sid.inpe.br/mtc-m21c/2020/10.28.15.15-TDI

UMA SOLUÇÃO HÍBRIDA DE PLANEJAMENTO EMBARCADO PARA AUMENTAR A AUTONOMIA OPERACIONAL DE SATÉLITES BASEADA EM REDE DE TAREFAS HIERÁRQUICAS E INFERÊNCIA DE ESTADOS

Filipe de Simone Cividanes

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelos Drs. Maurício Gonçalves Vieira Ferreira e Fabrício de Novaes Kucinskis, aprovada em 29 de outubro de 2020.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/43G3L3B>

INPE
São José dos Campos
2020

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

sid.inpe.br/mtc-m21c/2020/10.28.15.15-TDI

UMA SOLUÇÃO HÍBRIDA DE PLANEJAMENTO EMBARCADO PARA AUMENTAR A AUTONOMIA OPERACIONAL DE SATÉLITES BASEADA EM REDE DE TAREFAS HIERÁRQUICAS E INFERÊNCIA DE ESTADOS

Filipe de Simone Cividanes

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelos Drs. Maurício Gonçalves Vieira Ferreira e Fabrício de Novaes Kucinskis, aprovada em 29 de outubro de 2020.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/43G3L3B>

INPE
São José dos Campos
2020

Dados Internacionais de Catalogação na Publicação (CIP)

Cividanes, Filipe de Simone.

C499s Uma solução híbrida de planejamento embarcado para aumentar a autonomia operacional de satélites baseada em rede de tarefas hierárquicas e inferência de estados / Filipe de Simone Cividanes. – São José dos Campos : INPE, 2020.

xxxiv + 266 p. ; (sid.inpe.br/mtc-m21c/2020/10.28.15.15-TDI)

Tese (Doutorado em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2020.

Orientadores : Drs. Maurício Gonçalves Vieira Ferreira e Fabrício de Novaes Kucinskis.

1. Planejamento híbrido. 2. Autonomia de satélites. 3. Planejamento em rede de tarefas hierárquicas. 4. Planejamento clássico baseado em ações. 5. Planejamento automatizado de missão. I.Título.

CDU 629.7.01:629.78

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Serviço de Pós-Graduação-SEPGR

Pós-Graduação em ETE/Engenharia e Gerenciamento de Sistemas Espaciais.

DEFESA FINAL DE TESE DE FILIPE DE SIMONE CIVIDANES

No dia 29 de outubro de 2020, as 09h, por videoconferência, o(a) aluno(a) mencionado(a) acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O(A) aluno(a) foi APROVADO(A) pela Banca Examinadora, por unanimidade, em cumprimento ao requisito exigido para obtenção do Título de Doutor em Engenharia e Tecnologia Espaciais/Eng. Gerenc.de Sistemas Espaciais. O trabalho precisa da incorporação das correções sugeridas pela Banca Examinadora e revisão final pelo(s) orientador(es).

Título: " Uma solução híbrida de planejamento embarcado para aumentar a autonomia operacional de satélites baseada em rede de tarefas hierárquicas e inferência de estados**"**

Eu, Walter Abrahao dos Santos, como Presidente da Banca Examinadora, assino esta ATA em nome de todos os membros.

Membros da Banca

Dr. Walter Abrahao dos Santos, Presidente INPE.
Dr. Maurício Gonçalves Vieira Ferreira, Orientador(a) INPE.
Dr. Fabricio de Novaes Kucinskis, Orientador(a) INPE.
Dr. Ronaldo Arias, Membro da Banca INPE.
Dr. Rodrigo Rocha Silva, Convidado(a) FATEC.
Dr. Jose Eduardo Morello Lobo, Convidado(a), UMC.

Documento assinado eletronicamente por **Walter Abrahão dos Santos**, **Tecnologista**, em 05/11/2020, às 14:35 (horário oficial de Brasília), com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.mctic.gov.br/verifica.html, informando o código verificador **6030257** e o código CRC **ECA7DE62**.

Referência: Processo nº 01340.007475/2020-91 SEI nº 6030257

“Planning is the art and practice of thinking before acting.”

— Patrik Haslum

Dedico este trabalho à minha família, em especial, ao meu filho Miguel e ao meu pai Lucio.

AGRADECIMENTOS

A Deus primeiramente por ter me concedido saúde e sabedoria para conclusão deste trabalho.

Ao meu filho Miguel pelo seu carinho e amor incondicional. Peço sinceras desculpas pelos momentos ausentes, que não me permitiram oferecer toda atenção e alegria.

À minha esposa pelo companheirismo e por ter me tolerado com sabedoria no percorrer deste longo e difícil caminho de doutoramento. Peço desculpas pela ausência e pelo estranho mau humor em determinados momentos difíceis deste trabalho.

Aos meus pais, Eliana e Lucio, pela sólida educação e ensinamentos de persistência e humildade que muito colaboraram para conclusão deste trabalho. Peço sinceras desculpas pela ausência em certos momentos necessária para a conclusão do meu doutoramento.

Aos meus irmãos (Rafael e Luciana) e à Risolene pela amizade e conselhos valiosos nos momentos difíceis deste trabalho.

Aos meus orientadores Dr. Maurício e Dr. Fabrício que possibilitaram a realização desta pesquisa e pelas contribuições que muito enriqueceram este trabalho. Obrigado pela confiança e pelas oportunidades que me foram concedidas.

Ao grupo supervisão de bordo (SUBORD) do INPE pelos primeiros ensinamentos na área espacial e em particular nos conhecimentos relacionados a um software embarcado em satélites que muito ajudaram para realização deste trabalho.

A todos os amigos e colegas do INPE que de alguma forma colaboraram para conclusão deste trabalho de Doutorado.

RESUMO

Tradicionalmente, as operações de missões espaciais são baseadas em atividades planejadas pelo segmento solo. Devido a janelas de visibilidade restritas e atrasos na comunicação entre o satélite e as estações de operação em solo, alguns sistemas espaciais não podem ser completamente controlados por solo em tempo real. Como forma de superar isso, os operadores lançam mão da execução agendada de comandos temporizados, planejados com antecedência pelo centro de controle, para a operação em períodos fora de comunicação, que podem atingir até 90% do tempo da missão. Uma maneira de melhorar a operação é equipar o software de voo com um sistema autônomo de tomada de decisões, que implemente técnicas de planejamento automatizado de missão. Neste contexto, esta Tese propõe uma solução híbrida de planejamento embarcado baseado em rede de tarefas hierárquica e inferência de estados para aumentar o nível de autonomia operacional de satélites. A abordagem híbrida desta Tese é centrada nas regras de controle da decomposição hierárquica de tarefas aliada à abordagem generativa do planejamento clássico baseado em ações. Tal solução implica na criação de uma nova forma de representação do conhecimento a bordo do satélite, de um planejador computacionalmente 'leve' e compatível com as restrições impostas pelo hardware de voo qualificado para uso espacial, que possui baixa capacidade de processamento. Um planejador híbrido e modelo embarcado são propostos, implementados e testados como elementos de uma arquitetura de planejamento hierárquico organizada em camadas que permite unificar planejamento e escalonamento, mantendo-se a capacidade de revisão de planos. A solução foi submetida a cenários experimentais de planejamento embarcado a partir de um estudo de caso realístico voltado a uma missão de sensoriamento remoto. A modelagem do problema na nova representação do domínio e os experimentos realizados em um ambiente computacional representativo da área espacial puderam atestar a viabilidade da solução proposta. Os resultados de planejamento obtidos se mostraram promissores frente ao estado da arte do domínio embarcado de satélites. Conclui-se que o paradigma híbrido proposto permite acrescentar ganhos de desempenho computacionais usando as informações hierárquicas do domínio e, ao mesmo tempo, ser mais responsivo a situações não previstas através da abordagem generativa, mostrando-se uma solução interessante principalmente para aplicações de tempo real, como satélites.

Palavras-chave: Planejamento Híbrido. Autonomia de Satélites. Planejamento em Rede de Tarefas Hierárquicas. Planejamento Clássico Baseado em Ações. Planejamento Automatizado de Missão. Planejamento de Tempo Real. Arquitetura de Planejamento Automatizado a Bordo de Satélite.

A HYBRID ONBOARD PLANNING SOLUTION TO INCREASE SATELLITE OPERATIONAL AUTONOMY BASED ON HIERARCHICAL TASK NETWORK AND STATE INFERENCE

ABSTRACT

Traditionally, space mission operations are based on activities planned by the ground segment. Due to restricted visibility windows and delays in communication between the satellite and ground stations, some space systems cannot be completely controlled in real-time. As a way to overcome this, the operators resort to the scheduled execution of time-tagged commands, predefined by the ground segment, for the operation in periods out of communication, which can reach up to 90% of the mission time. One way to improve the operation is to equip the space flight software with an autonomous decision-making system that implements automated mission planning techniques. In this context, this thesis proposes a hybrid solution for onboard planning based on a Hierarchical Task Network (HTN) and state inference in order to increase the level of spacecraft autonomy. A hybrid planning approach centered on the control rules for the task reduction scheme is proposed, together with the generative paradigm of action-based classical planning. Such a solution implies the creation of a new form of knowledge representation onboard the satellite, of a computationally lightweight planner compatible with the constraints imposed by space-qualified hardware, which has low processing capacity. A hybrid planner and onboard model are proposed, implemented, and tested as elements of a layered architecture based on hierarchical planning that allows unifying planning and scheduling, maintaining the ability to review plans. The solution was submitted to experimental scenarios based on a realistic case study aimed at a Brazilian remote-sensing mission. The modeling of the problem and the experiments carried out in a representative space computational environment could attest to the viability of the proposed solution. The planning results obtained are promising given the state-of-the-art in the satellite onboard domain. It is concluded that the hybrid paradigm allows enhancing computational performance gains using hierarchical information of the domain and at the same time to be more responsive to unforeseen situations achieved by the generative approach, showing itself to be an advantageous solution mainly for real-time applications, such as satellites.

Keywords: Hybrid Planning. Satellite Autonomy. Hierarchical Task Network Planning. Action-based Classical Planning. Onboard Automated Mission Planning. Real-time Planning. Satellite Onboard Automated Planning Architecture.

LISTA DE FIGURAS

Pág.

Figura 1.1 – Modelo conceitual de um sistema com planejamento automatizado..... 3
Figura 1.2 – Cenário de operação espacial em ciclo fechado para um satélite com autonomia a bordo..... 9
Figura 2.1 – Histórico e previsão de lançamentos de nanossatélites. 14
Figura 2.2 – Correlação entre infraestrutura de suporte e atividades de operação. 15
Figura 2.3 – Dinâmica de operação de veículos espaciais da ESA..... 27
Figura 2.4 – Modelo conceitual da operação baseada em sequência de comandos. ... 29
Figura 2.5 – Modelo conceitual da operação baseada em objetivos (E4) e operação autônoma do segmento espacial (E5)..... 31
Figura 3.1 – Elementos presentes em um problema de Planejamento em IA. 35
Figura 3.2 – Exemplo de um grafo de transição de estados. 37
Figura 3.3 – Exemplo do domínio dos robôs portuários para transporte de mercadorias. 41
Figura 3.4 – Exemplo de uma ação na linguagem STRIPS. 43
Figura 3.5 – Exemplo de uma ação em PDDL 2.1. 44
Figura 3.6 – Entradas para um problema de planejamento em PDDL. 45
Figura 3.7 – Exemplo de um plano de ordem parcial e suas linearizações. 48
Figura 3.8 – Comparativo dos sistemas de planejamento. 56
Figura 3.9 – Planejamento e escalonamento em cascata. 59
Figura 3.10 – Exemplo de uma representação STN para uma atividade do domínio de satélites. 62
Figura 4.1 – Objetivo como uma restrição binária de um estado (um recurso/timeline do sistema) no tempo (domínio de tempo). 65
Figura 4.2 – Exemplo de um objetivo descrito como uma rede de tarefas hierárquicas a partir de uma árvore de decomposição. 69

Figura 4.3 – Visão geral da abordagem de reparo iterativo e os componentes envolvidos. 81
Figura 5.1 – Uma arquitetura hierárquica de planejamento embarcado baseada em rede de tarefas para aumentar a capacidade de resposta de satélites.... 95
Figura 5.2 – Fluxo de abstração das tarefas na hierarquia de camadas da HARPIA. ... 109
Figura 5.3 – Mecanismo de decomposição e execução de uma ação primitiva na camada de aplicação do OBSW. 112
Figura 6.1 – Domínio da linguagem HML + ISISml. 120
Figura 6.2 – Exemplo reduzido para o arquivo da representação hierárquica. 122
Figura 6.3 – Exemplo reduzido para o arquivo da representação estrutural. 124
Figura 6.4 – Exemplo de uma ação descrita em ISISml para habilitar o modo imageamento da câmera do satélite. 127
Figura 6.5 – Exemplo de eventos descritos na linguagem. 130
Figura 6.6 – Relação das propriedades e categoria dos recursos da linguagem estendida. 131
Figura 7.1 – Mecanismo de inspeção de busca progressiva e de ordem total do Liger. 142
Figura 7.2 – Tipos de estruturas de decomposição em HTN. 145
Figura 7.3 – Motor de inferência de estados para ações durativas. 152
Figura 7.4 – Exemplo de uma HTN para o domínio de um satélite. 154
Figura 8.1 – Proposta do estudo de caso no contexto dos objetivos desta Tese. 164
Figura 8.2 – Perspectiva do satélite Amazonia-1. 166
Figura 8.3 – Representação simplificada do módulo carga útil e da plataforma de serviço do Amazonia-1. 167
Figura 8.4 – Redes inicias de tarefas (Amazonia-1). 173
Figura 8.5 – Tarefas abstratas e métodos de decomposição na linguagem estendida (Amazonia-1). 177
Figura 8.6 – Lista de tarefas primitivas na linguagem estendida (Amazonia-1). 178

Figura 8.7 – Exemplo da especificação de elemento e variáveis de na linguagem estendida (Amazonia-1). 181
Figura 8.8 – Exemplo da especificação de domínios na linguagem estendida (Amazonia-1)..... 181
Figura 8.9 – Variáveis de estados no domínio do tempo para o Objetivo 2 do Cenário I. 190
Figura 8.10 – Uso de memória e energia em função do tempo para o Cenário II. 192
Figura 8.11 – Tempo de Planejamento da HARPIA para dez configurações da Versão III do Cenário II. 194
Figura 8.12 – Inicialização do modelo e adição dos objetivos na HARPIA. 195
Figura 8.13 – Console da HARPIA no modo *verbose* com a reprodução de parte do plano encontrado para o Cenário III. 196
Figura 8.14 – Taxas de sucesso da HARPIA segundo diferentes tamanhos de problemas. 198
Figura 8.15 – Análise da escalabilidade da HARPIA conforme o número de objetivos contidos no problema. 204
Figura 8.16 – Experimento de vinte cenários considerando nós visitados *versus* ações primitivas..... 206
Figura 8.17 – Plano com método HTN alternativo a uma restrição de um evento exógeno de visibilidade com a estação..... 208
Figura 8.18 – Trecho da codificação da HARPIA para verificação (à esquerda) e execução (à direita) de uma ação primitiva do modelo Amazonia-1. 210
Figura 8.19 – Diagrama de atividades representando a estratégia de monitoração e recuperação de falhas de execução da HARPIA..... 211
Figura 8.20 – Tempo de replanejamento da HARPIA para simulação de falhas ocorridas na execução..... 214

LISTA DE TABELAS

Pág.

Tabela 1.1 – Estrutura dos capítulos de fundamentação teórica e estado da arte. 12
Tabela 2.1 – Autonomia versus Automação no contexto de sistemas embarcados. 18
Tabela 2.2 – Missões espaciais e suas motivações para a adoção de autonomia. 20
Tabela 2.3 – Classificação de autonomia operacional segundo a ECSS. 25
Tabela 2.4 – Proposta de um novo nível de autonomia operacional. 32
Tabela 3.1 – Especificação da função de transição de estados. 38
Tabela 3.2 – Suposições restritivas impostas pelo planejamento clássico. 39
Tabela 3.3 – Comparação entre as abordagens de espaço de busca. 54
Tabela 3.4 – Relações temporais de Allen. 60
Tabela 4.1 – Linguagens de definição usadas no domínio espacial. 78
Tabela 4.2 – Abordagens e estratégias de planejamento no domínio espacial 84
Tabela 4.3 – Comparativo dos planejadores embarcados da área espacial. 85
Tabela 4.4 – Sumário comparativo das características essenciais dos sistemas de planejamento. 90
Tabela 4.5 – Sumário comparativo das características gerais dos sistemas de planejamento. 91
Tabela 6.1 – Transformações sofridas por um objetivo até ser representável por uma HTN. 114
Tabela 7.1 – Análise comparativa das abordagens de planejamento clássico, HTN e temporal. 138
Tabela 7.2 – Comparativo e estratégias de planejamento. 162
Tabela 8.1 – Consumo dos equipamentos modelados para o estudo de caso 170
Tabela 8.2 – Restrições temporais na operação dos equipamentos. 171
Tabela 8.3 – Decomposição hierárquica das tarefas compostas do estudo de caso... 175
Tabela 8.4 – Especificação dos elementos e domínios do estudo de caso (Amazonia-1). 179

Tabela 8.5 – Itens modelados da Representação Estática para o estudo de caso (Amazonia-1). 182
Tabela 8.6 – Precondições e efeitos das ações primitivas do estudo de caso (Amazonia-1)..... 183
Tabela 8.7 – Eventos modelados pela HML/ISISml para o estudo de caso (Amazonia-1)..... 185
Tabela 8.8 – Enumeração dos objetivos no âmbito da missão espacial considerando-se o modelo construído (Amazonia-1). 186
Tabela 8.9 – Descrição dos cenários experimentais I, II e II. 188
Tabela 8.10 – Resultados de desempenho da HARPIA para o Cenário I. 189
Tabela 8.11 – Solução encontrada pela HARPIA para Cenário I e Objetivo 2..... 190
Tabela 8.12 – Dados de desempenho da HARPIA para o Cenário II em três versões.. 193
Tabela 8.13 – Solução de refinamento encontrada pela HARPIA utilizando a heurística integrada. 200
Tabela 8.14 – Planos resultantes obtidos pela HARPIA para cenários de refinamento de estados. 201
Tabela 8.15 – Tempo médio de planejamento de problemas gerados aleatoriamente. 202
Tabela 8.16 – Variância e desvio padrão para os diferentes tempos de execução do experimento realizado. 205
Tabela 8.17 – Verificação de estados realizada pelo Gerenciador de Atividades da HARPIA..... 209
Tabela 8.18 – Descrição dos cenários de simulação de falhas na execução do plano do estudo de caso. 213
Tabela 8.19 – Tempo médio de replanejamento da HARPIA no âmbito de falhas de execução..... 215
Tabela 8.20 – Objetivo 5 com falha de execução na ação A11. 216
Tabela 8.21 – Plano de recuperação após falha de execução..... 217

Tabela 8.22 – Comparação de dados de desempenho das soluções de planejamento embarcado em satélites. 219
Tabela 8.23 – Ambiente computacional espacial das soluções de planejamento embarcado. 220
Tabela 8.24 – Sumário da taxa de sucesso da HARPIA para as simulações realizadas. 221
Tabela 8.25 – Síntese dos tempos de planejamento da HARPIA para as simulações realizadas..... 221
Tabela C.1 – Conjunto de estados dos equipamentos da carga útil do Amazonia-1 selecionados para os cenários experimentais. 265

LISTA DE ALGORITMOS

Algoritmo 3.1 – Planejamento no Espaço de Planos 50
Algoritmo 3.2 – Planejamento de Busca Progressiva 52
Algoritmo 3.3 – Planejamento de Busca Regressiva 53
Algoritmo 4.1 – Um algoritmo abstrato para planejamento em HTN 70
Algoritmo 4.2 – Algoritmo Reparo Iterativo 80
Algoritmo 7.1 – Algoritmo de Planejamento Liger 149
Algoritmo 7.2 – Reparo por Refinamento de Estados 157

LISTA DE SIGLAS E ABREVIATURAS

ACDH	*Attitude Control and Data Handling*
AML	*Aspen Modeling Language*
ANML	*Action Notation Modeling Language*
AOC	*Attitude and Orbit Control*
APSI	*Advanced Planning and Scheduling Initiative*
APSI-TRF	*APSI-Timeline-based Representation Framework*
ASE	*Autonomous Sciencecraft Experiment*
ASPEN	*Automated Scheduling and Planning Environment*
BNF	*Backus-Naur Form*
CASPER	*Continuous Activity Scheduling, Planning, Execution and Replanning*
CBERS	*China-Brazil Earth Resources Satellite*
CCD	*Charged-Coupled Device*
CLARAty	*Coupled Layer Architecture for Robotic Autonomy*
COMAV	Computador Avançado
COTS	*Commercial Off-The-Shelf*
CPU	*Central Processing Unit*
CSP	*Constraint Satisfaction Problems*
DDL	*Domain Definition Language*
DDR	*Digital Data Recorder*
DEC	Divisão de Eletrônica Espacial e Computação
DFS	*Depth-First Search*
DLR	*Deutsches Zentrum für Luft- und Raumfahrt*
DS-1	*Deep Space One*
DSQEE	*Datalogger SeQuence Execution Engine*
DSS	*Digital Signal Switch*
DT	*Data Transmitter*

DWR	*Dock Worker Robots*
EBNF	*Extended Backus Naur-Form*
ECSS	*European Cooperation for Space Standardization*
EO-1	*Earth Observing One*
ESA	*European Space Agency*
ESTEC	*European Space Research and Technology Centre*
ETCE	Engenharia, Tecnologia e Ciências Espaciais
EUROPA	*Extensible Universal Remote Operations Planning Architecture*
FCP	*Flight Control Procedure*
FDIR	*Fault, Detection, Isolation and Recovery*
GOAC	*Goal-Oriented Autonomous Controller*
GOESA	*Goal-based Enabling Software Architecture*
GPS	*Global Positioning System*
HARPIA	*Hierarchical Architecture for Real-time Planning in Intelligent Space Applications*
HML	*Hierarchical Modeling Language*
HSTS	*Heuristic Scheduling Testbed System*
HTN	*Hierarchical Task Network*
IA	Inteligência Artificial
IDEA	*Intelligent Distributed Execution Architecture*
INPE	Instituto Nacional de Pesquisas Espaciais
IPC	*International Planning Competition*
IPEX	*Intelligent Payload EXperiment*
ISIS	*Internal State Inference Service*
ISISml	*ISIS modeling language*
JSHOP2	*Java Simple Hierarchical Ordered Planner-2*
LEOP	*Launch and Early Orbit Phase*
LIFO	*Last In, First Out*
NDDL	*New Domain Definition Language*

MC	*Model Checking*
MDP	*Markov Decision Process*
MTL	*Mission Time Line*
NASA	*National Aeronautics and Space Administration*
OBCP	*On-Board Control Procedures*
OBDH	*On-Board Data Handling*
OBETTE	*On-Board Event Triggered Timeline Extension*
OBoTis	*On-Board TIme Selection*
OBSW	*On-Board SoftWare*
OBT	*On-board Time*
OGATE	*On-Ground Autonomy Test Environment*
PDDL	*Planning Domain Definition Language*
POCL	*Partial Order Causal Link*
POMDP	*Partially Observable Markov Decision Process*
POP	*Partial Order Planning*
PMM	Plataforma Multimissão
PS	*Planner/Scheduler*
P&S	*Planejamento e Escalonamento*
PUS	*Packet Utilization Standard*
RAM	*Random Access Memory*
RASSO	*Resources Allocation Service for Scientific Opportunities*
RAX	*Remote Agent Experiment*
RTEMS	*Real-Time Executive for Multiprocessor Systems*
RTU	*Remote Terminal Unit*
SCL	*Spacecraft Command Language*
SHOP2	*Simple Hierarchical Ordered Planner-2*
SIPE	*System for Interactive Planning and Execution*
SPARC	*Scalable Processor Architecture*
STL	*Standard Template Library*

STN	*Simple Task Network*
STRIPS	*Stanford Research Institute Problem Solver*
SUBORD	Supervisão de Bordo
TT&C	*Telemetry Telecommand and Control*
TVCR	*Timeline Validation Control and Repair*
TWT	*Traveling-Wave Tube*
VAMOS	*Verification of Autonomous Mission Planning On-board a Spacecraft*
VHPOP	*Versatile Heuristic Partial Order Planner*
V&V	Verificação e Validação
WFI	*Wide Field Imaging*
XML	*eXtensible Markup Language*
XIDDL	*XML IDEA Domain Definition Language*

SUMÁRIO

Pág.

1 INTRODUÇÃO 1

1.1. Motivação 6

1.2. Definição do problema 8

1.3. Objetivos 9

1.4. Metodologia 10

1.5. Organização 11

2 AUTONOMIA OPERACIONAL DE VEÍCULOS ESPACIAIS 13

2.1. Introdução à autonomia operacional de veículos espaciais 13

2.1.1. Autonomia *versus* automação 16

2.1.2. Motivações e benefícios 19

2.1.3. Riscos e considerações 21

2.1.4. Implicações ao segmento solo 23

2.2. Níveis de autonomia operacional em missões espaciais 25

2.3. Transferência do planejamento de operação ao segmento espacial 28

2.3.1. A operação baseada em sequência de comandos 28

2.3.2. A operação baseada em objetivos 31

3 FUNDAMENTAÇÃO TEÓRICA: PLANEJAMENTO AUTOMATIZADO EM INTELIGÊNCIA ARTIFICIAL 33

3.1. Planejamento automatizado 33

3.2. O planejamento clássico em inteligência artificial 35

3.2.1. Definição formal do problema 36

3.2.2. Suposições restritivas do modelo teórico 38

3.2.3. Domínio clássico 40

3.2.4. Linguagens de descrição 42

3.3. Conceitos do planejamento clássico 45

3.3.1. Espaço de busca 45

3.3.2. Ordenação das ações 47
3.3.3. Algoritmo de ordem parcial 49
3.3.4. Estratégia de comprometimento 51
3.3.5. Métodos e direções de busca 51
3.4. Tipos de sistemas de planejamento automatizado 54
3.5. Extensão do planejamento clássico 57
3.5.1. Planejamento e escalonamento 57
3.5.2. Abordagem em cascata ou integrada 58
3.5.3. Representação temporal 59
4 ESTADO DA ARTE DO DOMÍNIO ESPACIAL: TÉCNICAS DE PLANEJAMENTO AUTOMATIZADO COM POTENCIAL APLICAÇÃO A BORDO DE SATÉLITES. 63
4.1. Técnicas de planejamento automatizado para domínios não clássicos com aplicação na área espacial 63
4.1.1. Técnicas de planejamento em domínio determinístico 63
4.1.1.1. Planejadores temporais baseados em CSP 64
4.1.1.2. Planejadores hierárquicos baseados em HTN 67
4.1.2. Técnicas de planejamento em domínio não determinístico 72
4.1.2.1. Planejamento sob incerteza (MDP e MC) 72
4.2. Estado da arte em soluções de planejamento embarcado 76
4.2.1. Planejamento em lote *versus* contínuo 76
4.2.2. Linguagens de descrição do domínio espacial 77
4.2.3. HSTS e RAX-PS 79
4.2.4. CASPER 79
4.2.5. EUROPA 81
4.2.6. APSI-TRF 82
4.2.7. RASSO e LetMeDo 82
4.2.8. Demais planejadores 83
4.2.9. Análise comparativa 83
4.3. Estado da arte em sistemas embarcados com planejamento automatizado 85

4.4. Análise comparativa dos trabalhos 89
5 HARPIA: UMA ARQUITETURA HIEÁRQUICA DE PLANEJAMENTO EMBARCADO BASEADA EM REDE DE TAREFAS PARA AUMENTAR A AUTONOMIA OPERACIONAL DE SATÉLITES 92
5.1. Considerações da pesquisa 92
5.2. Hierarchical Architecture for Real-time Planning in Intelligent Space Applications 94
5.3. Camadas e componentes 97
5.4. A camada de decisão 97
5.4.1. Planejamento de missão 98
5.4.1.1. Tradutor de objetivos de missão 98
5.4.1.2. Planejador embarcado 99
5.4.2. Gerenciador de respostas à ciência (dados da carga útil) 101
5.4.3. Sumário 102
5.5. A camada reativa 103
5.5.1. Gerenciador de atividades 104
5.5.2. Tabela de eventos temporais endógenos 106
5.5.3. Sumário 107
5.6. Domínio representado por variáveis de estados 108
5.7. Nível de abstração das tarefas hierárquicas nas camadas 109
5.8. Interface com o software de voo e o padrão PUS 110
5.9. Integração de planejamento e execução 111
6 UMA EXTENSÃO DA ISISml PARA REPRESENTAÇÃO DO MODELO EMBARCADO BASEADA EM REDE DE TAREFAS HIERÁRQUICAS 113
6.1. Transformação do objetivo em rede de tarefas 113
6.2. Motivações para a representação em HTN 115
6.3. Considerações da pesquisa 116
6.4. Hierarchical Modeling Language: uma extensão à ISISml 119
6.4.1. Descrição estática 120

6.4.1.1. Representação hierárquica 121
6.4.1.2. Representação estrutural 123
6.4.2. Descrição comportamental 125
6.4.2.1. Ação 126
6.4.2.2. Evento 128
6.4.3. Representação de consumo de recursos 130
6.5. Representação do domínio 133
6.5.1. Especificação do domínio hierárquico 134
6.5.2. Especificação do problema de planejamento embarcado em satélites 135
7 UMA SOLUÇÃO HÍBRIDA USANDO PLANEJAMENTO HIERÁRQUICO E GENERATIVO COM INFERÊNCIA DE ESTADOS E GERENCIAMENTO DE TEMPO E RECURSOS DAS ATIVIDADES 137
7.1. Considerações da pesquisa 137
7.2. Formalismo do planejamento de missão em HTN 139
7.3. Visão geral da abordagem 141
7.4. Representação temporal 146
7.5. Algoritmo de planejamento embarcado 147
7.5.1. Estratégia de comprometimento 151
7.5.2. Processo de inferência de estados 152
7.6. Reparo e replanejamento 155
7.7. Algoritmo de reparo por refinamento de estados 156
7.8. Análise comparativa 161
8 ESTUDO DE CASO E CENÁRIOS EXPERIMENTAIS 164
8.1. Ambiente computacional embarcado 164
8.2. Missão espacial do estudo de caso 166
8.3. Considerações sobre a modelagem do Amazonia-1 168
8.4. Objetivos do estudo de caso 171
8.5. Estudo de caso da linguagem estendida 171
8.5.1. Especificação hierárquica 171

8.5.2. Especificação estrutural 178
8.5.3. Especificação comportamental 182
8.6. Cenários experimentais de planejamento automatizado 185
8.7. Sumário dos cenários de operação autônoma 187
8.7.1. Cenário I (equipamentos desligados) 188
8.7.2. Cenário II (diferentes objetivos) 191
8.7.3. Cenário III (replanejamento com novos objetivos) 194
8.7.4. Cenário IV (análise da capacidade de resposta) 197
8.7.5. Cenário V (análise de tempo médio de execução) 202
8.7.6. Cenário VI (análise de cenário nominal e com refinamento) 206
8.7.7. Cenário VII (falha em restrições de eventos exógenos) 207
8.7.8. Cenário VIII (simulação de falha na execução do plano) 208
8.8. Comparação de desempenho com demais abordagens 218
8.8.1. Análise comparativa 218
9 CONCLUSÃO 222
9.1. Contribuições da Tese 224
9.1.1. Áreas do conhecimento das contribuições 227
9.1.2. Contribuições para o INPE 228
9.2. Publicações 229
9.3. Trabalhos futuros 231
REFERÊNCIAS BIBLIOGRÁFICAS 234
APÊNDICE A – O DOMÍNIO DO ESTUDO DE CASO EM ISISml 249
A.1 O domínio hierárquico em ISISml 249
A.2 O domínio estrutural em ISISml 253
APÊNDICE B – O MODELO EMBARCADO DO ESTUDO DE CASO 255
B.1 Ações do modelo Amazonia-1 255
B.2 Eventos do modelo Amazonia-1 263
APÊNDICE C – SIMULAÇÕES DE TELEMETRIA PARA OS CENÁRIOS EXPERIMENTAIS 264

1 INTRODUÇÃO

Tradicionalmente, as missões espaciais não tripuladas, executadas por satélites e sondas, são controladas pelo paradigma de operação por sequências de comandos. O plano de voo é elaborado pela equipe de solo, o qual é transmitido periodicamente ao segmento espacial, por intermédio de uma sequência de comandos que serão executados pelo computador de bordo, de forma imediata ou em instantes predefinidos, na forma de comandos temporizados.

A telemetria de serviço é monitorada pela equipe de solo, que averigua a saúde da plataforma e as condições da segurança em bordo. Os planos de operação são produzidos com base na telemetria e outros planos são gerados segundo os objetivos da missão e condicionados às restrições da engenharia de operação (CIVIDANES et al., 2019).

Embora constitua o paradigma vigente, esta forma de operação está presa à ideia que todas as atividades do veículo espacial devem ser previstas e aprovadas pelo segmento solo. Isso traz limitações, como a impossibilidade de responder em tempo hábil a eventos não previstos e a incapacidade de rever os planos a partir de falhas ou mudanças inesperadas no funcionamento dos equipamentos (CIVIDANES et al., 2019).

Para lidar com essas limitações, estudos têm sido conduzidos em direção a um maior nível de autonomia operacional a bordo dos satélites (JONSSON et al., 2007; INDRA et al., 2008; BOZZANO et al., 2008; KUCINSKIS, 2012; VASSEV; HINCHEY, 2013; D'ANGELO et al., 2017; TIPALDI; GLIELMO, 2018).

A autonomia pode ser medida como a capacidade de a plataforma espacial atender aos objetivos da missão sem apoio do centro de controle durante certo período de tempo (ECSS, 2008). Isso se traduz basicamente em delegar parte das atividades hoje realizadas no segmento solo para o software de bordo, de acordo com o nível de autonomia operacional desejado para certa missão.

Nesse caso, o segmento espacial deve possuir alguma capacidade de tomada de decisões, permitindo que solo consiga controlar o satélite a partir de objetivos a serem

cumpridos, ao invés de sequências de comandos. O software de voo deixa de ser um mero executor de comandos e passa a ter algum nível de inteligência computacional em bordo, de forma a não depender exclusivamente da intervenção do centro de controle para direcionar suas ações e decisões (CIVIDANES et al., 2019).

Segundo Amigoni et al. (2010), uma plataforma espacial autônoma deve ser capaz até certo nível de observar novos objetivos da missão, planejar suas atividades conforme as metas impostas, executar e monitorar as ações planejadas, detectar a presença de falhas e, nesses casos, implantar possíveis estratégias de recuperação. Para isso, um novo software de controle deve ser estruturado em bordo (LEMAI et al., 2006).

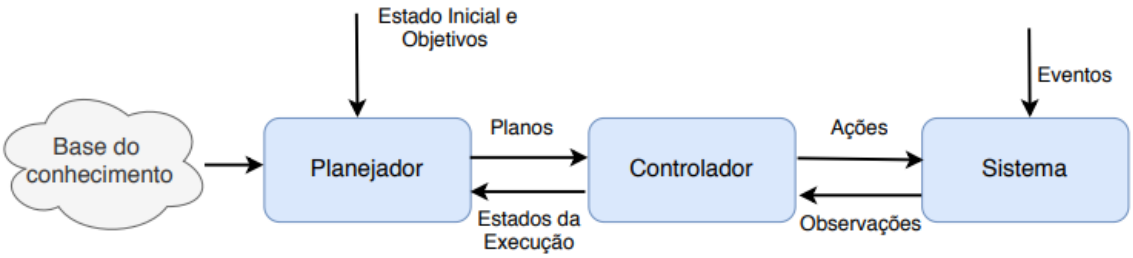
As etapas de planejamento e escalonamento são elementos cruciais para alcançar a autonomia em bordo do veículo espacial (KNIGHT et al., 2001, RUI et al., 2003 e AMIGONI et al., 2010). Elas são consideradas o cérebro de um satélite autônomo (HE et al., 2016). O planejamento constrói um plano de ações para alcançar os objetivos impostos, enquanto o escalonamento aloca tempo e recursos às atividades do plano.

O Planejamento Automatizado é a área da Inteligência Artificial (IA) que estuda modelos e algoritmos computacionais para análise, produção e execução de planos de ação para o atendimento a objetivos dados, de forma automatizada. Em sua forma mais geral, envolve diversos fatores como decisões lógicas, restrições temporais, consumo de recursos, cooperação, otimização, entre outros aspectos.

Como há diferentes formas de representar ações, eventos, estados e objetivos, é importante evidenciar um modelo conceitual para a compreensão geral de um problema de planejamento, representada pela Figura 1.1.

Em um sistema de planejamento automático a inteligência está relacionada ao mecanismo de busca utilizado e ao conhecimento do domínio da instância do problema. Uma arquitetura baseada em modelos de planejamento é um sistema controlado por software que emprega a representação do conhecimento sobre algum domínio específico para explorá-lo utilizando mecanismos de raciocínio apropriados.

Figura 1.1 – Modelo conceitual de um sistema com planejamento automatizado.



Fonte: Adaptado de Ghallab et al. (2004).

Neste contexto, foram estudadas por diversas agências espaciais, em especial, pela *National Aeronautics and Space Administration* (NASA), arquiteturas de software do segmento espacial que habilitem a autonomia em bordo. Um trabalho pioneiro e de referência na área de planejamento embarcado é a arquitetura *Remote Agent Experiment* (RAX - BERNARD et al., 1998) concebida para operar autonomamente a sonda espacial *Deep Space-1* (DS-1) a partir de objetivos recebidos de solo.

O *Autonomous Sciencecraft Experiment* (ASE - CHIEN et al., 2005), também da NASA, foi o primeiro software relacionado a uma missão de sensoriamento remoto a incorporar conceitos de autonomia, evidenciando que a autonomia não está restrita a missões *Deep Space*. Seu maior diferencial em relação ao antecessor RAX foi a capacidade de observar novos objetivos a bordo e replanejar as operações durante a execução do plano, de maneira '*online*'.

