teve que ser trocado. O requisito especifica que o número máximo de aeronaves na SCA é 4 pois existem 4 IAFs na SCA. Após as alterações no modelo, passam

a existir 2 IAFs na SCA, logo o requisito foi alterado para se adequar a este novo cenário. O requisito alterado B1' é apresentado a seguir.

- Requisito B1' – Não são permitidas mais do que 2 aeronaves realizando o procedimento de pouso na SCA. (Ausência – Global)
- Requisito B1' em Linguagem Controlada – Nunca ocorre que o número de aeronaves dentro da SCA seja maior do que 2.

$$\Box(\neg(\text{sca.numAircrafts} > 2)) \quad (22)$$

6.2.5 Conclusões do Capítulo

O primeiro ponto de discussão destas diretrizes é a fonte de requisitos. Existem ao menos duas fontes relevantes: documentos que especificam o sistema e a expectativa dos desenvolvedores do sistema. Espera-se que a expectativa dos demais envolvidos com o sistema (clientes, usuários) estejam contidas na expectativa dos desenvolvedores.

O levantamento do conjunto de requisitos deve considerar quais os objetivos da especificação. No caso, a especificação é feita para a verificação formal de requisitos do sistema. Portanto deve-se levar em consideração quais requisitos são relevantes. Por exemplo, propriedades como desempenho da aeronave (visto no estudo de caso B) a princípio não fazem parte do escopo da verificação realizada neste trabalho. Requisitos de sistema correspondem a especificações que definem o que o sistema deve fazer ou deve ser capaz de fazer, independente do nível de abstração da especificação.

Diretrizes para a etapa de modelagem foram analisadas dos trabalhos de Arcaini, Gargantini e Riccobene (2017) e no estudo de caso A e aplicadas ao estudo de caso B. O modelo do sistema representado por uma máquina de estados finita é adequado para sistemas reativos. A FSM é construída a partir do comportamento do sistema, que define os estados e as transições entre os estados. Cada elemento com comportamento distinto deve ser definido por um módulo distinto. No caso específico do ciclo de especificação proposto neste trabalho, os módulos devem ser definidos usando as seções VAR, DEFINE, ASSIGN e INVAR e as relações entre os elementos do sistema deve ser modelada na módulo main do modelo para NuSMV.

As proposições que serão usadas para a verificação formal foram elaboradas com base nos padrões de especificação definidos.

Como o mapeamento de linguagem natural controlada para linguagem de especificação formal é definido, a elaboração destas proposições em linguagem de especificação formal poderia ser automatizada para o conjunto de requisitos especificados em linguagem natural controlada.

Nestas etapas, pode haver introdução de erros no modelo e nas proposições durante a coleta e a classificação de requisitos. Como o conjunto de requisitos usados para a modelagem do sistema e para elaboração das proposições não é necessariamente o mesmo, espera-se que estes erros sejam detectados com a verificação formal, porém existe a possibilidade de um mesmo erro ser introduzido no modelo e nas proposições. A consulta a desenvolvedores do sistema com relação a interpretação dos requisitos para sua correta classificação pode minimizar este risco.

Zhao e Rozier (2014) fazem uma discussão sobre métodos para avaliar se as especificações formais estão corretas (com técnicas de *sanity checking*), porém esta discussão será simplificada neste texto, na seção 7.3.3. Além deste tipo de técnica, são definidas proposições que representam as meta-propriedades que o modelo deve possuir.

O modelo do estudo de caso B e um dos requisitos foram reformulados para que o processo de verificação formal com o verificador NuSMV fosse finalizado. Especialistas no sistema deveriam ser consultados para avaliar se estas alterações no modelo são coerentes com o sistema modelado. Com base no documento de especificação do conceito operacional do SATS, conclui-se que estas alterações são coerentes com o SATS, porém caso os desenvolvedores do sistema não estejam de acordo com as alterações, outras hipóteses teriam que ser propostas para simplificar o modelo. Para a modelagem do SATS, foi suposta a hipótese de que os pilotos possuem informação de posição de todas aeronaves dentro da SCA. Esta hipótese é necessária para modelagem do sistema e deverá ser considerada nas etapas posteriores do ciclo de especificação, como será apresentado no capítulo 7.

7 VERIFICAÇÃO FORMAL

Neste capítulo são apresentadas as diretrizes e táticas relacionadas à verificação formal do modelo. As atividades nestas etapas do ciclo de especificação são:

- uso do verificador NuSMV;
- exemplo de como analisar os resultados;
- ajustes nas proposições em LTL e CTL ou no modelo;
- ajustes nos requisitos e especificação de sistema em linguagem natural.

7.1 Verificação com NuSMV e Resultados de Verificação

A ferramenta de verificação formal NuSMV é um verificador de modelos simbólico, utilizado para verificação de sistemas. O verificador lê um modelo descrito por meio de um arquivo *script* e pode executar o modelo de modos diferentes: simulação interativa, simulação automática e verificação. No modo de simulação interativa, o usuário pode escolher o próximo estado a partir de um conjunto de estados possíveis. Na simulação automática, o NuSMV escolhe de maneira automática o próximo estado. No modo de verificação, verifica-se se o modelo satisfaz proposições em LTL ou CTL. Caso a proposição seja satisfeita, uma mensagem de sucesso é apresentada; caso contrário, um contra-exemplo é apresentado. O contra-exemplo é um exemplo de execução possível do modelo em que a proposição não é satisfeita.

No ciclo de especificação, o NuSMV é utilizado para verificar se o modelo satisfaz as proposições em LTL e CTL mapeadas dos requisitos do sistema. A ferramenta apresenta um resultado para cada proposição. Este resultado pode indicar que o modelo satisfaz ou não a proposição. Caso o modelo satisfaça a proposição, deve-se avaliar se não é um falso positivo, ou como definido em Arcaini, Gargantini e Riccobene (2017), ser satisfeita de maneira vazia (*vacuously*). Ser satisfeita de maneira vazia significa que a proposição está especificada de maneira errada e falta um comportamento no modelo que evidencie este erro. Caso o modelo não satisfaça a proposição, deve-se analisar o contra-exemplo e definir o motivo da inconsistência entre modelo e proposição. O termo inconsistência será usado para se referir a não satisfabilidade de uma proposição pelo modelo.

Portanto, os possíveis resultados da verificação formal são:

- Proposição satisfeita

- Proposição correta
- Satisfeita de maneira vazia
- Proposição não-satisfeita
 - Proposição incorreta
 - Modelo incorreto
 - Proposição e modelo incorretos

Uma proposta para avaliar se a proposição foi satisfeita de maneira vazia é analisar se a expressão pode ser avaliada como falsa em algum estado do modelo. Considerando o cenário em que a proposição possui uma precondição, caso a precondição nunca seja verdadeira, o modelo sempre irá satisfazer a proposição. Em proposições com o operador de implicação do tipo

$$p \implies q \tag{23}$$

se p sempre for falsa (a precondição nunca é satisfeita), a proposição será sempre satisfeita pelo modelo. Logo é interessante verificar se p é eventualmente satisfeita em alguma execução do modelo, com a expressão em CTL:

$$EF(p) \tag{24}$$

A expressão 24 verifica se a expressão p pode ser verdadeira em alguma execução. Caso não possa, $p \implies q$ será sempre verdadeira, o que pode indicar que o modelo ou a proposição possuem erros.