Os dois sistemas de planejamento foram organizados no conceito de três camadas. Há uma camada para o planejador/escalonador, uma intermediária de monitoração e controle e uma terceira para execução do plano, mais próxima ao hardware. Porém, utilizou-se nelas uma semântica de modelo diferente para cada camada, o que futuramente se mostrou uma solução inapropriada (CEBALLOS et al. 2011, p. 2; AMIGONI et al., 2010; VERMA et al., 2005). Já antevendo o problema das diferentes representações do conhecimento nas três camadas, a principal motivação do estudo da *Intelligent Distributed Execution Architecture* (IDEA - MUSCETTOLA et al., 2002) foi a unificação do modelo de conhecimento.

Em detrimento à organização em três camadas, adotou-se uma solução multiagentes com planejamento distribuído para tratar diferentes aspectos da aplicação e visando melhorar o tempo de resposta, o que traz segundo Lemai et al. (2006, p. 5) e Muñoz et al. (2018), maior complexidade para a coordenação das tarefas. Todavia, McGann et al. (2009) chamam a atenção para o fato de que o IDEA não tem mecanismos de resolução de conflitos entre os diferentes agentes de planejamento.

Tais propostas inclusive os trabalhos desenvolvidos (KUCINSKIS, 2007; KUCINSKIS, 2012) pelo Instituto Nacional de Pesquisas Espaciais (INPE) empregaram planejadores temporais a partir técnicas restritas à abordagem de problemas de satisfação de restrições – do inglês *Constraint Satisfaction Problems* (CSP).

De acordo com (CHIEN et al., 2012; MAYER et al., 2016), a combinação de planejadores temporais aliado a técnicas de CSP representa hoje a principal abordagem de planejamento para o domínio espacial, sobretudo se o observado o histórico de suas soluções precursoras (FRANK; JONSSON, 2003; AI-CHANG et al., 2004; FRATINI et al., 2008; CESTA et al., 2009; KNIGHT et al., 2014; BARREIRO et al., 2012) voltadas à automatização de operações em solo.

Parte do sucesso dos planejadores temporais do domínio embarcado espacial está relacionado com a sua flexibilidade para lidar situações não previstas. Qualquer mudança no estado da plataforma espacial pode desencadear o processo de planejamento, a fim de gerar um novo plano viável (PENG et al., 2018).

Apesar disso, essa abordagem carrega deficiências. Nela, torna-se difícil prever o tempo necessário para o reparo, pois a duração de cada plano pode não ser determinada (LONG et al., 2018). Seu custo computacional também é considerado alto para uma implementação embarcada, segundo Beaumet et al. (2011).

De acordo com os experimentos que divulgaram seus resultados (MUSCETTOLA et al., 1998; KNIGHT et al., 2001; KUCINSKIS; FERREIRA, 2013), os algoritmos levaram da ordem de dezenas de segundos ou até horas para replanejar um plano válido. A

experiência adquirida pelo INPE neste assunto mostra que dependendo das restrições do problema, o tempo de resposta pode não ser viável para um satélite em órbita.

Por outro lado, o tempo para reprogramação das atividades é de interesse particular ao Brasil, uma vez que as missões espaciais desenvolvidas pelo INPE foram até hoje voltadas a aplicações científicas e de sensoriamento remoto. Tais missões buscam na autonomia essencialmente o maior poder de reação do satélite.

Com objetivo de melhorar o tempo de resposta e atender às necessidades de uma aplicação espacial, a solução proposta nesta Tese propõe uma abordagem de planejamento baseada em *Hierarchical Task Network* (HTN - EROL et al., 1994) e inferência de estados.

Para lidar com estados não previstos do satélite, uma nova técnica de reparo em HTN denominada "reparo por refinamento de estados" foi elaborada. Ela se baseia numa abordagem híbrida que alia características do planejamento hierárquico à abordagem generativa do planejamento clássico.

Seu objetivo principal é dar mais capacidade de resposta ao planejador a situações não previstas pelo domínio hierárquico, sem depender de métodos alternativos de decomposição, aliviando ao mesmo tempo uma das grandes críticas imputadas à representação em HTN: o alto custo para a descrição do domínio hierárquico.

Ela também permite tratar as incertezas do ambiente, usando uma técnica determinística de planejamento que tem o potencial de trazer mais confiabilidade ao modelo. Segundo Chien et al. (2010), o domínio espacial deve ser altamente confiável para não produzir planos de operação que possam colocar em risco um bem tão valioso como um satélite.

Diferentemente dos demais estudos, a estratégia proposta neste trabalho traz mais confiabilidade ao codificar a experiência dos especialistas na solução a partir da representação em HTN. A adição de informações hierárquicas também leva a ganhos de desempenho e permite superar o estado da arte de sistemas embarcados do segmento espacial.

Para obter uma conexão mais forte com a prática de engenharia, propõe-se a implementação da solução proposta e de seus elementos dentro de uma arquitetura de planejamento embarcado organizada em camadas hierárquicas cujo objetivo é sua integração a um software de voo de satélites.

Da mesma forma, para a geração e representação do modelo a bordo de satélites, concebeu-se uma linguagem estendida baseada em HTN, tomando como referência outra linguagem do INPE: a *Internal State Inference Service Modeling Language* (ISISml – KUCINSKIS, 2012).

A ideia da pesquisa foi estendê-la para acomodar a solução hierárquica das atividades operacionais da missão, mantendo-se as melhores características da linguagem precursora.

A fim de atestar a solução de planejamento embarcado proposta nesta Tese, realizaram-se estudos experimentais realísticos voltados a um satélite de sensoriamento remoto do INPE cujo domínio foi descrito a partir da linguagem estendida. O domínio escolhido para o estudo de caso tem importância ao Brasil, considerando o histórico de missões espaciais desenvolvidas pelo INPE até hoje.

1.1. Motivação

O número de trabalhos na área de autonomia[1] que se propõem a ser embarcados em satélites ainda é muito restrito, sendo esta uma área com potencial de novas contribuições. O presente trabalho pretende colaborar na investigação na área de planejamento automatizado de missão em sistemas embarcados da área espacial, contribuindo para a crescente utilização de técnicas de planejamento e escalonamento em problemas reais.

[1] Esta tese foca em trabalhos que utilizam soluções de planejamento em Inteligência Artificial para aumentar a autonomia operacional de satélites. Estes trabalhos são descritos no Capítulo 4.

Devido às suposições restritivas impostas pelo modelo teórico[2] do planejamento clássico, existe uma lacuna entre os sistemas acadêmicos e do mundo real que precisam ser continuamente conectados, de modo a integrar a teoria do planejamento à complexidade dos problemas práticos, transferindo as inovações da teoria aos planejadores aplicados.

A presente Tese tem como motivação contribuir para o estreitamento entre o problema e a aplicação, notadamente no domínio embarcado de satélites, aliando técnicas do planejamento clássico a de planejadores aplicados a partir de uma solução híbrida de planejamento embarcado.

Em relação à motivação de uma aplicação espacial com planejamento automatizado, destaca-se que um satélite com maior autonomia operacional poderia diminuir drasticamente o tempo de reação para obter mais observações sobre o evento ocorrido, provendo maior celeridade na transmissão de alertas às autoridades competentes.

No entanto, as soluções de planejamento embarcado empregadas na área espacial até hoje (MUSCETTOLA et al., 1998; KNIGHT et al., 2001; KUCINSKIS; FERREIRA, 2013), seja por experimentos com herança de voo ou de laboratório (que empregaram processadores qualificados para o espaço), não permitiram o replanejamento das atividades em tempo real.

Diante deste cenário, tem-se a motivação de investigar novas soluções de planejamento embarcado que possa aumentar a capacidade de resposta de satélites na geração de novos planos a bordo.

Isso permitiria dar uma resposta em tempo hábil a eventos de interesse dos usuários da missão, que precisam de reações céleres, tal como um desastre ambiental, no caso

[2] De acordo com Nau (2007), trata-se de um modelo bastante simplificado para lidar com aplicações de interesse prático. Esse modelo será abordado com mais detalhes no Capítulo 3.

de um satélite de monitoramento ambiental, ou um evento científico de curta duração no âmbito de missões científicas.

Em missões de observação da Terra isso traria maior eficiência no combate às queimadas clandestinas, ao desmatamento ilegal, aos acidentes ambientais petrolíferos, dentre outros eventos nocivos ao ecossistema e à sociedade, como enchentes e alagamentos. Em um satélite ágil de sensoriamento remoto, por exemplo, o menor tempo de reação permitiria obter novas observações na mesma órbita em que o evento fora detectado.

Já em uma missão científica que busca eventos imprevisíveis e transientes (*e.g.*, perturbação ionosférica) a dependência com o segmento solo faz com que o tempo de reação seja possivelmente maior que a própria duração do evento.

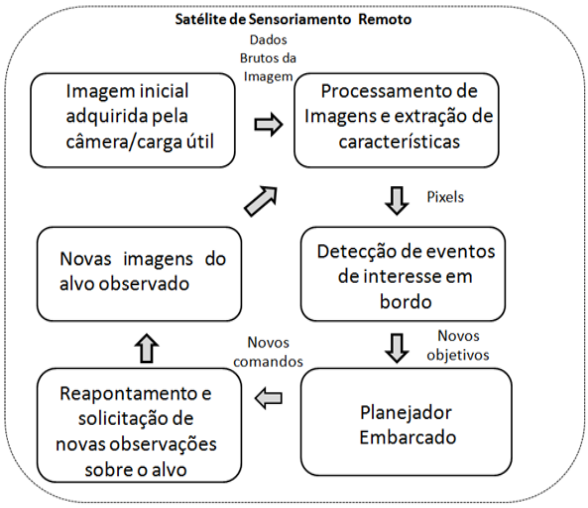
A maior capacidade de resposta do segmento espacial poderia trazer mais observações a esses fenômenos ocorridos no espaço, aumentando o retorno dos dados científicos da missão.

1.2. Definição do problema

A fim de caracterizar o problema que essa Tese pretende tratar, a Figura 1.2 exemplifica um cenário de operação, onde um satélite de sensoriamento remoto consegue reagir a eventos de interesse da missão sem a intervenção direta do segmento solo.

Um evento ambiental, como uma queimada, rompimento de barragem ou alagamento, é identificado em órbita a partir da análise dos dados da carga útil. O novo evento detectado gera um novo objetivo a ser cumprido pelo satélite, o qual é encaminhado ao planejador embarcado. Este último produz uma sequência de comandos cujo resultado final é permitir que o satélite faça novas observações sobre o alvo desejado.

Figura 1.2 – Cenário de operação espacial em ciclo fechado para um satélite com autonomia a bordo.



Fonte: Produção do autor.

1.3. Objetivos

O presente trabalho objetiva explorar a adoção de uma solução híbrida e unificada de planejamento e escalonamento embarcados, que faça uso das melhores características de planejamento hierárquico (HTN) e da abordagem generativa baseada em ações do planejamento clássico, cujo propósito é aumentar a autonomia operacional de satélites.

Tal objetivo implica na criação de uma nova forma de representação do conhecimento a bordo, de um planejador computacionalmente 'leve[3]' e compatível com as restrições impostas pelo hardware de voo qualificado para uso espacial, que possui baixa capacidade de processamento.

Para obter uma conexão mais forte com a engenharia espacial, um planejador híbrido e modelo embarcado são propostos, implementados e testados como elementos de uma arquitetura de planejamento automatizado organizado em camadas que se propõe a ser integrável a um software de voo de satélites.

[3] O termo computacionalmente leve se refere ao baixo custo computacional, ou seja, que não demande muito processamento computacional para a sua execução.

Tal arquitetura possui as seguintes características: planejamento baseado em rede de tarefas e inferência de estados; domínio hierárquico para representação de diferentes níveis de abstração, e organização em camadas que permite monitoração e controle das atividades, além de prover maior compatibilidade com um software de voo de satélites.

O objetivo é implementar a proposta de solução híbrida considerando o estado da arte atual de trabalhos de planejamento embarcado de missão em satélites. Além de características gerais de soluções e modelos de planejamento empregados, serão confrontados dados de desempenho obtidos pelos trabalhos que tiveram seus resultados divulgados.

1.4. Metodologia

A metodologia empregada nesta Tese pode ser classificada como a de uma pesquisa aplicada, pois está associada à solução de um problema de aplicação prática cujo estudo é dirigido para maior autonomia de satélites.

A metodologia é definida nas seguintes etapas:

(i) estudo sobre conceitos de autonomia operacional de satélites;

(ii) pesquisa em modelos e técnicas de Planejamento em Inteligência Artificial mais adequados ao domínio embarcado, considerando as particularidades da área espacial;

(iii) estabelecimento do estado da arte em linguagens e soluções de planejamento automatizado a bordo de satélites;

(iv) concepção de uma solução híbrida de planejamento embarcado baseada em rede de tarefas hierárquicas e inferência de estados para aumentar a autonomia operacional de satélites;

(v) proposta de extensão de uma linguagem de descrição do INPE visando adaptá-la para acomodar o domínio hierárquico previsto pela solução embarcada deste trabalho;

(vi) definição de uma arquitetura hierárquica de planejamento embarcado organizado em camadas, que prevê os elementos que realizam a solução híbrida proposta neste trabalho;

(vii) caracterização de uma missão espacial para estudo de caso e sua modelagem na linguagem estendida;

(viii) execução de cenários experimentais para atestar a viabilidade da solução híbrida em um ambiente computacional representativo da área espacial; e

(ix) análise e discussão dos resultados obtidos.

1.5. Organização

O restante desta Tese está estruturado como segue.

O Capítulo 2 introduz uma visão geral sobre a maior autonomia a bordo de veículos espaciais e seus conceitos principais.

O Capítulo 3 apresenta a fundamentação teórica relacionada à presente Tese: Planejamento em Inteligência Artificial, enfatizando-se os conceitos do planejamento clássico.

O Capítulo 4 traz as técnicas de planejamento com potencial aplicação a bordo de satélites, descrevendo-se o estado da arte dos principais modelos e soluções de planejamento empregados no domínio espacial a partir de análises comparativas dos trabalhos realizados.

O Capítulo 5 estabelece uma visão geral da arquitetura de planejamento embarcado baseada em rede de tarefas e inferência de estados que prevê os elementos necessários à solução híbrida proposta nesta Tese.

O Capítulo 6 traz a linguagem hierárquica estendida a partir da ISISml para a representação do conhecimento usada pela solução de planejamento proposta, apresentado-se também uma gramática da linguagem definida.

O Capítulo 7 apresenta a solução híbrida de planejamento embarcado baseado em rede de tarefas hierárquicas, abordando-se a técnica de replanejamento concebida para lidar com o não determinismo do ambiente.

O Capítulo 8 caracteriza a missão alvo para o estudo de caso e avalia os resultados experimentais obtidos segundo os cenários executados.

O Capítulo 9 sumariza os resultados e as contribuições alcançadas nesta Tese, apresentando potenciais temas de pesquisa que podem ser derivados como trabalhos futuros.

Para facilitar a compreensão dos capítulos subsequentes, a Tabela 1.1 sumariza a organização da fundamentação teórica e do estado da arte desta Tese.

Tabela 1.1 – Estrutura dos capítulos de fundamentação teórica e estado da arte.

Capítulo	Classificação	Tema
Capítulo 2	**Fundamentação Teórica** do Domínio da Pesquisa Aplicada	Autonomia Operacional de Veículos Espaciais
Capítulo 3	**Fundamentação Teórica** da Técnica Aplicada	Planejamento Automatizado em Inteligência Artificial
Capítulo 4	**Estado da Arte** de Técnicas de Planejamento Automatizado Embarcado Aplicado ao Domínio Espacial	Técnicas e Trabalhos de Planejamento Automatizado inseridos no Domínio Embarcado de Satélites

Fonte: Produção do autor.

2 AUTONOMIA OPERACIONAL DE VEÍCULOS ESPACIAIS

Esse Capítulo traz uma visão geral sobre conceitos de autonomia operacional de veículos espaciais. Para facilitar a clareza do texto, o termo 'autonomia operacional dos veículos espaciais', poderá ser referenciado apenas como 'autonomia do segmento espacial', ou simplesmente por 'autonomia espacial'.

2.1. Introdução à autonomia operacional de veículos espaciais

Uma tendência das missões espaciais é considerar o aumento do nível de autonomia em bordo dos veículos espaciais (JONSSON et al., 2007; TIPALDI; GLIELMO, 2018). Isso se deve basicamente ao crescente nível de complexidade das missões espaciais, à demanda contínua por redução de custos dos programas espaciais, à necessidade de obter menor tempo de reação a eventos não previsíveis e à necessidade de melhorar o retorno das missões em termos de quantidade e qualidade dos dados científicos coletados.

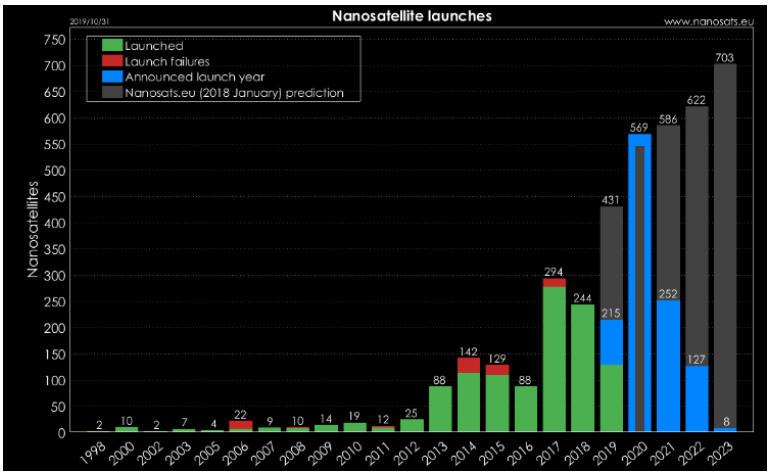
Há cerca de vinte anos, Teston et al. (1997) já previam que num futuro próximo as atividades de operação de missões evoluiriam do controle baseado em solo para monitoramento e controle embarcado, e que futuramente as funções mais complexas seriam gradativamente migradas para o computador embarcado. Quanto maior a complexidade de operação, mais decisivo será o papel da autonomia para o sucesso da missão (VERMA et al., 2005; KOLCIO et al., 2014).

Para Jonsson et al. (2007), em alguns casos, a autonomia espacial constitui fator preponderante para o sucesso da missão, como por exemplo nas missões de exploração planetária que têm atraso excessivo na comunicação e pouco tempo de visibilidade com as antenas em solo. Já as missões que monitoram desastres ambientais buscam autonomia visando essencialmente aumentar o poder de reação do satélite (vide GOETZ et al., 2013).

A autonomia também desempenha um papel crucial para reduzir a quantidade de recursos humanos e financeiros, especialmente importante para os países em

desenvolvimento, segundo Rui et al. (2005). O alto custo para manter uma equipe operacional qualificada nos grandes centros de controle pode levar a outros tipos de satélites a requererem autonomia, rompendo a ideia que ela seja algo factível somente em missões de grandes orçamentos. Isso se torna mais evidente para o cenário atual, em que há maior expectativa de vida útil das missões e previsão do aumento no número de lançamentos - veja a Figura 2.1 que mostra o quantitativo para a classe de satélites miniaturizados.

Figura 2.1 – Histórico e previsão de lançamentos de nanossatélites.



Fonte: Kulu (2019).

Nesse sentido, a autonomia deixou de ser apenas uma capacidade desejável. Há cerca de duas décadas, a comunidade da área espacial tem envidado esforços para dotar o segmento espacial com a autonomia necessária para atender diferentes tipos de aplicações espaciais.

O conceito básico de autonomia espacial parte do pressuposto que o veículo espacial deve assumir tarefas que minimizem a dependência dos operadores em solo. Tal conceito difere do modelo de operação tradicional "*human-in-the-loop",* pois exige pouca ou nenhuma intervenção humana.

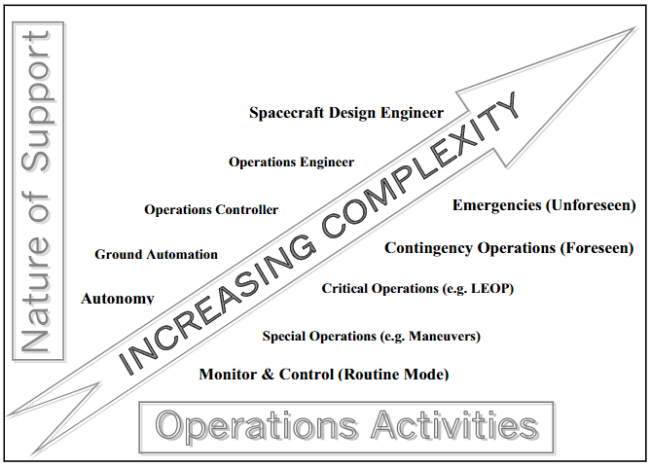
Isso foi feito inicialmente pela comunidade espacial na forma de automação dos comandos temporizados enviados a partir da equipe de operação em solo. A unidade de supervisão de bordo do satélite fica responsável por armazenar e despachar no instante programado o comando à aplicação pretendida.

Mais recentemente, comandos executados em resposta à ocorrência de eventos predeterminados passaram a ser empregados. Todas essas formas de comandar a plataforma espacial contribuem para o aumento da autonomia, mas compartilham da mesma limitação: demandam o planejamento em solo, realizado com grande antecedência.

Um exemplo de operações de rotina que tem histórico de autonomia nas missões atuais são as atividades de monitoramento e suporte à engenharia de bordo, segundo um levantamento feito por Truszkowski et al. (2009). Neste grupo de atividades, pode-se mencionar a tarefa de monitoramento de parâmetros de *housekeeping* que controla a saúde da plataforma espacial.

O alto grau de autonomia empregado nessa tarefa pode estar relacionado ao baixo nível de conhecimento necessário para realizá-la. Isso pode ser melhor observado pela Figura 2.2 que correlaciona a infraestrutura de suporte com as atividades de operação. Nela, pode-se constatar que a maior complexidade na operação implica em menos autonomia e a necessidade de mais conhecimento humano especializado.

Figura 2.2 – Correlação entre infraestrutura de suporte e atividades de operação.



Fonte: Van der Ha (2003).

Em complemento, a experiência do INPE mostra que também há outras tarefas mais complexas, desempenhadas com alto grau de autonomia nas missões da atualidade, tais como: controle de atitude, sequência inicial após a separação com o veículo

lançador, controle térmico, gerenciamento de modos do satélite, etc. Isso evidencia que o tempo de reação necessário para o satélite executar uma tarefa exerce grande influência no critério da adoção de autonomia.

Embora as atividades de operação da carga útil também possam requerer uma resposta rápida, elas continuam de inteira responsabilidade da equipe de solo, em parte por serem (dado o grande número e variedade de fatores a se considerar numa tomada de decisão), difíceis de modelar matematicamente ou através de uma árvore de decisão simples.

Van der Ha (2003) ressalva que embora a ascensão da autonomia tenha sido talvez um pouco mais lenta do que o previsto, resta pouca dúvida sobre a utilidade e o custo benefício dos conceitos de autonomia e automação na operação de satélites.

2.1.1. Autonomia *versus* automação

Na literatura é possível encontrar diferentes definições para os conceitos de autonomia e automação. Nesse cenário, é importante distingui-los no contexto de 'automação de comandos' e 'autonomia do software de voo'. Nos casos de automação de comandos, o satélite é apenas um executor de comandos e não tem o raciocínio ou discernimento da situação, estando limitado à execução de comandos predefinidos pelo segmento solo.

Basicamente, a automoção visa reduzir esforço e custo de operação através da automatização no processo de geração de comandos em bordo (CIVIDANES et al., 2019). Há quem defina este conceito limitado de autonomia como ‘automação’ que, segundo Cameron e Marshall (1998), consiste na prática de se empregar computador para se conduzir procedimentos repetitivos que são executados passo por passo (VASSEV; HINCHEY, 2013).

Na automação, o sistema provê instruções detalhadas de como realizar determinada tarefa e como resultados esperados poderiam ocorrer de forma previsível em reposta a um conjunto de comandos sequenciais.

Isso vem sendo explorado há décadas com o uso de telecomandos ditos 'temporizados' (*time-tagged telecommands*) e mais recentemente, através de comandos em resposta à ocorrência de eventos (*event-action commands*) e comandos disparados a partir de uma posição orbital (*position-tagged telecommands*) do satélite.

Até recentemente a noção de autonomia espacial era restrita a esses casos de autonomia limitada. Isso tem se mostrado adequado a missões de menor complexidade, que interagem com ambientes previsíveis, onde os comportamentos podem ser predeterminados em solo. Frost (2010) reforça que os sistemas automatizados não concebem escolhas próprias e quando se depararam com situações não planejadas ficam inertes, aguardando a intermediação humana.

Para alcançar um grau mais elevado de autonomia, entretanto, o segmento espacial deve ser capaz de tomar decisões rápidas e complexas, sem a intervenção do segmento solo, em resposta a algum evento identificado em bordo.

Para isso, um novo software de voo deve ser implementado a bordo. De acordo com (NARDONE et al., 2016), um (*On-Board Software* - OBSW) desempenha papel relevante na implementação de capacidades autônomas.

Com a adoção de autonomia no software de voo, um sistema espacial pode ser controlado, comandando-o para atingir um conjunto de objetivos (JONSSON et al., 2007), ao invés de uma sequência de comandos. O próprio sistema transforma os objetivos em sequências de comandos que cumprem cada objetivo.

Em termos de conceitos, Axmann (2010, p. 22) delimita autonomia como a capacidade de o sistema agir por sua conta própria dentro de certos limites. Para Beaumet (2008), o grau de autonomia de um satélite define sua capacidade de tomar decisões sem a intervenção de um operador.

A Tabela 2.1 apresenta de forma sumária as principais características entre os conceitos de Autonomia e Automação no contexto de sistemas embarcados da área

espacial. Outros detalhes sobre a definição de níveis de autonomia serão abordados na Seção 2.2.

Tabela 2.1 – Autonomia versus Automação no contexto de sistemas embarcados.

Descrição	Autonomia	Automação
Paradigma	Orientada a objetivos, adaptativa e independente[4].	Orientado a tarefas e inflexíveis. Utiliza procedimentos detalhados associados a eventos predefinidos.
Ambiente	Opera em ambientes não totalmente previsíveis.	Ambiente precisa ser bem caracterizado.
Executor	O executor trabalha com comandos gerados a partir de um planejador embarcado.	O executor trabalha com comandos residentes no software ou aqueles enviados pelo centro de controle.
Recursos	Gerenciamento dinâmico de recursos a bordo pelo planejador.	Não há gerenciamento de recursos a bordo. Os recursos são alocados, previamente e com margens, pela equipe de solo.
Modelo do Satélite	Há um modelo embarcado que simula as reações comportamentais do satélite.	Não há modelo do satélite a bordo. Qualquer análise futura do estado do satélite é realizada em solo seja mentalmente pelos operadores ou através de ferramentas computacionais de apoio.

Fonte: Adaptado de Cividanes et al. (2019).

É importante observar que o grau de autonomia depende das necessidades de cada missão (ROUFF, 2003, p. 8), e como colocado por Jonsson et al. (2007), não é algo do tipo "tudo ou nada".

Torna-se possível empregar menos autonomia em operações mais críticas, como a fase de *Launch and Early Orbit Phase* (LEOP); e adotar um grau mais elevado durante a operação de rotina, por exemplo. Isso se traduz na possibilidade de se definir níveis diferentes de autonomia ao longo do ciclo da missão (JONSSON et al., 2007).

[4] Independente até certo ponto, uma vez que atua em limites considerados seguros e pré-estabelecidos.

Dessa forma, cada plataforma espacial pode ser dotada de um nível específico de autonomia a partir da distribuição de responsabilidades entre o segmento solo e o satélite, que de forma sinérgica devem cumprir os objetivos da missão espacial.

2.1.2. Motivações e benefícios

Há tipos diferentes de missões que demandam autonomia espacial. De modo geral, as missões com maior necessidade de autonomia são aquelas que apresentam um ou mais dos seguintes aspectos (CHIEN; MORRIS, 2014): necessidade de menor tempo de reação a eventos; limitação na banda de comunicação de *downlink;* falta de previsibilidade do ambiente a ser explorado; restrita janela de visibilidade junto ao centro de controle; e latência na comunicação com a estação terrena. Os aspectos supracitados podem ser identificados em missões de sensoriamento remoto, missões científicas, interplanetárias e de comunicações[5] (TIPALDI; GLIELMO, 2015).

A Tabela 2.2 reúne as principais motivações que levam à adoção de autonomia, segundo as características das missões espaciais.

[5] Embora listados pelos autores citados, cabe a ressalva de que os satélites de comunicações do tipo geoestacionário não possuem grande demanda por autonomia, uma vez que estão em visada constante com a estação. O mesmo não se aplica aos de baixa altitude, como a constelação Iridium.

Tabela 2.2 – Missões espaciais e suas motivações para a adoção de autonomia.

Missão	Razões para a autonomia	Latência do sinal com a Terra	Tempo de Visibilidade
Científica (Órbita Baixa)	Aumentar a capacidade de reação a fenômenos de curta duração detectados em órbita	Em tempo quase real	Em alguns casos, apenas 10% do tempo
Espaço Profundo (Exploração Espacial)	Latência na comunicação, custo e complexidade de operação em solo	Dependendo da missão, pode chegar a horas	Muito Limitado
Sensoriamento Remoto (Órbita Baixa)	Aumentar a capacidade de reação a eventos ambientais detectados em órbita	Em tempo quase real	Em alguns casos, apenas 10% do tempo
Exploração Planetária (*Rovers*)	Latência na comunicação, custo e complexidade de operação em solo	Em geral, da ordem de dezenas de minutos	Muito limitado
Satélite de Comunicação (Geoestacionário)	Poucas razões, graças à visada constante; basicamente, redução de custo de operação	Em tempo quase real	100% do tempo
Satélite Miniaturizado (Órbita Baixa)	Principalmente redução de custo de operação, considerando o baixo custo da plataforma	Em tempo quase real	Em alguns casos, apenas 10% do tempo

Fonte: Produção do autor.

Sumarizam-se a seguir os benefícios alcançados a partir da autonomia no segmento espacial conforme (VAN DER HA, 2003; CHIEN et al., 2005; LEMAI et al., 2006; WOODS et al., 2006; JÓNSSON et al., 2007; CESTA et al., 2010; WILLE et al., 2013; GOETZ et al., 2013; TIPALDI; GLIELMO, 2015; TIPALDI ; GLIELMO, 2018):

- Aumentar a capacidade de resposta da missão;
- Aumentar a utilidade dos dados científicos retornados;
- Reduzir custos operacionais;
- Reduzir a interrupção de serviços da plataforma espacial;

- Minimizar a intervenção de solo;
- Reduzir margens de segurança na alocação de recursos; e
- Trazer maior independência à plataforma espacial.

2.1.3. Riscos e considerações

Algumas considerações devem ser brevemente colocadas em relação aos obstáculos que devem ser transpostos para aumentar a autonomia do segmento espacial.

Um primeiro aspecto está relacionado às limitações computacionais do ambiente espacial, uma vez que a autonomia implica necessariamente em mais tarefas desempenhadas a bordo. Isso pode ser parcialmente contornado com o avanço tecnológico dos processadores espaciais (VAN DER HA, 2003; TRUSZKOWSKI et al., 2009; BUTLER; PENNOTTI, 2013) aliado à possibilidade de se alocar hardware dedicado para tarefas específicas.

Outro desafio importante refere-se à autonomia implicar num profundo conhecimento do domínio espacial e do contexto operacional. Isso em razão da redução da participação humana nos processos de tomada de decisão (TIPALDI; GLIELMO 2015, p. 4).

De acordo com Grey (2012, p. 176), o risco de operar sem a intervenção humana decorre não só da incerteza que vem com o uso de um novo sistema autônomo, mas também no valor elevado de uma missão espacial; qualquer falha pode acarretar em severas repreensões à equipe. Isso traz resistência por parte de gerentes, pois veem a autonomia como risco potencial à missão.

Por outro lado, sabe-se que a partir do esforço inicial de se projetar, implementar e validar sistemas autônomos, é plenamente possível alcançar maior grau de maturidade quanto ao emprego da autonomia espacial. Recomenda-se incrementar o nível de autonomia gradativamente, amadurecendo o projeto da plataforma espacial autônoma à medida que se obtém herança de voo nessa tecnologia.

Outra insegurança citada pelos gerentes refere-se ao não determinismo[6] que um sistema autônomo pode trazer (vide D'ANGELO et al., 2017, e CODETTA-RAITERI; PORTINALE, 2015). De acordo com Kucinskis (2007, p. 28), saber exatamente como um sistema irá se comportar aumenta a segurança com relação à sua robustez, mas reduz o número de situações às quais o sistema poderá gerar uma resposta adequada.

Práticas tradicionais de Verificação e Validação (V&V) não são apropriadas para software de voo com autonomia (TIPALDI; GLIEMO 2015; NARDONE et al., 2016). De acordo com Eickhoff (2012, p. 204), arquiteturas de sistemas espaciais autônomos devem ser verificadas e validadas de forma muito completa. São necessários bancos de testes capazes de simular cenários detalhados.

A autonomia traz esforço adicional nas fases (V&V) da missão especialmente para o software de voo. É nessa linha que os trabalhos Brat e Jonsson (2005), Bozzano et al. (2011), Kucinskis (2011), Bensalem et al. (2014), Nardone et al. (2016), exploraram suas pesquisas.

Alguns trabalhos citados abaixo, que estão fora do escopo desta Tese, dão ênfase na validação do software de voo com autonomia em bordo; outros focam na validação do planejador embarcado, e alguns contemplam áreas correlatas, como resiliência e dependabilidade.

Vassev e Hincey (2013) mencionam o uso potencial de métodos formais para modelagem e validação do comportamento do software de voo com autonomia em bordo. Em Murray et al. (2013), apresenta-se um relatório técnico sobre resiliência aplicada a sistemas espaciais com autonomia na plataforma.

Já em Muñoz et al. (2014), trabalho em colaboração com a *European Space Agency* (ESA), propõe-se um ambiente de teste do segmento solo para validar arquiteturas de software com autonomia espacial para missões robóticas, denominado *On-Ground*

[6] Há técnicas voltadas a ambiente não determinístico que serão melhor descritas no Capítulo 4, cujos algoritmos utilizam modelos de natureza probabilística.

Autonomy Test Environment (OGATE), projeto relacionado ao *Goal-Oriented Autonomous Controller* (GOAC), da ESA, que será abordado com mais detalhes adiante.

De acordo com Wertz e Larson (1999, p. 890), todo esforço adicional advindo da autonomia deve ser considerado investimento, pois custos operacionais podem ser reduzidos. Pode-se ratificar isso pelo *satélite Earth Observing One* (EO-1), em que se estimou a economia de um milhão de dólares por ano, a partir do início de sua operação autônoma, conforme Chien et al. (2010).

Embora Wertz e Larson (1999, p. 29) sejam taxativos ao afirmar "com a maior capacidade computacional a bordo é claramente possível criar satélites completamente autônomos", sabe-se das barreiras técnicas e culturais que devem ser suplantadas. A partir do momento em que o retorno científico transpor o esforço adicional que leva à autonomia do satélite, esta tecnologia deverá ser cada vez mais empregada em missões futuras.

2.1.4. Implicações ao segmento solo

Gao et al. (2013) mencionam que devem ser consideradas todas as implicações para o segmento solo quando há aumento no nível de autonomia do segmento espacial.

Uma preocupação já expressada na literatura (CIVIDANES et al., 2019) é a perda da competência por parte dos operadores decorrente de longos períodos de operação autônoma de missões espaciais. Para evitar tal situação, recomenda-se treinar os operadores com simuladores, a fim de manter a competência não só nas operações rotineiras, mas especialmente em situações de anormalidade.

Um benefício imediato para o segmento solo é a potencial redução na carga de trabalho dos técnicos, engenheiros e cientistas, o que resulta num menor custo de operação. Os operadores não precisariam trabalhar em regime ‘24x7’ para esperar uma resposta e planejar novas operações imediatamente. Eles também despenderiam menos tempo com as tarefas rotineiras do dia a dia, podendo dedicar-se a atividades

mais complexas de planejamento em longo prazo (INDRA et al., 2008, p. 1; OLIVE 2012; PAN, 2016).

Em geral, os operadores se tornam mais propensos à adoção de maior nível de autonomia naquelas missões em operação há algum tempo. As possíveis explicações são o risco mais aceitável, dado que a missão se encontra próximo ao final de sua vida útil e o maior discernimento sobre as reações comportamentais da plataforma. Corroborando isso, pode-se destacar que a missão *Earth Observing One* (EO-1) da NASA permitiu a adoção de um planejador embarcado somente após o término de sua vida útil originalmente prevista (CHIEN et al., 2010 p.2).

No tocante aos riscos inerentes à operação espacial, Wertz e Larson (1999, p.24) destacam que a autonomia pode reduzir a complexidade da operação, o que diminui as chances de falhas humanas na operação em solo. Isso vem ocorrendo, por exemplo, com os subsistemas de controle de atitude e de controle térmico que operam cada vez mais de forma autônoma.

Tominaga (2010) frisa que um simples erro nas atividades de operação pode levar à execução de procedimentos falhos, causando danos irreversíveis em satélites, que podem comprometer os seus funcionamentos, colocando em risco todo o investimento alocado na missão. Por outro lado, Kucinskis (2012) menciona que sistemas autônomos não estão sujeitos a fatores humanos, como cansaço e variações de humor. Isso tende a elevar gradualmente a confiabilidade da missão na medida em que as tecnologias que habilitam a autonomia amadurecem.

A autonomia também permite a reconfiguração automática em bordo e a monitoração e execução das políticas de segurança selecionadas em tempo real (TIPALDI; GLIEMO, 2015). Ela endereça funções autônomas de diferentes maneiras, o que provê ao segmento espacial a capacidade de continuar operando a missão em casos de situações críticas sem depender da intervenção de centro controle, conforme previsto em (ECSS, 2008).

2.2. Níveis de autonomia operacional em missões espaciais

Para determinação do nível de autonomia operacional, consideram-se: tipo de missão, objetivos e prioridades, características orbitais, tempo de visibilidade com a estação terrena, conceitos de operações e restrições de comunicação (ESTEVE et al., 2012 e OLIVE, 2012). A Tabela 2.3 contém os graus de autonomia operacional estabelecidos segundo a *European Cooperation for Space Standardization* (ECSS) cujas normas são adotadas pela Coordenação de Engenharia do INPE.