A análise do contra-exemplo é feita em duas etapas: uma em termos do modelo – interpretação da execução do modelo em que as variáveis assumiram valores que não satisfazem a proposição – e em termos do sistema – interpretação do cenário de operação do sistema em que o contra-exemplo ocorre. Por meio desta análise, é possível definir se o modelo está errado, a proposição está errada ou ambos estão errados. Caso o cenário de operação e a execução do modelo sejam válidos, provavelmente a proposição está mal especificada ou o requisito não é válido. Caso a execução do modelo não seja válida, o modelo está errado; deve-se analisar se o modelo foi elaborado corretamente, o que indicaria que a especificação de sistema possui erros de projeto, ou se o modelo não possui todos comportamentos da especificação do sistema.

Em seu modo padrão o *software* do NuSMV é executado pelo terminal e verifica se o modelo satisfaz as proposições. Zhao e Rozier (2014) sugerem executar o NuSMV

com a opção “-df” para melhorar a performance do verificador. Esta opção desabilita a computação do conjunto de estados alcançáveis. Segundo o manual do NuSMV, esta computação pode melhorar o desempenho de modelos com espaço de estados esparsos, porém pode ser muito custosa. Em uma análise comparativa, ambos estudos de caso gastam menos tempo caso o NuSMV seja executado com a opção “-df”. Outra opção utilizada nos estudos de caso foi a configuração do nível de verbosidade para 1. Neste nível, o programa exibe qual proposição está sendo verificada no instante.

7.1.1 Estudo de Caso A – Coordenação AAC

No trabalho de Zhao e Rozier (2014) as proposições referentes ao requisitos A9 e A10 não foram satisfeitas pelo modelo. No presente trabalho, o resultado foi semelhante. O requisito A9 será investigado nesta e nas seções seguintes.

O requisito A9 especifica que caso o controlador transfira o controle para o TSAFE, as aeronaves cujo controle foi transferido não devem realizar um comando proveniente do controlador. Um contra-exemplo que o NuSMV apresenta é esquematizado na Figura 7.

Figura 7 - Contra-exemplo do requisito A9

Neste contra-exemplo, o controlador recebe um alerta do TSAFE acima do limiar. O controlador decide enviar um comando para a aeronave e no instante seguinte, transferir o controle para o TSAFE. No instante que o TSAFE está enviando o comando para a

aeronave, esta já iniciou o comando enviado pelo controlador, que transferiu o controle do voo, o que viola a proposição.

Em termos do modelo, este cenário ocorreu pois o valor da variável `tsafe_sndcmd` foi alterada para verdadeiro, porém a aeronave não possui esta informação. Em termos do sistema, este cenário pode ocorrer caso a aeronave não possua informação referente a qual sistema a está controlando antes da realização de um comando.

Outro contra-exemplo é apresentado em Zhao e Rozier (2014). O controlador recebe um alerta de conflito acima do limiar para as aeronaves 1 e 2, e decide enviar um comando para a aeronave 1. Logo após, o TSAFE recebe um alerta de conflito entre as aeronaves 1 e 3 abaixo do limiar. Portanto, o TSAFE informa ao controlador que vai assumir o controle das aeronaves 1 e 3. O controlador transfere o controle e o TSAFE envia uma resolução para a aeronave 3. Porém, como a aeronave 1 não recebeu um comando do TSAFE, seu comando com mais alta prioridade é o comando proveniente do controlador. A aeronave 1 realiza o comando do controlador, mesmo com a transferência de controle para o TSAFE, o que viola a proposição.

Em termos do modelo, a execução é possível pois apesar de estar sob controle do TSAFE, a variável `tsafe_cmd` da aeronave 1 era falsa. Em termos do sistema real, o cenário ocorre porque a aeronave 1 não possuía comando de prioridade maior que a do controlador.

Com esta interpretação é possível entender porque o modelo não satisfaz a proposição e com base na análise, ajustes no modelo são propostos para que a proposição seja satisfeita pelo modelo. Neste caso, concluiu-se que: o modelo não possui um comportamento necessário para satisfazer a proposição; e a proposição especifica um requisito do sistema. Portanto o ajuste será feito apenas no modelo.

7.1.2 Estudo de Caso B – SATS

Por meio da verificação formal, foi possível avaliar que o modelo do SATS satisfaz as proposições referentes aos requisitos B2 a B5, porém o modelo foi simplificado para reduzir o tempo de verificação a um tempo razoável, visando o uso do ciclo de especificação como ferramenta para o projeto de sistemas, como discutido em Zhao e Rozier (2014).

As alterações a seguir foram propostas devido ao tempo de verificação. No modelo, o SATS recebe apenas 3 voos, *i.e.*, após cada aeronave atingir o estado de pouso, ela não é reiniciada como se estivesse se aproximando da SCA. Da mesma maneira, ao realizar o procedimento de arremetida, a aeronave faz a transição para a região atribuída, mas sai da simulação, como se tivesse pousado. O procedimento de arremetida como especificado em Abbott et al. (2004) só foi modelado com 3 aeronaves, por isso não foi possível verificar a

proposição referente ao requisito B1. Uma versão alterada do requisito B1 foi verificada. Esta alteração exigiu adaptações no modelo e na especificação de sistema do SATS. Estas alterações foram discutidas na seção 6.2.4.

7.2 Ajuste – Especificação Formal e Modelo

Após a execução do NuSMV o resultado da verificação pode gerar dois conjuntos: proposições satisfeitas e proposições não satisfeitas pelo modelo. Esta etapa trabalha com os resultados satisfeitos de maneira vazia e com o conjunto de proposições não satisfeitas, de modo a alterar o modelo ou a proposição para torná-los consistentes.

Se na análise do resultado de verificação concluiu-se que a proposição estava errada, deve-se avaliar se foi corretamente mapeada seguindo o padrão de especificação em que foi classificada e se o padrão escolhido é o padrão correto.

Caso a conclusão da análise indique que o modelo está errado, o modelo deverá ser alterado. A diretriz para ajuste da especificação formal consiste em descobrir as causas das inconsistências entre modelo e proposições. Esta diretriz analisa o contra-exemplo fornecido pelo NuSMV. A partir deste contra-exemplo, deve-se analisar quais variáveis assumiram o valor responsável pela inconsistência, que serão referenciadas como variáveis dependentes, e identificar quais variáveis assumiram valores que conduziram o sistema ao estado do sistema com a inconsistência, que serão referenciadas como variáveis independentes. A alteração deve introduzir as variáveis independentes à transição das variáveis dependentes.

Além da análise dos contra-exemplos, deve-se avaliar se as hipóteses supostas no processo de modelagem não influenciaram no resultado (satisfação ou não satisfação) da verificação formal. Se uma proposição é satisfeita se e somente se a hipótese for considerada, então a hipótese ou parte dela deve ser classificada como ajuste da especificação formal. Caso algum ajuste seja feito, é necessário verificar novamente o modelo.