Tabela 2.3 – Classificação de autonomia operacional segundo a ECSS.

Nível	Descrição	Funções
E1	Execução de missão controlada pelo segmento solo, autonomia limitada para aspectos de segurança	Controle em tempo real para operações nominais. Execução de comandos temporizados para aspectos de segurança
E2	Execução de planos planejados em solo	Capacidade de armazenar telecomandos temporizados com programação a bordo
E3	Execução em bordo de operação de forma adaptativa	Operações autônomas baseadas em eventos Execução de procedimentos de controle de operação em bordo
E4	Execução em bordo de operação orientada a objetivos	Operação baseada em objetivos com replanejamento a bordo

Fonte: Adaptado de ECSS (2008).

A norma ECSS-E-ST-70-11C (ECSS, 2008) define quatro níveis de autonomia no que diz respeito à execução de operações nominais de uma missão espacial. Os níveis de E1 a E4 indicam em ordem crescente o aumento da autonomia e a menor dependência do segmento solo.

Nas missões recentes da ESA, o planejamento de operações é realizado basicamente por *Flight Control Procedures* (FCP), *Mission Time Line* (MTL) e *On-Board Control Procedures* (OBCP), segundo Tipaldi e Glielmo (2018).

O primeiro nível (E1) é comumente utilizado em missões espaciais que têm bastante tempo de visibilidade com a estação terrena e, portanto, permite o controle em tempo

real do satélite. Adota-se o conceito de *Flight Control Procedures* que basicamente consiste no envio de sequências de telecomandos e verificação da telemetria recebida. Os FCPs são executados passo a passo pelo operador de solo, segundo Wander e Förstner (2013). Para este nível, cita-se como exemplo um satélite geoestacionário (*e.g.,* comunicações ou meteorológico) que fica em constante visada com a estação.

O segundo nível (E2) é onde reside a maioria das missões espaciais da atualidade, no qual se têm programação de comandos temporizados em bordo, utilizando um *Mission Time Line*, a partir do planejamento de operações realizado em solo; é bastante utilizado nas missões em que não há muito tempo de visibilidade junto à estação.

A maioria das missões da atualidade, assim como os programas espaciais do INPE, estão entre o nível E2 e E3. Para detalhes acerca dos níveis de autonomia adotados pelas últimas missões do INPE, vide Kucinskis e Ferreira (2010, p. 4).

O terceiro nível (E3) prevê procedimentos de controle de operação em bordo (OBCP). De acordo com Olive (2012), na comunidade europeia somente veículos espaciais que dispõem de tecnologias avançadas e inseridos em missões científicas, ou de observação da Terra, implementaram OBCPs.

Os OBCPs tornam mais adaptativa a execução de procedimentos em bordo, sendo, segundo Tipaldi e Gliemo (2018), uma forma primitiva de operação em ciclo fechado. São procedimentos autônomos escritos na forma de arquivos que se assemelham a *scripts*, implementados em uma linguagem de programação de alto nível (em C ou Ada, por exemplo) e encapsulados no padrão *Packet Utilization Standard* (PUS - ECSS, 2016) ainda no segmento solo (WANDER; FÖRSTNER, 2013; TIPALDI; GLIELMO, 2018).

Depois de encaminhado ao segmento espacial, são processados e executados por uma máquina virtual em bordo. Conforme prevê o ECSS (2010), são ativados diretamente por telecomandos ou por eventos específicos ocorridos em bordo.

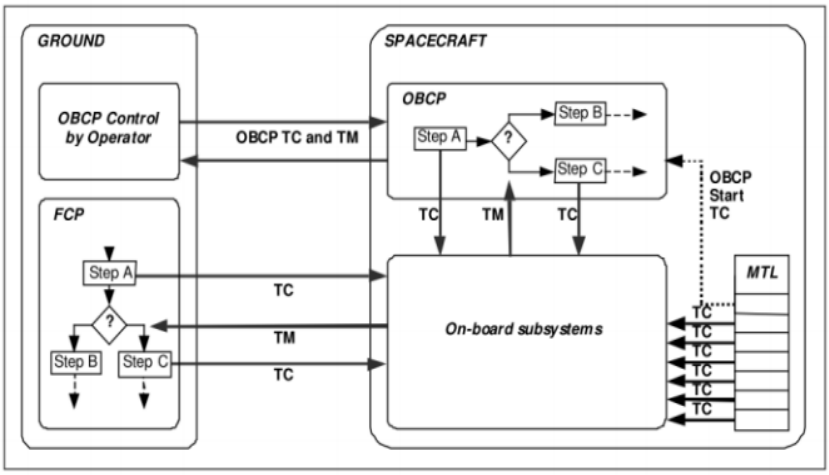
Tais *scripts* são utilizados, entre outras situações, para a execução de 'sequências de lançamento' (que comportam a abertura de painéis solares, aquisição inicial do Sol, o

ato de ligar pela primeira vez os equipamentos da plataforma, etc.), comutar um equipamento de voo de nominal para redundante, alterar modos de operação da carga útil e tomar ações de *Fault, Detection, Isolation and Recovery* (FDIR).

Os OBCPs também são adequados para missões que requerem maior automação operacional a bordo de veículos espaciais. Isso traz flexibilidade para os operadores prepararem o envio de sequências de operações mais complexas. Porém, D'Angelo et al. (2017) afirmam que essa solução não é adequada para satélites que operam em contextos dinâmicos e não previsíveis. Segundo Wander e Förstner (2013), OBCPs são processos automatizados, mas não autônomos.

A Figura 2.3 mostra a sistemática da execução dos OBCPs em ambiente seguro e de forma segregada, e como são enviados e executados os FCPs e MTLs em bordo.

Figura 2.3 – Dinâmica de operação de veículos espaciais da ESA.



Fonte: Tipaldi e Bruenjes (2015).

Apenas duas[7] missões espaciais da NASA, colocaram à prova o maior grau de autonomia do segmento espacial (E4) e ainda assim, em caráter científico e de demonstração tecnológica. Tal nível apresenta maior complexidade do que os níveis E2

[7] Recentemente, uma terceira missão (CubeSat) também fez uso do maior grau de autonomia, um novo experimento, mas utilizando o mesmo planejador embarcado usado na missão anterior da NASA. Isso será abordado com mais detalhes no Capítulo 4.

e E3, pois estes tomam decisões a partir de um número bastante restrito de condições, onde a sequência de ações é implementada antecipadamente em solo (TIPALDI; GLIELMO, 2018) de maneira similar aos telecomandos agendados.

Na autonomia real, de acordo com Thummala et al. (2003, p. 25), o satélite reage a um evento não somente baseado em regras, mas utilizando outras capacidades presentes em bordo, como agentes de planejamento embarcado e motores de inferência. Sendo mais específico, utilizando-se técnicas de Planejamento Automatizado em IA a bordo de satélites. Para tanto, deve-se transferir parte do planejamento de operações hoje realizado exclusivamente em solo para o segmento espacial.

2.3. Transferência do planejamento de operação ao segmento espacial

A transferência do planejamento de operação para o segmento espacial implica que o satélite receba não mais ‘comandos a executar’, mas sim ‘objetivos'. Isso constitui um novo paradigma denominado “operação de missão espacial baseada em objetivos” conforme a classificação da ECSS.

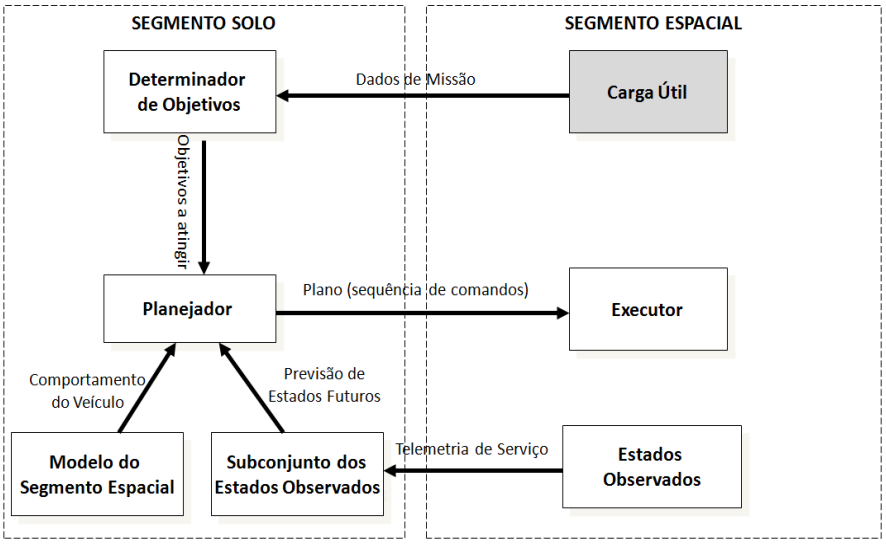
Para o melhor entendimento desta forma de operação, descrevem-se a seguir os paradigmas existentes. As duas seções subsequentes foram baseadas no trabalho de Kucinskis (2012).

Ao final, propõe-se um novo nível conceitual de autonomia tomando como base a classificação original da ECSS, conforme Cividanes et al. (2019).

2.3.1. A operação baseada em sequência de comandos

O paradigma atual de operação espacial é baseado na geração de sequências de comandos previamente definidos pelos operadores da missão. A Figura 2.4 traz uma representação conceitual deste paradigma. Sua finalidade não é mapear fielmente os elementos contidos na arquitetura de operação de uma missão espacial, mas ser útil para um melhor entendimento. Nela, um 'Determinador de Objetivos', representado pelos usuários da missão, define os objetivos a serem atingidos pelo segmento espacial.

Figura 2.4 – Modelo conceitual da operação baseada em sequência de comandos.



Fonte: Adaptado de Kucinskis (2012).

Cabe ao planejador transformar os objetivos em sequências de comandos a serem transmitidas ao satélite. O planejador consulta o histórico de estados passados do veículo espacial, que lhe foram transmitidos via telemetria, e realiza, a partir de um modelo comportamental do satélite, uma inferência para determinar os estados futuros - aqueles que se espera encontrar em algum momento futuro de operação. Esse ciclo de operação se repete durante a vida útil da missão a partir dos contatos subsequentes do satélite com a estação terrena.

Os níveis de E1 a E3 adotam a operação baseada em comandos, que demandam o planejamento em solo, realizado com grande antecedência, e sem considerar possíveis mudanças no estado do satélite. Se novas oportunidades forem identificadas em órbita, não é possível adaptar o plano devido às restrições temporais impostas pela comunicação solo-bordo.

É preciso que ocorram ao menos dois contatos subsequentes com a estação: um primeiro para que a equipe de operação seja informada do evento, e outro para que esta envie ao satélite o plano para reagir ao evento. Isso caracteriza operação em ciclo aberto, com alta dependência do centro de controle e baixa autonomia do software de voo para tomada de decisões a bordo.

A seguir, destacam-se as principais limitações desse paradigma de operação:

- O plano é estabelecido para um horizonte de tempo futuro com base no modelo comportamental do satélite. Esse horizonte pode chegar a dias de antecedência[8], dificultando prever com exatidão o estado inferido. Quanto maior a distância temporal entre o momento do planejamento e a execução do plano, maior a incerteza dos estados, o que pode prejudicar a qualidade dos planos gerados;
- Como o planejamento é feito antecipadamente e sem precisão dos estados futuros, a equipe de operação adota elevadas margens de segurança na alocação de recursos, o que acarreta na subutilização da plataforma espacial;
- Devido às restrições da banda de comunicação e da disponibilidade de memória em bordo, as telemetrias utilizadas para o planejamento futuro são defasadas temporalmente cerca de algumas horas e contêm dados amostrados, o que faz que a equipe de operação trabalhar com um subconjunto de estados do satélite;
- Sem acesso aos dados de telemetria em tempo real, qualquer falha na execução da sequência de comandos faz com que todo o plano de operação seja perdido, sem possibilidade de replanejar as atividades. Isso ainda pode levar o satélite a um modo seguro[9], no qual a maioria dos equipamentos ficam desligados, gerando interrupção dos serviços da plataforma espacial.

Apresenta-se a seguir o paradigma de operações baseada no envio de objetivos, ao invés de comandos. Ele representa a base sobre a qual se implementa a operação autônoma de um veículo espacial.

---

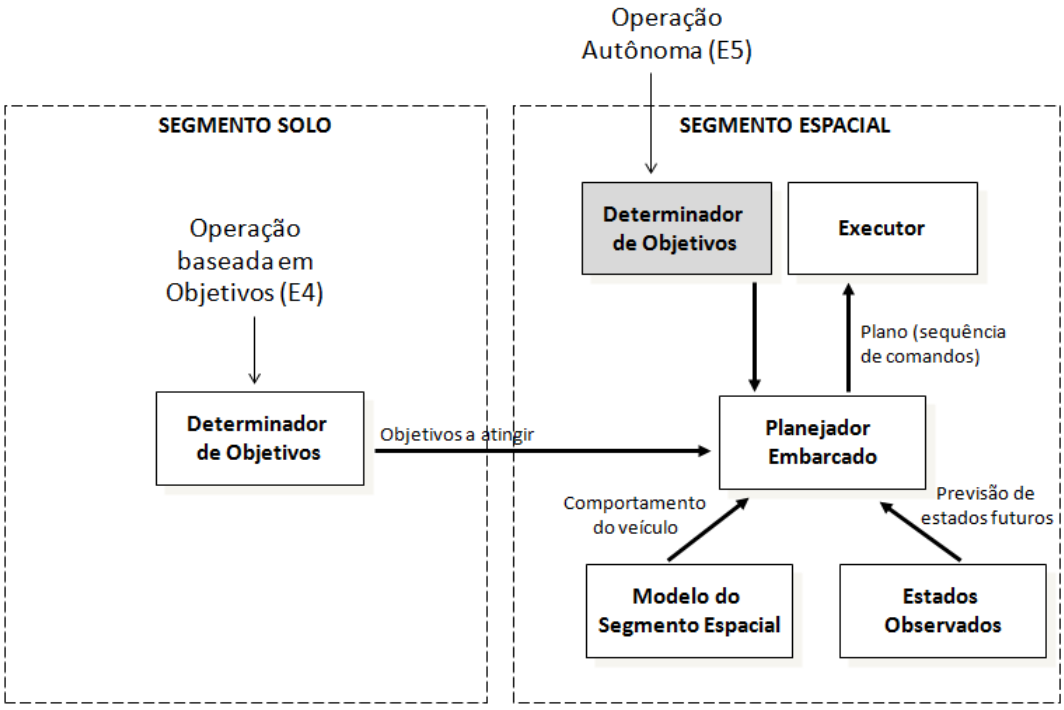
[8] No caso do satélite CBERS-4 do INPE, o planejamento de operações pode chegar a mais de uma semana de antecedência.

[9] Tipicamente, em modo segurança são mantidos ligados o subsistema de comunicação, suprimento de energia e funções de controle térmico, enquanto demais subsistemas não essenciais ficam desligados. Além disso, o satélite pode ser apontado para uma atitude específica em relação ao Sol e à Terra.

2.3.2.A operação baseada em objetivos

O paradigma de operação baseada em objetivos, vide Dvorak et al. (2008), está conceitualmente ilustrado pela Figura 2.5. Como dito antes, a ideia central é a transferência do processo de planejamento, ou ao menos parte dele, para o segmento espacial. O satélite passa a receber objetivos em alto nível a atingir (o que fazer), ao invés de comandos a executar - como fazer. Fica a cargo dele a determinação de quais comandos levarão ao cumprimento dos objetivos.

Figura 2.5 – Modelo conceitual da operação baseada em objetivos (E4) e operação autônoma do segmento espacial (E5).



Fonte: Adaptado de Kucinskis (2012).

O planejador em conjunto com o software de voo tem acesso completo e, em tempo real, aos estados do satélite, o que potencialmente melhora a qualidade do processo de inferência de estados futuros. Além disso, os estados podem ser averiguados regularmente, disparando-se correções caso sejam detectados desvios entre o que foi previsto e o que foi aferido.

Reduzem-se assim as limitações impostas pelo planejamento antecipado em solo e também pela falta de flexibilidade dos planos – uma vez que um plano pode ser

adaptado a bordo, na medida do necessário. Tal forma de operação tem potencial de maximizar o uso dos recursos em bordo.

É possível, entretanto, evoluir além da operação baseada em objetivos. Como pode ser observado em cinza na Figura 2.5, ao adicionar um determinador de objetivos ao segmento espacial, abre-se caminho para a operação autônoma do veículo espacial. Isso traz novas possibilidades à missão.

Assumindo, por exemplo, que um algoritmo embarcado identifique a presença de queimadas nas imagens de um satélite de sensoriamento remoto, o segmento espacial pode requisitar outras observações sobre o alvo e encaminhá-las ao planejador embarcado na forma de novos objetivos.

Isso aumenta substancialmente a capacidade de resposta do satélite e traz um nível de autonomia superior à classificação prevista pela ECSS. A Tabela 2.4 apresenta uma proposta de um novo nível de autonomia (E5), em que o segmento espacial é capaz de definir seus próprios objetivos. Há sua representação também na Figura 2.5.

Tabela 2.4 – Proposta de um novo nível de autonomia operacional.

Nível	Descrição	Funções
E5	Operação autônoma do segmento espacial com autodeterminação dos objetivos em bordo	Possibilita que o planejador modifique autonomamente o plano, retirando ou adicionando objetivos a partir de eventos ou oportunidades de interesse da missão percebidos em órbita pelo determinador embarcado de objetivos

Fonte: Cividanes et al. (2019).

Para exemplos de determinador embarcado de objetivos, veja os trabalhos (FRANCIS et al., 2015; CHIEN et al., 2009; CASTANO et al., 2008; CASTANO et al., 2007). Eles são em geral algoritmos embarcados que visam o reconhecimento de padrões, como os usados para a classificação e identificação de eventos de interesse em imagens de satélites.

3 FUNDAMENTAÇÃO TEÓRICA: PLANEJAMENTO AUTOMATIZADO EM INTELIGÊNCIA ARTIFICIAL

Este Capítulo traz os tópicos fundamentais de Planejamento Automatizado em Inteligência Artificial relacionados à presente Tese, enfatizando-se os conceitos da abordagem de planejamento clássico e suas principais linguagens.

3.1. Planejamento automatizado

É comum a necessidade de se planejar ações para alcançar os objetivos diários fazer parte do cotidiano das pessoas. O planejamento é um processo humano de tomada de decisão que busca alcançar um determinado resultado a partir de um conjunto sequencial de ações previsíveis. Sua automatização tem sido um dos objetivos da pesquisa realizada na área de IA (RUSSELL; NORVIG, 2004).

O Planejamento Automatizado é um ramo de IA que pode ser aplicado em diversas áreas que exigem comportamento autônomo e deliberativo. Dentre elas, estão os problemas da área de logística, sistemas de diagnóstico, veículos subaquáticos, robôs e satélites (CARDOSO, 2018). O planejamento em IA é um processo computacional que delibera sobre a escolha, organização e ordenação de ações para atingir objetivos (GHALLAB et al., 2004). Este processo simula o comportamento de um ambiente quando aplicada uma sequência de ações.

As características do ambiente controlado são mapeadas por um conjunto de estados, que representam configurações específicas do ambiente. Cada ação aplicada ao ambiente pode modificar um ou mais de seus estados. O objetivo final é encontrar uma sequência correta de ações, de modo a formar um caminho entre o estado atual e o estado que se pretende alcançar, chamado de “estado objetivo”.

Como a busca é realizada antecipando o resultado de diferentes ações, este processo de avaliação pode levar algum tempo até que uma possível solução seja encontrada. Caso uma sequência parcial de ações for inconsistente, o planejador precisa retornar à última escolha e selecionar outra ação aplicável presente no domínio.

Para que o planejador possa tomar decisões sobre qual ação escolher e saber raciocinar durante a busca, deve existir um modelo que descreva o ambiente a partir de representações de seus estados e uma coleção de ações aplicáveis. Este modelo não deve especificar as ações necessárias para atingir os objetivos, mas sim definir o efeito esperado de cada ação ou evento que pode ocorrer no sistema modelado (TIPALDI; GLIELMO, 2018).

As ações são usualmente descritas pelas suas precondições e efeitos. Nas precondições indicam-se quais estados devem ser verdadeiros para que a ação possa ser executada, enquanto nos efeitos expressa-se como o estado atual do ambiente será alterado se a ação for aplicada. A estrutura que representa a sequência de ações é denominada “plano” e o algoritmo que a produz é denominado “planejador”. Um plano solução é aquele que, ao ser executado, garante a realização do objetivo pretendido.

Cada domínio possui especificidades que devem ser consideradas na escolha do algoritmo de planejamento e modelos de representação do conhecimento. Alguns métodos de planejamento são mais apropriados para lidar com restrições temporais e consumo de recursos, existem aqueles mais propícios para representação de falhas e ocorrência de eventos, enquanto outros são voltados para encontrar planos ótimos.

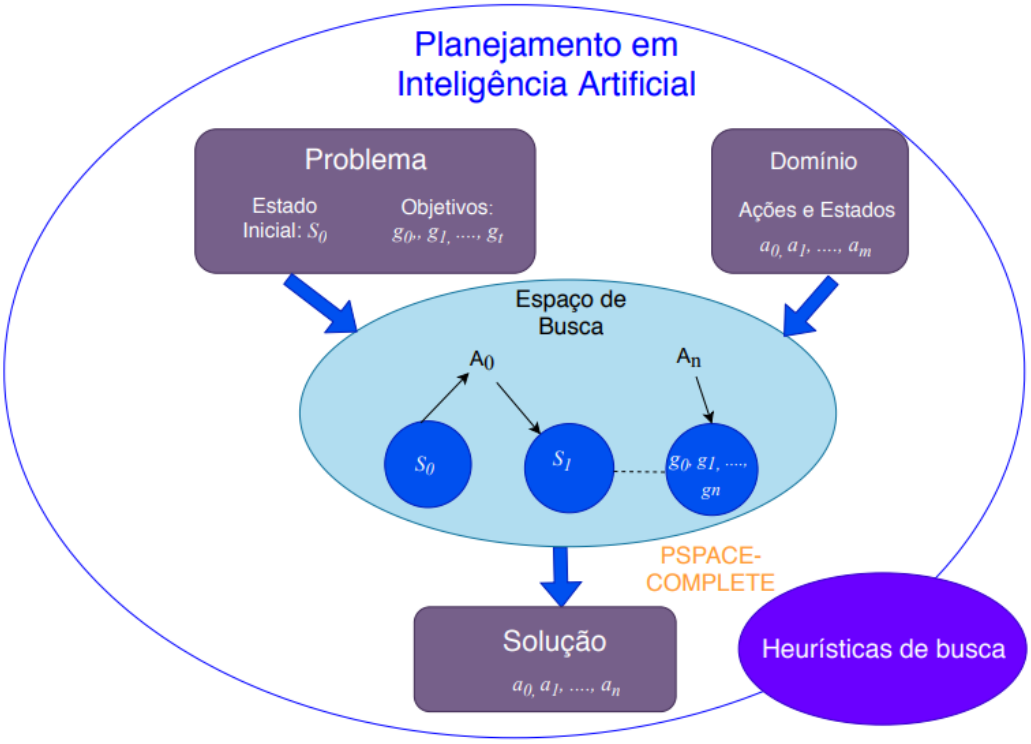
Em domínios que exigem uma resposta mais rápida, alcançar uma solução ótima pode não ser a melhor abordagem. Já em problemas grandes que fazem uso intenso de memória, implementar algum mecanismo de controle no feixe de busca pode ser interessante.

Este contexto evidencia que não existe a melhor estratégia para resolver qualquer tipo de problema. Para lidar com as várias formas de planejamento e as necessidades de diferentes aplicações, técnicas vêm sendo propostas pela comunidade científica ao longo dos anos. No próximo Capítulo serão abordadas as estratégias com histórico ou potencial de uso em planejadores da área espacial.

3.2. O planejamento clássico em inteligência artificial

Nessa seção introduz-se o Planejamento clássico em IA e seus conceitos principais. Ele representa um modelo teórico para construção de sistemas de planejamento, mas também serve como base para futuras extensões de problemas de planejamento. Os problemas clássicos incluem apenas ações determinísticas e assumem informações completas sobre os estados do ambiente. Mesmo com esse modelo simplificado, a busca por um plano está pelo menos na classe de complexidade *PSpace-complete*[10] (BYLANDER, 1994). A Figura 3.1 traz uma representação dos principais elementos existentes em um problema de Planejamento, que serão melhores explorados nas seções subsequentes.

Figura 3.1 – Elementos presentes em um problema de Planejamento em IA.



Fonte: Produção do autor.

[10] Um problema de decisão é PSPACE-completo se pode ser resolvido usando uma quantidade de memória que é polinomial no comprimento de entrada (espaço polinomial) e se todos os outros problemas que podem ser resolvidos no espaço polinomial podem ser transformados nele em tempo polinomial.

3.2.1. Definição formal do problema

Um modelo conceitual de planejamento clássico pode ser formalmente definido pela tripla $P = < \Sigma, S_0, G >$, onde (GHALLAB et al., 2004): Σ denota um sistema de transição de estados, S_0 é um estado inicial e $G = \{ g_0, g_1, ..., g_t \}$ corresponde a um conjunto de estados objetivo do problema.

O sistema de transição de estados é definido pela seguinte 4-tupla: $\Sigma = <S, A, \varepsilon, \gamma>$, onde: $S = \{s_0, s_1, ..., s_n\}$ é um conjunto finito de estados; $A = \{a_0, a_1, ..., a_m\}$ corresponde a um conjunto finito de ações e $\varepsilon = \{e_0, e_1, ..., e_n\}$ representa um conjunto de eventos externos ao sistema. Note-se que S_0 e G podem variar conforme o problema, já o sistema de transição de estados permanece o mesmo dentro de um domínio de planejamento.

Uma ação é um acontecimento cuja ocorrência causa uma transição do instante *t* para um instante *t+1*, fazendo com que o ambiente evolua do estado corrente s_t para um estado subsequente s_{t+1} (PEREIRA, 2007). Ela é definida pela tupla: {*pre(a), eff(a)*}, onde *pre(a)* é um conjunto de precondições da ação *a,* e *eff (a)* representa um conjunto de efeitos. Uma ação é dita aplicável a um estado *s* se satisfaz sua precondição.

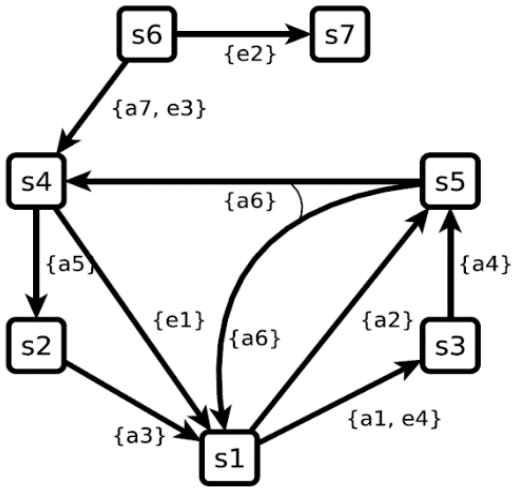
A diferença entre ações e eventos, é que estes últimos apesar de contribuírem para evolução do sistema, não estão sob o controle do agente[11] e não são automaticamente incluídos no problema. Ou seja, eventos são externos ao sistema, embora afetem seu estado. Em um problema de planejamento clássico, a formulação do domínio se reduz à tripla: $\Sigma = <S, A, \gamma>$ devido à suposição do determinismo do ambiente.

A função de transição de estados do modelo estendido é denotada por $\gamma : S \times (A \cup \varepsilon) \rightarrow 2^S$. Este sistema de transição pode ser representado por um grafo dirigido cujos nós são os estados em *S.* Se $s' \in \gamma(s,u)$, onde *u é* o par (a,e), com $a \in A$, e $e \in \varepsilon$, então o

[11] O termo agente na área de planejamento em IA é por vezes utilizado como forma de referenciar o sistema de planejamento. Por exemplo, no domínio espacial, o agente é o sistema de planejamento que controla o satélite ou a plataforma espacial.

grafo contém um arco de *s* a *s'* que é rotulado com *u.* Cada um desses arcos é chamado de transição de estado (CAMPOS, 2008). Tal representação por grafos pode ser observada pelo exemplo da Figura 3.2, onde os diferentes estados são denotados por *s,* as ações por *a* e os eventos por *e*.

Figura 3.2 – Exemplo de um grafo de transição de estados.



Fonte: Campos (2008).

Um domínio descrito por um grafo orientado e rotulado também pode ser denominado grafo de transições de estados. Com um grafo de transições de estados, é possível formular uma tabela (considere o exemplo da Tabela 3.1) que especifica a função de transição γ responsável por computar o próximo estado.

Observe pela Tabela 3.1 que *Nop* denota uma ação neutra sem operação, na qual a transição é ocasionada exclusivamente por um evento exógeno: $\gamma(s, n_{op}, e) = \gamma(s, e)$. De forma análoga, ε denota um evento neutro quando a transição é provocada exclusivamente por uma ação: $\gamma(s, a, \varepsilon) = \gamma(s, a)$.

Dado um problema de planejamento clássico, um planejador consiste essencialmente num algoritmo que realiza uma busca no grafo de transições de estados cujo objetivo é procurar um caminho que leve o planejador do estado inicial S_0 até um estado objetivo S_n. Quando se encontra um caminho, o planejador produz a sequência de ações que delimita esse caminho, do contrário, a resposta é a indicação do fracasso.

Tabela 3.1 – Especificação da função de transição de estados.

S	A	ε	2^S
s_1	a_2	ε	s_5
s_1	a_1	e_4	s_3
s_2	a_3	ε	s_1
s_3	a_4	ε	s_5
s_4	a_5	ε	s_2
s_4	n_{op}	e_1	s_1
s_5	a_6	ε	s_1, s_4
s_6	n_{op}	e_2	s_7
s_6	a_7	e_3	s_4

Fonte: Produção do autor.

Uma solução formal para o problema de planejamento clássico *P* é composta por uma sequência linear de ações que satisfaça o objetivo (G), tal que:

$$\varepsilon = \{\emptyset\};$$

$$s_i \in S;$$

$$S_1 = \gamma(s_0, a_i),\ S_2 = \gamma(s_1, a_j),\ \ldots,\ S_n = \gamma\,(s_{n-1}, a_k);$$

$$\pi = \{a_i, a_j, \ldots, a_k\},\ e\ S_n \in G;$$

Considerando a existência de vários objetivos, o plano final é composto não apenas por uma sequência, mas por um conjunto de sequência de ações encadeadas. Assumindo $G = \{ g_0, g_1 \}$, por exemplo, e considerando os planos resultante $\pi_1 = \{a_1, \ldots, a_k\}$ e $\pi_2 = (a_1, \ldots, a_j)$, sua concatenação representa uma solução para esse problema: $\pi_1 * \pi_2 = (a_1, \ldots, a_k, a'_1, \ldots, a'_j)$.

As propriedades de uma solução de planejamento serão abordadas na Seção 3.4.

3.2.2.Suposições restritivas do modelo teórico

Existe um modelo teórico, não operacional, utilizado como referência para o problema de planejamento clássico (NAU, 2007) que assume suposições restritivas sobre o tipo do plano, metas desejadas, interação com o sistema controlado, componente temporal e principalmente quanto à descrição do ambiente em que se deseja planejar.

De acordo com Nau (2007), esse modelo simplificado apresenta oito propriedades restritivas, a saber: domínio finito, completamente observável, determinístico, estático, objetivos restritos, planos sequenciais, tempo implícito e planejamento *offline*. As definições de cada suposição estão enumeradas (A_0-A_7) na Tabela 3.2. Ao longo desta Tese, algumas dessas enumerações são mencionadas para se referir a suposições restritivas do modelo clássico.

Tabela 3.2 – Suposições restritivas impostas pelo planejamento clássico.

Propriedade	Suposição Restritiva	Descrição da Restrição
A_0	Σ finito	O espaço de estados é formado por um conjunto finito de estados, $S = \{s_0, s_1, ..., s_n\}$ para algum n conhecido
A_1	Σ totalmente observável	É sempre possível ao planejador conhecer o estado atual do sistema. A ação de 'sentir' sempre retorna a informação adquirida.
A_2	Σ determinístico	Uma ação leva a um único estado possível e não há incerteza sobre os efeitos das ações no ambiente
A_3	Σ estático	Não sofre influência de eventos exógenos e o estado corrente do ambiente muda somente como consequência de ações executadas pelo agente
A_4	Objetivos restritos	Objetivos (G) são restritos e especificados antecipadamente. Metas estendidas e funções de utilidade não são consideradas
A_5	Planos sequenciais	Os planos (π) representam uma sequência de ações linearmente ordenadas
A_6	Planos atemporais	A duração de cada atividade ou ação é irrelevante, não sendo considerada no problema
A_7	Planejamento *offline*	O processo de planejamento é feito de forma *offline*, não há atualização sobre o estado de execução do plano ao planejador. O planejador não está diretamente conectado ao ambiente, trabalhando em malha aberta

Fonte: Produção do autor.

O relaxamento de qualquer uma das propriedades transforma o problema em planejamento do tipo não clássico, aproximando-se mais do modelo encontrado por ambientes que lidam com problemas práticos de engenharia.

Em sistemas espaciais, por exemplo, as ações nem sempre são deterministas (*e.g.,* a comunicação com um equipamento pode falhar); algumas variáveis de estados podem assumir infinitos valores; o satélite sofre influência de fatores externos (*e.g.*, períodos de eclipse descarrega a bateria); o tempo determina não só quando um subsistema deve ser ligado, como também o consumo de recursos como propelente, energia ou espaço para armazenamento de dados; os comandos derivados do plano solução são executados pelo computador em tempo real, o que caracteriza planejamento *online*.

Este contexto evidencia que o modelo clássico é muito restritivo para lidar com a maioria dos problemas de interesse prático (NAU, 2007). Em contrapartida, trabalhar com suposições relaxadas traz mais complexidade e dependendo das condições impostas pode tornar o problema intratável.

Usualmente os trabalhos científicos realizam suas pesquisas sob esse modelo restritivo. O objetivo é simplificar a representação do problema e promover o *benchmark* entre os algoritmos propostos. Embora isso não impulsione diretamente o desenvolvimento de planejadores aplicados, soluções gerais do planejamento clássico podem ser estendidas à resolução de problemas reais. A discussão sobre como obter uma conexão mais forte com a prática de engenharia é tema recorrente na comunidade de planejamento.

3.2.3. Domínio clássico

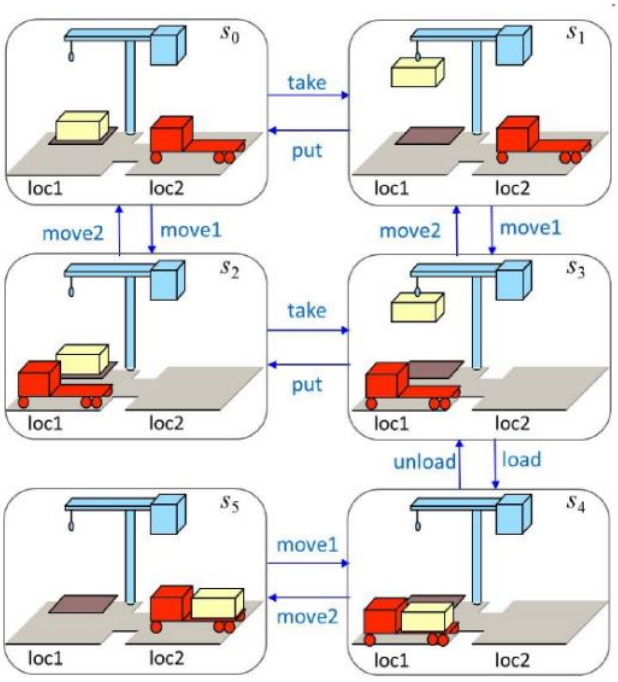
Um problema que segue o modelo restrito é o domínio dos Robôs Portuários (*Dock Worker Robots* - DWR). Trata-se de um problema conhecido pela comunidade de planejamento e pode ser visto com uma extensão do tradicional problema clássico conhecido como “mundo dos blocos”.

Os objetos do domínio DWR incluem: mercadorias (*containers*) dispostas em uma pilha, um robô-caminhão que pode transportar os *containers,* movendo-os de um local para outro, e um guindaste para carregar ou descarregar as mercadorias (JIANG; XU, 2017).

O estado do problema se altera com as diferentes ações executadas, conforme o exemplo da Figura 3.3. Quando o problema é iniciado, um planejador tenta escolher tarefas que construam uma sequência de ações, modificando o atual estado do problema até alcançar o estado objetivo. Se a sequência parcial de ação for inconsistente, o planejador deve retornar à última configuração para selecionar outra ação aplicável.

Retornando à formalização apresentada na Seção 3.2.1, temos a seguinte formulação para o exemplo da Figura 3.3: situação inicial*:* s_0 = *container* está no local 1 e o caminhão está no local 2; estado objetivo: S_5 = o *container* deve estar no local 2. Considerando o domínio: $\Sigma = <S, A, \varepsilon, \gamma>$, ele pode ser instanciado como: $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$; $A = \{$ *take, put, load, undload, move* $\}$; $\varepsilon = \{\emptyset\}$; γ = conforme destacado pelas setas da Figura 3.3.

Figura 3.3 – Exemplo do domínio dos robôs portuários para transporte de mercadorias.



Fonte: Jiang e Xu (2017).

Mesmo com as restrições do modelo clássico nem sempre a resolução do problema é trivial. Considere a configuração deste exemplo, contendo: cinco localizações, três pilhas e cem *containers*; isso implica em um espaço de busca de 10^{277} estados

(GHALLAB et al., 2004). Uma heurística que guie o processo de busca torna-se fundamental. Heurísticas são funções que ajudam a orientar e acelerar a busca por uma solução para um determinado problema.

Segundo Jiang e Xu (2017), uma heurística para seleção de ação é um método que classifica um conjunto de ações na ordem de sua relativa preferência. Usualmente, uma heurística é modelada por uma função *h*, a qual pode ser usada para computar uma avaliação $h(a_i)$ para cada ação candidata no conjunto de ações, onde se define a convenção que cada ação preferida $a_i \in A$ é uma ação em que a sua função $h(a_i)$ corresponde ao menor valor, como por exemplo, $Select\ (A) = argmin\ \{ h(a_i) \mid a_i \in A \}$.