7.2.1 Estudo de Caso A – Coordenação AAC

Esta seção irá apresentar uma proposta de solução para a inconsistência do requisito A9.

Com relação ao contra-exemplo relatado em Zhao e Rozier (2014), os autores sugerem que o TSAFE envie um comando para a aeronave que realizará a manobra de resolução e um comando para a outra aeronave “manter a rota”. Esta alteração não foi feita no modelo em Zhao e Rozier (2014). O modelo foi alterado para este estudo de caso, porém não alcançou o resultado desejado. A alteração foi feita na transição das variáveis `TSAFE_commandX`,

especificando o envio de comandos TSAFE para ambas aeronaves envolvidas no conflito. O modelo alterado continuou não satisfazendo a proposição, apresentando o contra-exemplo descrito pela figura 7.

O cenário descrito pelo contra-exemplo foi analisado para impedir sua ocorrência. O cenário ocorre pois a aeronave não possui informação da transferência de seu controle do controlador para o TSAFE antes de executar o comando proveniente do controlador. Para evitar este cenário, a seguinte alteração foi proposta: antes de executar um comando proveniente do controlador, a aeronave deve verificar se o controlador possui o controle da aeronave e se o controlador não irá transferir o controle para o TSAFE. A alteração no modelo é apresentada a seguir. O código do módulo **Aircraft** original de Zhao e Rozier (2014) é apresentado na listagem 7.1.

```

1 next(CTR_command_done) :=
2 case
3 !TCAS_command & !TSAFE_command & CTR_command : TRUE;
4 TRUE: FALSE;
5 esac;
```

Listagem 7.1: Módulo Aircraft – Original

Neste caso, a aeronave realiza o comando se não houver comandos de maior prioridade disponíveis. A alteração proposta é apresentada na listagem 7.2.

```

1 next(CTR_command_done) :=
2 case
3 !TCAS_command & !TSAFE_command & CTR_command & enable_ctr_cmd : TRUE;
4 TRUE: FALSE;
5 esac;
6
7 DEFINE
8 enable_ctr_cmd := (CTR_on_control & !Control_hands_off);
```

Listagem 7.2: Módulo Aircraft – Alterado

Neste caso, a aeronave realiza o comando se não houver comandos de maior prioridade disponíveis, se o controlador estiver controlando a aeronave e se a transferência de controle não foi proposta. A variável **CTR_on_control** é uma booleana que indica se a aeronave está sob controle do módulo **Controller** e não do TSAFE, e a variável **Control_hands_off** é uma booleana que indica se o controlador irá transferir o controle da aeronave para o TSAFE. O valor de **Control_hands_off** é o mesmo valor de **tsafe_sndcmd**, apresentado na seção 6.2.5.

Esta alteração combinada com a alteração proposta em Zhao e Rozier (2014) resultou em um modelo que satisfaz a proposição referente ao requisito A9. É necessário verificar se as proposições que eram satisfeitas antes das alterações apresentam o mesmo resultado após as alterações. Realizando a verificação formal, verificou-se que todas as outras proposições

continuam sendo satisfeitas pelo modelo.

7.2.2 Estudo de Caso B – SATS

Do conjunto de proposições verificadas, requisitos B1 a B5, a verificação do modelo não apresentou proposições não satisfeitas. Analisando as hipóteses propostas na modelagem do sistema, nota-se que a hipótese de que as aeronaves possuem informação sobre a ocupação das regiões da SCA influencia neste resultado. Caso essa hipótese não fosse proposta, a aeronave não possuiria a informação sobre qual o nível inferior disponível em seu MAHF. Como proposto em Abbott et al. (2004) a aeronave deve usar como referência a posição de sua aeronave-líder, porém no caso de um procedimento de arremetida, a aeronave não deve usar como referência a aeronave-líder para avaliar a disponibilidade do MAHF. No modelo, isto foi representado pela SCA estar configurada como parâmetro do módulo *Aircraft*.

Neste caso, a hipótese introduziu um comportamento no modelo que permitiu a satisfação de uma proposição. Parte desta hipótese deve ser então proposta para introdução na especificação do sistema. Caso esta hipótese implicasse na elaboração de expressão em LTL ou CTL, a hipótese seria introduzida no conjunto de requisitos do sistema.

7.3 Ajuste – Linguagem Natural

Nesta etapa, os ajustes feitos nas proposições e no modelo devem ser refletidos nos requisitos de sistema e na especificação de sistema em linguagem natural. Os ajustes feitos nas proposições devem resultar em ajustes nos requisitos de sistema e os ajustes no modelo devem resultar em ajustes na especificação de sistema. O ajuste deve ser contextualizado em um cenário de operação do sistema para então propor alterações na especificação em linguagem natural. Estas alterações podem ser adição ou alteração de requisitos e especificação.

7.3.1 Estudo de Caso A – Coordenação AAC

A solução proposta por Zhao e Rozier (2014) de adicionar um comando de “manter rota” foi discutida com os desenvolvedores do AAC. Eles aceitaram a proposta e adicionaram o comando de “manter rota” ao conceito operacional do AAC.

No caso da proposta apresentada na seção anterior, a especificação em linguagem natural seria: “A aeronave deve possuir informações sobre quem a está controlando no momento

e se haverá troca de controle” e “O piloto deve executar um comando proveniente do controlador caso não haja comandos provenientes do TSAFE e do TCAS e se a aeronave estiver sob controle do controlador e não estiver no meio de um processo de transferência de controle do controlador para o TSAFE”. A primeira expressão deve ser introduzida como uma nova especificação de sistema. A segunda define uma propriedade do sistema, portanto é introduzida como requisito de sistema. A tabela 22 apresenta o conjunto de requisitos do AAC após a aplicação do ciclo de especificação.

Mesmo que as alterações feitas resultem em um modelo que satisfaz as proposições baseadas nos requisitos de propriedades do sistema, deve-se avaliar se a mudança é viável no mundo real. Para isso é necessário consultar fontes de requisitos. A princípio este requisito não é incoerente com nenhuma descrição disponível em Erzberger e Heere (2010). Antes de consolidar este como um novo requisito do AAC, seria necessário consultar os desenvolvedores do AAC.

Este novo conjunto de requisitos pode ser verificado em uma nova iteração do ciclo de especificação. O requisito A12 seria classificado no padrão de especificação de resposta global.

7.3.2 Estudo de Caso B – SATS

Como apresentado, a hipótese de que a aeronave possui informações sobre a ocupação das regiões da SCA deve ser considerada como ajuste do modelo. O contexto de operação ocorre durante o procedimento de arremetida. Neste procedimento, a aeronave deve seguir para a região mais inferior disponível do MAHF atribuído. Sem a hipótese, a aeronave não poderia avaliar qual a disponibilidade destas regiões. Talvez na especificação do conceito operacional do SATS, a expectativa seja de que o piloto avalie visualmente a disponibilidade destas regiões, porém essa expectativa não está explícita, portanto é necessário um requisito que exija que os pilotos tenham algum meio para consultar a posição de outras aeronaves dentro da SCA.

Deve-se propor a introdução de um novo requisito que remeta a esta hipótese. Como a hipótese é necessária apenas para o procedimento de arremetida, uma proposta seria de que “a aeronave deve possuir informação sobre as aeronaves que estão nas regiões dos IAFs e MAHFs”. Este requisito não entra em conflito com outros requisitos de Abbott et al. (2004) e devido à verificação formal é possível afirmar que o requisito é coerente com o comportamento esperado do sistema. Este requisito seria introduzido a especificação de sistema.

Tabela 22 - Requisitos do AAC após aplicação do ciclo de especificação

No	Requisito
A1	Enquanto o TSAFE não emite um alerta de conflito com previsão abaixo do limiar, o controlador é responsável pela formulação de resoluções e pela resolução de conflitos.
A2	Após o TSAFE emitir um alerta, enquanto o tempo para perda de separação for acima do limiar, o controlador deve escolher uma dentre estas três opções: (1) inibir o TSAFE <i>Resolution</i> de enviar uma resolução para o conflito; (2) autorizar o TSAFE <i>Resolution</i> a enviar uma resolução para a aeronave; (3) não agir no momento para o determinado conflito.
A3	Caso o controlador autorize o TSAFE <i>Resolution</i> a enviar uma resolução, o envio é feito imediatamente.
A4	Caso a estimativa para perda de separação caia para abaixo do limiar sem que o controlador tenha tomado alguma ação, a responsabilidade pela resolução do conflito é transferida automaticamente para o TSAFE <i>Resolution</i> .
A5	Após o TSAFE enviar uma resolução para a aeronave, uma mensagem no monitor do controlador indica quando o conflito foi resolvido e o controle da aeronave é transferido para o controlador, que fica responsável pela orientação da aeronave para a rota original.
A6	O piloto deve ser treinado para responder ao TSAFE e entender as condições que disparam um alerta TSAFE e TCAS.
A7	Todos alertas detectados pelo TSAFE serão resolvidos eventualmente.
A8	Caso o TSAFE assuma o controle de uma aeronave, eventualmente o controle será transferido para o controlador.
A9	Caso o controlador transfira o controle de uma aeronave para o TSAFE, esta aeronave não pode executar comandos do controlador.
A10	As resoluções do TSAFE mais antigas serão sempre executadas antes de resoluções mais recentes.
A11	Se um comando do TSAFE for enviado a uma aeronave, o controlador deve transferir o controle desta aeronave para o TSAFE.
A12	O piloto deve executar um comando proveniente do controlador caso não haja comandos provenientes do TSAFE e do TCAS e se a aeronave estiver sob controle do controlador e não estiver no meio de um processo de transferência de controle do controlador para o TSAFE.

7.3.3 Conclusões do Capítulo

A etapa de verificação do ciclo de especificação consiste na utilização da ferramenta de verificação NuSMV. Verifica-se se o modelo satisfaz as proposições e aquelas que apresentarem incoerentes devem ser analisadas. As proposições incoerentes correspondem às proposições satisfeitas de maneira vazia e proposições não satisfeitas pelo modelo. A análise é feita por meio de interpretação do contra-exemplo fornecido pela ferramenta e por meio de simulações de execução do modelo.

O trabalho de Arcaini, Gargantini e Riccobene (2017) apresenta meta-propriedades que todos modelos em NuSMV devem ter. Zhao e Rozier (2014) discutem a validade do resultado de verificação, para os casos do modelo satisfazer ou não satisfazer a proposição. Mesmo que a proposição seja satisfeita, não há garantia do modelo (e do sistema, por extensão) apresentar o comportamento esperado pelos desenvolvedores. Isto pode ser contornado por uma meta-propriedade discutida do modelo. Uma das meta-propriedades de Arcaini, Gargantini e Riccobene (2017) – a MP10 – define que nenhuma proposição deve ser satisfeita de maneira vazia. Uma proposta para verificar isso foi apresentada na seção 7.1. Deve-se elaborar proposições com a expressão 24 para todas proposições com construção do tipo $p \implies q$.

Para o conjunto de proposições incoerentes, deve-se avaliar se proposição ou modelo devem ser ajustados. Esta avaliação é feita pela análise do contra-exemplo. No caso de proposições satisfeitas de maneira vazia, deve-se avaliar o contra-exemplo da verificação da meta-propriedade referente a proposição.

Após a alteração, deve-se verificar novamente a proposição. Caso seja satisfeita, deve-se verificar se outras proposições verificadas continuam com os mesmos resultados. Caso algum resultado seja alterado, deve-se voltar a etapa de verificação formal e entender como as alterações influenciaram no modelo.

Com esta abordagem, é possível identificar em fases iniciais do desenvolvimento de um projeto requisitos que potencialmente não sejam coerentes com o comportamento esperado do sistema.

A diretriz para ajustes na descrição em linguagem natural usa como base as alterações realizadas nas proposições e no modelo. A interpretação da correção em termos do sistema real é necessária. Após a interpretação, deve-se propor um requisito e avaliar se este novo requisito não é incoerente com outros requisitos do sistema. Esta avaliação deve ser realizada considerando o maior número de fontes de requisitos possível.

Hipóteses apresentadas nas etapas de modelagem devem ser analisadas. Se for concluído que as proposições só são satisfeitas devido a suposição da hipótese, a hipótese deve ser

adicionada à especificação do sistema.

Se proposições foram ajustadas ou introduzidas, estas proposições poderiam ser mapeadas para requisitos especificados em linguagem natural controlada. Como o mapeamento é definido pelo padrão de especificação, este processo poderia ser automatizado.

Ao fim desta etapa, o ciclo é fechado e pode ser executado novamente. Um conjunto de requisitos verificados formalmente é produzido a cada iteração. Os requisitos introduzidos em uma iteração do ciclo, devido a ajustes, poderiam ser usados em uma nova iteração do ciclo de especificação. Pode-se manter o nível de abstração por meio de inclusão de mais requisitos ou pode-se alterar o nível de abstração, aproximando o modelo do nível de implementação por meio da substituição de requisitos por outros mais concretos. Ao alterar o nível de abstração é necessário garantir que as proposições satisfeitas no nível mais abstrato continuem sendo satisfeitas.

8 RESULTADOS E DISCUSSÕES

O resultado esperado após aplicação do ciclo de especificação é um conjunto de requisitos especificados em linguagem natural verificados formalmente. O método de especificação formal consiste num ciclo composto por etapas. Os objetivos de cada etapa do ciclo são alcançados pela aplicação de um conjunto de diretrizes. A seguir, os objetivos e diretrizes propostos para cada etapa são apresentados.

8.1 Etapa de especificação em linguagem natural

Objetivo da etapa: obter um conjunto de requisitos do sistema classificados em padrões de especificação.

Para alcançar este objetivo deve-se considerar as diferentes fontes de requisitos do sistema. Existem ao menos dois tipos de fontes relevantes: documentos em geral e os profissionais envolvidos no projeto do sistema.

Após determinar a fonte, deve-se considerar qual perspectiva do sistema será analisada, para coleta específica de requisitos relevantes para a verificação que será feita.