3.2.4. Linguagens de descrição

Planejadores automatizados requerem uma descrição precisa das tarefas de planejamento[12] (JIMÉNEZ et al., 2012). Uma linguagem de planejamento é uma ferramenta utilizada para descrever um modelo das ações que podem ser realizadas no ambiente controlado. Ela traz a especificação do estado do ambiente e dos objetivos a serem alcançados.

A linguagem *STanford Research Institute Problem Solver* (STRIPS – FIKES; NILSSON, 1971) foi a primeira iniciativa de uma sintaxe voltada ao planejamento clássico. Nela, estados são modelados como conjuntos de proposições que são verdadeiras em tais estados, e as ações que podem alterar a validade de certas proposições. Uma ação STRIPS é representada por três listas: precondição, adição e exclusão [13].

Essa linguagem especifica o mundo como condições lógicas e representa um estado do ambiente como uma conjunção de literais (átomos) totalmente instanciados[14]. Em lógica de predicados de primeira ordem, um literal é uma sentença atômica ou uma

[12] Os autores se referiam ao conjunto de tarefas disponíveis ao planejador para compor um plano.
[13] 'Adição' e 'Exclusão' referem-se a proposições que se tornam respectivamente verdadeiras ou falsas após a execução da ação.
[14] Conceitualmente, as ações podem ser totalmente instanciadas (*ground*) ou não. Ações totalmente instanciadas são aquelas que têm valores especificados. Isto é, com termos compostos exclusivos por constantes, sem parâmetros (na descrição das precondições e efeitos).

sentença atômica negada. A STRIPS se apoia na hipótese do mundo fechado (*Closed-World Assumption*).

Nesta linguagem, os literais não conhecidos ou não descritos explicitamente no domínio são considerados falsos. Apesar de suas limitações, a teoria de 'ações' do STRIPS ainda é utilizada em muitos planejadores recentes e é a base da maioria dos planejadores clássicos (KUCINSKIS, 2007; CANTONI, 2010). A Figura 3.4 exemplifica uma ação em STRIPS cujo objetivo é voar em um avião de um lugar para o outro. Note-se a presença de literais nas precondições e efeitos na referida figura.

Figura 3.4 – Exemplo de uma ação na linguagem STRIPS.

Ação (Voar(p, de, para) ,

PRECOND: Em (p, de) ∧ Avião (p) ∧ Aeroporto(de) ∧ Aeroporto(para)

EFEITO: ¬Em(p, de) ∧ Em (p, para)

)

Fonte: Adaptado de Russell e Norvig (2004).

A *Planning Domain Definition Language* (PDDL – FOX; LONG, 2003), inspirada no STRIPS, representa um padrão no planejamento clássico. O objetivo é que ela se posicione como uma linguagem comum entre o problema a ser resolvido e o método de resolução. Atualmente a PDDL representa a linguagem oficial da competição internacional (*International Planning Competition* – IPC) de planejamento realizada bienalmente.

A ideia dos idealizadores foi prover uma padronização quanto à formalização dos problemas, que ainda não existia até a sua criação em 1998. Nas edições seguintes do IPC, a linguagem continuou a ser utilizada e novas versões foram criadas para se adequar aos constantes desafios da área. A cada versão[15] subsequente, outras características foram sendo incorporadas (CANTONI, 2010), incluindo: tipos, funções,

[15] A última versão disponível é a PDDL 3.1.

representação numérica, ações durativas, axiomas, objetos, literais temporais, funções de otimização, preferências, objetivos estendidos, etc. Para outros detalhes sobre versões da PDDL, consulte Cantoni (2010).

Assim como as linguagens precursoras, a PDDL se baseia em ações. A Figura 3.5 traz um exemplo[16] de uma ação que modela o ato de ligar um instrumento de um satélite. Nela, estão envolvidos os seguintes predicados: *on_board*, *power_avail*, *power_on*, *calibrated*, os quais são responsáveis por modelar fatos lógicos. Os parâmetros são instâncias de objetos e nesta ação estão relacionados a dois tipos de objetos (*instrument* e *satellite*) definidos na descrição do domínio.

Figura 3.5 – Exemplo de uma ação em PDDL 2.1.

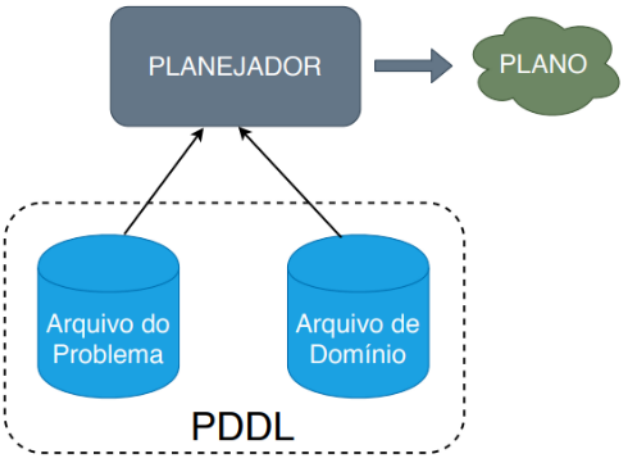
```
(:action switch_on
 :parameters (?i - instrument ?s - satellite)
 :precondition (and (on_board ?i ?s)
                    (power_avail ?s)
                )
 :effect (and (power_on ?i)
              (not (calibrated ?i))
              (not (power_avail ?s))
         )
)
```

Fonte: Long e Fox (2003).

Em linhas gerais, o domínio da PDDL é dividido em duas partes (LONG; FOX, 2003): descrição do domínio (tipos, predicados e ações comportamentais) e definição do problema (objetos, estados iniciais e condições do objetivo), conforme mostra a Figura 3.6. Embora a descrição dos dois arquivos não seja mandatória para a linguagem, é necessária para a maioria dos planejadores. Praticamente todos os planejadores modernos clássicos são baseados em alguma versão da PDDL.

[16] O objetivo é mostrar uma visão geral de uma ação em PDDL e não todos os conceitos envolvidos na linguagem.

Figura 3.6 – Entradas para um problema de planejamento em PDDL.



Fonte: Produção do autor.

Outra linguagem relevante do planejamento clássico é a *Action Description Language* (ADL - PEDNAULT, 1989) que na linha do tempo está entre a STRIPS e a PDDL. Demais linguagens voltadas a planejadores aplicados são abordadas no Capítulo 4.

3.3. Conceitos do planejamento clássico

Esta seção traz as duas principais abordagens de espaço de busca do planejamento clássico.

3.3.1. Espaço de busca

O espaço de pesquisa representa o conjunto de todos os estados alcançáveis a partir do estado inicial por qualquer sequência de ações. Dependendo das propriedades do domínio, a topologia do espaço de busca pode ser denotada por uma árvore, caso exista apenas uma maneira de evoluir de um estado para outro, do contrário, por um grafo. Quase todos os métodos de planejamento são essencialmente procedimentos de busca. Há dois tipos de espaço de busca percorrido por planejadores clássicos (MINTON et al., 1994; GHALLAB et al., 2004; NAU, 2007; COLES et al., 2010): planejamento no espaço de estados (*state-space planning*) e planejamento no espaço de planos (*plan-space planning*).

O espaço de estados: o espaço de busca é uma árvore ou um grafo que representa possíveis estados do problema, onde cada nó denota um estado e os arcos correspondem a uma possível transição de estado. A partir do estado inicial, diferentes nós filhos podem ser expandidos durante a busca. Se uma ação é aplicável, ela se torna o estado atual, de qual a busca continua até encontrar uma possível solução. A grande vantagem dessa abordagem é o maior controle sobre o estado do ambiente, que proporciona melhor desempenho (COLES et al., 2010). Uma heurística nessa abordagem pode ser baseada na seleção de nós específicos ou no custo estimado do caminho mais econômico até o objetivo. Por estar relacionada com estados, tem menos flexibilidade para lidar com restrições temporais se comparada à abordagem de espaço de planos.

O espaço de planos: o espaço de busca pode ser representado como uma árvore ou um grafo, onde os nós são planos parciais e os arcos representam operações de refinamentos. A partir de um nó inicial, um plano aproximado e incompleto é progressivamente expandido, inserindo-se ações ou restrições até que um plano válido livre de falhas seja encontrado (NAU, 2007). Em geral, evita-se adicionar ao plano parcial qualquer restrição que não seja estritamente necessária (CAMPOS, 2008), seguindo o princípio do comprometimento mínimo (ver Seção 3.3.4). A principal desvantagem do espaço de planos é não ter uma representação explícita dos estados. Nela, uma heurística[17] não consiste apenas em escolher o próximo nó, mas também o próximo solucionador de conflitos.

Uma diferença de maior destaque entre as duas abordagens está no propósito das ações durante a busca. No espaço de estados, as ações atuam sobre o ambiente visando alterar seus estados, já no espaço de planos as operações atuam sobre os planos, sendo o foco na resolução de vínculos causais e restrições de ordenação. Estes últimos estão intimamente relacionados aos algoritmos de planejamento de ordem parcial, abordados a seguir.

[17] Veja o trabalho de Jiang et al. (2016), para um exemplo de heurística dessa abordagem.

3.3.2. Ordenação das ações

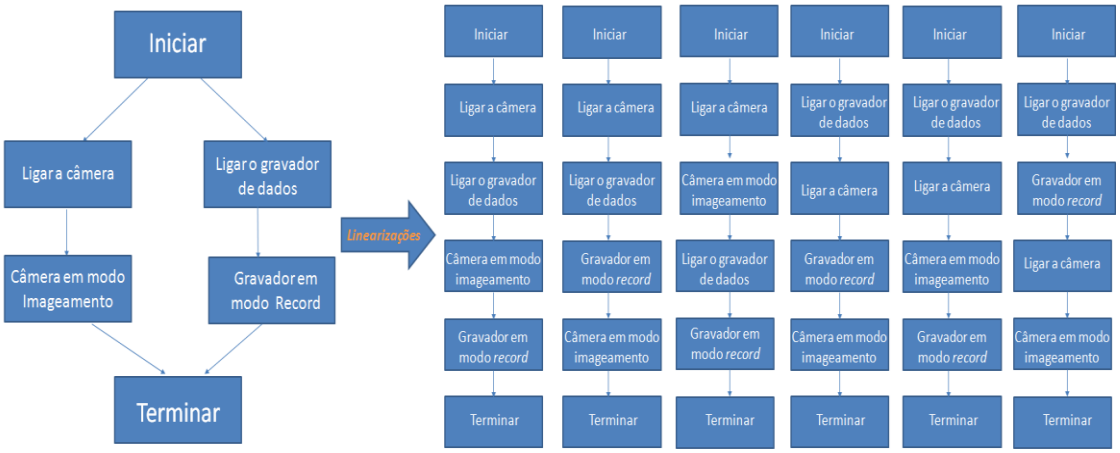
Existe um conceito relacionado à forma de ordenação das tarefas durante a busca pelo plano cuja classificação pode ser de ordem total ou parcial. No planejamento de ordem total, a cada passo do algoritmo o plano está sempre ordenado com relação a todos os passos restantes. Já no planejamento de ordem parcial somente alguns passos são de fato ordenados. Nele, o planejador pode inserir ações sem especificar qual delas deve executar primeiro, o que possibilita o paralelismo das ações - caso o ambiente permita. O plano nessa abordagem pode ser composto por um conjunto parcialmente ordenado de ações.

O processo que constitui um plano de ordem total derivado de um plano parcial é chamado de *Linearização* do plano. Toda linearização de uma solução de ordem parcial é uma solução de ordem total cuja execução a partir do estado inicial alcançará um estado objetivo. Cabe ao planejador escolher uma linearização ou delegar essa etapa a outro agente envolvido no problema. A Figura 3.7 traz um exemplo[18] de plano parcial à esquerda e seis possíveis linearizações correspondentes à direita.

É importante observar que isso não flexibiliza ao algoritmo determinar qualquer tipo de ordenação entre as ações. No exemplo da Figura 3.7, as linearizações respeitam ordens estabelecidas: uma câmera só é colocada no modo '*imaging*', uma vez que ela esteja ligada, assim como o gravador só é acionado para o modo '*record*' se estiver ligado. Planejadores da área espacial têm adotado a estratégia de ordem parcial.

[18] Cabe mencionar que embora em algumas situações a ordem de ligar a câmera ou gravador não seja determinante em uma missão de sensoriamento remoto, em um plano de voo realizado em solo é comum adotar uma padronização quanto à ordem de operação desses equipamentos.

Figura 3.7 – Exemplo de um plano de ordem parcial e suas linearizações.



Fonte: Produção do autor.

Uma conhecida abordagem de planejamento que explora o ordenamento parcial é a *Partial Order Causal Link* (POCL). Nela, a solução é composta essencialmente pela escolha de ações, vínculos causais e ordenação dessas ações. De acordo com (CAMPOS, 2008; LEMAI, 2004; e SAPENA et al., 2015), a estrutura do plano nessa abordagem tem os seguintes componentes:

- Um **conjunto de ações**. Estas ações representam os passos do plano e são obtidas a partir do conjunto de todas as ações possíveis para o domínio. Um plano vazio detém apenas as ações: Iniciar (A_0) e Terminar (A_∞). A_0 não tem precondições e apresenta como efeito a descrição do estado inicial do problema. A_∞ não tem efeitos e apresenta como precondição a descrição do objetivo do problema.

- Um conjunto **de restrições de ordenação**. Cada restrição de ordenação tem a forma $A \prec B$, isto significa que a ação A deve ser executada antes de B, mas sem a necessidade de ser imediatamente antes, podendo haver ações intermediárias.

- Um conjunto **de precondições abertas**. Uma ação *a* cuja precondição *p* ainda não foi atendida é chamada de precondição aberta. Quando adicionada uma nova ação no plano, todas as suas precondições devem ser colocadas como precondições abertas. Para resolver isso, é preciso encontrar uma ação *b* preexistente no plano (ou inserir

uma nova) de forma que ela preceda a ação *a* e produza o efeito *p*. Os planejadores em geral têm o compromisso de reduzir o conjunto de precondições abertas a cada passo.

- Um conjunto de **vínculos causais.** Um vínculo causal entre as ações *A* e *B* no plano é representado como $A \xrightarrow{p} B$. Isto afirma que *p* é um efeito da ação *A* e uma precondição de *B.* Para manter esse intervalo protegido, o planejador deve impor que *p* seja verdadeiro sobre o intervalo das ações *A* e *B*.

Um plano parcial não pode ser estendido a partir de uma nova ação C_{threat} que gere conflito. Um exemplo de conflito entre um vínculo causal e uma ação ocorre quando o efeito da ação C_{threat} for ¬*p* e se C_{threat} ocorrer depois de *A* mas antes de *B.* Se um determinado passo destrói a precondição realizada por um vínculo causal existente, o chamamos de passo ameaça.

Uma ameaça tem o potencial de destruir um plano inteiro. Ela pode ser resolvida de duas maneiras (SAPENA et al., 2015): *Promoting* ou *Demoting*. A primeira impõe uma restrição de ordem no formato $B \prec C_{threat}$ e a segunda impõe uma restrição de ordem no formato $C_{threat} \prec A$. No domínio espacial, técnicas de reparo iterativo têm sido adotadas para lidar com passos ameaças.

3.3.3. Algoritmo de ordem parcial

O conceito de planejamento em ordem parcial (do inglês: *Partial Order Planning* - POP) foi usado em vários planejadores, como: *Versatile Heuristic Partial Order Planner* (VHPOP – SIMMONS; YOUNES, 2003) e UCPOP (PENBERTHY; WELD, 1992). Trata-se de uma clássica implementação de planejamento no espaço de planos.

O POP surgiu na década de mil novecentos e noventa como forma de superar a limitação do ordenamento linear das ações imposta pela abordagem de ordem total. Os maiores benefícios do POP são a boa capacidade de adaptação a falhas (BECHON et al., 2014) e a facilidade de lidar com planejamento temporal (SAPENA et al., 2015).

O Algoritmo 3.1 contém o mecanismo geral de funcionamento de um algoritmo de Planejamento no Espaço de Planos (PSP – *Plan Space Planning*), que é uma generalização de algoritmos POP. Uma falha no Algoritmo 3.1 constitui um objetivo aberto ou uma ameaça. A cada passo do algoritmo na versão recursiva é verificado se um plano parcial atende a uma possível solução, do contrário aplica-se novamente uma estratégia de refinamento.

Este processo se repete de forma recursiva até encontrar uma solução ou falhar. Uma possível heurística consiste em solucionar as falhas usando o menor número de 'resolvedores' possíveis. Note-se que a função *ObjetivosAbertos (π)* busca todas as submetas do plano π não estabelecidas por um vínculo causal; *Ameacas(π)* retorna todas as ações que ameaçam algum vínculo causal*;* e a função *Resolver(f, π*) seleciona um método para resolver a falha *f.*

Algoritmo 3.1 - Planejamento no Espaço de Planos (PSP)

```
1 Função PSP(π)
2  Início
3       falhas ← ObjetivosAbertos(π) U Ameacas(π);
4       se falhas = Ø então retorne (π); // sem falhas, o plano solução foi encontrado
5       seleciona-se alguma falha f ∈ falhas ;
6       resolvedores ← Resolver(f, π);  // define-se um conjunto de resolvedores
7       se resolvedores = Ø retorne insucesso; // não há mais resolvedores para esta falha
8       não deterministicamente escolha um resolvedor ρ ∈ resolvedores;
9       π ← Refinar (ρ, π ); //adicionam-se operações de refinamento ao plano para resolver f
10      retorne PSP(π ); // a busca pelo plano é feita de forma recursiva
11 Fim
```

Fonte: Adaptado de Dvorak et al. (2014a).

O principal entrave de um algoritmo PSP é ser considerado lento frente a outros algoritmos (SIMMONS; YOUNES, 2003; BECHON et al., 2014). Para Sapena et al. (2015), gerar um plano custa mais caro do que gerar um estado devido à necessidade de verificação de conflitos. Apesar disso, muitos planejadores, inclusive os da área espacial, se baseiam nessa abordagem aliada à técnica do comprometimento mínimo (INGRAND; GHALLAB, 2017), descrita a seguir.

3.3.4. Estratégia de comprometimento

Há basicamente duas estratégias adotadas por planejadores: comprometimento antecipado (*early-commitment*) ou comprometimento mínimo (*last-commitment)*. O conceito de comprometimento no planejamento clássico é aplicado usualmente a decisões relacionadas a restrições de ordenação das ações e na instanciação de ações, mas fora do planejamento clássico também pode ser estendido a outros aspectos, como na alocação antecipada ou não de recursos usados pelas ações.

No comprometimento antecipado, uma decisão sobre ordenamento das tarefas nunca é postergada. A vantagem dessa abordagem é possibilitar o maior controle dos estados durante a busca. Já no comprometimento mínimo há um relaxamento dessa decisão, o que confere maior flexibilidade ao planejador (INGRAND; GHALLAB, 2017). Inicialmente são tomadas apenas decisões óbvias ou mais importantes. A ideia consiste em não fazer compromisso com ordenações até que seja estritamente necessário. O ponto forte é sua maior expressividade, por outro lado, seus algoritmos tendem a ser mais sofisticados.

3.3.5. Métodos e direções de busca

Os mecanismos de buscas em planejamento podem ser progressivos (*forward search*) ou regressivos (*backward search*). Os algoritmos da classe progressiva partem do estado inicial e percorrem o espaço de busca a partir da função de transição de estados até chegar, dentre diferentes alternativas, ao estado objetivo. As ações aplicáveis são adicionadas ao final do plano solução. Sua grande vantagem é ter conhecimento completo do estado a qualquer instante durante a busca. O Algoritmo 3.2 traz sua lógica de funcionamento. A busca em profundidade e a busca em largura são exemplos clássicos de implementações da estratégia progressiva.

Algoritmo 3.2 - Planejamento de Busca Progressiva

```
1 Função PlanejamentoProgressivo (s_o, S_g, Σ)
2 Entrada: Estado Inicial: s_o, Estado Objetivo: S_g, Domínio: Σ
3 Saída: Plano Soluçãoπ
4 Início
5     s ← s_o; // estado inicial
6     π ←Ø; // plano solução inicializa vazio
7     repita
8     início
9         se s ∈ S_g então retorne π; // encontrou um plano que atenda ao estado objetivo
10        A ← { a | a é uma ação totalmente instanciada de Σ e a é aplicável a s e a ∈ A(s) };
11        se A = Ø então retorne falha; // não há mais ações disponíveis
12        Não deterministicamente escolha a ∈ A;
13        π ← π.a; // adiciona a ação ao final do plano solução
14        s ← γ(s, a); // computa o próxima estado
15    fim // do laço
16 Fim
```

Fonte: Adaptado de Kuter e Nau (2004).

Já no procedimento regressivo, a direção é inversa, isto é, do estado meta ao estado inicial. A busca começa pelo objetivo e aplica-se a função inversa de transição de estados sucessivamente, gerando submetas. As ações relevantes são colocadas no início do plano solução (linha 13). A busca termina quando um desses planos satisfaz as condições do estado inicial ou quando esgotadas as ações apropriadas. O Algoritmo 3.3 traz sua lógica de funcionamento.

Algoritmo 3.3 - Planejamento de Busca Regressiva

```
1 Função PlanejamentoRegressivo (s_o, S_g, Σ)
2  Entrada: Estado Inicial: s_o, Estado Objetivo: S_g e Domínio: Σ
3  Saída: Plano Solução π
4  Início
5     π ← Ø;
6     s ← s_o;
7     repita
8     início
9        se s ∈ S_g então retorne π;
10       A ← {a | a é uma ação totalmente instanciada de Σ e a é relevante para S_g e a ∈ A(s)};
11       se A = Ø então retorne falha; // não há mais ações disponíveis
12       Não deterministicamente escolha a ∈ A;
13       π ← a.π;  //adiciona a ação no início do plano solução
14       s ← γ^-1(S_g, a); // aplica o próximo estado a partir da função inversa de transição de estados
15    fim // do laço
16 Fim
```

Fonte: Adaptado de Ghallab et al. (2004).

É possível perceber que o método regressivo não se aplica à busca no espaço de planos, pois o nó inicial seria o próprio plano que se está buscando. A busca regressiva e progressiva são métodos usualmente aplicados em planejamento de espaço de estados. A Tabela 3.3 faz uma síntese comparativa entre as duas abordagens de espaço de busca do planejamento clássico.

Tabela 3.3 – Comparação entre as abordagens de espaço de busca.

Características	Espaço de Estados	Espaço de Planos
Estratégia de busca	Planejamento Progressivo Planejamento Regressivo	Planejamento de Ordem Parcial
Nós	Estados Intermediários	Planos parciais
Transições	Ações	Operações de Refinamento
Ação	O efeito esperado da ação (mudança de estado) é sobre o ambiente	O efeito esperado da operação (restrições de ordenação) é sobre o plano
Objetivo	Os planejadores buscam um caminho entre o estado inicial e o estado objetivo	Os planejadores buscam um plano que satisfaça certas condições restritivas
Vantagens	Controle total dos estados do ambiente	Flexibilidade para lidar com situações não previstas
Desvantagens	Dificuldade para lidar com falhas	Lento
Exemplo de técnica aplicável	Planejadores baseados em STRIPS	Planejador Temporal

Fonte: Produção do autor.

3.4. Tipos de sistemas de planejamento automatizado

Há três classificações de sistemas de planejamento quanto à descrição de seu domínio:

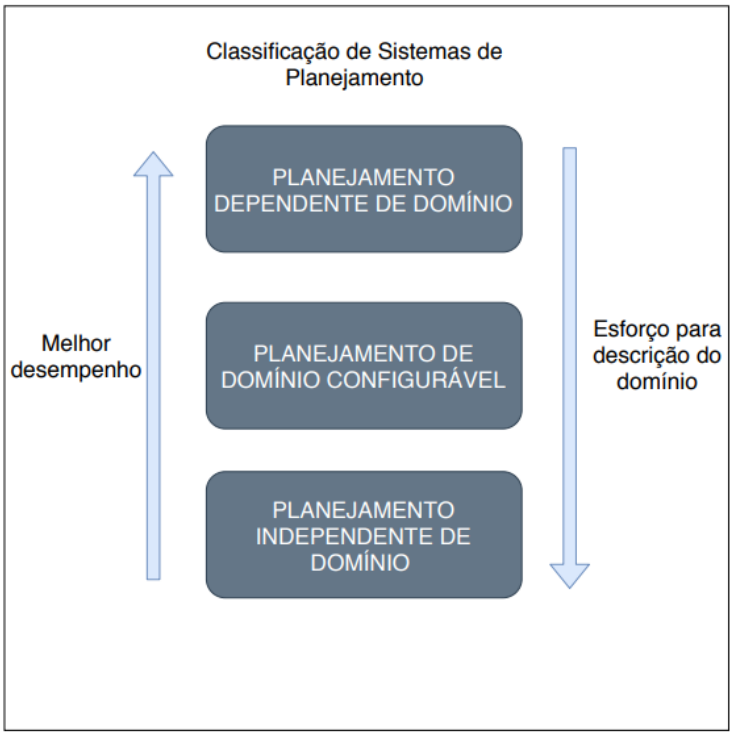
Um **sistema de planejamento dependente de domínio** é construído para operar sobre um domínio específico e apresenta desempenho drasticamente superior aos planejadores independentes de domínio (JIANG; XU, 2017). Muitas das aplicações práticas estão inseridas nessa classe. O sucesso se deve principalmente à presença de conhecimento específico embutido no domínio. No entanto, planejadores dessa categoria são mais difíceis de serem adaptados a outros domínios ou mesmo de utilizados em projetos futuros.

Um **sistema de domínio configurável** combina algumas das principais vantagens existentes nas duas abordagens. Adota-se conhecimento específico na modelagem do problema, mas o algoritmo que realiza a busca pelo plano é independente de domínio. Isto é, o planejador é genérico o suficiente para resolver problemas de outros domínios. Uma desvantagem dessa abordagem é que a modelagem do problema se torna mais complexa se comparada à solução independente de domínio. Um planejador HTN tradicional geralmente pertence a essa categoria.

Um **sistema de planejamento independente de domínio** é construído, em tese, para operar em qualquer domínio. Por essa razão, a definição de uma linguagem formal para especificar os problemas se torna mais evidente do que na abordagem dependente de domínio. Um exemplo clássico da abordagem independente de domínio é a linguagem PDDL e os diversos planejadores desenvolvidos para operar sobre ela. O maior desafio dessa abordagem é tornar a linguagem expressiva o suficiente para atender uma ampla variedade de problemas, mas restritiva o bastante para que algoritmos eficientes operem sobre ela. Na prática, é inviável que os planejadores independentes de domínio funcionem bem para qualquer tipo de domínio de planejamento. Sua principal desvantagem é apresentar um desempenho inferior se comparado às demais abordagens, justamente por não levar vantagem em nenhuma propriedade especial do domínio.

A Figura 3.8 traz uma ilustração comparativa das abordagens com relação ao desempenho e ao esforço humano necessário para descrição do domínio. Em termos de abrangência para resolução de diferentes problemas, os de domínio configurável levam vantagem, seguido dos sistemas independentes de domínio e, por último, a abordagem específica de domínio.

Figura 3.8 – Comparativo dos sistemas de planejamento.



Fonte: Adaptado de Nau (2007).

Uma vez descritas as classificações dos sistemas quanto à descrição do domínio, é conveniente apresentar algumas definições (D_1 a D_3) acerca das principais propriedades de uma solução de planejamento (GHALLAB et al. 2004):

D_1 (Correto): um planejador é dito correto (*sound*) se toda sequência de ações que ele retorna é de fato uma solução verdadeira. O algoritmo POP é um exemplo de um procedimento correto.

D_2 (Completo): se existir uma solução do problema, garante que pelo menos uma será sempre encontrada pelo planejador. O método de busca progressiva é um típico exemplo de uma solução completa (*complete*).

D_3 (Ótima): Se houver alguma medida de otimização, o planejador encontra sempre a solução ótima. Soluções ótimas são geralmente custosas computacionalmente, dado que usualmente é necessário visitar diferentes caminhos possíveis no espaço de busca.

Um exemplo de planejadores que exploram políticas ótimas são os domínios baseados em probabilidades abordados na Seção 4.1.2.1.

3.5. Extensão do planejamento clássico

Esta Tese tem interesse pelos planejadores aplicados, especialmente as técnicas compatíveis com o domínio embarcado da área espacial. Em geral, os planejadores aplicados são bastante expressivos e eficazes na solução de problemas específicos, pois podem ser adaptados através de procedimentos externos[19] em parte graças ao seu formalismo menos restritivo.

Em contrapartida, planejadores clássicos superam os planejadores aplicados na solução de problemas gerais, baseiam-se em padrões bem definidos e têm uma forte comunidade de pesquisa que constantemente produz novidades. Talvez a diferença mais preponderante é que os planejadores aplicados produzem planos para serem executados em um sistema físico, onde as ações podem deixar de ter os efeitos esperados. Já o planejamento clássico assume que as ações são sempre corretas e completas, o que torna a execução do plano incondicional. A seleção de ações no planejamento clássico também raramente leva em conta os recursos consumidos e o tempo atribuído a cada atividade. Em decorrência disso, os planejadores aplicados precisam lidar com algumas variáveis não tratadas no modelo clássico, discutidas a seguir.

3.5.1. Planejamento e escalonamento

Os planejadores teóricos do planejamento clássico conseguem modelar uma variedade de problemas acadêmicos. Em aplicações mais realistas e na maioria dos sistemas reais, entretanto, as atividades precisam ser complementadas com informações de

[19] Procedimentos externos computam uma lógica do problema, cuja declaração não está representada no modelo de domínio. De modo geral, são lógicas mais complexas que não podem ser descritas pela linguagem de especificação do domínio.

tempo e recursos. A decisão sobre a alocação de tempo e recurso às atividades do plano caracteriza um problema típico de escalonamento.

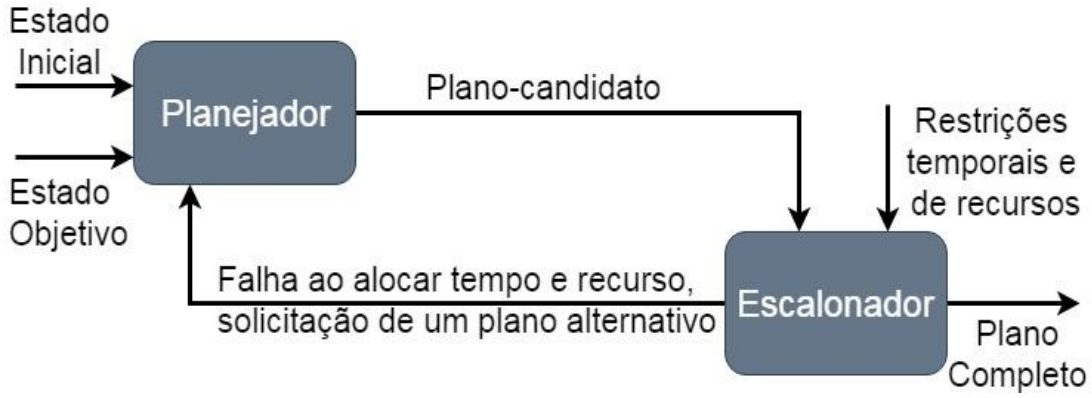
Os estudos sobre técnicas de escalonamento originaram-se na área de Pesquisa Operacional e vêm sendo incorporados a projetos em IA. Um problema de escalonamento envolve a atribuição de recursos, limitados em quantidade, para tarefas distintas durante determinado período de tempo, de forma a otimizar um ou mais objetivos. Há duas abordagens utilizadas em sistemas de planejamento que são tratadas a seguir.

3.5.2. Abordagem em cascata ou integrada

A maioria dos estudos de planejamento mantém os estágios de planejamento e escalonamento (P&S) de forma sequencial por simplificação: primeiro definem-se quais as ações, e depois as mesmas são posicionadas temporalmente e têm alocados a elas os recursos necessários para sua consecução. Porém, considerar as restrições impostas já na busca pelas ações, como nos sistemas P&S integrados (vide MUSCETTOLA et al., 1998), tende a melhorar o desempenho (QI et al., 2017).

Nesse caso, a sequência encontrada pelo planejador já considera o consumo de recursos e os momentos corretos de execução de cada comando. Na abordagem sequencial (não integrada), se um plano não for escalonável, o processo retorna novamente ao planejador para escolha de tarefas alternativas. Tal mecanismo, representado pela Figura 3.9, pode se repetir ciclicamente, tornando o processo lento e ineficiente.

Figura 3.9 – Planejamento e escalonamento em cascata.



Fonte: Adaptado de Kucinskis (2007).

Em aplicações reais, as etapas de P&S tendem a ser integradas (SMITH et al., 2000). Nos planejadores embarcados essas etapas têm sido unificadas. Um estudo mais amplo sobre a integração de P&S pode ser visto em: Garrido e Barber (2001).

O primeiro sistema a considerar tempo e recursos a partir de um escalonador integrado ao planejamento foi o planejador temporal denominado IxTeT (GHALLAB; LARUELLE, 1994; LEMAI, 2004). É uma abordagem considerada uma extensão do planejamento no espaço de planos e rica em representações temporais.

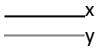
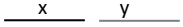
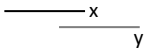
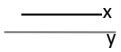
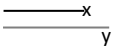
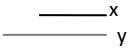
A seguir, descrevem-se formas existentes para manipulação temporal em sistemas de escalonamento aplicados à área de IA. Tais formas lidam com o relaxamento das suposições A_5 e A_6 (plano sequencial e atemporal) da Tabela 3.2.

3.5.3. Representação temporal

Para simplificar o raciocínio durante a busca pelo plano, a duração é abstraída no modelo clássico e todas as ações são consideradas transições instantâneas entre estados. Apesar disso, a representação do tempo nos problemas reais é crítica porque as ações e relações entre elas ocorrem ao longo do tempo. As informações temporais estabelecem quando os estados do domínio devem apresentar os valores desejados, ou em que período determinadas tarefas devem ser realizadas. Há dois formalismos comumente aplicados: a álgebra temporal e a álgebra de intervalos de Allen, conhecida também como intervalos de Allen (ALLEN, 1983).

O primeiro opera sobre restrições binárias qualitativas[20] entre instantes fixos, sem duração, denominados *'time points'*, com base nos seguintes símbolos de relação temporal: {<,>, =}. Nele, um ponto pode ser o início ou fim de algum evento, assim como um momento qualquer no tempo. O segundo opera sobre intervalos e restrições binárias qualitativas, utilizando treze relações primitivas: *{Equal; Before; Meet; Overlap; FinishedBy; Contain; Start; During; StartBy; Finish; OverlapBy; MeetBy; After}* que pela relação inversa se reduz a sete, como mostra a Tabela 3.4.

Tabela 3.4 – Relações temporais de Allen.

Relação	**Símbolo**	**Inverso**	**Intervalo**
X *equal* Y	eq	eq	
X *before Y*	b	b_i	
X *meets Y*	m	m_i	
X *overlaps* Y	o	o_i	
X *during* Y	d	d_i	
X *starts* Y	s	s_i	
X *finishes* Y	f	f_i	

Fonte: Allen (1983).

No modelo de Allen, as ações e eventos passam a ocorrer dentro de períodos de tempo que possuem relacionamentos temporais entre si. As restrições binárias são utilizadas para estabelecer as relações temporais entre tarefas. Da mesma forma, são usadas por (VILAIN; KAUTZ, 1986) para estabelecer as relações entre seus pontos no tempo.

[20] Uma restrição temporal é dita qualitativa se ela se refere a uma ordenação relativa das ações (*e.g.,* a ação *A* deve terminar antes de *B)*. Uma restrição temporal é quantitativa para uma posição absoluta no tempo (*e.g,* a ação *A* deve começar no tempo 0).

A forma de representação temporal mais utilizada até o momento no domínio espacial é o Intervalo de Allen. Segundo Mayer et al. (2016), os planos nessa representação são mais fáceis de serem reparados, pois certos deslocamentos nas ações (preservando os intervalos) podem ser gerados, sem necessitar revisar todo o plano.

Os planejadores temporais fazem uso da representação em intervalos e pontos no tempo. Foi a primeira abordagem a suportar nativamente a componente temporal no problema. Constituem uma extensão do planejamento clássico, onde os aspectos temporais das ações, eventos e objetos também são considerados.

O planejamento temporal pode ser resolvido com métodos semelhantes ao planejamento clássico. A principal diferença é que a definição de um estado deve incluir informações sobre o tempo absoluto atual e até que ponto a execução de cada tarefa ativa deve ocorrer, considerando a possibilidade das atividades se sobreporem temporalmente.

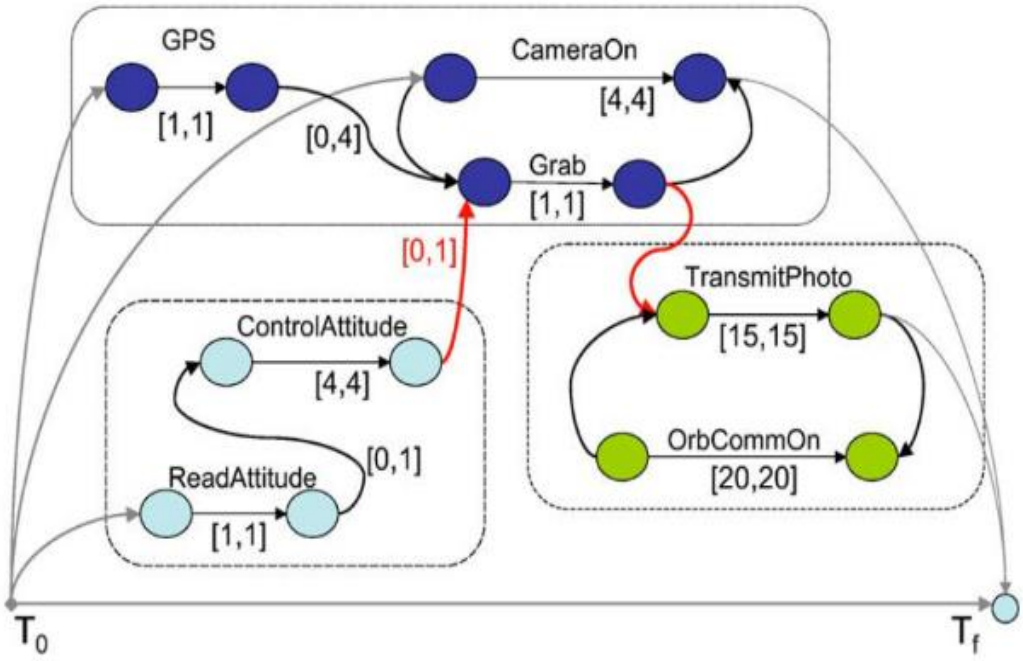
Restrições temporais são representadas em problemas de planejamento usualmente por meio de redes temporais. A ideia é representar as relações temporais usando um grafo, onde os nós são eventos temporais e os arcos entre os nós denotam o tempo mínimo e máximo que podem ocorrer entre os eventos (DECHTER et al., 1991).