Os requisitos devem ser separados em requisitos de sistema e especificação de sistema. Os requisitos de sistema descrevem o que o sistema deve fazer, e especificação de sistema descrevem como o sistema deve ser organizado e implementado para atender aos requisitos de sistema.

Após a definição dos requisitos relevantes, os requisitos de sistema devem ser classificados em um dos padrões de especificação listados na seção 6.1.

8.2 Etapa de especificação formal

Objetivo da etapa: obter um modelo do sistema representado por uma máquina de estados finita e proposições especificadas em LTL e CTL de requisitos do sistema.

Deve-se utilizar a especificação de sistema para modelagem do sistema representado por uma FSM. Cada entidade do sistema com comportamento distinto deve ser representado por um elemento definido em um módulo distinto do modelo. Os módulos são definidos por variáveis, que compõem o estado do elemento, e pelas transições de cada uma destas variáveis. As transições são caracterizadas pelo gatilho e pelo valor correspondente. O gatilho é definido pela composição do valor de outras variáveis do módulo e por valores de variáveis de outros módulos.

Especificamente para o verificador NuSMV, as entidades devem ser definidas em módulos

distintos do modelo. A definição de cada módulo é segmentada em seções. As seções que devem ser usadas são VAR, DEFINE, ASSIGN e INVAR.

A seção VAR define as variáveis do módulo. As variáveis são usadas para observação do estado do elemento. As variáveis têm um tipo; a escolha do tipo deve refletir os valores que a variável pode assumir.

A seção DEFINE codifica expressões que representam combinação de valores de determinadas variáveis.

A seção ASSIGN é utilizada para a definição das transições de estado com os operadores init e next. Os operadores possuem como parâmetro a variável cujo valor será definido. O operador init define o valor inicial da variável e o operador next define o valor da variável no próximo passo de execução.

A seção INVAR é utilizada para complementar a transição de estados. É possível criar transições não-determinísticas.

A linguagem possui um módulo específico denominado `main` que declara todos elementos do modelo e estabelece a relação entre eles. Os elementos do modelo são definidos como variáveis deste módulo e as relações são definidas por meio de referência a variáveis de elementos nos parâmetros de outros elementos.

Os requisitos devem ser mapeados para proposições em LTL ou CTL utilizando o padrão de especificação em que foram classificados. Os termos usados para elaboração das variáveis proposicionais devem ser os mesmos utilizados no modelo. Os padrões de especificação propostos são apresentados a seguir, associados à forma de escrita da propriedade em linguagem natural controlada e à proposição em linguagem de especificação formal.

- Ausência - Global – Um evento $\{P\}$ nunca ocorre.
 - Nunca ocorre que $\{P\}$ seja verdadeiro.
 - $\Box(\neg\{P\})$
- Ausência - Após $\{Q\}$ até $\{R\}$ – Um evento $\{P\}$ não ocorre após um evento $\{Q\}$ ocorrer até que um evento $\{R\}$ ocorra.
 - Após $\{Q\}$ até $\{R\}$, não ocorre que $\{P\}$ seja verdadeiro.
 - $\Box(\{Q\} \wedge \neg\{R\} \implies (\neg\{P\}U\{R\}))$
- Resposta - Global – Um evento $\{S\}$ ocorre em resposta a um evento $\{P\}$.
 - Se $\{P\}$ ocorreu então em resposta $\{S\}$ ocorre eventualmente.

- $\Box(\{P\} \implies \Diamond\{S\})$
- Precedência - Após $\{Q\}$ Até $\{R\}$ – Um evento $\{S\}$ precede um evento $\{P\}$, após um evento $\{Q\}$ ocorrer até que um evento $\{R\}$ ocorra.
 - Após $\{Q\}$ Até $\{R\}$, se $\{P\}$ ocorreu, então $\{S\}$ deve ter ocorrido.
 - $\Box(\{Q\} \wedge \neg\{R\} \implies (\neg\{P\}U(\{S\} \vee \{R\})))$
- Resposta Imediata - Global – Um evento $\{S\}$ ocorre imediatamente após um evento $\{P\}$ ocorrer.
 - Se $\{P\}$ ocorreu então em resposta $\{S\}$ ocorre imediatamente.
 - $\Box(\{P\} \implies \bigcirc\{S\})$
- Resposta Imediata a Transição - Global – Um evento $\{S\}$ deve ocorrer após a transição de um estado $\{P\}$ para $\{R\}$.
 - Se $\{P\}$ é verdade e $\{R\}$ ocorre, então imediatamente $\{S\}$ ocorre como resposta.
 - $\Box(\{P\} \wedge \bigcirc\{R\} \implies \bigcirc(\{R\} \wedge (\bigcirc\{S\})))$
- Múltiplas Respostas - Global – Apenas um dos eventos $\{S_1\}$, $\{S_2\}$... $\{S_n\}$ deve ocorrer em resposta ao evento $\{P\}$.
 - Se $\{P\}$ ocorreu, então em resposta $\{S_1\}$; ou $\{S_2\}$; ... ; ou $\{S_n\}$ eventualmente ocorre.
 - $AG(\{P\} \implies EF\{S_x\})$
 - $\Box(\{P\} \implies \Diamond(\{S_1\} \vee \{S_2\} \vee \dots \vee \{S_n\}))$
 - Neste caso, haverá n proposições em CTL e uma proposição em LTL.

8.3 Etapa de verificação formal e análise do resultado da verificação;

Objetivo da etapa: verificar se o modelo satisfaz as proposições e análise do resultado de verificação de cada proposição.

O verificador de modelos NuSMV é usado para a verificação formal do modelo da etapa anterior. Caso a ferramenta não consiga finalizar o processo de verificação, hipóteses devem ser propostas para simplificação do modelo.

As proposições são verificadas e as que não forem satisfeitas devem ser investigadas a partir do contra-exemplo fornecido pela ferramenta. Deve-se realizar interpretação com

termos do modelo e com termos do sistema real. A interpretação em termos de modelo é necessária para ajustes na especificação formal e/ou no modelo e a interpretação em termos do sistema real é necessária para avaliar se a especificação formal está errada, se o modelo está errado ou se ambos estão errados.

Além disso, deve-se avaliar se as proposições não são sempre verdadeiras ou sempre falsas, o que resultaria em falsos positivos. Nestas condições, é provável que a proposição tenha sido especificada incorretamente ou algum comportamento esperado do sistema não foi modelado. A seção 7.3.3 apresenta uma diretriz para avaliar se o modelo apresenta meta-propriedade que buscam evitar este cenário.

Nesta etapa as hipóteses propostas durante a etapa anterior para simplificação do modelo devem ser analisadas para avaliar se poderiam ser descartadas e aumentar a fidelidade do modelo.

8.4 Etapa de ajuste na especificação formal

Objetivo da etapa: obter modelo e proposições de modo que o modelo satisfaça as proposições.

Nesta etapa, deve-se realizar ajustes nas proposições e no modelo baseados na análise do resultado da verificação formal.

A diretriz proposta consiste em definir a causa do erro (inconsistência entre modelo e proposição) e propor alterações baseadas nesta causa.

Para alterações no modelo, a partir da análise do contra-exemplo fornecido pelo NuSMV, deve-se avaliar quais são as variáveis dependentes e quais são as variáveis independentes. As variáveis dependentes assumem o valor inconsistente na execução apresentada pelo contra-exemplo, configurando o estado inconsistente com a proposição. As variáveis independentes causam a transição das variáveis dependentes para o estado inconsistente. As variáveis independentes devem ser introduzidas diretamente à transição das variáveis dependentes para evitar que a execução alcance este estado inconsistente.

Para alterações nas proposições, deve-se avaliar se foram classificadas no padrão de especificação mais adequado e se foram mapeadas corretamente. Caso nenhuma alteração ajuste a proposição, deve-se avaliar se o requisito desta proposição é coerente com o esperado do sistema.

Após as alterações, deve-se verificar novamente a proposição referente a alteração. Caso seja satisfeita pelo modelo, deve-se verificar se outras proposições já verificadas mantêm os resultados anteriores à alteração. Caso algum resultado seja alterado, deve-se voltar à etapa de verificação formal e analisar como as alterações influenciaram o modelo.

As hipóteses propostas para o processo de modelagem devem ser analisadas para avaliar

se alguma proposição foi satisfeita se e somente se a hipótese existe. Em caso positivo, a hipótese deve ser avaliada como um ajuste no modelo e considerá-lo na etapa de ajustes na linguagem natural.

8.5 Etapa de ajuste na especificação em linguagem natural

Objetivo da etapa: obter um conjunto de requisitos verificado formalmente e uma especificação formal coerente com os requisitos.

A diretriz para ajustes na descrição em linguagem natural com base nos ajustes feitos em etapas anteriores consiste na interpretação destes ajustes em termos do sistema real. Os ajustes feitos nas proposições devem resultar em ajustes nos requisitos de sistema e os ajustes no modelo devem resultar em ajustes na especificação formal. Após a interpretação, deve-se propor uma especificação em linguagem natural referente ao ajuste e avaliar se esta especificação não é incoerente com outros requisitos do sistema. Deve-se considerar o maior número de fontes de requisitos nesta avaliação.

8.6 Aplicação do Ciclo de Especificação nos Estudos de Caso

A aplicação do ciclo de especificação no estudo de caso do AAC foi bem sucedida pois ao final da aplicação do ciclo foi obtido um conjunto de requisitos com menos erros do que o conjunto de requisitos anterior à aplicação do ciclo e porque resultados de verificação semelhantes ao de Zhao e Rozier (2014) foram obtidos. Devido à aplicação do ciclo, foi possível detectar um comportamento indesejado na coordenação do AAC que já havia sido discutida em Zhao e Rozier (2014). O modelo do sistema foi alterado para remover este comportamento indesejado e um ajuste foi proposto para o sistema AAC baseada no ajuste no modelo.

O conjunto de táticas foi utilizado na verificação formal do SATS. Um dos requisitos do sistema e parte do modelo foram alterados para a verificação formal, devido ao desempenho da ferramenta. Com base no documento de especificação do conceito operacional do sistema estas alterações não deveriam invalidar a verificação, entretanto o ideal seria validar estas alterações com profissionais envolvidos no projeto, visto que estes profissionais são fontes de requisitos e especificação do sistema. Durante o processo de modelagem, uma hipótese foi proposta para o modelo do sistema. Esta hipótese é necessária para que o modelo se comporte como o esperado. Esta hipótese foi então proposta como um ajuste no conceito operacional do SATS. Por meio do uso do ciclo de especificação, é possível afirmar que a especificação do conceito operacional do sistema apresentada em Abbott et al. (2004)

somada ao ajuste proposto é coerente com os requisitos do sistema.

A aplicação do ciclo de especificação em um sistema segue o *workflow* apresentado na figura 1. Todas as atividades deste *workflow* correspondem a diretrizes de etapas do ciclo de especificação.

9 CONCLUSÕES

Um método de especificação e verificação formal de requisitos de sistemas de gerenciamento de tráfego aéreo foi proposto. Este método tem como entrada requisitos especificados em linguagem natural e resulta em requisitos ajustados após aplicação de técnica de verificação formal. Este método consiste em um ciclo de especificação composto por etapas. Diretrizes para cada etapa do ciclo foram propostas baseadas em estudos de caso de propostas de sistemas ATM.

A principal contribuição foi o estabelecimento de diretrizes para elaboração de expressões escritas em linguagem de especificação formal a partir de requisitos escritos em linguagem natural que podem ser verificadas formalmente. Os requisitos são especificados por meio de classificação em padrões de especificação. Outra contribuição deste trabalho foi a seleção de padrões de especificação de Autili et al. (2015) e a proposta de novos padrões baseados em características de sistemas de tráfego aéreo.

As diretrizes de cada etapa foram elaboradas por meio de análise do trabalho de Zhao e Rozier (2014) e de seleção e elaboração de padrões de especificação. Estas diretrizes foram aplicadas como estudo de caso no sistema AAC e no sistema SATS.

No estudo de caso do sistema AAC, o mesmo conjunto de requisitos e modelo do sistema de Zhao e Rozier (2014) foi usado. Neste trabalho, duas proposições não foram satisfeitas. No presente trabalho, os requisitos foram escritos em linguagem natural controlada e foram mapeados em proposições especificadas em LTL e CTL. O modelo foi então alterado para satisfazer todas proposições especificadas, inclusive as expressões referentes aos requisitos não satisfeitos no trabalho original. Estas alterações no modelo resultaram em propostas de alteração na especificação do sistema. A aplicação do ciclo de especificação foi bem sucedida, pois alcançou resultados de verificação semelhantes aos apresentados em Zhao e Rozier (2014) e obteve como resultado um conjunto de requisitos que passaram por um processo de verificação formal.

No estudo de caso do sistema SATS, requisitos escritos em linguagem natural foram levantados do documento de conceito operacional do SATS (ABBOTT et al., 2004) e o sistema foi modelado usando a especificação de sistema apresentada no documento de conceito operacional e elementos do modelo proposto no trabalho de Doweck, Munoz e Carreno (2004). Diretrizes de modelagem foram propostas com a análise do modelo do sistema AAC em Zhao e Rozier (2014), diretrizes do manual do verificador NuSMV e diretrizes apresentadas em Arcaini, Gargantini e Riccobene (2017). Estas diretrizes de modelagem foram utilizadas para modelar o SATS. O modelo completo do sistema foi alterado e hipóteses foram propostas para reduzir o número de estados do modelo e garantir

que o NuSMV fosse capaz de verificar o modelo em tempo hábil. Com estas alterações, foi possível verificar parte dos requisitos do SATS. Um dos requisitos foi alterado para reduzir o número de estados do modelo.

A demonstração da aplicação do ciclo de especificação em estudos de caso foi uma das contribuições deste trabalho por apresentar o processo que antecede a verificação formal, *i.e.*, como obter uma especificação formal de requisitos e especificação em linguagem natural.

Outra contribuição do trabalho foram as diretrizes de modelagem de sistemas. A técnica de verificação formal escolhida foi a de verificação de modelos que verifica proposições escritas em linguagens de especificação formal para um determinado modelo. O objeto de estudo deste trabalho corresponde aos requisitos do sistema, que são mapeados para proposições, logo as diretrizes foram propostas para o caso em que não existe um modelo do sistema.