Um tipo especial de redes temporais é a *Simple Task Network* (STN). Nela, cada restrição temporal aceita somente um valor de intervalo de tempo - não se permite, por exemplo, disjunção entre intervalos. Alguns trabalhos da área espacial (vide AMIGONI et al., 2010) usam STN para verificar a consistência e formular o caminho mínimo entre os nós. De acordo com Amigoni et al. (2010), a consistência de uma STN pode ser verificada em tempo polinomial[21], usando algoritmos de exploração de caminhos em grafos com arestas contendo pesos (*e.g., all-pair-shortest-paths*[22]*,*

[21] Mais precisamente a complexidade é de tempo cúbico $O(n^3)$.
[22] Também conhecido como algoritmo de *Floyd-Warshall.*

algoritmo de Dijkstra, A* entre outros). A Figura 3.10 traz um exemplo[23] de uma atividade relacionada a um satélite de sensoriamento remoto contida em uma STN.

Figura 3.10 – Exemplo de uma representação STN para uma atividade do domínio de satélites.



Fonte: Amigoni et al. (2010).

A principal limitação da STN é não suportar o gerenciamento de recursos (QI et al., 2017). Para tratar isso, a maioria dos planejadores do domínio espacial adicionou um escalonador (usando a técnica de CSP) associado ao conceito de timelines, como será visto na Seção 4.1.1.1.

O próximo capítulo mostra como esses recursos foram utilizados pelos planejadores aplicados, enfatizando-se o estado da arte do domínio espacial.

[23] O objetivo é ilustrar uma representação clássica de uma STN e não se aprofundar nas tarefas deste exemplo.

4 ESTADO DA ARTE DO DOMÍNIO ESPACIAL: TÉCNICAS DE PLANEJAMENTO AUTOMATIZADO COM POTENCIAL APLICAÇÃO A BORDO DE SATÉLITES

A pesquisa de planejamento automatizado produziu algumas técnicas muito poderosas que podem ser generalizadas para trabalhar em domínios não clássicos. São exemplos o planejamento temporal, planejamento baseado em decomposição hierárquica de tarefas e o planejamento sob incerteza (NAU, 2007).

Tais abordagens são consideradas uma extensão do planejamento clássico e algumas delas frequentemente adotadas por planejadores aplicados. O presente capítulo enfatiza as técnicas com histórico ou potencial de aplicação no domínio espacial, abordando-se as linguagens e os planejadores de bordo existentes e uma análise comparativa entre os estudos já realizados.

4.1. Técnicas de planejamento automatizado para domínios não clássicos com aplicação na área espacial

O planejamento embarcado visa encontrar uma sequência de comandos que leve o satélite a satisfazer os objetivos da missão de forma automatizada. A seguir são descritas as técnicas com potencial aplicação no domínio espacial. Nesta Tese, elas foram organizadas em dois grupos, considerando se o modelo de planejamento empregado lida explicitamente com domínios não determinísticos.

4.1.1. Técnicas de planejamento em domínio determinístico

Formalmente, um domínio de planejamento é dito determinístico se $|\gamma(s,a)| \leq 1$ para $\forall s$ e $\forall a$ (KUTER; NAU, 2004; NAU, 2007). Técnicas que lidam com ambientes determinísticos na área espacial em geral lidam com os objetivos de duas formas. A primeira é traduzindo o objetivo recebido na forma de restrições de estados do satélite. A segunda é representá-lo como uma tarefa de alto nível, que seja decomposta em uma rede de tarefas hierarquicamente menores. Tais formas de representação são tratadas respectivamente como 'Problemas de Satisfação de

Restrições' (CSP) e 'Rede de Tarefas Hierárquicas' (HTN), ambos os estudos na área de IA.

4.1.1.1. Planejadores temporais baseados em CSP

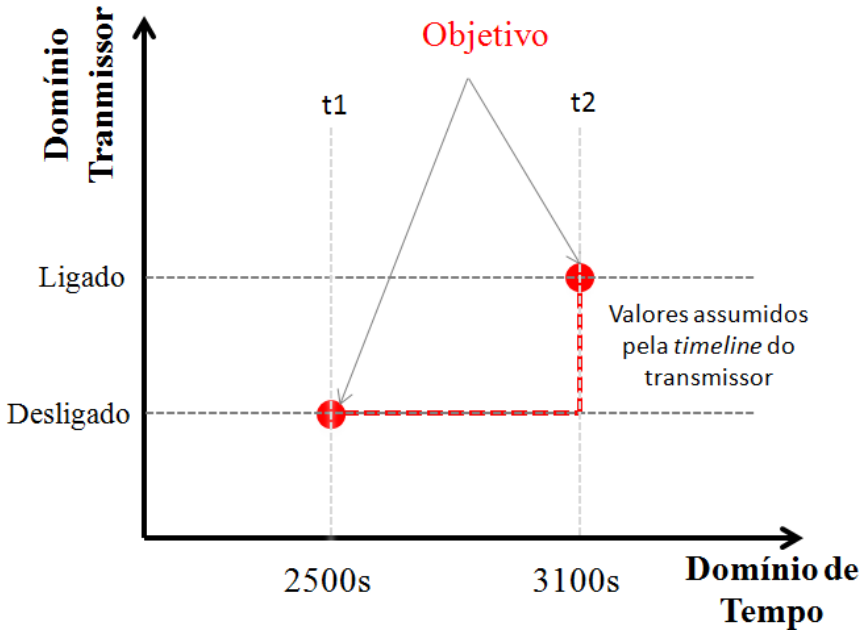
A maioria dos sistemas de planejamento embarcado da área espacial tratam os objetivos como um Problema de Satisfação de Restrições (CSP). Formalmente, um CSP é denotado por um conjunto finito de variáveis $X = \{x_1,...,x_n\}$; um conjunto finito de domínios de cada variável $\{D_1,...,D_n\}$, tal que $x_i \in D_i$; e um conjunto finito de restrições $\{C_1,...,C_m\}$. O objetivo final é atribuir valores para cada variável de forma que todas as restrições impostas sejam satisfeitas.

Uma solução ótima é aquela que atende todas as restrições e ao mesmo tempo maximiza ou minimiza os valores da função objetivo. Um CSP é dito consistente se tal solução existir. Há dois métodos principais de algoritmos que solucionam um CSP (SMITH et al., 2000): busca construtiva e busca local (iterativos). De forma sumária, eles diferem em sua técnica de otimização: o primeiro tenta encontrar uma solução ótima, já o segundo tem seu foco na geração de planos rápidos (WOJTKOWIAK et al., 2013). A busca local tem benefícios práticos, pois permite tomar como entrada qualquer plano independentemente de suas falhas e inconsistências.

Um ponto chave do planejador CSP, que difere do planejamento clássico, é a importância do mecanismo de inferência de estados. Em um domínio modelado na forma de CSP, os estados são definidos pelos valores de um conjunto de variáveis, e os objetivos consistem de um conjunto de restrições que tais valores devem obedecer.

Para que o CSP seja instanciado pelo planejador, é preciso que os objetivos sejam transformados em restrições sobre estados de variáveis do sistema a partir de um domínio no tempo. A ideia de representar um objetivo na forma de restrições do satélite pode ser vista com maior completude em (DVORAK et al., 2009). A Figura 4.1 ilustra o seguinte objetivo: 'o transmissor de dados da carga útil deve ser ligado no momento 2500s e desligado em 3100s'.

Figura 4.1 – Objetivo como uma restrição binária de um estado (um recurso/timeline do sistema) no tempo (domínio de tempo).



Fonte: Cividanes et al. (2019).

A partir da Figura 4.1, pode-se formular um CSP com as informações seguintes.

Dados os domínios:

$D_{transmissor}$ = *{ligado, desligado}*

D_t = *{0, ...}*

E as variáveis:

transmissor $\in$ $D_{transmissor}$

momento $\in$ D_t

A restrição que estabelece o objetivo é representada como*:*

$R_{transmissor,momento}$ = *(ligado, 2500; desligado, 3100).*

Outros objetivos podem ser instanciados a partir de restrições que envolvam um ou mais estados. Por exemplo, o objetivo: *'ligar uma câmera no momento n'* em uma missão de Observação da Terra envolve a alocação de diferentes recursos, a saber: potência, taxa de utilização da memória do gravador de dados, apontamento do

satélite e consumo de combustível para eventuais manobras, etc. A partir dessa forma de relacionamento entre estados e tempo é possível descrever um comportamento complexo através de conjuntos de restrições binárias que imponham ao CSP estados desejados para *timelines* e recursos em períodos específicos.

Um planejador baseado em *timelines* é definido como uma coleção de restrições sobre um conjunto de elementos temporais (pontos no tempo ou intervalos, discutidos na Seção 3.5.3), atribuídos a algumas atividades (CIVIDANES et al., 2019). Uma ação não é uma transição de estado única, mas uma coleção de eventos sincronizados que alteram as condições de variáveis de estado num certo intervalo de tempo.

As soluções são representadas como *timelines* ou sequência de estados, definindo o comportamento desejado do sistema. Isso pode ser resolvido aplicando técnicas genéricas de CSP, embora algoritmos específicos para lidar com o gerenciamento de recursos possam ser utilizados.

Cada *timeline* representa a evolução de um estado de uma variável do sistema num dado horizonte, como o exemplo da Figura 4.1. Qualquer atividade que altere recursos ou estados pode ser alocada numa *timeline*. Sua adoção em sistemas de planejamento e escalonamento demonstrou ser promissora em diversas aplicações reais, como em sistemas espaciais autônomos[24] (MAYER et al., 2016).

Este conceito é aderente às aplicações espaciais, pois sua abstração está mais próxima da forma em que os problemas e as restrições são naturalmente representados na operação real de um satélite (FRATINI et al., 2014). Sua característica permite aos planejadores temporais baseados em CSP unificarem as etapas de planejamento e escalonamento (FRATINI et al., 2008).

[24] Cabe a ressalva que alguns autores acabam estendendo o grau de maturidade alcançado pela tecnologia de planejamento em solo de missões espaciais às aplicações espaciais autônomas, o que ainda não é uma realidade. O uso de planejamento embarcado seja por experimento em voo ou pesquisa aplicada ainda é uma temática carente de publicações e resultados.

Como uma atividade do plano pode estar associada a um intervalo de tempo, ao invés de uma ocorrência temporal exata, a solução possui flexibilidade para enfrentar a incerteza do ambiente durante sua execução (MAYER et al., 2016). Para informações adicionais sobre uso de *timelines* em aplicações espaciais, consulte (CHIEN et al., 2012; MAYER et al., 2016).

4.1.1.2. Planejadores hierárquicos baseados em HTN

Ao invés de buscar um estado objetivo como no CSP, as técnicas de planejamento em HTN tentam realizar um objetivo a partir da decomposição de tarefas de alto nível em subtarefas até chegarem às tarefas primitivas (CIVIDANES et al., 2019). Para cada tarefa não primitiva, o planejador escolhe um método aplicável e o instancia para decompor a tarefa em subtarefas. Os métodos descrevem os meios pelos quais as tarefas podem ser reduzidas. Eles podem ter precondições para sua execução e efeitos de suas ações no ambiente.

Além disso, pode haver restrições ou regras de como tarefas de mais baixo nível podem ou não se relacionar. As restrições definem a forma com que as variáveis devem ser instanciadas e quais condições devem ser verdadeiras para que as tarefas sejam executadas. Um plano válido é encontrado quando todas as restrições forem satisfeitas e restarem somente tarefas que podem ser executadas diretamente sem precisar de decomposição.

Dentre outros formalismos, o problema de planejamento em HTN pode ser representado por 3-tuplas *(s, U, D),* onde (QI et al., 2017): *s* representa os estados; *U* é uma rede de tarefas; *D = (O, M)* denota o domínio do problema, onde: *M* é um conjunto de métodos e *O* é um de conjunto operadores que indica como as tarefas primitivas podem ser executadas.

A técnica de HTN requer um conhecimento do domínio por especialistas para descrever a hierarquia das tarefas – funcionando, de certa forma, como heurísticas de busca –, mas sem depender de um mecanismo de busca refinado para encontrar as decomposições possíveis e ações associadas.

Os métodos de decomposição facilitam muito a solução do problema de planejamento, pois geram apenas planos que são soluções para o problema. A HTN é adequada para domínios nos quais alguma representação hierárquica é desejável ou conhecida antecipadamente (GEORGIEVSKI; AIELLO, 2015).

É útil para aplicações práticas devido à sua capacidade de diminuir o espaço de busca do problema. O planejador HTN é eficiente em tempo de execução (QI et al., 2017), o que é promissor para o ambiente embarcado. Sua aplicação em veículos espaciais autônomos permite diminuir bastante a complexidade do problema de planejamento (ZHANG et al., 2006).

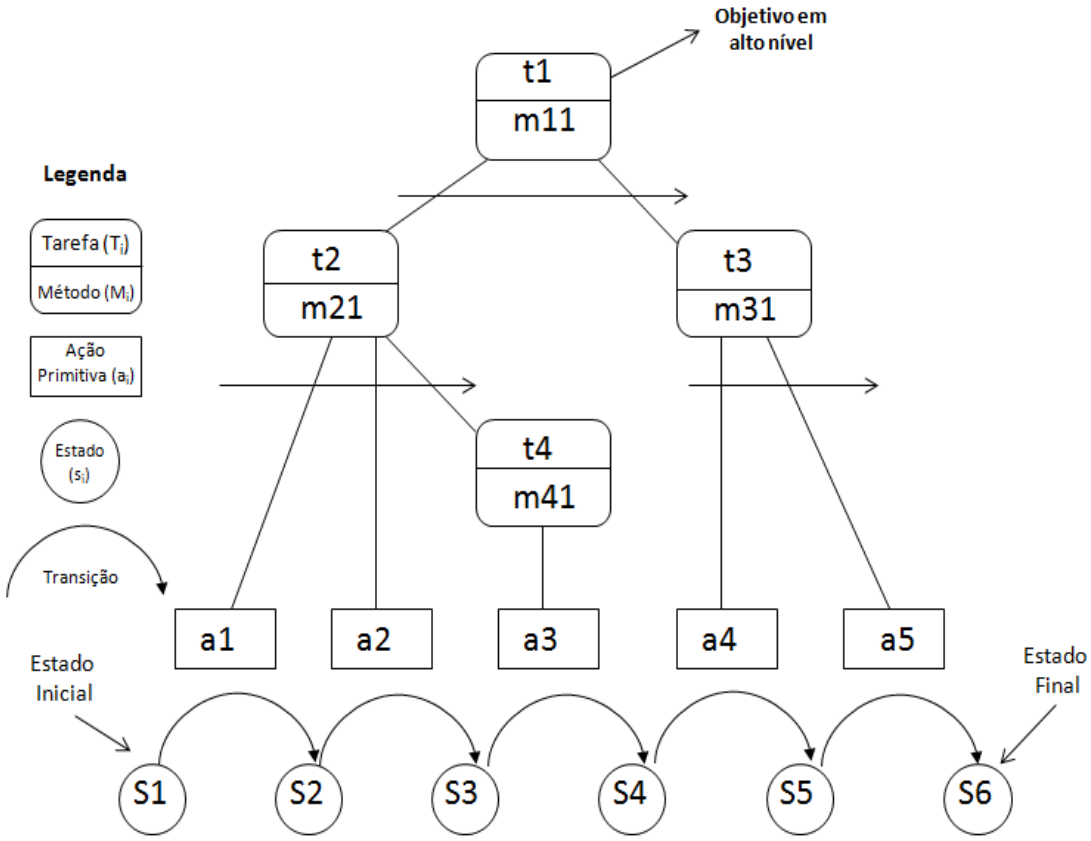
Considerando o contexto histórico, os primeiros planejadores HTN construídos foram do tipo 'espaço de planos' (*e.g.,* Nonlin - TATE, 1977), já os mais modernos seguem a abordagem de 'espaço de estados'. O planejador proposto nesta Tese se baseia na abordagem de espaço de estados. Para conhecimento de uma definição formal dessas abordagens, consulte Georgievski e Aiello (2015).

O planejamento em HTN pode ser de ordem total (*e.g., Simple Hierarchical Ordered Planner* - *SHOP,* NAU et al., 2003) ou parcial (*e.g., Flexible Acting and Planning Environment* - FAPE, Dvorak et al., 2014).

A diferença entre eles é que o de ordem parcial permite que cada método se decomponha em um conjunto de subtarefas parcialmente ordenadas. Isso traz mais expressividade ao planejador, em contrapartida pode incrementar a complexidade do algoritmo.

A Figura 4.2 mostra um exemplo de um planejamento HTN de ordem total. Nela, apresenta-se um objetivo expresso na forma de uma rede de tarefas e o mecanismo de busca das atividades que satisfaçam este objetivo. A seta representa a ordenação das subtarefas que neste exemplo é dado a partir de uma busca progressiva cuja direção é da esquerda para a direita.

Figura 4.2 – Exemplo de um objetivo descrito como uma rede de tarefas hierárquicas a partir de uma árvore de decomposição.



Fonte: Cividanes et al. (2019).

O Algoritmo 4.1 apresenta a lógica base de um planejador HTN abstrato cuja implementação é genérica para acomodar tanto a estratégia de ordem total, como a parcial. Nele, têm-se os seguintes parâmetros: *s* representa os estados do sistema, *U é* o conjunto de rede de tarefas, *C* é o conjunto de restrições, *O* é um conjunto de operadores e *M* representa uma coleção de métodos. A linha 4 verifica se a tarefa é primitiva para tentar proceder com uma possível solução do problema, caso atenda todas as restrições impostas (linha 6). Se for uma tarefa composta (linha 12), escolhe-se um método aplicável, que não tenha sido selecionado anteriormente. Em seguida (linha 19), invoca-se a mesma função passando como parâmetro as novas subtarefas e restrições geradas no passo anterior (linha 18).

Algoritmo 4.1 – Um algoritmo abstrato para planejamento em HTN

```
1  Função HTN-Abstrato (s, U, C, O, M )
2    se (U,C ) demonstrou não ter uma solução então
3        retorne falha;
4    senão se U é uma tarefa primitiva então
5       se (U,C ) tem uma solução então
6         não deterministicamente deixe π ser uma solução;
7         retorne π;
8       fim
9       senão
10        retorne falha;
11      fim
12   senão
13      escolha uma tarefa composta u ∈ U ;
14      ativo ← {m ∈ M | task(m) é unificável com t_u };
15      se ativo != Ø então
16          não deterministicamente escolha qualquer m ∈ ativo;
17          σ ← chame mgu para m e t_u que renomeie todas as variáveis de m;
18          (U',C') ← δ(σ (U,C), σ(u), σ (m));
19        retorne HTN-Abstrato(s, U', C', O, M);
20      fim
21      senão
22        retorne falha;
23      fim
24 fim
```

Fonte: Adaptado de Cardoso (2018).

Em um extenso estudo realizado por Georgievski e Aiello (2015), observou-se que praticamente todos os planejadores HTN implementam a busca em profundidade que é do tipo de busca desinformada (cega), e apenas um dos trabalhos citados no estudo utilizou a busca informada.

Algoritmos de busca informada (*e.g.*, A*) consideram informações adicionais para avaliar a distância do nó atual ao nó objetivo. Estratégias de pesquisa informada podem encontrar soluções com maior eficiência do que as não informadas.

No entanto, o propósito do planejamento em HTN é explorar a existência de um caminho presente na rede de tarefas que leve à realização do objetivo, e não visa necessariamente encontrar uma solução ótima (*e.g.*, caminho mínimo). Isso é feito,

por exemplo, pelo algoritmo A* a partir de uma função que avalia peso ou custo dos caminhos.

Com relação à expressividade da representação em HTN, Erol et al. (1994) provam a partir de análises de complexidade que o problema descrito em HTN é mais expressivo que as linguagens de planejamento clássico. Foi mostrado que um problema especificado em STRIPS pode ser representado por HTN, mas o contrário não se aplica. A desvantagem da HTN frente ao planejamento clássico é a sua necessidade de especificar além das ações elementares, uma coleção de métodos de decomposição.

Fora do domínio espacial, existem planejadores HTN independentes de domínio. Alguns exemplos relevantes são *System for Interactive Planning and Execution* (SIPE[25] – WILKINS et al., 1995), JSHOP2 desenvolvido na linguagem JAVA cuja versão é uma extensão do seu antecessor SHOP[26], escrito em LISP.

Entretanto, suas linguagens, assim como a forma de representação do problema não são compatíveis com o ambiente computacional espacial. Esses planejadores também são inadequados para lidar com problemas da área espacial, pois tipicamente não consideram conjuntamente: tempo, recursos agregados e a possibilidade de falhas nas ações de planejamento. De acordo com Jiménez et al. (2012), um dos grandes problemas de planejadores de "prateleira" é que eles possuem baixo poder de escalabilidade e são falhos em propor soluções de boa qualidade.

O contexto acima evidencia que os planejadores de prateleira não são aderentes aos objetivos desta Tese.

Recomenda-se o trabalho de Georgievski e Aiello (2015) para uma revisão detalhada sobre o estado da arte em planejamento hierárquico em HTN.

---

[25] O SIPE foi o primeiro planejador HTN a estender a representação temporal das atividades usando STN.
[26] A principal diferença entre as versões (SHOP e SHOP2) é que o SHOP2 acomoda a decomposição de métodos HTN em ordem parcial. JSHOP2 é uma implementação em Java da versão SHOP2 escrita originalmente em LISP.

4.1.2.Técnicas de planejamento em domínio não determinístico

Até o momento apresentaram-se representações de planejamento cujas ações têm efeitos determinísticos. Todavia, há casos em que pode ser útil supor que certas ações do ambiente tenham mais de um resultado possível. Isso pode ser adequado em situações onde o efeito de uma ação pode variar devido a mudanças aleatórias no ambiente ou à interferência de outros agentes. Em tese, são indicadas para domínios cujo grau de incerteza do ambiente é maior, como os encontrados por sondas espaciais de exploração planetária.

Para tratar o planejamento sob incerteza, duas técnicas têm sido alvos de pesquisa, descritas a seguir. Elas lidam em geral com o não determinismo limitado do ambiente. Nesta definição, apesar das ações produzirem múltiplas saídas, seus efeitos no ambiente podem ser previstos antecipadamente, o que possibilita especificá-los na descrição do domínio. Isso dá a oportunidade de tratar o não determinismo do ambiente codificando efeitos alternativos de uma ação.

Embora o termo "não determinístico" seja amplamente aceito e utilizado na área de planejamento, cabe salientar que o modelo não é de fato não determinístico. O termo não determinístico se refere ao ambiente de planejamento e o modelo como forma de prever possíveis falhas resultantes das ações. Trata-se de uma alternativa para superar o modelo clássico que considera a execução da ação incondicional.

Como será visto adiante, a presente Tese propõe que o não determinismo do ambiente deve ser tratado pelo planejador. Isso faz com que o esforço da modelagem do problema seja transferido ao algoritmo de planejamento.

4.1.2.1. Planejamento sob incerteza (MDP e MC)

Considerando o problema de planejamento clássico (representado pela tripla $P = \langle S_0, G, A \rangle$), uma versão não determinística é idêntica com a exceção de que cada ação *A'* pode ter um conjunto de efeitos adicionais, os quais podem modelar falhas ou eventos

exógenos. Na prática, um domínio não determinístico expressa a incerteza que o agente tem acerca do comportamento do ambiente.

As principais fontes de incerteza do ambiente estão relacionadas ao relaxamento das propriedades restritivas do modelo clássico, especificamente: conhecimento parcial sobre o estado do mundo[27], não determinismo e ambiente não estático.

Técnicas de planejamento sob incerteza lidam com o não determinismo do ambiente a partir de modelos probabilísticos, planos de contingência ou usando técnicas de verificação de modelos. Dentre elas, destaca-se o processo de decisão *markoviano* (*Markov Decision Process* - MDP) e *Model Checking* (MC).

Em um domínio representado por MDP existe uma função de utilidade ligada aos estados, com recompensas, penalidades e distribuição de probabilidades. O objetivo consiste em maximizar a recompensa acumulada obtida a partir da visitação dos estados com base na probabilidade dos efeitos das ações.

Em outras palavras, os modelos probabilísticos permitem a representação de ações com consequências associadas à distribuição de probabilidades. Isto possibilita modelar a dinâmica do ambiente onde alguns resultados têm maior chance de ocorrer do que outros.

Os planos não são especificados como uma sequência de ações, mas como políticas que mapeiam estados em ações. A solução é uma política ótima que descreve a melhor ação para cada estado. Tal política pode ser encontrada a partir de uma variedade de métodos, incluindo algoritmos de programação dinâmica, tais como iteração de valor ou iteração de política (EDDY; KOCHENDERFER, 2019).

Formalmente, um MDP é definido pela 4-tupla: *<S, A, T, R>,* onde: *S* é um conjunto de estados que o processo pode estar; *A* é um conjunto de ações executáveis; *T* é um

[27] A observabilidade total refere-se à capacidade do agente ter acesso completo a todas as variáveis de estado do ambiente. Em certos cenários de planejamento, algumas variáveis de estados podem nunca ser observáveis, configurando-se a 'observabilidade parcial' do ambiente.

conjunto de probabilidades de transição condicional entre estados, sendo $T : S \times A \rightarrow \Pi(S)$; e $R : S \times A \rightarrow \mathbb{R}_+$ corresponde à função de recompensa para tomada de decisão quando se está em um estado $s \in S$ (KAELBLING et al., 1998).

A estrutura do MDP pressupõe que após executar uma ação que considera um resultado incerto, o sistema poderá observar completamente o estado resultante (SMITH et al., 2000).

Em outras palavras, o MDP assume que o ambiente de planejamento é completamente observável. Para lidar especificamente com ambiente parcialmente observável, existe uma extensão do MDP denominada *Partially Observable Markov Decision Process* (POMDP).

Embora o MDP busque soluções ótimas, ele torna o espaço de busca do problema grande (SMITH et al., 2000; LEMAI, 2004) e tem sido aplicado a problemas menores, dada a dificuldade em encontrar uma solução viável em termos de tempo e recursos computacionais (CODETTA-RAITERI; PORTINALE, 2015).

A falta de dados estatísticos para definição das probabilidades (PUTERMAN et al., 1994) também é um obstáculo, especialmente se a aplicação for embarcada em um veículo espacial. Aliás, em aplicações práticas pode ser muito difícil obter dados probabilísticos fidedignos. Deve-se observar ainda que o modelo MDP tradicional não dispõe de uma representação explícita para manipulação temporal (SMITH et al., 2000), diferentemente do planejamento baseado em verificação de modelo.

Já as soluções do MC, por outro lado, não usam probabilidades e são definidas em termos de estados alcançáveis e planos de contingência. A ideia do planejamento baseado na verificação de modelos é resolver problemas de planejamento a partir de um modelo teórico. O domínio é descrito como um sistema de transição de estados finitos não determinístico baseado em possibilidades. O objetivo é expresso por meio de uma fórmula de lógica temporal (NARDONE et al., 2019).

Dado um sistema de transição de estado e uma fórmula de lógica temporal, o planejamento baseado em MC gera planos que controlam a evolução do sistema para

todos os comportamentos do ambiente que tornam a fórmula temporal verdadeira (GHALLAB et al., 2004). Trata-se de uma técnica de verificação formal de sistemas que visa provar propriedades com base na exploração exaustiva de estados do sistema.

No entanto, diferentemente do MDP, as políticas do MC contêm apenas os estados envolvidos nas transições para atingir os objetivos (VICTORIA et al., 2012). Por isso, MC tem obtido melhores resultados que o MDP apesar de também sofrer com a explosão de estados (KUTER; NAU, 2004). Isso acaba sendo mais problemático no MDP devido à sua representação explícita do domínio (VICTORIA et al., 2012).

A solução obtida na verificação por modelos tende a ser mais confiável devido ao uso de métodos formais, o que traz mais garantias sobre a validade dos planos gerados (PEREIRA, 2007).

É possível perceber que MDP e MC possuem abordagens distintas para lidar com o não determinismo do ambiente em problemas de planejamento. Enquanto o primeiro é voltado a um problema de otimização, o segundo representa um problema de satisfatibilidade.

Ambas as abordagens sofrem com o alto custo para descrição do domínio. Se comparado ao formalismo das técnicas não determinísticas, as representações por HTN e CSP são mais intuitivas de se modelar o problema para não especialistas em IA.

Outro entrave é que para uma mesma entrada pode haver resultados distintos, dificultando os testes de robustez em solo. Como a execução de um plano pode produzir mais de um caminho, a definição de uma solução torna-se mais difícil se comparada ao planejamento clássico. Outra consequência é o aumento do espaço de busca do problema devido à necessidade de considerar diferentes efeitos nas ações.

Em resumo, as técnicas atuais de planejamento que suportam nativamente as incertezas do ambiente não são aderentes ao domínio embarcado. Se aplicados de forma realista, esses planejadores tendem a ser computacionalmente intratáveis e, portanto, possuem até o momento pouca relevância prática.

Além disso, as técnicas supracitadas lidam apenas com não determinismo limitado do ambiente. Em sistemas reais, como aplicações espaciais, não é possível enumerar todas as falhas (ou ações indesejáveis) resultantes de ações.

No levantamento bibliográfico realizado por Cividanes et al. (2019), encontraram-se dois trabalhos que exploraram MDP e MC no domínio espacial: D'angelo et al. (2017) e Bozzano et al. (2008).

4.2. Estado da arte em soluções de planejamento embarcado

Uma vez apresentadas as técnicas computacionais, podemos descrever os principais planejadores desenvolvidos pela comunidade da área espacial. Eles são considerados o cérebro dos sistemas de planejamento. Antes de abordá-los, faz-se uma breve introdução acerca de duas abordagens de planejamento e de linguagens voltadas ao domínio espacial.

4.2.1. Planejamento em lote *versus* contínuo

Há duas abordagens tradicionais já adotadas por aplicações embarcadas da área espacial: planejamento em lote ou contínuo (CHIEN et al., 1999; KINGHT et al., 2001). Embora o princípio desses conceitos esteja relacionado intimamente ao mecanismo de atuação do planejador, ele também exerce influência no tamanho do horizonte temporal, sobretudo nos planejadores temporais.

No planejamento realizado por lote (*batch),* utilizado, por exemplo, em (MUSCETTOLA et al., 1998; NOGUEIRA et al., 2017), o tempo é dividido em vários horizontes de planejamento, cada qual com um período de tempo, em geral, significativo. Quando o tempo se aproxima ao final do horizonte do plano corrente, o planejador projeta o estado futuro que o satélite estará no final da execução do plano atual. Com isso, formula-se um novo problema conforme os objetivos e os estados inferidos do próximo horizonte. É uma abordagem considerada custosa computacionalmente (NOGUEIRA et al., 2017). Nela, não há atualização de estados e nem deliberação sobre novos objetivos durante a execução, o que limita sua capacidade de reação.

Em contrapartida, o planejamento contínuo permite o replanejamento durante a execução do plano a partir de uma mudança no contexto da plataforma espacial. Ele visa minimizar o tempo de reparo de forma que, a qualquer momento, uma atualização incremental das metas, do estado atual ou do horizonte (com incrementos de tempo pequenos) possa mudar o estado do plano corrente. Isso deve prevalecer nos sistemas embarcados devido à menor latência na resposta. Ademais, mantendo um horizonte mais curto, a projeção dos estados futuros torna-se mais precisa, gerando planos de melhor qualidade (CIVIDANES et al., 2019).

O conceito do planejamento em lote se aproxima à abordagem clássica (planejamento *offline*) e o contínuo dos planejadores aplicados devido à característica de monitoração *online*. Apesar de estes conceitos terem sido aplicados a planejadores temporais, eles podem ser estendidos a outras classes. Na sua forma mais geral, a definição de planejamento contínuo baseia-se no monitoramento contínuo, no qual o planejador deve persistir a diferentes execuções, mesmo na presença de eventos inesperados.

4.2.2. Linguagens de descrição usadas no domínio espacial

Assim como no planejamento clássico, é preciso de uma linguagem para descrição do domínio espacial. Ela deve descrever o comportamento do satélite durante a sua operação a partir de representações de suas ações, estados e objetivos.

Linguagens do planejamento clássico como a PDDL não foram desenvolvidas para execução em ambientes embarcados. De acordo com Smith et al. (2008), a natureza proposicional da PDDL, as construções restritas para descrever mudanças e consumo de recursos e a capacidade limitada para modelar restrições temporais, tornam difícil ou até mesmo impossível o seu uso em aplicações reais.

Considerando o cenário acima e uma vez que não existe uma linguagem padronizada para a área espacial (FRATINI et al., 2014), os projetistas têm empregado linguagens específicas para cada aplicação. Elas são em geral baseadas em intervalos de tempo, restrição de valores e quantidade de recursos consumidos.

Uma linguagem pioneira é a *ASPEN Modeling Language* (AML) do planejador de operações executado em solo *Automated Scheduling and Planning Environment* (ASPEN – KNIGHT et al., 2014; FUKUNAGA et al., 1997), cuja representação é voltada a ações, hierarquia de atividades e recursos utilizados em planejadores baseados em CSP.

A maioria (vide Tabela 4.1), no entanto, é baseada no conceito de *timelines,* sendo ricas em representações de restrições temporais. São exemplos: a *New Domain Definition Language* (NDDL) do planejador *Extensible Universal Remote Operations Planning Architecture* (EUROPA – BARREIRO et al., 2012); a *Domain Definition Language* (DDL) utilizada pelo framework de planejamento denominado *Advanced Planning and Scheduling Initiative* (APSI – FRATINI; CESTA 2012); e ISISml – KUCINSKIS; FERREIRA, 2011) da arquitetura *Goal-based Enabling Software Architecture* (GOESA – KUCINSKIS; FERREIRA, 2013) e a XIDDL (XML IDEA *Domain Definition Language* – ASCHWANDEN et al., 2006) da arquitetura IDEA.

A *Action Notation Modeling Language* (ANML - SMITH et al., 2008) é uma inciativa da NASA voltada para criar uma linguagem unificada para o planejamento baseado em *timelines*.

Tabela 4.1 – Linguagens de definição usadas no domínio espacial.

Nome	**Principais Componentes**	**Sistema Planejamento**	**Abordagem de Planejamento**
AML	Ações e Timelines	ASPEN	CSP (Planejamento Temporal)
NDDL	Timelines	EUROPA	CSP (Planejamento Temporal)
DDL	Timelines	APSI	CSP (Planejamento Temporal)
ISISml	Ações e Timelines	GOESA	CSP (Planejamento Temporal)
XIDDL	Timelines	IDEA (Multiagente)	CSP (Planejamento Temporal)

Fonte: Produção do autor.

A próxima seção descreve os planejadores de bordo. Eles evoluíram a partir dos planejadores executados em solo. No entanto, há características importantes que tornam seu projeto mais desafiador que seus antecessores: estão limitados às especificidades do ambiente computacional da área espacial e devem lidar com a capacidade de planejamento em tempo real.

4.2.3. HSTS e RAX-PS

Embora o *Heuristic Scheduling Testbed System* (HSTS – MUSCETTOLA, 1994) seja executado em solo, ele pode ser considerado precursor dos planejadores de bordo e pai dos planejadores temporais modernos, pelo menos no domínio espacial. Este *framework* permite representar o ambiente como um conjunto de variáveis de estados que se modificam no tempo. Ele descreve a dinâmica do domínio, definindo as atividades e a modelagem de recursos do sistema. Foi utilizado para o planejamento de observações e operação do Telescópio Hubble.

O RAX-PS, inspirado no HSTS, manipula os objetivos na forma de um CSP. Trata-se de um planejador temporal baseado em *timelines* (DVORAK et al., 2014) e não utiliza a abordagem de HTN (CHIEN et al., 1998). O mecanismo de planejamento utiliza uma variedade de algoritmos de propagação de restrições para localizar e resolver conflitos até que uma solução seja encontrada. Uma limitação está no tempo de resposta devido à característica do planejamento em lote adotado.

4.2.4. CASPER

O *Continuous Activity Scheduling Planning Execution and Replanning* (CASPER), inspirado no ASPEN, representa o maior caso de sucesso de planejamento embarcado na área espacial. Ele utiliza a abordagem de planejamento contínuo por meio da técnica de reparo iterativo (CHIEN et al., 1999a e BU et al., 2016). Nela, os conflitos são detectados, em seguida o algoritmo seleciona um conflito associado a um possível método de reparo (TRAN et al., 2004).

O processo é repetido até encontrar um plano válido livre de falhas. Este planejador temporal mantém um plano consistente com as informações atualizadas, conforme o ciclo: mudanças no estado inicial ou dos objetivos; propagar os efeitos das mudanças; e invocar algoritmos de reparo baseado em regras para remoção dos conflitos. Sua premissa é suportar a atualização contínua do plano corrente (PENG et al., 2018).

O Algoritmo 4.2 traz a lógica do algoritmo de reparo iterativo. Basicamente, essa técnica seleciona iterativamente um conflito de escalonamento e executa alguma ação (adicionar, mover ou remover) para tentar solucioná-lo. Ela se aplica aos planejadores do tipo espaço de planos e funciona como um solucionador de falhas (equivalente à função *Resolver* do Algoritmo 3.1). Outros detalhes sobre o algoritmo podem ser encontrados em Chien et al. (1999).