Ao final da aplicação do ciclo de especificação, o conjunto de requisitos pode ser refinado para obtenção de uma descrição num nível mais próximo do nível de implementação. Nesta abordagem, o ciclo é aplicado novamente até a definição de um conjunto de requisitos mais concretos verificados formalmente. Este conjunto de requisitos pode ser usado para a orientação da implementação do sistema; a expectativa é de que o conjunto de requisitos possua menos erros de projeto do que um conjunto de requisitos não verificados.

A elaboração de proposições em especificação formal a partir de requisitos em linguagem natural controlada e a especificação de requisitos em linguagem natural controlada a partir de proposições em especificação formal ajustadas devido aos resultados de verificação formal podem ser automatizadas devido ao uso de padrões de especificação.

O ciclo de especificação proposto possui as limitações resultantes das técnicas, linguagens e ferramenta escolhidas. O modelo do sistema deve ser elaborado de modo a evitar o problema de explosão de estados. As expressões especificadas formalmente permitem expressar requisitos reativos de sistemas por usar linguagens que usam termos temporais. Para requisitos de outros tipos, outras linguagens deveriam ser usadas.

9.1 Trabalhos Futuros

Com os resultados obtidos do ciclo de especificação e suas limitações, são propostas quatro alternativas para trabalhos futuros: automação do ciclo de especificação; inclusão de linguagens de especificação formal probabilísticas e de tempo-real; utilização do modelo elaborado para uso no verificador NuSMV como artefato para desenvolvimento de software; geração de casos de teste a partir dos resultados de verificação formal.

A tradução de requisitos em linguagem natural controlada para proposições em especificação formal e vice-versa pode ser automatizada. Com a utilização de técnicas de processamento de linguagem natural, a classificação em um padrão de especificação poderia ser automatizada também. O padrão SBVR (OBJECT MANAGEMENT GROUP, 2017) define um vocabulário para especificar requisitos de sistema em linguagem controlada para um domínio específico. Caso o SBVR fosse introduzido ao ciclo de especificação, os requisitos seriam especificados em linguagem natural controlada. A única etapa manual no ciclo de especificação seria o de análise dos resultados de verificação formal e o de ajustes na especificação formal. As outras etapas seriam automatizadas.

Autili et al. (2015) apresentam padrões de especificação qualitativos, em tempo real e probabilísticos. Os requisitos em linguagem natural controlada são mapeados para expressões em linguagens de lógica temporal (como LTL e CTL), lógica temporal probabilística (PLTL) e lógica temporal em tempo real (TCTL). Esta alternativa aumentaria a expressividade do ciclo de especificação. Os autores de Zhao e Rozier (2014) expandem seu trabalho com relação ao AAC utilizando especificações probabilísticas em outros trabalhos.

A utilização do modelo gerado para desenvolvimento de código visa a aproximação de descrição em linguagem natural e implementação. O ciclo de especificação gera um conjunto de requisitos e um modelo representado em máquina de estados finita. Trabalhos podem ser conduzidos para definição de diretrizes de tradução desta representação em *software* (código) e verificar se o *software* respeita a especificação formal e os requisitos verificados.

Fraser, Wotawa e Ammann (2009) apresentam uma vantagem da técnica de verificação de modelos: geração de casos de testes a partir do resultado da verificação de um verificador de modelos. Logo, estes resultados podem ser usados tanto em fases mais conceituais do projeto quanto em fases posteriores, como implementação e testes.

Referências

- ABBOTT, T. S. et al. **Small Aircraft Transportation System, Higher Volume Operations Concept: Normal Operations**. Virginia, EUA, Agosto 2004. 29 p.
- ARCAINI, P.; GARGANTINI, A.; RICCOBENE, E. Nuseen: a tool framework for the nusmv model checker. **2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)**. p. 476–483, March 2017. DOI: 10.1109/ICST.2017.54.
- AUTILI, M. et al. Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar. **IEEE Transactions on Software Engineering**, v. 41, n. 7, p. 620–638, July 2015. DOI: 10.1109/TSE.2015.2398877.
- AZZOPARDI, M. A. **Computational Air Traffic Management**. 2015. Tese (Doutorado) — School of Engineering, Cranfield University, Bedfordshire, Reino Unido.
- BAYEN, A. M. **Computational Control of Networks of Dynamical Systems: Application to the National Airspace System**. 2004. Tese (Doutorado) — Stanford University, Stanford, CA, USA. AAI3111687.
- BEN-ARI, M.; PNUELI, A.; MANNA, Z. The temporal logic of branching time. **Acta Informatica**, v. 20, n. 3, p. 207–226, 1983. DOI: 10.1007/BF01257083.
- BOWEN, J. The ethics of safety-critical systems. **Commun. ACM**, New York, NY, USA, ACM, v. 43, n. 4, p. 91–97, abr. 2000. 7 p. DOI: 10.1145/332051.332078.
- BROWN, M. D. **Air Traffic Control Using Virtual Stationary Automata**. 2006. Dissertação (Mestrado) — Massachusetts Institute of Technology, Cambridge, Massachusetts, EUA.
- CERONE, A.; LINDSAY, P. A.; CONNELLY, S. Formal analysis of human-computer interaction using model-checking. **Third IEEE International Conference on Software Engineering and Formal Methods (SEFM'05)**. p. 352–361, Sept 2005. DOI: 10.1109/SEFM.2005.19.
- CIMATTI, A. et al. Nusmv 2: an opensource tool for symbolic model checking. **Proceedings of the 14th International Conference on Computer Aided Verification**. p. 359–364, 2002.
- CLARKE, E. M.; EMERSON, E. A.; SISTLA, A. P. Automatic verification of finite-state concurrent systems using temporal logic specifications. **ACM Trans. Program. Lang. Syst.**, New York, NY, USA, ACM, v. 8, n. 2, p. 244–263, abr. 1986. 20 p. DOI: 10.1145/5397.5399.

DOWEK, G.; MUNOZ, C.; CARRENO, V. A. **Abstract model of the SATS concept of operations: Initial results and recommendations**. Hampton, VA, EUA, Março 2004. 46 p.

DURAND, N.; BARNIER, N. Does ATM Need Centralized Coordination? Autonomous Conflict Resolution Analysis in a Constrained Speed Environment. **ATM seminar 2015, 11th USA/EUROPE Air Traffic Management R&D Seminar**. jun. 2015.

DWYER, M. B.; AVRUNIN, G. S.; CORBETT, J. C. Property specification patterns for finite-state verification. **Proceedings of the Second Workshop on Formal Methods in Software Practice**. p. 7–15, 1998. DOI: 10.1145/298595.298598.

EMERSON, E. A.; HALPERN, J. Y. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. **J. ACM**, New York, NY, USA, ACM, v. 33, n. 1, p. 151–178, jan. 1986. 28 p. DOI: 10.1145/4904.4999.

ERZBERGER, H. **Transforming the NAS: The Next Generation Air Traffic Control System**. California, EUA, Outubro 2004. 34 p.