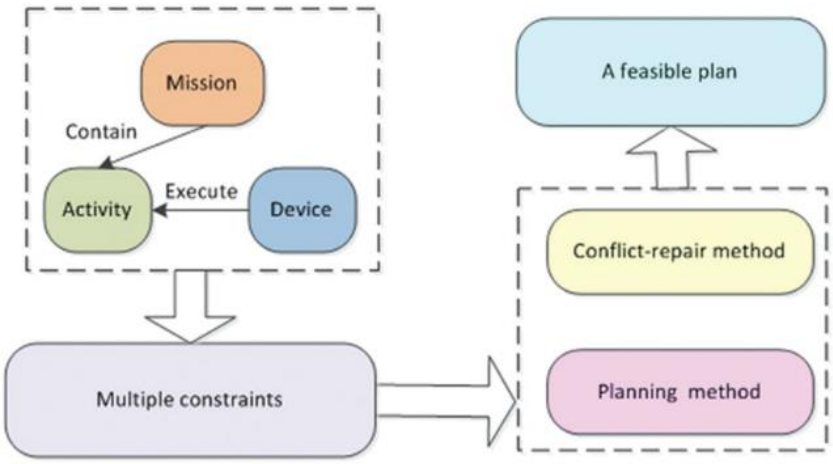
Algoritmo 4.2 – Algoritmo Reparo Iterativo

```
1  Função ResolvaConflitos (max_iteracoes)
2  Início
3    iteracoes = 1;
4      repita (conflitos != Ø  e iteracoes <  max_iteracoes)
5        selecione um conflito;
6        selecione um método (mov, add ou rem) para resolver o conflito
7        case 'mover'
8          selecione um 'culpado' para mover;
9          selecione tempo para mover um culpado;
10       case 'adicionar'
11         selecione uma nova atividade para adicionar;
12         selecione o tempo de início da nova atividade;
13       case 'remover'
14         selecione uma atividade para deletar;
15     se não houve progresso então
16       desfaça a última ação;
17     iterações ← iterações + 1;
18 Fim
```

Fonte: Adaptado de Chien et al. (1999).

É importante frisar que a técnica de reparo iterativo usada no CASPER se baseia em um planejador CSP de busca local. A Figura 4.3 traz uma visão geral (em alto nível de abstração) dos componentes envolvidos nesse paradigma.

Figura 4.3 – Visão geral da abordagem de reparo iterativo e os componentes envolvidos.



Fonte: Bu et al. (2016).

Essa técnica depende de métodos de reparo predefinidos escritos por especialistas que são específicos para cada tipo de falha. Isso se aproxima da maior deficiência da técnica voltada a domínios não determinísticos (Seção 4.1.2), no qual é preciso enumerar explicitamente todos os tipos de falhas. O reparo iterativo também não evita modificações inúteis e não permuta todas as combinações possíveis.

4.2.5. EUROPA

O EUROPA é um *framework* da NASA implementado em C++, descendente do HSTS, para representação e solução de problemas de satisfação de restrições com ênfase em redes de restrições temporais (ASCHWANDEN et al., 2006). Seu objetivo é prover a concepção de um sistema de planejamento para gerenciamento de operações em solo que fosse facilmente integrável a outros sistemas e encapsulasse toda a lógica de planejamento e escalonamento.

O algoritmo principal é o de busca em profundidade (*depth-first*), sem uso de retrocesso[28], para resolução de falhas, incluindo etapas de propagação de restrições

[28] O algoritmo do EUROPA-2 permite pontos de retrocesso.

temporais usando STN. As operações de refinamento são intercaladas com a propagação de restrições na rede temporal do plano parcial.

Este planejador temporal provê interfaces para que um cliente possa modificar um plano, e consultá-lo quanto à sua viabilidade, consistência, decisões abertas, entre outros aspectos (FRANK; JONSSON, 2003). Ele foi estendido em uma segunda versão (BARREIRO et al., 2012) e aplicado a vários domínios, incluindo a operação em solo de missões interplanetárias e de constelação de satélites. Foi referência para construção de planejadores embarcados, como da arquitetura IDEA descrita adiante.

4.2.6. APSI-TRF

O APSI resultou na definição de um *framework* denominado *Timeline-based Representation Framework* (TRF). Trata-se de um *framework* que provê um arcabouço teórico necessário para construção de sistemas de planejamento e escalonamento que operam sobre *timelines*. O APSI é uma instância prática do TRF que utiliza a linguagem DDL para suportar sistemas de planejamento da ESA. Ao contrário de outros sistemas, não existe uma noção explícita de 'ação', a representação é baseada apenas no paradigma de *timelines*.

4.2.7. RASSO e LetMeDo

O planejador denominado *Resources Allocation Service for Scientific Opportunitites* (RASSO – KUCINSKIS; FERREIRA, 2007) do INPE seguiu a linha proposta pelo RAX e CASPER, fazendo uso de técnicas de Planejamento e Escalonamento da área de IA mais especificamente CSP, para permitir o replanejamento a bordo. O RASSO visava a realocação de recursos, especificamente, energia e memória, para fornecer maior tempo de execução ao experimento cujo estudo de caso estava voltado ao domínio de satélites científicos.

O planejador embarcado *LetMeDo* foi desenvolvido como parte da arquitetura GOESA do INPE. Ele tem como objetivo ser de propósito geral e sem empregar heurísticas de busca, pois não se propõe a ter desempenho otimizado. O algoritmo para resolução do

CSP é da classe de busca local, mas pode ser configurado também como de busca aleatória. Apresenta técnica similar à do reparo iterativo do CASPER. Para escapar de platôs e mínimos locais são adicionadas estratégias de perturbações ao plano.

4.2.8. Demais planejadores

Demais pesquisas para controlar satélites a partir de um planejador embarcado foram realizadas, porém sem considerar um ambiente computacional compatível com o da área espacial. Em Amigoni et al. (2010) e Rui et al. (2005), investiga-se planejamento distribuído por meio de uma arquitetura multiagentes.

Amigoni et al. (2010) tratam a HTN como uma STN e depois procedem com algoritmos CSP para alocação de recursos. Particularmente em Indra et al. (2008), concentra-se no formalismo de Redes Petri para modelar diferentes abstrações da plataforma espacial, segundo diferentes horizontes temporais. Em Jiang e Xu, (2017) apresenta-se uma técnica interessante para converter automaticamente o modelo PDDL na forma de restrições de tabela a partir de um planejador baseado em CSP.

Alguns trabalhos (HE et al., 2016; AXMANN, 2010; PENG et al., 2018; SHE et al., 2018) estudaram problemas voltados ao escalonamento de imagens em satélites de sensoriamento remoto com autonomia a bordo. Eles propõem métodos para otimizar ou resolver o problema de escalonamento de imagens e, em alguns casos, considerando-se recursos do satélite. Entretanto, o propósito dessas pesquisas não é produzir uma sequência de ações ao computador de bordo, distanciando-se de uma solução de Planejamento em IA. Os experimentos simulados não utilizam ambientes computacionais da área espacial e raramente consideram a estrita restrição de tempo real.

4.2.9. Análise comparativa

As Tabelas 4.2 e 4.3 apresentam um quadro comparativo das características essenciais dos planejadores visando delinear o estado da arte. Embora os planejadores ASPEN, EUROPA e APSI sejam voltados originalmente para gerenciar operações de solo, eles

tiveram versões adaptadas para serem embarcadas em aplicações espaciais. Foram consideradas nessa análise as características de suas versões embarcadas à exceção do ASPEN.

Por meio da Tabela 4.2, nota-se que os planejadores que consideraram o ambiente real de operação de um satélite ficaram restritos a técnicas voltadas a ambiente determinístico. Elas se mostraram viáveis para execução a bordo de computadores espaciais, que possuem severas limitações de processamento e memória.

Tabela 4.2 – Abordagens e estratégias de planejamento no domínio espacial.

Nome	Tipo de Planejamento	Estratégia	Algoritmo	Linguagem
RAX-PS	Temporal (CSP)	Planejamento em Lote (*Batch*)	Busca em Profundidade	PDDL
ASPEN	Temporal (CSP)	Reparo Iterativo	Busca Construtiva e Local	AML
CASPER	Temporal (CSP)	Reparo Iterativo	Busca Local	Baseada na AML
EUROPA	Temporal (CSP)	Reparo Iterativo	Busca em Profundidade	NDDL
APSI	Temporal (CSP)	Reparo Iterativo	Não Discutido	DDL
LetMeDo	Temporal (CSP)	Similar ao Reparo Iterativo	Busca Local	ISIS

Fonte: Produção do autor.

A maioria utiliza *timelines* para o gerenciamento de recursos aliado à álgebra de Allen para manipular restrições temporais, conforme as Tabelas 4.2 e 4.3. Observa-se pela Tabela 4.3, que os modelos embarcados não são geralmente unificados (*e.g.*, um modelo distinto para planejamento, diagnose e execução).

Tabela 4.3 – Comparativo dos planejadores embarcados da área espacial.

Planejador	RT	PL	HB	TL	CR	MPU	PSI	STN	POP	MD
RAX-PS	IA	CE	Sim	Sim	Não	Não	Sim	Sim	Sim	Sim
ASPEN	IA	CE	Sim	Sim	Não	Sim	Sim	Não	Sim	Sim
CASPER	IA	CE	Sim	Sim	Sim	Não	Sim	Não	Sim	Sim
EUROPA	IA	DI	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
APSI-TRF	IA	DI	Sim	Sim	Sim	Sim	Sim	ND	Sim	Sim
LetMeDo	TP	CE	Não	Sim	Sim	Sim	Sim	Não	Não	Sim

RT = Representação Temporal, PL = Planejamento, HB = Heurísticas de Buscas, TL = *Timelines,* CR = Capacidade de Replanejamento, MPU = Modelo de Planejamento Unificado, IA = Intervalo de Allen, TP = *Time Points,* CE = Centralizado, DI = Distribuído; PSI = Planejamento e Escalonamento Integrado; MD = Modelo Determinístico; ND = Não Discutido.

Fonte: Produção do autor.

Em virtude da severa limitação computacional de processadores qualificados, os algoritmos de planejamento têm empregado soluções incompletas e de menor custo computacional em detrimento a soluções ótimas. A maioria deles implementou a busca no 'espaço de planos' do planejamento clássico.

Embora as linguagens desenvolvidas sejam independentes de domínio, os trabalhos indicam que as aplicações embarcadas têm informações específicas do domínio (*e.g.,* no algoritmo de busca, na construção de heurísticas, na resolução de falhas, etc.). Isso faz com que as aplicações se classifiquem como sistemas de planejamento dependentes de domínio.

4.3. Estado da arte em sistemas embarcados com planejamento automatizado

Alguns planejadores de bordo desenvolvidos (*e.g.,* CASPER, RAX-PS, EUROPA, APSI) foram incorporados em um software de controle de sistemas espaciais (*e.g.*, BERNARD et al., 1998; CHIEN et al., 2005; CEBALLOS et al., 2011) que possuem outros componentes necessários para a solução final de planejamento.

Descrevem-se a seguir os principais sistemas[29] criados por NASA, ESA, *Deutsches Zentrum für Luft-und Raumfahrt* (DLR) e INPE. São abordados aqueles estudos que buscaram produzir ações de forma automatizada ao computador embarcado, visando controlar autonomamente o veículo espacial.

O RAX (BERNARD et al., 1998; RAJAN et al., 2000) foi utilizado para o planejamento experimental automatizado da sonda DS-1, quando ela estava a 65 milhões de milhas da Terra. Este experimento pioneiro[30] alcançou sucesso na aplicabilidade do planejamento embarcado em ciclo fechado. O software é constituído basicamente por três módulos: Planejador e Escalonador (intitulado RAX-PS), Executivo e Identificação de Modos e Recuperação. O RAX mantém um modelo distinto para planejamento, execução e diagnose.

O ASE (CHIEN et al., 2005; SHERWOOD et al., 2004), descrito no início desta Tese, foi utilizado para controlar autonomamente o satélite EO-1. Ele se apresenta como uma arquitetura de três camadas, consistindo de um planejador, um executivo e uma camada de monitoração e controle do sistema. Seu principal avanço em relação ao antecessor RAX é propiciar o planejamento contínuo, que permite replanejar as operações em tempo de execução a partir da abordagem de reparo iterativo do CASPER.

A GOESA (KUCINSKIS, 2012) visa habilitar a operação orientada a objetivos. Tem como propósito ser uma arquitetura aplicável a outras plataformas espaciais, e integrável a um software de voo. Ela tem como cerne um motor de inferência de estados denominado *Internal State Inference Service* (ISIS). A GOESA opera sobre uma

[29] No contexto deste trabalho, os termos "sistema" ou "software" de planejamento são por vezes tratados como soluções de planejamento, a fim de evitar repetições excessivas. Já o termo "arquitetura" se refere a uma solução em nível arquitetural de um sistema de planejamento.

[30] É considerado um trabalho de referência na área de autonomia espacial, pois se tornou o primeiro software a realizar uma demonstração de planejamento embarcado com operação espacial orientada a objetivos. É citado em diversos artigos e pesquisas na área como uma aplicação prática de alta relevância.

linguagem de representação de domínio própria (ISISml) baseada em timelines que unifica planejamento e escalonamento.

O *Intelligent Payload EXperiment* (IPEX – ALTINOK et al., 2015) foi desenvolvido para validação de tecnologias de autonomia, tais como processamento de dados a bordo e operações autônomas. Um dos objetivos do experimento foi validação do CASPER em outros cenários de operação. O IPEX está inserido em uma missão CubeSat de sensoriamento remoto lançada em 2013 e operada por treze meses. O CASPER original foi customizado para integrar o software de voo deste *CubeSat.*

Os três projetos descritos a seguir encontram-se no escopo de missões robóticas de exploração planetária.

A IDEA (MUSCETTOLA et al., 2002; DIAS et al., 2003) continuou os estudos do RAX e propôs uma arquitetura multiagente, onde cada agente de planejamento atua em um nível de abstração do sistema, de forma a intercalar planejamento e execução. O objetivo segundo os autores é prover maior abstração e mais capacidade de resposta da aplicação. O principal avanço da IDEA em relação ao RAX e ASE é propor um modelo unificado de planejamento e execução. Os planejadores embutidos da IDEA são baseados no EUROPA da NASA.

O GOAC (CEBALLOS et al., 2011) reúne esforços de diferentes instituições contratadas pela ESA-ESTEC (*European Space Research and Technology Centre*). O estudo propõe uma arquitetura multiagente similar à proposta da IDEA. Seu raciocínio é orientado por modelos baseados estritamente em timelines. Ele possui um conjunto de variáveis de estados e descreve as transições permitidas, bem como uma série de regras de sincronização. O sistema de planejamento utilizado no GOAC segue a abordagem de 'dividir para conquistar' (CEBALLOS et al., 2011, p. 5) e é centrado no *framework* APSI da ESA.

A arquitetura *Coupled Layer Architecture for Robotic Autonomy* (CLARAty - ESTLIN et al., 2003; NESNAS et al., 2003) se diferencia ao definir apenas duas camadas: decisão e funcional. Segundo Volpe et al. (2001), estabelecer três camadas implica manter

diferentes modelos comportamentais do robô de forma redundante, além do controle e/ou representação descentralizado do sistema.

Tanto GOAC como IDEA foram submetidas a estudos de casos para fins de prova de conceito a partir de um ambiente computacional de simulação de uma superfície rochosa, destino final dos *rovers* para os quais as arquiteturas foram desenvolvidas. Não foram encontrados dados de desempenho nos trabalhos pesquisados.

Há outros trabalhos de robótica não descritos aqui por apresentar menor relevância com o escopo deste trabalho. À exceção das limitações de comunicação e do ambiente de operação (*e.g.*, superfície de Marte), o domínio de *rovers* da área espacial se aproxima muito a da área de robótica - uma temática bastante explorada na comunidade de planejamento.

Outras abordagens (WOODS et al., 2006; LENZEN et al., 2014) apresentam técnicas que permitem o reparo do plano a bordo, mas sem dispor de um planejador embarcado, distanciando-se da finalidade central proposta nesta Tese.

Dentre elas, destacam-se o (i) *Verification of Autonomous Mission Planning On-board Spacecraft* (VAMOS - LENZEN et al., 2014) que combina a capacidade de replanejamento em tempo real com auxílio de sistemas de planejamento em solo que têm mais poder computacional; e (ii) o *Validation Control and Repair* (TVCR - WOOD et al., 2006) composto por um validador de plano, um monitor de execução e um gerador de reparo de planos. Sua principal limitação, segundo Bozzano (2011), é realizar quantidade limitada de replanejamento e não ser de uso geral.

Excluindo-se os trabalhos mais recentes, uma descrição mais detalhada sobre os projetos embarcados pode ser vista no trabalho de Kucinskis (2012). A presente Tese deu continuidade à pesquisa em sistemas de planejamento do domínio espacial, fazendo um profundo levantamento de seus principais atributos, que estão sumarizados a seguir.

4.4. Análise comparativa dos trabalhos

As Tabelas 4.4 e 4.5 mostram um quadro comparativo com atributos considerados relevantes acerca dos sistemas de planejamento. Tais tabelas ajudam a delinear o estado da arte e entender os possíveis pontos em aberto. Elas foram divididas em duas tabelas reunindo características essenciais e gerais respectivamente. Alguns dos acrônimos presentes nas tabelas estão definidos na lista de siglas deste documento.

Tabela 4.4 – Sumário comparativo das características essenciais dos sistemas de planejamento.

Nome	DA	OO	EP	MU	CO	PL
RAX (1999)	Científica (Espaço Profundo)	Sim	Planejador Temporal (CSP)	Modelos Distintos	PS, EXEC e *Livingstone*	Único
IDEA (2002)	Sondas Espaciais (Robótica)	Não	Planejador Temporal (CSP)	Modelo Unificado	Multi-agentes	Vários Planejadores
ASE (2003)	Observação da Terra	Sim	Planejador Temporal (CSP)	Modelos Distintos	CASPER e SCL	Planejador Único
TVCR (2006)	Sondas Espaciais (Robótica)	Não	Serviço de Replanejamento (*Timelines*)	-	Validador, Reparador e Monitor	-
RASSO (2007)	Científica (Órbita baixa)	Não	Planejador Temporal (CSP)	-	RASSO	Planejador Único
GOAC (2011)	Sondas Espaciais (Robótica)	Sim	Planejador Temporal (CSP)	Modelo Unificado	Multi-agentes	Vários Planejadores
GOESA (2012)	Observação da Terra	Sim	Planejador Temporal (CSP)	Modelo Unificado	ISIS Letmedo	Planejador Único
IPEX (2013)	Observação da Terra	Sim	Planejador Temporal (CSP)	Modelos Distintos	DSQEE CASPER	Planejador Único
VAMOS (2014)	Observação da Terra	Não	Replanejamento (*Timelines*)	-	OBETTE OBoTis	-
HARPIA (2020)	Observação da Terra	Sim	Planejador Hierárquico (HTN)	Modelo Unificado	*Liger* Gerenciador de Atividaes	Planejador Único

DA = Domínio de Aplicação; OO = Operação baseada em Objetivos; EP = Estratégia de Planejamento; Modelo Unificado = Modelo embarcado Unificado; PL = Planejador.

Fonte: Produção do autor.

Tabela 4.5 – Sumário comparativo das características gerais dos sistemas de planejamento.

Nome	HQ	IE	EI	PS	TL	RF	AR	OC
RAX (1999)	Sim RAD6000 *PowerPC*	Sim	Sim (ESL)	Sim	Sim	Sim (*Livingstone)*	Não	Planejamento, Controle e Execução
IDEA (2002)	Não	Sim	Sim	Sim	Sim	Não	Sim	Planejamento e Execução
ASE (2003)	Sim *Mongoose V* - R3000	Sim	Sim (SCL)	Sim	Sim	Não	Não	Planejamento, Controle e Execução
TVCR (2006)	Não	Sim	Não	Não	Sim	Não	Não	Não
RASSO (2007)	Sim ERC32	Sim	Não	Sim	Sim	Não	Não	Não
GOAC (2011)	Não	Sim	Não	Sim	Sim	Não	Sim	Planejamento e Execução
GOESA (2012)	Sim ERC32	Sim	Sim	Sim	Sim	Não	Sim	Não
IPEX (2013)	COTS Atmel ARM9	Sim	Não	Sim	Sim	Não	Não	Planejamento, Controle e Execução
VAMOS (2014)	COTS *PowerPC*	Parcial	Não	Não	Sim	Não	Sim	Não
HARPIA (2020)	Sim ERC32	Sim	Não	Sim	Não	Não	Não	Deliberativa, Reativa e Aplicação

HQ = Hardware Qualificado para uso espacial; IE = Inferência de Estados; EI = Executor Inteligente; PS= Planejamento e Escalonamento integrado; TL = abordagem por *TimeLines;* RF = Recuperação autônoma de Falhas; AR = arquitetura reutilizável; OC = Organização em Camadas.

Fonte: Produção do autor.

5 HARPIA: UMA ARQUITETURA HIEÁRQUICA DE PLANEJAMENTO EMBARCADO BASEADA EM REDE DE TAREFAS PARA AUMENTAR A AUTONOMIA OPERACIONAL DE SATÉLITES

Este Capítulo traz uma descrição geral da arquitetura de planejamento embarcado proposta nesta Tese, descrevendo-se o propósito de cada componente e suas formas de interação cujo objetivo final é produzir novos planos de missão em tempo real, sem depender da intervenção direta da equipe de solo. Ela é considerada uma extensão da arquitetura GOESA do INPE.

A principal diferença entre elas é o foco. A GOESA é uma arquitetura de uso geral cujo componente cerne é o motor de inferência de estados voltado a planejadores baseados em restrições de estados (CSP). Ela propõe componentes gerais que podem ser estendidos por outros sistemas que visam à operação baseada em objetivos.

A arquitetura desta Tese incorpora ideias da GOESA, estendendo-a para uma organização em camadas com níveis hierárquicos, aplicando-se outro paradigma de planejamento (HTN). O objetivo é definir os elementos que compõem a solução híbrida de planejamento proposta nesta Tese. Sua implementação visa obter uma conexão mais forte com a pesquisa aplicada à engenharia espacial.

5.1. Considerações da pesquisa

A partir do capítulo anterior foi possível compreender as características fundamentais dos sistemas embarcados com autonomia desenvolvidos pela comunidade da área espacial. Foram constatados os pontos fortes, assim como as lições aprendidas a partir de experiências negativas de certas soluções. A seguir, colocam-se os principais aspectos considerados para a concepção da abordagem de planejamento desta Tese.

Representação do conhecimento: diferenças na semântica de diferentes modelos para representar um mesmo sistema de planejamento levam a erros de difícil detecção. Cada componente pode fazer uma percepção diferente sobre o estado do satélite, o que dificulta a consistência e o sincronismo das atividades em operação. A propensão

de utilizar vários modelos certamente é herdada da organização em camadas. Ao estabelecer um modelo único, o sistema deve representar diferentes níveis de abstração e, principalmente, saber como raciocinar sobre elas.

Organização: conforme as Tabelas 4.4 e 4.5, sobre a disposição dos componentes no software de controle, certas arquiteturas adotadas pelos sistemas de planejamento seguiram o legado de três camadas, outras foram organizadas apenas em módulos ou serviços, enquanto as demais adotaram a abordagem multiagentes. Ressalva-se que a tecnologia atual para implementação de agentes não é diretamente aplicável ao ambiente computacional da área espacial.

Planejador centralizado: a maioria dos sistemas possui um planejador centralizado, o que traz uma solução mais inteligível. Já o IDEA e GOAC propuseram uma abordagem multiagentes visando obter níveis diferentes de atuação (deliberação/execução) do planejador. Isso traz grande esforço para a coordenação dos objetivos, a fim de evitar conflitos entre planos de outros agentes. É considerada uma solução mais complexa, sendo indicada para ambientes cujos agentes estão fisicamente distantes, como a operação coordenada de uma constelação de satélites autônomos.

Planejamento & Escalonamento: a maioria das soluções (vide Tabela 4.5) unificou as etapas de planejamento e escalonamento. A ideia de integrá-las consiste em evitar interações sucessivas entre planejador e escalonador na busca por um plano, o que pode levar à degradação do desempenho, conforme visto na Seção 3.5.2.

Ambiente computacional: alguns sistemas (*e.g.*, IDEA, GOAC e TVCR) que empregaram inicialmente soluções computacionais tipicamente comerciais acabaram não realizando experimentos em processadores espaciais. Isso realça a importância de se considerar as limitações computacionais ainda nas fases iniciais do estudo.

Representação para ambientes determinísticos: os sistemas usaram representações deterministas de planejamento visando, sobretudo, melhor desempenho e menor esforço para descrição do domínio – propriedades relevantes para aplicações práticas.

Desempenho: existe uma constante preocupação acerca do desempenho. Algumas abordagens tentaram melhorar o desempenho utilizando o paradigma multiagentes (*e.g.*, IDEA e GOAC) e outras empregaram heurísticas de busca (*e.g.,* CASPER), a fim de aumentar a velocidade do algoritmo.

Com base nos aspectos considerados, sumarizam-se as seguintes motivações do estado da arte: integrar planejamento e escalonamento, dispor de um modelo único para planejamento e execução, alcançar planejamento em tempo real com capaciade de revisão de planos e estabelecer um modelo embarcado com níveis de abstração de operação.

Dado o cenário acima, a arquitetura de planejamento proposta nesta Tese se diferencia de abordagens similares pela sua organização em camadas que provê níveis hierárquicos de abstração do domínio a partir de uma solução híbrida, que permite unificar planejamento e replanejamento a partir de um motor de busca baseado em HTN que realiza inferência de estados e recursos.

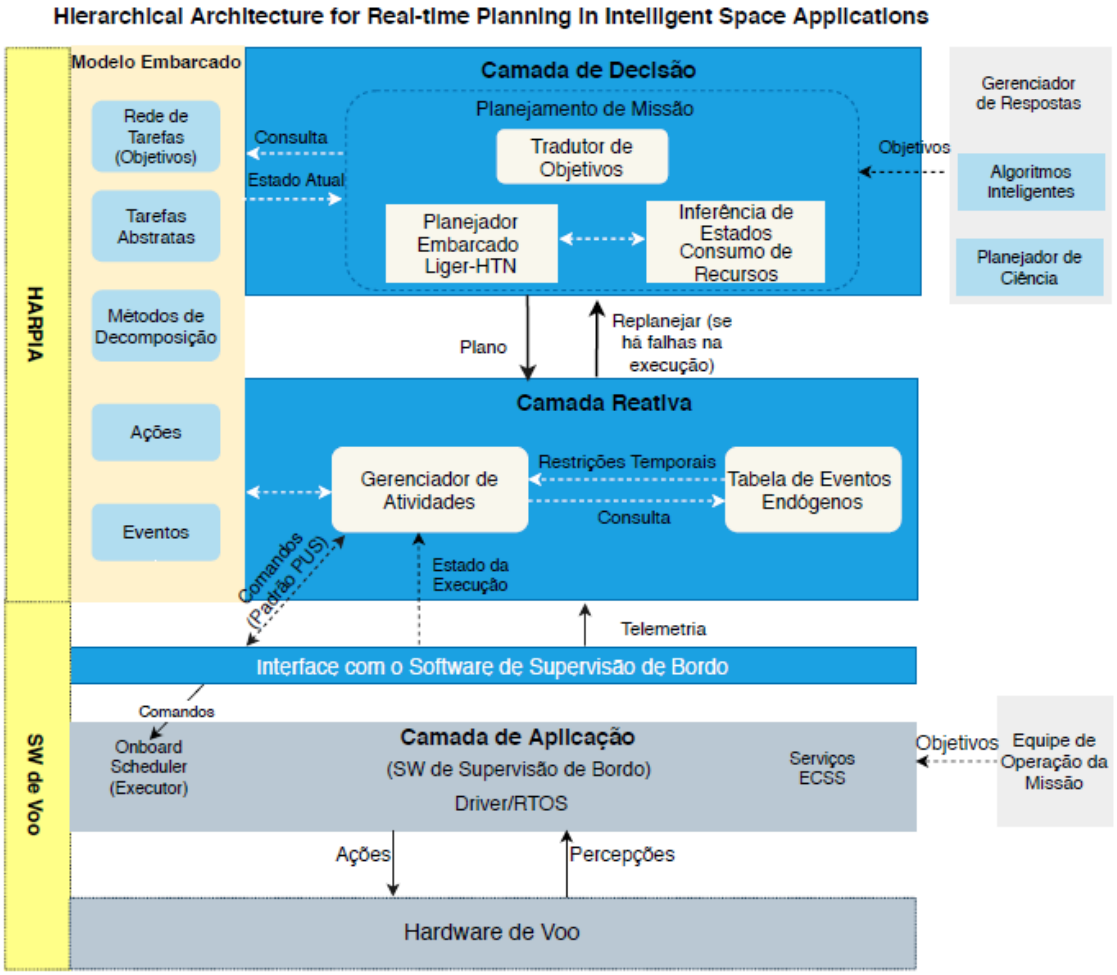
Na próxima seção apresenta-se uma visão geral da ***H****ierarchical* ***A****rchitecture for* ***R****eal-time* ***P****lanning in* ***I****ntelligent Space* ***A****pplications* (HARPIA), abordando suas características elementares e seu princípio básico de funcionamento. Ela se baseia em modelos comportamentais e interfaces que podem facilitar sua reutilização em outras plataformas espaciais.

5.2. Hierarchical architecture for real-time planning in intelligent space applications

A arquitetura de software embarcado deste trabalho visa transferir para bordo algum grau de inteligência computacional capaz de aumentar a autonomia operacional de satélites. Suas características gerais incluem planejamento em rede de tarefas hierárquicas, inferência de estados que unifica planejamento e escalonamento e uma nova forma de representação de conhecimento a bordo, baseada em uma extensão de uma linguagem do INPE desenvolvida no trabalho de Kucinskis (2012).

À arquitetura representada pela Figura 5.1, deu-se o nome de HARPIA em homenagem a uma personagem da mitologia grega[31]. Ela é organizada na forma de camadas, visando maior modularidade e simplificação. O cerne da arquitetura é o planejador embarcado e os elementos que compõem a solução de planejamento proposta nesta Tese.

Figura 5.1 – Uma arquitetura hierárquica de planejamento embarcado baseada em rede de tarefas para aumentar a capacidade de resposta de satélites.



Fonte: Produção do autor.

[31] Trata-se de uma personagem híbrida da mitologia grega, formada por metade mulher e metade ave de rapina. No entanto, frisa-se que essa alusão à personagem híbrida está relacionada com a solução de planejamento proposta na presente Tese e não à arquitetura em si.

A HARPIA é organizada pelas camadas decisão e reativa e uma terceira denominada 'aplicação[32]' que representa o próprio software de supervisão de bordo. Cada camada tem responsabilidades específicas, sendo formada por componentes, que realizam um ciclo 'Sentir-Pensar-Agir' similar ao conceito de agentes inteligentes.

- **Sentir**: envolve a obtenção de dados de camadas inferiores e o mapeamento dessas informações em uma representação utilizável pelo software embarcado.
- **Pensar*:*** envolve a obtenção dos dados dos sensores e informações da plataforma espacial para definição do que deve ser feito para cumprir os objetivos desejados.
- **Agir**: refere-se à capacidade de tomada de decisão do componente embarcado e produz resultados que serão encaminhados a outros elementos da arquitetura.

Conforme Zheltoukhov e Stankevich (2017) há duas classificações de arquiteturas de controle autônomo: horizontal ou vertical. Na primeira cada camada acessa os dados do meio físico diretamente, enquanto na segunda o fluxo de informação precisa respeitar a hierarquia entre as camadas adjacentes, de modo que somente a camada inferior tenha acesso aos dados do ambiente físico.

A organização da HARPIA apresenta uma forma diferenciada da classificação vertical, pois de qualquer camada é possível obter informações atualizadas do estado do satélite usando a telemetria em tempo real. Todavia, no que se refere à troca de informação entre os componentes, a hierarquia e o nível de abstração entre as camadas são preservados.

[32] Um software de voo tradicional fornece as funções necessárias para as operações nominais e de contingência do satélite. Dentre outras responsabilidades, ele gerencia os modos do satélite, os subsistemas, a interface com solo, controle térmico, a saúde da plataforma e a execução dos comandos a bordo.

Isso contorna uma das limitações da decomposição puramente vertical quanto à separação da aquisição e uso de informações, de acordo com Zheltoukhov e Stankevich (2017).

Outra vantagem dessa organização em camadas é que ela permite aos especialistas em IA se concentrarem nas camadas superiores que envolvem as atividades de planejamento e escalonamento, deixando os detalhes da iteração com o hardware da camada de aplicação, para a equipe de supervisão de bordo.

Ademais, a estrutura com hierarquia da HARPIA propicia a integração dos componentes em IA a um software de voo convencional cuja organização também é hierárquica, conforme Tipaldi et al. (2018).

5.3. Camadas e componentes

A seguir apresentam-se os elementos centrais da arquitetura. Cada componente possui uma estrutura comum composta pelas capacidades de percepção, ação e habilidades específicas. Apesar de uma estrutura elementar, o funcionamento interno dos componentes é diferente, o que confere aos elementos da arquitetura um comportamento específico. Os componentes são estratificados para explorar as funcionalidades de camadas adjacentes.

Definiram-se duas camadas na HARPIA: a camada de decisão e reativa, as quais diferem em termos de nível de abstração de informação, tempo de respostas a estímulos do ambiente. As duas devem trabalhar de forma integrada e intercalada no tempo, raciocinando sobre um modelo embarcado unificado que atendam às necessidades de planejamento em tempo real proposto pela solução híbrida.

5.4. A camada de decisão

A Camada de Decisão é considerada uma camada deliberativa. É nela que o problema de planejamento e escalonamento é tratado e onde novos objetivos podem ser identificados a partir da tomada de decisão dos algoritmos inteligentes cujo propósito é analisar os dados da carga útil em tempo real.

A programação de atividades do satélite é realizada nessa camada, considerando o estado atualizado da plataforma e os objetivos recebidos dos usuários - sejam eles enviados pela equipe de solo ou detectados em bordo. A camada de decisão é constituída pelo Planejador de Missão e pelo Gerenciador de Respostas à Ciência, descritos a seguir.

5.4.1. Planejamento de missão

Este componente fornece o serviço de planejamento construindo planos de ações para alcançar os objetivos. Partindo de um estado inicial e objetivos impostos, ele é capaz de construir um plano de ações para um determinado problema a partir de suas percepções do ambiente, executar estas ações e avaliar a satisfação do objetivo.

Como dito antes, o planejador é o elemento central da HARPIA. Ele habilita a operação autônoma do satélite, permitindo que comandos sejam gerados a bordo sem a intervenção de solo. O planejamento de missão é constituído por dois módulos o 'Tradutor de Objetivos' e pelo 'Planejador Embarcado', conforme a Figura 5.1.

5.4.1.1. Tradutor de objetivos de missão

O 'Tradutor de Objetivos' recebe os objetivos tanto do segmento solo, como do 'Gerenciador de Respostas à Ciência'. É responsável por mapear os objetivos recebidos em tarefas de alto nível que são tangíveis ao planejador embarcado a partir da consulta ao modelo embarcado. Ele deve traduzir um objetivo na forma de tarefas hierárquicas que o sistema deve executar a partir da consulta à representação do conhecimento em HTN da HARPIA.

Os objetivos podem ser transformados em tarefas, como: '*obter uma imagem de um alvo de interesse*', ou *'fazer reprodução (Playback) das imagens gravadas em bordo'*. Uma tarefa pode ser desdobrada em várias ações que levem o satélite ao cumprimento do objetivo recebido.

É prevista a resolução de conflitos entre os objetivos identificados em órbita e aqueles enviados pelo segmento solo, considerando-se suas prioridades. A HARPIA pode eleger

níveis de prioridade dos objetivos, como 'nominal' ou 'prioritário'. Um algoritmo que não foi implementado neste trabalho poderia explorar a priorização de tarefas, a fim de que novas oportunidades possam ser integradas ao plano, seja removendo ou postergando tarefas de menor prioridade.

Essa é uma característica positiva da representação em HTN. Um novo objetivo é inserido como uma nova atividade do plano que será mapeada por uma rede de tarefas. Nesse caso, há possibilidade de certo objetivo falhar, mas outros permanecerem válidos.

O tradutor de objetivos da HARPIA também é responsável por vincular as informações temporais dos objetivos às subtarefas de maior nível hierárquico antes de adicioná-las na agenda de tarefas. Uma vez feito isso, os tempos são propagados às tarefas filhas de maneira *'top-down'* à medida que o algoritmo de planejamento é executado. A depender da missão, outros parâmetros inerentes aos objetivos podem ser gerenciados por este componente antes de encaminhar o problema ao planejador.

5.4.1.2. Planejador embarcado

A diferença básica entre um executor de comandos de um software de voo convencional e um planejador embarcado é que o primeiro lida diretamente com comandos a partir do plano de operação realizado por solo, ao passo que o segundo trata os objetivos, de forma a encontrar a sequência de comandos que leve o satélite ao cumprimento dos objetivos impostos.

De forma sintética, o planejador embarcado proposto recebe os objetivos de alto nível e o estado atualizado da plataforma espacial a partir da consulta a um motor de inferência de estados, que gerencia o modelo embarcado – que, por sua vez, descreve o comportamento esperado da plataforma. Antes de iniciar a busca pelo plano, o processo de inferência de estados utiliza as telemetrias do satélite para atualizar as variáveis de estados do modelo embarcado. Com a sequência de tarefas encontrada pelo planejador, as encaminha à camada reativa para monitoração e controle. Em seguida, o planejador aguarda um 'status' da execução do plano da camada reativa.

A escolha do processo de planejamento embarcado é uma tarefa relevante devido à limitação de capacidade computacional de um satélite. Na HARPIA, adotou-se um planejador baseado em rede de tarefas hierárquicas em detrimento aos planejadores temporais predominantes na área espacial.

O planejamento com técnicas de redução de tarefas trata a viabilidade da solução ao problema de forma incremental, sendo adequado para problemas de aplicações reais devido ao seu princípio de decomposição de tarefas hierárquicas e à sua expressividade para representação do conhecimento.

Durante a execução do algoritmo, serão verificadas as precondições entre as tarefas que compõem a rede. Sempre que uma ação durativa tiver restrições de recursos, serão consultados os estados futuros do satélite a partir de um mecanismo de inferência de estados.