ERZBERGER, H.; HEERE, K. Algorithm and operational concept for resolving short-range conflicts. **Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering**, v. 224, n. 2, p. 225–243, 2010. DOI: 10.1243/09544100JAERO546.

ERZBERGER, H.; LAUDERDALE, T. A.; CHU, Y.-C. Automated conflict resolution, arrival management, and weather avoidance for air traffic management. **Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering**, v. 226, n. 8, p. 930–949, 2012. DOI: 10.1177/0954410011417347.

FATWANTO, A. Translating software requirements from natural language to formal specification. **Computational Intelligence and Cybernetics (CyberneticsCom), 2012 IEEE International Conference on**. p. 148–152, July 2012. DOI: 10.1109/CyberneticsCom.2012.6381636.

FEDERAL AVIATION ADMINISTRATION. **NextGen Implementation Plan 2016**. Washington, DC, EUA, Junho 2016. 84 p.

FRASER, G.; WOTAWA, F.; AMMANN, P. E. Testing with model checkers: a survey. **Software Testing, Verification and Reliability**, v. 19, n. 3, p. 215–261, 2009. DOI: 10.1002/stvr.402.

FRASER, M. D.; KUMAR, K.; VAISHNAVI, V. K. Informal and formal requirements specification languages: bridging the gap. **IEEE Transactions on Software Engineering**, v. 17, n. 5, p. 454–466, May 1991. DOI: 10.1109/32.90448.

GIBSON-ROBINSON, T. et al. FDR3 — A Modern Refinement Checker for CSP. **Tools and Algorithms for the Construction and Analysis of Systems**. p. 187–201, 2014.

HANH, T. T. B.; HUNG, D. Verification of an air-traffic control system with probabilistic real-time model-checking. **UNU-IIST report**, 2007.

HOARE, C. A. R. Communicating sequential processes. **Commun. ACM**, New York, NY, USA, ACM, v. 21, n. 8, p. 666–677, ago. 1978. 12 p. DOI: 10.1145/359576.359585.

HOLZMANN, G. **Spin Model Checker, the: Primer and Reference Manual**. First. [S.l.]: Addison-Wesley Professional, 2003.

KAUR, A.; GULATI, S.; SINGH, S. Analysis of three formal methods-z, b and vdm. **International Journal of Engineering**, v. 1, n. 4, 2012.

KHARCHENKO, V. et al. Analysis of the flight object concept. **Proceedings of the National Aviation University**, n. 2, p. 7–11, 2015.

LALISH, E.; MORGANSEN, K. A. Distributed reactive collision avoidance. **Autonomous Robots**, v. 32, n. 3, p. 207–226, 2012. DOI: 10.1007/s10514-011-9267-7.

LARRUCEA, X.; COMBELLES, A.; FAVARO, J. Safety-critical software [guest editors' introduction]. **IEEE Software**, v. 30, n. 3, p. 25–27, May 2013. DOI: 10.1109/MS.2013.55.

MAYER, E.; FRÖHLICH, J. A dds-based middleware for cooperation of air traffic service units. In: **COMMUNICATION Technologies for Vehicles: 6th International Workshop, Nets4Cars/Nets4Trains/Nets4Aircraft 2014**, Offenburg, Germany, May 6-7, 2014. Proceedings. Cham: Springer International Publishing, 2014. p. 94–102.

MCCONNELL, S. **Code Complete, Second Edition**. Redmond, WA, USA: Microsoft Press, 2004.

MUÑOZ, C.; CARREÑO, V.; DOWEK, G. Formal analysis of the operational concept for the small aircraft transportation system. In: **RIGOROUS Development of Complex Fault-Tolerant Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 306–325. DOI: 10.1007/11916246_16.

OBJECT MANAGEMENT GROUP. **Semantics of Business Vocabulary and Business Rules Version 1.4**. Needham, MA, EUA, Maio 2017. 328 p.

OLIVEIRA, Í. R. de. **Análise de risco da operação de espaçamento temporal aerotransportado por meio de um modelo em rede de Petri estocástica e dinamicamente colorida**. 2007. Dissertação (Mestrado) — Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil.

OLIVEIRA, Í. R. de; CUGNASCA, P. S. Checking safe trajectories of aircraft using hybrid automata. In: **COMPUTER Safety, Reliability and Security: 21st International Conference, SAFECOMP 2002 Catania, Italy, September 10–13, 2002 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 224–235.

PIZZO, W. N. **Avaliação da disponibilidade de sistemas computacionais críticos para o controle do espaço aéreo por meio de modelo analítico da teoria de filas**. 2008. Dissertação (Mestrado) — Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil.

PNUELI, A. The temporal logic of programs. **Proceedings of the 18th Annual Symposium on Foundations of Computer Science**. p. 46–57, 1977. DOI: 10.1109/SFCS.1977.32.

POPESCU, V. M. **Airspace analysis and design by data aggregation and lean model synthesis**. 2013. Tese (Doutorado) — Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Georgia, Atlanta, EUA.

SAALTINK, M. The z/eves system. In: **ZUM '97: The Z Formal Specification Notation: 10th International Conference of Z Users Reading, UK, April 3–4, 1997 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. p. 72–85. DOI: 10.1007/BFb0027284.

SESAR JOINT UNDERTAKING. **European ATM Master Plan Executive View Edition 2015**. Luxemburgo, Fevereiro 2016. 126 p.

SPIVEY, J. M. **The Z Notation: A Reference Manual**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

STECKLEIN, J. M. et al. Error cost escalation through the project life cycle. **14th Annual International Symposium**. 2004.

STROHMEIER, M.; LENDERS, V.; MARTINOVIC, I. On the security of the automatic dependent surveillance-broadcast protocol. **IEEE Communications Surveys Tutorials**, v. 17, n. 2, p. 1066–1087, Secondquarter 2015. DOI: 10.1109/COMST.2014.2365951.

STROHMEIER, M. et al. Realities and challenges of nextgen air traffic management: the case of ads-b. **IEEE Communications Magazine**, v. 52, n. 5, p. 111–118, May 2014. DOI: 10.1109/MCOM.2014.6815901.

YOUSAF, S. et al. Formal analysis of arrival procedure of air traffic control system. **Life Sci J**, v. 9, n. 4, 2012.

YOUSAF, S.; ZAFAR, N. A.; KHAN, S. A. Formal analysis of departure procedure of air traffic control system. **Software Technology and Engineering (ICSTE)**,

2010 2nd International Conference on. v. 2, p. V2–301–V2–305, Oct 2010. DOI: 10.1109/ICSTE.2010.5608802.

ZAFAR, N. A. Safety control management at airport taxiing to take-off procedure. **Arabian Journal for Science and Engineering**, v. 39, n. 8, p. 6137–6148, 2014. DOI: 10.1007/s13369-014-1176-6.

ZHAO, Y.; ROZIER, K. Y. Formal specification and verification of a coordination protocol for an automated air traffic control system. **Science of Computer Programming**, v. 96, Part 3, p. 337 – 353, 2014. DOI: <http://dx.doi.org/10.1016/j.scico.2014.04.002>. Special Issue on Automated Verification of Critical Systems (AVoCS 2012).