Um plano estará válido se for executável a partir do estado inicial e alcançar a lista de tarefas que é fornecida como entrada no problema, usando os métodos definidos como parte do domínio. Se ao final restarem somente tarefas primitivas que alteram o estado do satélite caracteriza-se o sucesso, do contrário, configura-se uma falha do processo de planejamento.

O objetivo é usar o mesmo algoritmo de planejamento em todas as camadas da HARPIA. Isso elimina a necessidade de mecanismos de sincronização entre os diferentes planejadores, permitindo o mesmo nível de raciocínio.

Outra característica importante é que o planejador da HARPIA não faz distinção se uma solicitação se refere a um novo objetivo ou a um pedido de replanejamento decorrente de uma falha percebida na camada reativa.

Se os métodos de decomposição falharem, o algoritmo aplica uma técnica de replanejamento. Ela pode ser usada tanto para o cenário em que o satélite não esteja em uma configuração inicial prevista, como nos casos de desvios durante a execução.

A ideia é utilizar uma abordagem determinista de planejamento aliada a uma técnica de reparo de estados que permite lidar com o não determinismo ilimitado (desde que exista uma solução para o problema considerando a técnica de reparo utilizada e o domínio construído) do ambiente. Isso propicia à arquitetura uma integração mais sutil entre as etapas de decisão e execução.

5.4.2. Gerenciador de respostas à ciência (dados da carga útil)

O Gerenciador de Respostas à Ciência tem a capacidade de observar novos objetivos a partir da análise dos dados da carga útil em tempo real. Ele é responsável por fazer a operação do veículo espacial ocorrer em ciclo fechado, potencialmente com mínima intervenção da equipe de solo. É composto por 'Algoritmos Inteligentes' e 'Planejadores de Ciência', como ilustrado na Figura 5.1.

Os algoritmos inteligentes buscam eventos de interesse para a missão de forma autônoma. Eles podem ser eventos ambientais, como um incêndio, no caso de uma missão de sensoriamento remoto ou uma rocha de especial interesse em uma missão de exploração planetária *in-situ,* por exemplo.

É importante que os algoritmos embarcados possam ser selecionados e ter seus parâmetros atualizados pelo centro de controle, caso seja necessário, ao longo da missão a partir da validação dos resultados pelos usuários em solo, o que faz aprimorar eventualmente o desempenho dos algoritmos.

Já os Planejadores de Ciência referem-se a subproblemas específicos da carga útil. Em um satélite de sensoriamento remoto pode haver um planejador para gerenciar as observações em andamento, resolvendo conflitos de janelas, períodos de eclipse, e considerando as prioridades de cada solicitação, por exemplo.

Tal solução permite dividir o espaço de busca do planejador embarcado central que é genérico, a partir de planejadores de ciência que podem possuir heurísticas específicas, alcançando resultados de melhor desempenho.

Além do potencial de melhorar o tempo de resposta na geração dos planos, os objetivos serão encaminhados ao planejador livres de conflitos. Isso também traz maior usabilidade da arquitetura ao restringir necessidades individuais da carga útil cuja finalidade pode variar conforme a missão.

Deve-se notar que o Planejador de Ciência não tem a finalidade de gerar comandos ao software de controle – função exclusiva do planejador da HARPIA. Essa solução também não implica em planejamento descentralizado, pois não há distribuição de objetivos entre os planejadores.

Embora previsto, este componente não será implementado. Para sua construção, é comum recorrer a um processador adicional devido à alta taxa de dados requerida para o processamento em tempo real. Observe na Figura 5.1 que esse componente não é integrante da HARPIA.

5.4.3.Sumário

A seguir, descrevem-se de forma sumária as principais atividades realizadas na camada de decisão:

1. Obtém-se o estado atual do satélite; as variáveis de estados descritas no domínio são atualizadas pelas telemetrias correspondentes antes de iniciar o processo de planejamento;

2. O objetivo é transformado em uma rede de tarefas após a consulta ao domínio. Neste momento, formula-se um problema de planejamento;

3. O motor de busca do planejador da HARPIA inicia o processo da busca no espaço da rede de tarefas;

4. As precondições e os efeitos das tarefas que alteram modos e recursos dos equipamentos são previstos por meio da antecipação do comportamento do satélite, considerando o modelo embarcado. Os efeitos (por exemplo da ação

ligar a câmera do satélite por 5000 segundos), são inferidos de forma *'offline'* – dado que as telemetrias não realimentam o modelo durante o planejamento;

5. Durante a busca do plano, já se considera a alocação de recursos e tempo às atividades, o que implica na integração de planejamento e escalonamento;

6. Ações que consomem recursos do satélite, como memória do gravador de dados são alocadas em tempo de planejamento. Qualquer problema nessa fase configura uma falha no processo;

7. Caso o satélite não esteja no estado previsto pelos métodos de decomposição, o algoritmo procede com técnicas de reparo no plano;

8. Uma vez que um plano de ações primitivas que alteram estados do veículo espacial é encontrado, este é encaminhado à camada reativa;

9. Em seguida, esta camada fica à espera de um '*status*' de execução. No caso de insucesso, tenta-se um replanejamento a partir do estado atualizado da plataforma;

10. O replanejamento é baseado em refinamentos de estados que são inseridos na busca pelo plano à medida que precondições das tarefas não sejam atendidas.

5.5. A camada reativa

A camada reativa é constituída pelo Gerenciador de Atividades e pela Tabela de Eventos Temporais Endógenos. Ela faz interface entre as camadas de decisão e de aplicação, conforme a Figura 5.1.

Trata-se de uma camada intermediária que lida com tempo de repostas menores e que interage com o software de supervisão de bordo. Ela raciocina sobre o mesmo modelo utilizado pelo planejador, porém aqui não há o conceito de objetivos ou tarefas abstratas.

Os dados de telemetria da camada de aplicação são utilizados como insumo para controlar se os efeitos das tarefas primitivas que compõe o plano foram executados corretamente.

Como a HARPIA tem acesso às percepções do hardware em tempo real a partir da telemetria, entende-se que o sistema de planejamento trabalha de forma *online* com o ambiente, intercalando-se as etapas de planejamento e execução similar à abordagem de planejamento contínuo discutida na Seção 4.2.1.

O Gerenciador de Atividades deve usar suas percepções para descobrir o que acontece enquanto um plano é executado. Ele deve identificar se algo inesperado ocorrer, como o fato de um equipamento não ter sido ligado ou mesmo a constatação de uma falha, considerando os efeitos esperados de cada tarefa.

Destaque-se que o executor de comandos temporizados se encontra na camada de aplicação e não faz parte da camada reativa. Tal solução difere dos sistemas atuais que o mantém junto ao controlador. Isso traz compatibilidade com a proposta de estabelecer níveis de autonomia e garante maior reusabilidade da arquitetura.

5.5.1. Gerenciador de atividades

O Gerenciador de Atividades se comunica com a Tabela de Eventos Temporais Endógenos. Ele faz interface entre planejador e camada de aplicação, a qual mantém o plano de operação por meio de um executor de comandos temporizados.

Uma vez que o Gerenciador de Atividades esteja com um plano de ações recebido da camada de decisão, ele deve consultar a Tabela de Eventos Temporais Endógenos, a fim de incorporar no plano as restrições temporais intrínsecas à operação dos equipamentos.

Isso não implica na segregação do processo de planejamento e escalonamento, visto que o plano recebido já vem com as ações ordenadas temporalmente e tem alocado a elas os recursos necessários para sua consecução.

Trata-se de um processo de refinamento, onde certas ações do plano são deslocadas temporalmente em função da ocorrência de eventos endógenos. Isso preserva a característica de uma arquitetura hierárquica, na qual um plano se desdobra em um nível de abstração mais detalhado na medida em que atravessa suas camadas inferiores.

Em seguida, é de sua responsabilidade consolidar as ações dos planos recebidos e seus instantes temporais refinados e verificar se o plano é executável.

O próximo passo do gerenciador de atividades é despachar o plano com as ações primitivas para a camada de aplicação e monitorá-los.

Este componente deve identificar os eventuais conflitos que possam surgir após uma atualização do estado do satélite. Sua finalidade é monitorar as ações do plano, verificando se foram executadas conforme o esperado.

É possível detectar basicamente dois tipos de falhas: (i) precondições inválidas das tarefas e (ii) efeitos inválidos das tarefas.

Na presente Tese esse comportamento será simulado (Seção 8.7.8) com a injeção de falhas de estados do satélite durante a execução do plano no nível de ações primitivas.

Tal simulação é importante uma vez que podem surgir diferenças entre o sistema físico controlado e o comportamento operacional previsto pelo modelo durante a fase de planejamento.

Alguns sistemas de planejamento do domínio espacial adotaram modelos procedurais (uma linguagem intermediária) para controle de execução do plano. Recebem comandos de alto nível do planejador e os traduzem para comandos de baixo nível por meio de um executor baseado em *scripts* e regras de controle.

Isso impõe caminhos de execução do tipo *hardcoding,* segundo Finzi et al. (2004), o que prejudica sua adaptação a novos contextos. Além disso, implica em

representações de conhecimento diferentes, já desencorajada por experiências anteriores.

Na HARPIA, evita-se a adoção de um nível intermediário no processo de execução. As ações primitivas do planejador HTN são mapeadas para comandos que podem ser executados diretamente pelo computador.

Fazendo um paralelo com a forma pela qual o satélite é controlado nas missões atuais, na qual um plano de voo é desdobrado em tarefas operacionais compostas por telecomandos e tempos associados, o plano final da HARPIA também é convertido em comandos temporalmente posicionados, de maneira análoga aos telecomandos programados a bordo.

5.5.2.Tabela de eventos temporais endógenos

Essa tabela mantém os eventos temporais que descrevem efeitos indiretos de comandos que restringem a operação do satélite. Ao comandar que se ligue o transmissor de dados, por exemplo, pode haver um intervalo mínimo para que ele se torne operacional devido ao tempo necessário para o pré-aquecimento da válvula do transmissor.

Na presente arquitetura, a ação de ligar o transmissor produz indiretamente um evento temporal endógeno. Para esse tipo de representação, a HARPIA mantém na sua linguagem a descrição de eventos temporais associados a determinadas ações. A partir disso, constitui-se o que chamamos de Tabela de Eventos Temporais Endógenos.

Para modelar esse tipo de operação, os sistemas de planejamento embarcado do Capítulo 4 empregaram uma modelagem rica em restrições temporais vinculadas a atividades, aproveitando-se da flexibilidade dos planejadores temporais. Nelas, entretanto, eventos temporais advindos dos objetivos são mesclados com restrições operacionais dos equipamentos.

A HARPIA adotou uma modelagem temporal mais moderada na fase deliberativa pelos seguintes motivos:

- Simplificar a atribuição de restrições temporais na camada de decisão, onde o algoritmo é mais complexo, sobretudo, pela natureza hierárquica das tarefas;
- Prover maior flexibilidade ao planejador, visto que os tempos relacionados aos objetivos podem ser eventualmente relaxados, mas as restrições operacionais dos equipamentos não.
- Evitar modelar estados não existentes na operação real do equipamento. No exemplo do transmissor, um estado "*warm_up*" poderia ser modelado para simular uma restrição temporal antes de efetivar a transição de modo.

5.5.3. Sumário

A seguir, descrevem-se de forma sumária as principais atividades realizadas na camada reativa:

1. O gerenciador de atividades recebe o plano de ações já ordenadas e adiciona as restrições temporais dos eventos endógenos;

2. Comparando a predição dos estados do satélite na camada de decisão com o monitoramento da telemetria em tempo real, durante a execução do plano, a camada reativa é responsável por verificar se houve desvios entre o que foi inferido em tempo de planejamento e o estado atual da plataforma;

3. A abstração das ações é de menor nível nessa camada. Há somente tarefas primitivas, suas precondições e efeitos no satélite.

4. Se as precondições e os efeitos das tarefas primitivas ocorrerem conforme esperado, o plano é executado com sucesso;

5. Se durante a execução ocorrer uma falha, a execução do plano é interrompida, e um pedido de replanejamento com estado atualizado do satélite é enviado à camada deliberativa.

5.6. Domínio representado por variáveis de estados

O domínio previsto para HARPIA não é representado de forma explícita[33] como alguns modelos de planejamento. Isso requer grande esforço na fase de modelagem e para os casos em que o ambiente possa assumir infinitos valores de estados, descrevê-los se torna impraticável. Por essa razão, o presente domínio é visto como uma entidade que pode estar em diferentes estados. Isso é mais aderente ao modelo espacial no qual um estado do satélite é descrito por um conjunto de variáveis que recebem valores de domínios finitos cujos valores variam no tempo.

No domínio da HARPIA, especificam-se as ações do satélite que podem ser executadas para alterar o estado atual e suas condições de aplicabilidade. Isso permite que uma solução seja encontrada a partir de uma rede de tarefas inicial, na qual o sistema alcançará o estado objetivo, aplicando-se sucessivas ações decorrentes de métodos de decomposição HTN presentes no domínio. O objetivo consiste em encontrar um refinamento da tarefa inicial, ao invés de satisfazer a descrição de estados de um objetivo. O domínio completo da HARPIA é formado por elementos especificados por uma nova forma de representação a bordo, descrita no próximo Capítulo.

É importante observar que este domínio não se propõe a descrever falhas de ações. Na HARPIA, as falhas de estados relacionadas ao cumprimento dos objetivos são tratadas com técnicas de refinamento e não de forma explícita no modelo pelo especialista. Já as falhas no nível de sistemas do satélite (*e.g.,* velocidade excessiva das rodas de reação, superaquecimento da câmera, baixo nível de carga da bateria, etc.) devem ser tratadas pelo componente de FDIR do OBSW, e não no nível de planejamento. Isso traz maior compatibilidade com um software de voo.

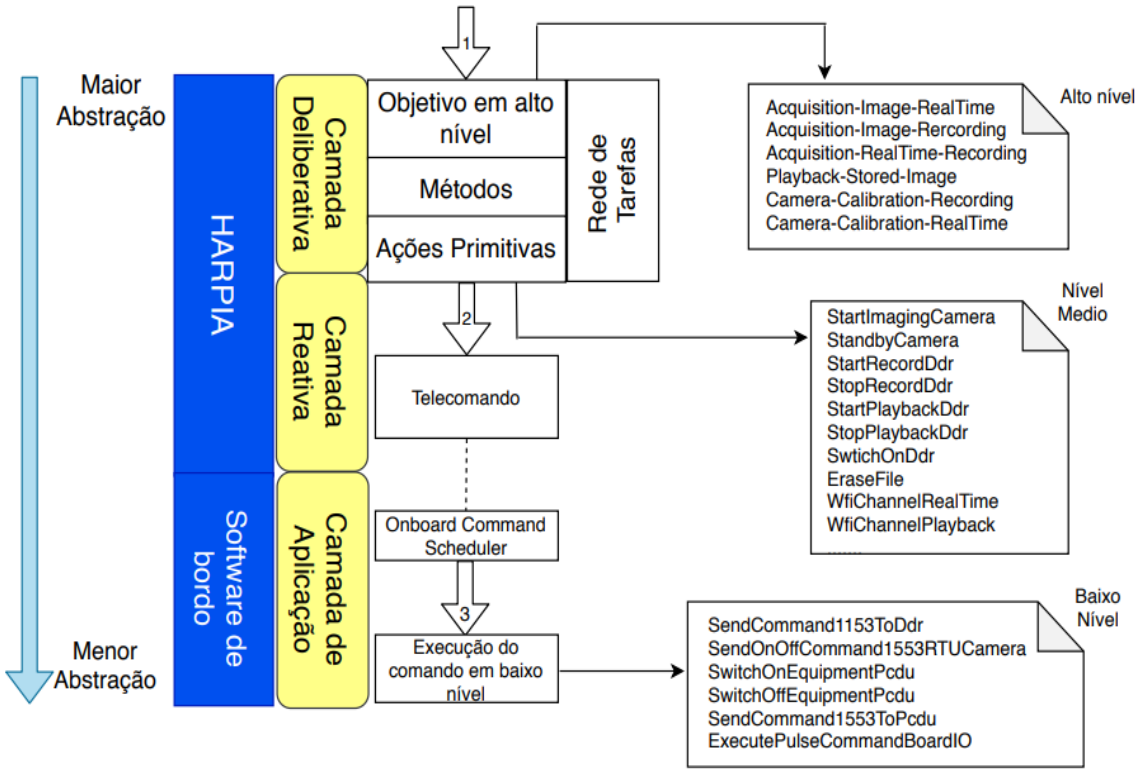
[33] Isso elimina a necessidade de manter uma máquina de estados correta, grande e finita do satélite, que em termos práticos seria intratável.

5.7. Nível de abstração das tarefas hierárquicas nas camadas

A base de raciocínio para a operação de um sistema de planejamento é seu domínio embarcado, sendo o conjunto de ações o coração do modelo. Em contraste às abordagens similares, as ações da HARPIA têm diferentes abstrações.

A Figura 5.2 traz exemplos de abstrações de atividades (exemplos de refinamento para uma missão de sensoriamento remoto) que atravessam a hierarquia das camadas, desde o objetivo recebido em alto nível (*e.g.,* "Aquisição de imagem em tempo real") até ele ser executado na forma de comandos de baixo nível (*e.g.,* comandos relacionados ao barramento MIL-STD-1553B) pelo software de voo.

Figura 5.2 – Fluxo de abstração das tarefas na hierarquia de camadas da HARPIA.



Fonte: Produção do autor.

Observa-se que apenas a camada deliberativa interpreta todas as abstrações de tarefas hierárquicas. Na seta 2 o objetivo em alto nível é mapeado como uma rede de

tarefas inicial cujo processo de planejamento resulta no nível de abstração de tarefas primitivas.

Na camada reativa o raciocínio da arquitetura se restringe às ações primitivas e a telecomandos. Pela seta 2, as ações primitivas são abstraídas como telecomandos. A partir da camada de aplicação apenas telecomandos e comandos são interpretados. O último passo cujo processo é indicado pela seta 3 é dado pela implementação do telecomando que resulta na abstração de comandos de mais baixo nível excetuáveis pelo processador.

Fazendo-se uma analogia com a abordagem clássica, os telecomandos mapeados pela camada reativa funcionam na HARPIA como os operadores do planejamento clássico, pois são eles que indicam como uma tarefa primitiva deve ser executada no ambiente.

O fluxo de abstração mostrado acima permite demonstrar que as ações contidas no plano elaborado pela arquitetura estão próximas da plataforma espacial e não exigem deliberações adicionais para que sejam executadas pelo computador de bordo.

Isso é muito útil para manter o modelo unificado, garantir a reusabilidade e evitar a necessidade de um executor inteligente na camada de aplicação.

5.8. Interface com o software de voo e o padrão PUS

A arquitetura preconiza uma fronteira bem delimitada com o software de voo. Em função disso, os itens modelados devem estar relacionados a informações disponíveis por meio de uma interface com o OBSW. Isso é realizado a partir de: (i) variáveis de estados associadas a parâmetros de telemetria e (ii) ações 'indivisíveis' que se relacionam com telecomandos. O objetivo é que essa interface seja encapsulada pelo padrão PUS da ESA.

Em softwares de voo que adotam o PUS, é comum existir um roteador de pacote centralizado que gerencia o tratamento em bordo dos pacotes na camada de aplicação, sejam eles uma telemetria ou um telecomando. Isso é intermediado por um componente do OBSW normalmente intitulado *Interface Ground* (I/F *Ground*).

O padrão PUS da ECSS define pacotes de telemetria e telecomando no nível de aplicação, considerando a interface de comunicação solo-bordo (ECSS, 2016). O PUS trata a utilização de pacotes para fins de monitoração e controle de subsistemas e da carga útil de veículos espaciais. Hoje, representa um padrão utilizado praticamente em todas as missões espaciais da Europa, assim como nas do INPE.

O PUS está no escopo da norma ECSS-E-ST-70-41C (ECSS, 2016) que estabelece conceitos operacionais usados para monitorar e controlar um satélite durante as fases de integração, teses e operação em voo.

A norma define um conjunto de serviços que satisfazem os conceitos operacionais identificados por ela. Cada serviço tem um conjunto de *requests* e *reports* associado. O primeiro está vinculado a telecomando e o segundo à telemetria.

Os serviços definidos pela norma incluem: parâmetros de *housekeeping*, relatório de eventos, armazenar e recuperar dados a bordo, transferir grande quantidade de dados, gerenciamento de memória, gerenciamento de tempo, entre outros.

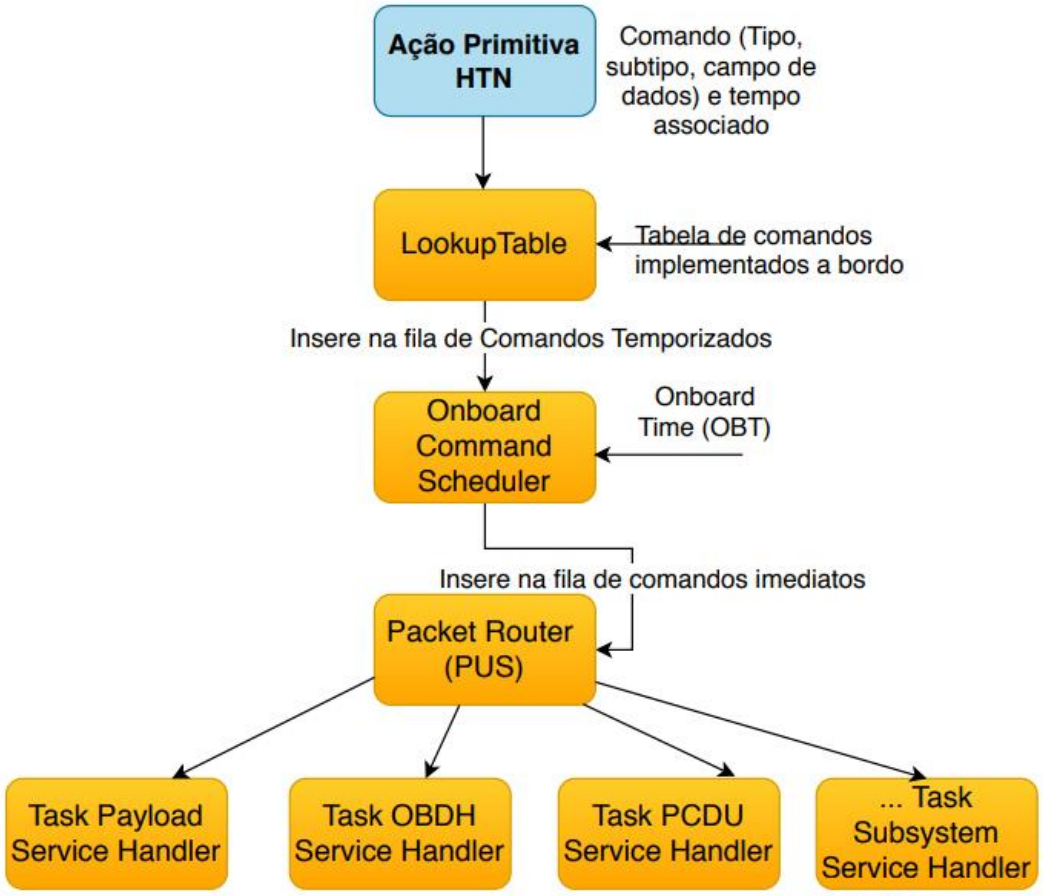
5.9. Integração de planejamento e execução

Um assunto em alta na comunidade de planejamento (vide GHALLAB et al., 2014 e GHALLAB et al., 2016) é como transformar a solução em execução de planos. Grande parte dos trabalhos foca apenas no raciocínio deliberativo e não apresenta soluções para a fase de execução (*acting*). Para pesquisas aplicadas, como a que se propõe esta Tese, isso se torna relevante.

Para proceder com o desdobramento de ações primitivas em um plano executável, a camada reativa da HARPIA deve consultar na tabela de telecomandos (*LookupTable*), aqueles que correspondem às ações de planejamento, inserindo-os na fila de comandos temporizados (*Onboard Command Scheduler*) do OBSW, conforme representado na Figura 5.3. A partir disso, a responsabilidade não é mais da HARPIA e sim do software de voo.

Uma vez que o OBT (*On-Board Time*) do computador de bordo indicar o momento da execução dos comandos agendados, estes serão enviados ao gerenciador de pacotes (*Packet Router*). Baseado no PUS, a aplicação destino é numericamente identificada e o pacote é despachado à fila de mensagens que recebe os comandos destinados aos subsistemas correspondentes. Isso mantém compatibilidade com as missões atuais do INPE que adotam a norma da ECSS e traz flexibilidade para sua reutilização em outras plataformas espaciais.

Figura 5.3 – Mecanismo de decomposição e execução de uma ação primitiva na camada de aplicação do OBSW.



Fonte: Produção do autor.

6 UMA EXTENSÃO DA ISISml PARA REPRESENTAÇÃO DO MODELO EMBARCADO BASEADA EM REDE DE TAREFAS HIERÁRQUICAS

Este capítulo apresenta a linguagem de descrição usada pela solução de planejamento implementada pela HARPIA. Ela estende a linguagem ISISml do INPE para representar o domínio hierárquico proposto nesta Tese.

A ISISml foi desenvolvida no escopo da GOESA cuja modelagem é voltada a planejadores CSP baseados em *timelines.*

A proposta do presente trabalho é ampliá-la para acomodar a representação em HTN, mantendo-se as melhores características de cada abordagem. De forma a promover uma melhor compreensão das regras e da sintaxe da linguagem, formaliza-se sua gramática.

Antes de descrever uma visão geral da linguagem estendida, mostram-se como o planejamento de operação pode ser representado por objetivos em uma missão espacial e como estes são transformados em uma rede de tarefas na HARPIA.

6.1. Transformação do objetivo em rede de tarefas

Um objetivo representa em geral atividades operacionais da carga útil ou de engenharia que podem ser executadas pelo satélite. Para a HARPIA tratar os objetivos, é necessário estabelecer como representá-los, de forma a permitir sua manipulação a bordo. A Tabela 6.1 traz um exemplo de um objetivo especificado em alto nível e suas respectivas transformações até que possa ser manipulado pelo software de voo.

O objetivo transformado, seja pelo segmento solo ou espacial, é colocado como uma atividade a realizar (*e.g.,* “Obter uma imagem”), contendo momentos de início e fim. Este objetivo é diretamente representável como uma tarefa de mais alto nível em uma HTN cuja representação é descrita no domínio da HARPIA.

Não faz parte do objetivo desta Tese converter as localizações geográficas em instantes temporais de imageamento. O trabalho de Eddy e Kochenderfer (2019) menciona que a partir da trajetória do veículo espacial é possível determinar as janelas nas quais o satélite tem sua linha de visão direta voltada ao local da imagem de interesse.

Segundo os autores, um algoritmo[34] de busca pode ser empregado para encontrar os períodos da trajetória em que haverá uma linha direta de visada do satélite para o centro da imagem, às quais todas as demais restrições geométricas da imagem sejam atendidas. Consulte o trabalho de Eddy e Kochenderfer (2019) para outros detalhes.

Tabela 6.1 – Transformações sofridas por um objetivo até ser representável por uma HTN.

Estabelecido por	Objetivo	Transformações realizadas
Usuário final da missão	"Quero uma imagem da Baía de Almirantado, na Antártica"	Localização convertida em dados geográficos.
Operador do satélite	Obter uma imagem entre a (latitude Y1, longitude X1) e (latitude Y2, longitude X2)	Dados geográficos convertidos em referências temporais (período em que o satélite passa sobre a área a imagear), em função dos dados orbitais.
Sistema Espacial ou Segmento solo	Obter uma imagem entre os momentos A e B	O executor trabalha com comandos residentes no software ou aqueles enviados pelo centro de controle.
Software de voo/HARPIA	Ligar e manter a câmera, gravador de dados ligados entre os momentos A e B	O objetivo é mapeado para uma rede de tarefas do domínio HTN embarcado juntamente com as informações temporais adquiridas através de ferramentas computacionais de apoio.

Fonte: Adaptado de Kucinskis (2012).

[34] O algoritmo utiliza como entrada um conjunto de localizações geográficas a serem gravadas e a trajetória do satélite.

6.2. Motivações para a representação em HTN

As principais motivações que levaram à solução de planejamento desta Tese a adotar a representação em HTN foram:

- A maioria dos problemas reais é facilmente descrita por conhecimento humano usando a representação hierárquica entre tarefas. Isso não é diferente em um plano de operação de satélites, onde operadores e especialistas conhecem em alto nível as tarefas a serem executadas e algumas das relações entre elas;
- Segundo Ingrand e Ghallab (2017), a HTN é uma das poucas representações que possibilita unificar planejamento e execução usando um único modelo.
- A organização hierárquica ajuda a simplificar a modelagem, que é um dos principais problemas que os engenheiros precisam enfrentar para implantar ferramentas automatizadas e, ao mesmo tempo, permite planos mais compreensíveis para o especialista. Segundo Georgievski e Aiello (2015), esse melhor entendimento do domínio possibilita, em alguns casos, a minimizar falhas na sua descrição;
- A representação em HTN apresenta desempenho superior a linguagens como STRIPS e PDDL (CARDOSO, 2018);
- Tem domínio altamente escalável e planejamento adaptável (DVORAK et al., 2014a); e
- Boa capacidade para representação níveis abstração – propriedade desejável na operação de um satélite.

Com relação às justificativas para este trabalho estender uma linguagem preexistente do INPE para representação em HTN, destacam-se: (i) não existe uma linguagem comum para descrição em HTN (GEORGIEVSKI; AIELLO, 2015); (ii) linguagens do planejamento clássico (*e.g.,* PDDL) não dão suporte para a descrição em HTN; e (iv) prover continuação às pesquisas já realizadas pelo instituto.

6.3. Considerações da pesquisa

Como visto na Seção 4.2.2, em função das particularidades do domínio espacial e das limitações das linguagens do planejamento clássico, as instituições da área espacial construíram linguagens próprias voltadas a aplicações de automatização de operações executadas em solo. Elas são em geral menos orientadas à ação e ricas na representação de restrições temporais, cuja sintaxe é voltada essencialmente para planejadores baseados em CSP.

Infelizmente, as linguagens concebidas não foram pensadas como uma forma padrão para modelar problemas e domínios, o que dificulta a reusabilidade e disseminação de informações, modelos e soluções de software, segundo Fratini et al. (2014). Ademais, elas se apoiam na ideia do planejamento *offline*, que não são aderentes às aplicações de domínio embarcado.

De modo geral, as linguagens temporais possuem as seguintes deficiências: (i) apresentam baixo nível abstração; (ii) é uma técnica custosa computacionalmente (BEAUMET et al., 2011); (iii) é limitada para codificar o conhecimento do especialista e (iv) têm baixo poder de escalabilidade (JIANG; XU, 2017).

Embora essas desvantagens possam ser toleráveis em *frameworks* de solo, elas se tornam uma questão desafiadora em satélites com autonomia, os quais precisam lidar com restrições de computadores de bordo, planejamento em tempo real e modelo integrado ao software de voo, que possui altos níveis de criticidade e preocupações de confiabilidade e segurança.

Iniciativas posteriores (*e.g.*, SMITH et al., 2008) buscaram aumentar a expressividade da linguagem, no entanto o foco da sintaxe se manteve voltado a planejadores temporais.

Apesar de ter relevância à comunidade de planejamento, a expressividade tem seu custo: os modelos se tornam menos inteligíveis e com uma grande classe de restrições, encontrar um plano pode ser computacionalmente custoso, o que é nocivo aos sistemas espaciais embarcados. Adicionalmente, isso está mais próximo de como

programar o problema usando diferentes conceitos de Planejamento Automatizado em IA do que modelar adequadamente o domínio para aplicações reais. Tal abordagem distancia os engenheiros de sistemas espaciais e o pessoal de operação (não são especialistas em linguagem de IA) do processo de modelagem.

Com intuito de colaborar com o estado da arte, esta Tese propõe a extensão da ISISml para uma linguagem de definição que busca uma maneira leve de modelar o comportamento do sistema espacial e, ao mesmo tempo, que atenda às demandas atuais de satélites com autonomia, como: maior abstração, desempenho e escalabilidade.

Em vez de exigir um conhecimento profundo sobre a linguagem de planejamento usada, os engenheiros espaciais podem realmente se concentrar no domínio hierárquico e no modelo espacial para lidar com os requisitos operacionais, tanto no nível de missão, como de sistemas.

Esse aspecto é especialmente importante para sistemas complexos, como satélites, que exigem conhecimento multidisciplinar de engenharia para modelar seu comportamento.

Para aumentar a inteligibilidade do modelo e obter melhor desempenho, propõe-se um domínio baseado em rede de tarefas. É apresentada uma representação hierárquica estendida que permite unificar planejamento e escalonamento e prover mecanismos de inferência de estados. Além disso, inclui-se a descrição de *'acting'* e *'sensing'* para garantir o planejamento *online*.

Como este domínio estende a ISISml, propõe-se uma divisão entre a solução hierárquica do especialista e os elementos modelados da aplicação. Como o foco está na modelagem do sistema e não nas declarações de variáveis, tipos e objetos, o modelo se torna mais inteligível, o que aumenta sua reusabilidade, reduzindo o grau de acoplamento entre os componentes da linguagem.

Isso permite que a linguagem proposta gerencie estados complexos do sistema espacial de maneira modular, alterando pequenas partes da descrição estrutural e

comportamental. Com isso, o modelo se torna mais escalável e de planejamento adaptável.

Isso traz aspectos positivos: (i) facilita o processo de manutenção corretiva do modelo em diferentes estágios da missão e (ii) se alinha com a maioria dos processos de desenvolvimento de software (vide ECSS-E-ST-40C - ECSS, 2009) da área espacial que tem geralmente uma abordagem incremental[35], segundo Tipaldi et al. (2018).

Com relação aos conceitos de planejamento, os diferenciais da linguagem frente a outras propostas similares estão destacados a seguir.

- Utiliza a representação por variáveis de estados, ao invés de lógica proposicional e de primeira ordem, como no planejamento clássico;
- Acomoda aspectos de planejadores hierárquicos, *timelines* e clássicos numa só linguagem descritora;
- Permite que os engenheiros codifiquem procedimentos operacionais usando tarefas de decomposição hierárquica;
- Apresenta descrições para planejamento, escalonamento e execução das tarefas;
- Não revela a componente temporal do problema em tempo de descrição de domínio. Cabe ao planejador embarcado definir qual tipo de representação temporal será utilizada.

Isso foi possível graças à ideia de considerar as propriedades preexistentes da ISISml somada a outras adaptações necessárias, obtendo-se uma nova forma de representação em HTN que alia algumas das melhores características de planejadores hierárquicos e temporais.

[35] Versões intermediárias de software são liberadas ao programa durante o ciclo de desenvolvimento para apoiar as atividades de testes e integração. Isto tende a não ser diferente em um sistema espacial com planejamento automatizado.

6.4. Hierarchical Modeling Language: uma extensão à ISISml

A solução de planejamento proposta nesta Tese utiliza como entrada um domínio definido por uma linguagem declarativa. O modelo descrito por ela deve incluir aspectos físicos e comportamentais da plataforma, para que esta consiga responder a estímulos internos e externos do ambiente, de forma a atender os objetivos impostos da missão.

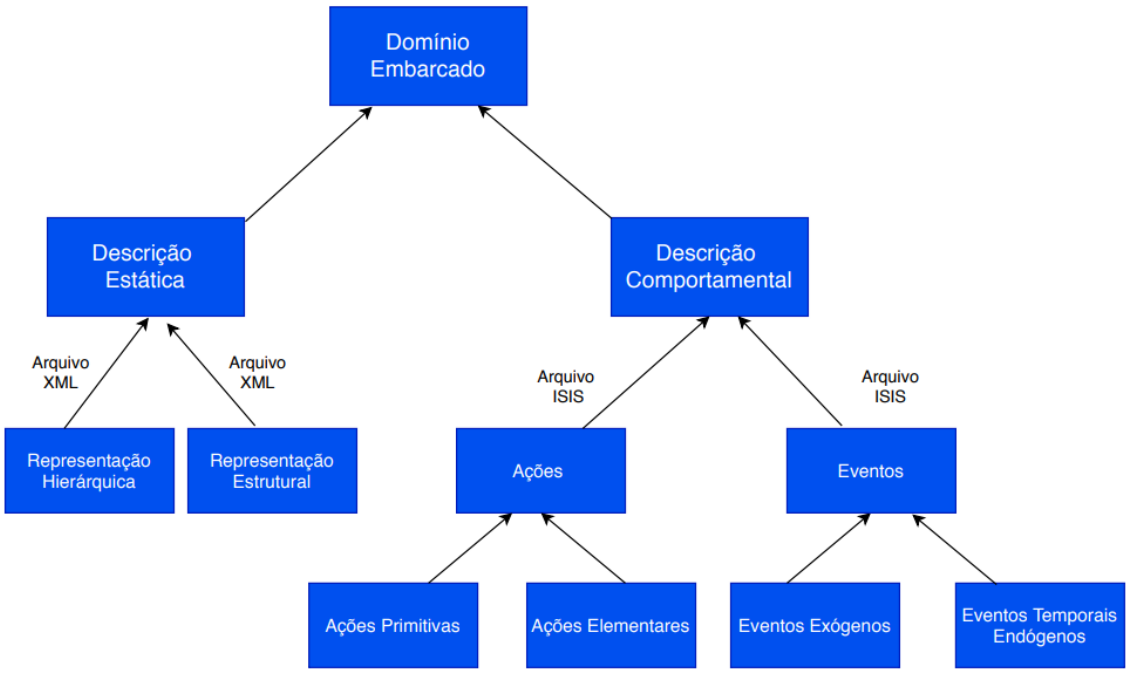
O objetivo é que a linguagem estendida proposta se mantenha simples para possibilitar a descrição do domínio, sem demandar um especialista da linguagem e que seja de fácil entendimento para os usuários finais, como engenheiros de sistemas espaciais.

A finalidade é propor uma nova abordagem de modelar sistemas espaciais que inclui a solução do especialista. O idioma foi construído de maneira que possa ser expandido por outros recursos de planejamento.

À Representação Hierárquica, foi dado o nome de *Hierarchical Modeling Language* (HML) por incluir na ISISml a descrição hierárquica no domínio. Embora seja possível especificar o domínio de alguns planejadores baseados estritamente em *timelines*, o foco é estender a capacidade dos modelos hierárquicos atuais, usando planejamento em HTN.

Conforme ilustrado pela Figura 6.1, o domínio embarcado (HML e ISISml) é definido por duas descrições complementares: estática e comportamental. O objetivo é propor uma divisão entre a solução de domínio configurável e as ações comportamentais.

Figura 6.1 – Domínio da linguagem HML + ISISml.



Fonte: Produção do autor.

Note-se que diferentemente do planejamento clássico, não se prevê um arquivo de definição do problema, pois os objetivos e estados iniciais são resolvidos em tempo de execução, considerando-se o satélite uma aplicação de tempo real.

6.4.1. Descrição estática

A Descrição Estática é especificada por duas representações: 'Representação Hierárquica' e 'Representação Estrutural'. Ambas são mantidas por arquivos independentes cuja formatação é baseada no padrão XML, o que facilita o reuso e exportação por outros sistemas de planejamento.

A 'Representação Hierárquica' refere-se à parte do modelo do sistema espacial que codifica a solução HTN a partir do conhecimento descrito pelo especialista. É a partir dessa descrição que se determina o espaço da rede de tarefas usado pelo algoritmo de busca e onde estão listados os objetivos da missão espacial.

A ‘Representação Estrutural’ mantém os elementos do sistema e o relacionamento estrutural entre eles. Ela reflete a organização física de subsistemas e equipamentos do segmento espacial.

6.4.1.1. Representação hierárquica

O arquivo da descrição hierárquica representa basicamente a descrição estática das tarefas hierárquicas do modelo. Ele é dotado de três partes: uma lista de tarefas abstratas e primitivas, uma lista de objetivos, e os métodos que contêm os esquemas de decomposição de tarefas. Ele é mantido por um arquivo no formato XML chamado *“hierarchical.xml*”.

O XML promove o intercâmbio com a possibilidade de se criar ferramentas gráficas de visualização e criação de planos hierárquicos, trazendo maior proximidade entre usuário final e linguagem. Um modelador poderia tanto importar dados em XML para uma ferramenta de visualização, como a partir da ferramenta exportar uma solução hierárquica no formato XML previsto pela linguagem.

A Figura 6.2 traz um exemplo reduzido do arquivo “*hierarchical.xml*” para o domínio de um satélite de sensoriamento remoto. Em suma, nesse arquivo se especifica:

- Conjunto de possíveis objetivos do problema, a serem selecionados e eventualmente customizados em tempo de execução;
- Tarefas primitivas que compõem o modelo;
- Tarefas compostas que fazem a abstração do domínio e seus métodos de decomposição;
- Condições para os métodos de decomposição das tarefas abstratas;
- Pontos de inferência de estados.

Figura 6.2 – Exemplo reduzido para o arquivo da representação hierárquica.

```
<hierarchical_description>
    <compoundtask name = "StartAcquisitionRealtime" id= "22" inference_moment = "true" methods = "ordered">
        <method namme = "StartAcquisitionRealtime_method">
            <precondition>
                <precond constraints = "PCDU.OrbitPeriod == sunlight"/>
                <precond constraints = "AOCS.mode == mission"        />
                <precond constraints = "DT.CommunicatingWithGround == withcommuication"/>
                <precond constraints = "OBDH.satelliteMode == routine"/>
                <precond constraints = "WFI.imageCloudy == cloudless"/>
            </precondition>
            <subtasks order = "ordered">
                <task  name  = "PrepareConfigCamera"        id = "24"  type = "compound"/>
                <task  name  = "SetDataTransmitterRealTime" id = "25"  type = "compound"/>
            </subtasks>
        </method>
    </compoundtask>

    <compoundtask name = "StopAcquisitionRealTime" id  = "23" inference_moment = "true" methods = "ordered">
        <method  name = "StopAcquisitionRealTime_method">
            <subtasks order = "ordered" >
                <task  name  = "TurnOffCamera"        id = "26"   type = "compound"/>
                <task  name  = "SwitchOffRtu"         id = "12"   type = "primitive"/>
                <task  name  = "SwitchOffDdr"         id = "10"   type = "primitive"/>
                <task  name  = "TurnOffDtRealTime"    id = "27"   type = "compound"/>
            </subtasks>
        </method>
    </compoundtask>
</hierarchical_description>
```

Fonte: Produção do autor.

Cada tarefa HTN possui um nome e um identificador único associado. As tarefas podem ser do tipo “*primitive*”, “*compound*” ou “*goaltask*” cuja definição é dada pelo elemento “*type*”. As tarefas abstratas devem declarar os possíveis métodos de decomposição.

Os métodos estão necessariamente ligados a uma tarefa abstrata. As precondições para decomposição dos métodos são opcionais e podem ser determinadas pelo atributo “*precondition*”. Cada restrição deve estar relacionada aos valores do domínio impostos aos elementos do sistema espacial, os quais estão definidos pela Representação Estrutural.

Um método se desdobra em uma lista de subtarefas que podem ser primitivas ou abstratas. O elemento “*order*” determina se as subtarefas devem ser ordenadas entre si, do contrário ("*unorder*"), o planejador é livre para escolher a ordem das ações. O mesmo atributo é utilizado para definir se o planejador deve respeitar a ordem dos métodos declarados pelas tarefas abstratas ou se o planejador pode considerar a ordem que lhe for mais conveniente.

O elemento "*inference_moment*" indica a opção de pontos de inferência a partir de tarefas abstratas. Esse recurso habilita os planejadores a projetar estados, eventos e consumo de recursos de suas subtarefas. O planejador pode também a partir disso criar pontos de estratégias de replanejamento.

Da análise do método "*StartAcquisitionRealtimeImage_method*" da Figura 6.2, depreende-se que ele é decomposto se o satélite estiver no período solar da órbita, se o controle de atitude estiver no modo missão e se o satélite estiver no modo rotina e em visibilidade com a estação. Nessas condições, ele se decompõe de forma ordenada em duas subtarefas: "*PrepareConfigCamera*" e "*SetDataTransmitterRealTime*".

Exemplos de outras partes deste arquivo são vistos adiante no Capítulo 8 que demonstra a aplicabilidade do domínio hierárquico proposto a partir de um estudo de caso.

A seguir mostra-se a representação estrutural e comportamental da ISISml. A presente Tese traz contribuições que foram adicionadas à ISISml (HML+ISISml) para melhor adaptação ao modelo HTN e à proposta deste trabalho. No entanto, a representação é toda baseada na sua linguagem precursora e por esse motivo será referenciada em alguns momentos apenas como 'ISISml'.

6.4.1.2. Representação estrutural

Se a 'Descrição Hierárquica' modela abstrações de ações e soluções de planos hierárquicos, a Descrição Estática é voltada para a modelagem das propriedades e características do sistema espacial. Ela traz a especificação dos elementos físicos da plataforma espacial que devem ser abstraídos pelo modelo.

A Descrição Estática é constituída por dois componentes: "Elementos" e "Domínios". Tal descrição é mantida por um arquivo XML denominado "*structural.xml*". A Figura 6.3 traz um exemplo reduzido desse arquivo.

Figura 6.3 – Exemplo reduzido para o arquivo da representação estrutural.

```
<structural_description>
    <domains>
        <domain name = "SatelliteMode">
            <value>satellite_launch</value>
            <value>satellite_emergency</value>
            <value>satellite_degraded</value>
            <value>satellite_routine</value>
            <value>satellite_oribt_maneuver</value>
        </domain>
        <domain name = "WfiModes">
            <value>wfi_power_off</value>
            <value>wfi_stand_by</value>
            <value>wfi_imaging</value>
            <value>wfi_calibration</value>
        </domain>
    </domains>

    <elements>
        <element name = "WFI">
            <timeline name = "Mode" type = "WfiModes" hkparameterid = "1" />
            <timeline name = "CloudyImage" type = "targetVisibility" hkparameterid = "2" />
        </element>
        <element name = "DT">
            <timeline name = "Mode" type = "DtModes" hkparameterid = "3" />
            <timeline name = "CommunicatingWithGround" type = "bool" hkparameterid = "4" />
        </element>
        <element name = "DDR">
            <timeline name = "Mode" type = "DdrModes" hkparameterid = "5" />
            <timeline name = "DssState" type = "DssStatus" hkparameterid = "6" />
            <resource name = "Memory"  capacityMax = "160000" capacityMin = "0" access = "exclusive" property =
                "continuous" category = "consumable" >
                <consumer name = "WFI" initialtotalconsumptionhkparameterid = "7"
                    initialrateconsumptionhkparameterid = "8"/>
            </resource>
        </element>
    </elements>
</structural_description>
```

Fonte: Produção do autor.

De forma sumária, esse arquivo especifica (KUCINSKIS, 2012):

- Os domínios das variáveis de estados;
- Os elementos que compõem o modelo;
- As variáveis de estados (timelines) e recursos pertencentes a cada elemento;
- Os consumidores, tipos, quantidade e forma de recursos;
- Identificadores dos parâmetros de *housekeeping* (unidades de informação de telemetria) e tipo de domínio associado a cada *timeline.*

Os elementos são componentes estruturais utilizados como forma de organizar informações sobre o domínio a ser modelado, podendo ser equipamentos ou subsistemas do satélite.

As *timelines* registram a evolução dos estados de variáveis em função do tempo, modelando propriedades dinâmicas da aplicação. Note-se que uma *timeline* não está necessariamente vinculada a um atributo dos subsistemas, podendo-se modelar abstrações lógicas de controle do problema.

Os recursos constituem um tipo especial de *timeline* na ISISml e representam os componentes consumíveis do domínio modelado. Eles determinam quando é possível executar certas atividades, de acordo com os recursos disponíveis a bordo.

Foram definidos os seguintes atributos relacionados aos recursos: "*capacityMin*" e *"capacityMax*", os quais definem a quantidade mínima e máxima; *"category*" denota se é um recurso consumido por taxa; "*property*" estabelece se a variável é discreta ou contínua; "*access*" delimita o tipo de acesso ao recurso. Este último é relevante para o planejador prever a proteção adequada (*e.g.*, uso de semáforos) quando o acesso for realizado de forma concorrente por diferentes ações.

Como forma de exemplificar o uso desses atributos, pode-se se basear no elemento "DDR" da Figura 6.3. Ele contém um recurso "*Memory*" cuja capacidade mínima e máxima é 0 e 160000 (*e.g.,* megabits) respectivamente.

A memória possui um consumidor intitulado 'WFI' cujo consumo é definido por taxa ("*consumable"*). Ela é de acesso exclusivo e se trata de uma variável contínua[36]. A descrição é finalizada com a indicação da telemetria correspondente que mantém a quantidade utilizada e a taxa de consumo inicial deste recurso.

Com relação aos valores domínios (parte superior da Figura 6.3), todos são tratados como enumerações no modelo embarcado construído. Já quanto às variáveis dos recursos, aquelas indicadas como "*discrete*" são tratadas como *uint32_t*, e "*continuous*" como variáveis do tipo "*float*".

6.4.2. Descrição comportamental

A Descrição Comportamental refere-se à parte do modelo do sistema espacial que descreve as modificações nos valores de seus componentes quando submetidos a determinadas ações ou eventos (KUCINSKIS, 2012). Ela descreve como as atividades alteram os estados do satélite, modelando o aspecto dinâmico da operação espacial.

[36] A taxa de dados é dada por um número decimal.

Esta descrição é especificada por uma sintaxe próxima à linguagem C e consta de arquivos cuja extensão é *".isis*". É constituída por dois conjuntos de arquivos: ações (*action*.isis) e eventos (*event*.isis).

Diferentemente das ações, os eventos não compõem diretamente o domínio do problema e não são declarados na "Representação Hierárquica".

6.4.2.1. Ação

O grau de abstração de uma ação é definido pelo modelador do problema, ponderando-se aspectos de eficiência do planejador, esforço para descrição do domínio e reusabilidade do modelo para descrever diferentes objetivos da missão.

Para melhor compreensão disso, considere o ato de ligar um equipamento do satélite. Para sua consecução, pode ser preciso enviar comandos a mais de um equipamento (*e.g.,* subsistema de potência e placas I/O). Um modelador pode abstrair os efeitos desses comandos como uma única tarefa do modelo de planejamento, embora na prática existam duas tarefas encaminhadas para equipamentos distintos.

Desta forma, nem toda ação que o satélite realiza na fase de execução é modelada como uma ação visível pelo planejador; detalhes não relevantes para a fase deliberativa também são propositalmente abstraídos, assim como aspectos do problema que se sabe como resolver no momento da execução.

NA HML/ISISML, uma ação pode ser uma ação sem hierarquia (elementar) ou vinculada a uma rede (primitiva). Isso confere à linguagem estendida tanto a capacidade para representação hierárquica, como da abordagem generativa do planejamento clássico.

De todo modo, uma ação primitiva ou elementar é sempre uma ação indivisível que não requer decomposição durante a fase deliberativa.

Toda ação primitiva modelada na Descrição Comportamental deve ser prevista na "Representação Hierárquica". De modo análogo, todos os valores de domínios de

recursos e *timelines* manipulados pelas ações devem ser especificados previamente na "Representação Estrutural".

As ações se correlacionam com algum elemento modelado pela ISISml. Desta forma, cada subsistema tem um conjunto de ações descritas em um arquivo independente (por exemplo, *CameraActions*.isis, *GravadorActions*.isis, etc).

Conforme o exemplo da Figura 6.4, uma ação na ISISml é constituída por três blocos: (i) precondições, (ii) efeitos (similar às pós-condições do planejamento clássico) e (iii) identificação de um serviço PUS.

Figura 6.4 – Exemplo de uma ação descrita em ISISml para habilitar o modo imageamento da câmera do satélite.

Arquivo *CameraActions.isis*

```
Action StartImaging
{
    Preconditions
    {
        PCDU.dcdcrtuline    = on;          // A RTU precisa estar ligada para que a WFI possa receber comandos
        WFI.mode            = wfi_stand_by;
        PCDU.wfiLine        = on
    }

    Effects
    {
        WFI.mode = wfi_imaging;
        PCDU.Power.Consume(element_wfi, 81, resource_power); // WFI consome 81 W em modo imaging
    }

    ServiceRequest
    {
        ServiceType     = 134;
        ServiceSubtype  = 1;
        DataLength      = 2;
        DataField       = 16;
    }
}
```

Fonte: Adaptado de Kucinskis (2012).

O primeiro e segundo bloco são insumos na fase de planejamento, enquanto que o terceiro provê informações de como o sistema espacial pode executar uma ação. Essa forma de descrever uma ação propõe a integração das etapas de planejamento e execução a partir de uma mesma linguagem descritora.

As precondições impõem valores condicionais aos estados da plataforma, enquanto os efeitos consomem recursos e alteram modos de subsistemas. Os efeitos correspondem ao modelo de transição de estados e denotam a expectativa do estado futuro do satélite após a execução da ação.

No exemplo da Figura 6.4, o efeito desta ação é que a câmera entre no modo imageamento e passe a consumir 81 W. Nada impede que a linguagem seja estendida para suportar outros tipos de operadores e expressões.

A correlação de uma ação por um comando identificado por tipo, subtipo e campo de dados visa prover compatibilidade com a norma da ECSS.

O campo de dados corresponde à carga útil de um pacote PUS. Por exemplo, um serviço para ligar ou desligar uma linha de potência depende da identificação do equipamento. Na ISISml, isso deve ser expresso pela carga útil, precedido pelo tamanho total dos dados a depender do tipo do parâmetro encapsulado no pacote.

No tocante aos efeitos das ações, é preciso definir qual subsistema detém o recurso, qual elemento é o consumidor e o tipo do recurso a ser modificado. Na ação deste exemplo, quem controla o recurso é o subsistema de fornecimento de energia, a câmera é o elemento que o consome, o valor a ser consumido é 81 W, sendo o recurso do tipo “potência”. Para casos em que a ação deixa de consumir um recurso (*e.g.,* a ação: ‘Interromper Gravação de Dados da Câmera’) deve-se estabelecer o consumo em zero.

6.4.2.2. Evento

Os eventos modelam como o planejador deve se comportar frente a estímulos internos e externos ao sistema espacial. Seu objetivo é dar mais expressividade à linguagem, promovendo maior robustez ao planejador para lidar com as limitações do planejamento clássico que considera o ambiente discreto e determinístico.

Em linhas gerais, os eventos modelados na HML + ISISml descrevem dois tipos de eventos (exógenos e endógeno) que têm uso e conceitos diferentes na solução de planejamento proposta nesta Tese.

Os eventos exógenos afetam o processo de planejamento, ativando ou interferindo na execução de determinadas ações do domínio. Eles modificam estados e recursos do satélite, mas estão fora do controle do software de voo. Por definição da linguagem precursora, a origem deles deve ser proveniente do ambiente espacial cuja ocorrência deve ser previsível.

O evento exógeno atua como uma ação instantânea, sendo disparado a partir de instantes futuros do horizonte de planejamento (considerando o instante inicial da fase deliberativa). Isso permite ao planejador lidar com a influência de agentes externos.

Eles podem ser tanto um evento nominal como o fato de o satélite entrar em visibilidade, como um evento ambiental, como o fato de o satélite estar sujeito à alta radiação espacial decorrente de um período de intensa atividade solar.

No caso de um evento não nominal, os efeitos são ações de reação para melhor aproveitamento dos recursos do satélite ou ainda uma ação de mitigação ao risco.

Já os eventos endógenos modelam restrições operacionais dos equipamentos, como por exemplo, o tempo de inicialização de um subsistema. Ele é disparado a partir de uma ação escolhida pelo planejador e produz como saída restrições temporais entre atividades do plano.

Sendo mais específico, os eventos endógenos impõem restrições de tempo adicionais à execução da ação. A Figura 6.5 traz um exemplo da especificação de eventos na HML.

Figura 6.5 – Exemplo de eventos descritos na linguagem.

Arquivo *events.isis*

```
ExogenousEvent StartCommunication
{
    DT.CommunicatingWithGround = 1; // em visibilidade
}

ExogenousEvent FinishCommunication
{
    DT.CommunicatingWithGround = 0; // fora de visibilidade
}

EndogenousEvent DataTransmissorReady
{
    RaisesAfter
    {
       // Apos ligar o Transmissor WDT, deve-se esperar 4 minutos, antes de habilitar o canal da WFI
       PCDU.ddrLine = lineOn;
    }

    TemporalEffects
    {
       OBTPlan += 240;
    }
}
```

Fonte: Adaptado de Kucinskis (2012).

É relevante destacar que a linguagem não prevê a descrição de eventos aleatórios associados a uma transição específica de estados, como definido pelos modelos que tratam falhas decorrentes de ambientes não determinísticos vistos na Seção 4.1.2.

6.4.3. Representação de consumo de recursos

A maioria dos planejadores embarcados vistos no Capítulo 4 tem a capacidade de resolver problemas com alocação de recursos. Os recursos são abstrações convenientes em aplicações de P&S e são especialmente relevantes no domínio espacial, dada a necessidade de gerenciar de forma eficiente os escassos recursos disponíveis a bordo.

Conforme observado em Georgievski e Aiello (2015), recursos podem ser classificados como discretos ou contínuos (propriedade intrínseca à variável) e em reusáveis ou consumíveis cuja definição se baseia em como o recurso é utilizado pela aplicação.

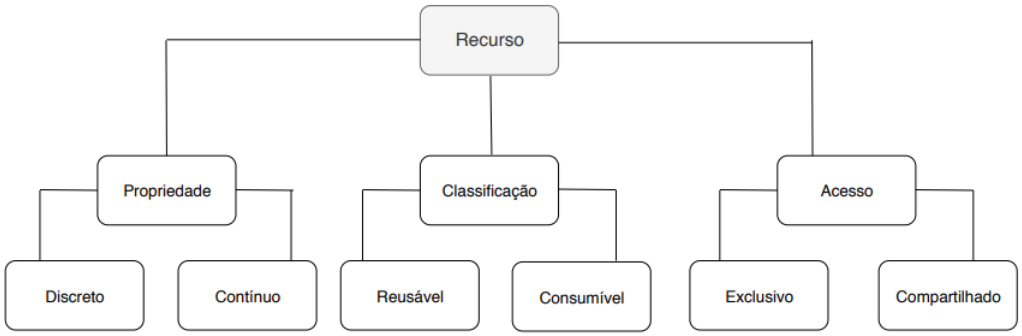
Um recurso consumível pode ser usado apenas por uma quantidade limitada de tempo (*e.g.,* memória do gravador de dados) e geralmente possui uma capacidade mínima e máxima associada, podendo ser reabastecido ou não.

Se um recurso consumível não puder ser restabelecido é dito de ‘consumo descartável’, como, por exemplo, o propelente utilizado pelo controle de atitude para realizar manobras orbitais.

Um recurso é dito ‘reusável’ se ele pode ser utilizado mais de uma vez. Um exemplo típico para o domínio espacial é a energia do subsistema de distribuição de potência utilizada por vários equipamentos.

O tipo do acesso ao recurso pode ser compartilhado ou exclusivo. A Figura 6.6 traz as relações entre os tipos de recursos supracitados que foram considerados na construção da extensão da ISISml.

Figura 6.6 – Relação das propriedades e categoria dos recursos da linguagem estendida.



Fonte: Produção do autor.

O consumo de recursos em problemas de planejamento está vinculado às ações. As ações pontuais de efeito imediato sobre estados usualmente não violam restrições de recursos, diferentemente das ações com duração. Uma falha ocorre sempre que uma ação contida num intervalo de tempo violar a capacidade mínima ou máxima estabelecida pelo modelo.

Um recurso na linguagem estendida é qualquer entidade física ou virtual da plataforma que tenha disponibilidade limitada quando utilizada por uma ação. Um recurso R é uma 3-tupla (i_R, C_R ,c_R), onde i denota um nível inicial de recurso, C traz a capacidade máxima e c estabelece o nível mínimo.

Segundo Kucinskis (2012), uma operação na ISISml pode alocar uma quantidade fixa de recursos, ou iniciar o consumo de recursos em taxas (unidades consumidas por tempo). A dinâmica de operação decorre de transações subsequentes que podem cessar o consumo do recurso, ou iniciar sua liberação também a uma dada taxa. Tal forma de consumir recursos mapeia mais fielmente o comportamento do sistema real.

Uma operação sobre *R* consumida por uma taxa representa uma modificação numérica considerando um intervalo de tempo específico. Essa transação é definida pela 4-tupla (R, s_t, e_t, δ), onde *R* é um recurso, s_t e e_t correspondem aos tempos de início e fim, e δ denota a função de consumo decorrente desta operação.

Considerando a ação do satélite "*apagar dados do gravador*", ela independe do tempo decorrido e tem efeitos imediatos sobre a disponibilidade do recurso "*memória do gravador*". Nesse caso, configura-se uma operação instantânea cuja representação é dada a partir da 3-tupla (R, t, δ), visto que $t = t_s = t_e$.

É importante frisar que o consumo de um recurso pode variar conforme o estado do subsistema. Por exemplo, um transmissor em '*standby*' consome menos potência do que transmitindo dados em tempo real. Toda dinâmica de consumo é determinada pelos efeitos das ações especificados na descrição comportamental.

Uma característica que mais difere a ISISml de outras representações baseadas em HTN (*e.g.*, a linguagem de especificação do SHOP2) é a forma de consumir recursos. Linguagens hierárquicas usualmente não lidam com recursos e quando o fazem alocam quantidade fixa de recursos mesmo para aquelas atividades que consomem recursos por unidade de tempo. Eles são descritos antecipadamente de forma estática nos efeitos das ações abstratas ou primitivas.

Como cada ação abstrata ou primitiva geralmente está vinculada hierarquicamente a uma ou mais tarefas, alocar uma quantidade fixa de recursos implicaria em menos reusabilidade e maior esforço para descrição do domínio, visto que cada objetivo pode ter restrições temporais distintas. Além disso, isso tem mais propensão a erros no

modelo, já que o cálculo do consumo é realizado antecipadamente para diversas atividades do domínio.

Por essa razão, a ISISml não estabelece uma quantidade fixa de recursos às tarefas. Em vez disso, especifica consumidores e as formas de consumo. Eles são atrelados às variáveis de estados e especificados conforme o tipo do recurso alocado (KUCINSKIS, 2012).

Isso alia características positivas de planejadores temporais (para a qual foi desenvolvida originalmente a ISISml), mas usando planejadores HTN proposto nesta Tese.

Isso também se aproxima da forma real de operação de um satélite do que a de sistemas de planejamento em HTN baseado puramente em estados que usualmente não considera a componente temporal no problema.

Ao estender essa propriedade da ISISml para o domínio baseado em rede de tarefas, traz-se uma nova de representação de consumo para planejadores hierárquicos que unifica planejamento e escalonamento, facilitando o processo de inferência de estados.

Esta abordagem somada a outras características permite ao planejador lidar com as principais deficiências da representação em HTN quando aplicada fora do planejamento clássico (inferência de estados, ocorrência de eventos, consumo de recursos e mecanismo para reparo de planos).

6.5. Representação do domínio

As linguagens do domínio espacial carecem de uma sintática bem definida ou mesmo de exemplos concretos. Os sistemas espaciais autônomos mencionam que se basearam em alguma linguagem voltada para *framework* de solo, mas sem apresentar as adaptações necessárias para o planejamento embarcado.

Isso se aproxima à forma convencional de programação do que a de sistemas de planejamento, dificultando a padronização, legibilidade e validação do modelo.

Para evitar isso, o presente trabalho utiliza o formalismo da gramática Backus-Naur Form (BNF) para definir regras de sintaxe da linguagem proposta.

A especificação BNF é derivada de um conjunto de regras, <símbolo> :: == __expressão__, onde <símbolo> é um símbolo não terminal que possui uma ou mais sequências de símbolos terminais e não terminais dada pela expressão cuja sequência de símbolos pode ser separada por um barra vertical '|', indicando uma escolha. Tal notação indica as possibilidades de substituição para o símbolo à esquerda. Um símbolo que nunca aparece ao lado esquerdo é dito terminal.

Para aumentar a expressividade da gramática, utilizou-se neste trabalho uma sintaxe adicional [37] à BNF: o sinal '*' indica que há zero ou mais símbolos possíveis e '+' indica que pode haver a instância de um ou mais símbolos. O sinal '#' denota que o símbolo aparece uma vez ou é optativo. Símbolos precedidos por & são variáveis numéricas e também símbolos terminais por simplificação na representação proposta.

6.5.1. Especificação do domínio hierárquico

A BNF é tipicamente utilizada para descrever linguagens de programação. Neste trabalho, aproveita-se de sua expressividade para descrever os principais elementos descritos em XML contidos na representação estrutural da linguagem estendida.

Isso facilita a clareza de como os elementos podem se relacionar e as regras de descrição. Para aumentar a legibilidade da gramática, suprimiram-se regras de formatação e sintaxe da própria XML, dando ênfase aos componentes da linguagem.

[37] Alguns desses sinais adicionais foram baseados na EBNF (*Extended* BNF).

Especificação 6.1 – Gramática BNF da Representação Estática

Descrição Estrutural

```
1 <domain-values-list>     ::= <domain-type+>
2 <domain-type>            ::= $domain-name $values+
3 <element-list>           ::= <element+>
4 <element>                ::= $element-name <timeline+> <resource*>
5 <timeline>               ::= $timeline-name &telemetry-id <domain-type>
6 <resource>               ::= $resource-name &max-capacity# &min-capacity# <attributes> <consumer*>
7 <consumer>               ::= $consumer-element-name &id-telemetry-current-consume &id-telemetry-initial-rate
8 <attributes>             ::= <category> <access-type> <property>
9<access-type>             ::= shareable | exclusive
10<category>               ::= consumable | reusable
11<property>               ::= discrete | continuous
```

Descrição Hierárquica

```
1 <primitive-task>         ::= $task-name
2 <abstract-task>          ::= ( (ordered | unordered) <inference-moment#>)  <method+>
3 <method>                 ::= $method-name <precond-list> <tasks-list+>
3 <task>                   ::= <primitive-task> | <abstract-task>
4 <task-list>              ::= (ordered | unordered) <task+>
5 <precond-timeline-list>  ::= <precond-timeline*>
6 <precond-timeline >      ::= $timeline-name <compare-op> $domain-value
7 <goal-list>              ::= <goal+>
8 <goal>                   ::= $goal-name
9 <compare-op>             ::= != | == | >= | <= | < | >
10 <inference-moment>      ::= true | false
```

O domínio comportamental da ISISml não será especificado pela BNF, pois sua sintaxe é similar a de outras linguagens tradicionais baseadas em ações e efeitos e pelo fato de o presente trabalho não apresentar modificações significativas em relação à sua versão não estendida.

6.5.2. Especificação do problema de planejamento embarcado em satélites

No planejamento clássico, os problemas são definidos na etapa de descrição do domínio seja pelo especialista ou por uma ferramenta automatizada. Em aplicações espaciais, muitas das informações que definem o problema não são conhecidas antes da execução da aplicação.

Para que o planejamento de missão baseado em objetivos seja caracterizado a bordo, é preciso especificar o problema, de modo que ele seja traduzido como uma entrada válida ao planejador embarcado. Isso deve ser feito por um sistema de planejamento para coletar as informações do estado corrente do satélite.

O objetivo da gramática da Especificação 6.2 é definir a representação do problema a bordo de satélites utilizada na solução de planejamento proposta nesta Tese. Observe-se que a representação utiliza partes da descrição estrutural e comportamental definidos anteriormente pela ISISml + HML.

Especificação 6.2 - Gramática BNF da Representação do Problema a bordo de Satélites

1	**<domain>**	::= $space-mission-name
2	**<def-problem>**	::= **<domain> <goal-list> <def-initial-states> <planning-horizon> <exogenous-events>**
3	**<def-initial-states>**	::= **<initial-timeline-list> <initial-resources-consumption-list>**
4	**<initial-timeline-list>**	::= **<initial-state-timeline^{+}>**
5	**<initial-state-timeline>**	::= $initial-state
6	**<initial-resources-consumption-list>**	::= **<initial-resources-consumption^{+}>**
7	**<initial-resources-consumption>**	::= $initial-resource
8	**<exogenous-event>**	::= **<orbit- prediction> <environmental-events>**
9	**<environmental-events>**	::= $cloudy_target $solar-radiation-activity $space-debri
10	**<predict-orbit>**	::= **<satellite-visibility> <period-orbit>**
11	**<satellite-visibility>**	::= $satellite-visibility-time | $satellite-non-visibility-time
12	**<orbit- prediction>**	::= $sunlight-time | $eclipse-time
13	**<planning-horizon>**	::= $time-interval
14	**<goal-list>**	::= **<goal^{+}>**
15	**<goal>**	::= $goal-name $goal-parameter^{+}

De acordo com a gramática acima, as seguintes informações definem o problema: lista de objetivos em HTN, estado inicial das variáveis de estados e da disponibilidade inicial dos recursos, horizonte de planejamento e previsão de eventos futuros. Dentre essas informações, apenas o conjunto de objetivos é listado durante a descrição do domínio.

Para descrição final do objetivo, entretanto, ainda é preciso de parâmetros adicionais representados pelo símbolo $goal-parameter. Para o objetivo ‘comandar uma manobra de correção de altitude de órbita’, por exemplo, é preciso basicamente dos seguintes parâmetros: atitude nominal à qual o satélite deve apontar, tempo de início e duração da manobra.

Por último, observe que alguns símbolos estabelecidos pela gramática podem depender do tipo da missão (órbita, plataforma, número de estações, conceitos de operação, etc.), como os símbolos que denotam os eventos ambientais, previsão de órbita, horizonte de planejamento, tempo de visibilidade do satélite, etc. Apesar disso, os símbolos em alto nível são os mesmos.

7 UMA SOLUÇÃO HÍBRIDA USANDO PLANEJAMENTO HIERÁRQUICO E GENERATIVO COM INFERÊNCIA DE ESTADOS E GERENCIAMENTO DE TEMPO E RECURSOS DAS ATIVIDADES

Este capítulo descreve o formalismo HTN, a abordagem de planejamento híbrida implementada pela HARPIA, o processo de inferência de estados e o mecanismo para lidar com reparo do plano após a constatação de uma falha ou evento não previsto.

Para facilitar a clareza do texto e evitar repetição massiva de "uma solução híbrida de planejamento", este capítulo pode se referir à solução simplesmente por HARPIA - a arquitetura de planejamento que realiza os elementos desta solução no âmbito do domínio de satélites.

7.1. Considerações da pesquisa

Antes de descrever a abordagem de planejamento embarcado proposta nesta Tese, são listadas as principais razões para evitar o uso de planejadores temporais baseados em CSP predominantes nos sistemas de planejamento vistos no Capítulo 4.

- O CSP é considerado custoso computacionalmente para uma implementação embarcada (BEAUMET et al., 2011);
- O tempo de planejamento baseado em CSP não é previsível (LONG et al., 2018), o que é nocivo para respostas de tempo real;
- O algoritmo de planejamento baseado em CSP tem baixa escalabilidade, vide o trabalho de Amigoni et al. (2010). O planejador CSP se torna ineficiente à medida que se aumenta o tamanho do problema com mais restrições (JIANG; XU, 2017).

Apesar de vantagens, os planejadores HTN também apresentam limitações importantes:

- **Alto custo para descrever os métodos HTN**. Isso ocorre porque o formalismo HTN exige que os especialistas forneçam métodos para cobrir todos os cenários possíveis que o algoritmo possa encontrar. Se o planejador HTN se deparar com

uma situação que o engenheiro não havia previsto, ele simplesmente falharia sem encontrar uma solução;

- **Fragilidade em ambientes dinâmicos**. O problema anterior se agrava em ambientes dinâmicos nos quais eventos fora do controle do planejador podem ocorrer, levando a novas situações não previstas pelo domínio; os planejadores em HTN não são adequados para trabalhar em ambientes abertos e dinâmicos;
- **Separação de Planejamento e Escalonamento.** Os planejadores HTN separam usualmente essas etapas. Isso geralmente acarreta na subutilização das "heurísticas de busca" presentes na rede - uma das principais virtudes desta abordagem.

A Tabela 7.1 faz uma análise das abordagens clássica, hierárquica e temporal.

Tabela 7.1 – Análise comparativa das abordagens de planejamento clássico, HTN e temporal.

Critério	Abordagem Generativa (Clássico)	Planejamento Hierárquico (HTN)	Planejamento Temporal (CSP)
Esforço na descrição do domínio	Baixo (apenas ações do modelo)	Alto (ações e métodos HTN)	Baixo (apenas ações do modelo)
Escalabilidade	Baixa	Alta (métodos HTN)	Baixa
Capacidade de inferência de estados futuros	Não tem	Baixa	Alta
Expressividade do domínio	Baixa	Alta (solução do Especialista no domínio)	Baixa
Capacidade de respostas a falhas	Alta (POP)	Baixa (alto custo para implementar planejamento de contingência)	Alta
Capacidade de lidar com recursos e tempo	Não tem	Baixa	Alta

Fonte: Produção do autor.

Para suprir os problemas supracitados do HTN, mas mantendo suas vantagens, é aqui proposta uma abordagem que alia planejamento hierárquico com a capacidade de lidar com situações não previstas do planejamento generativo. Embora a ideia de uma abordagem híbrida não seja nova, as poucas propostas existentes, descritas a seguir, utilizam HTN com o planejamento no espaço de planos de ordem parcial, o que tende a degradar o desempenho. Na HARPIA, a busca progressiva faz com que os estados evoluam sempre de forma determinística, levando à redução do tempo de execução. Esse aspecto é fundamental para satélites que utilizam processadores espaciais tipicamente lentos.

7.2. Formalismo do planejamento de missão em HTN

A solução de planejamento desta Tese toma como base o trabalho de Erol (1994) o primeiro formalismo de planejamento hierárquico baseado em HTN proposto na literatura.

A HARPIA se diferencia ao propor um formalismo híbrido a partir de um modelo hierárquico aliado à abordagem generativa do planejamento clássico, mas sem utilizar vínculos causais (POCL, descritos no item 3.3.2). Visando o escalonamento das atividades, o formalismo também estende o modelo HTN de Erol para acomodar a componente temporal no problema.

Uma rede de tarefas *U* é um par (T, ψ), onde T é uma coleção finita e não vazia de tarefas (constituída por tarefas abstratas e elementares) e ψ um conjunto de restrições. Cada tarefa do modelo é uma 4-tupla $T = (s_t, e_t, P, C)$, onde:

- s_t é o tempo inicial da tarefa;
- e_t é o tempo final da tarefa (opcional – nem todas ações são durativas);
- P_t é o pai da tarefa;
- C_t é a classe da tarefa
  - ✓ *Nominal* inserida por decomposição HTN ou *Reparadora* por inserção de tarefas.

O símbolo ψ especifica restrições sobre T que devem ser satisfeitas durante o processo de planejamento. As restrições podem ser de recursos, relações temporais entre as tarefas ou precondições para sua realização.

Um método de decomposição é uma 4-tupla (*m*, t_a, $LIST_{precond}$, *subtasks*) que codifica uma maneira de decompor uma tarefa abstrata (t_a) em tarefas de mais baixo nível a partir de subtarefas de *m.* Para sua decomposição, é preciso satisfazer um conjunto de precondições ($LIST_{precond}$).

Uma tarefa primitiva *E* é realizada por um operador *O* denotado pela tripla ($LIST_{precond}$, $LIST_{effects}$ e S_{id}), onde $LIST_{precond}$ é uma lista de precondições que deve ser satisfeita antes da ação, $LIST_{effects}$ corresponde aos efeitos no ambiente, e S_{id} corresponde à identificação de um tipo e subtipo de um telecomando implementado a bordo, útil para a fase de execução.

Uma instância HTN do problema de planejamento de missão da solução proposta nesta Tese é definida pela 8-tupla P = (*V, D, A, E, O, M,* U_i, S_i), onde:

- *V,* um conjunto finito de variáveis que representa todos os estados e recursos do satélite;
- *D,* um conjunto finito de domínios de valores para cada variável, com $V_i \in D_i$;
- *A*, um conjunto finito de tarefas abstratas;
- *E*, um conjunto finito de tarefas primitivas, tal que $A \cap E = \emptyset$;
- *O*, um conjunto finito de operadores;
- $M \subseteq A \times U$, um conjunto finito de métodos de decomposição; sendo eles totalmente ordenados com $m \in M = (a,U)$;
- $U_i \in A$, uma rede de tarefas inicial a ser executada; ela representa o nível mais alto de abstração da árvore de decomposição;
- $Si \in V$, o estado inicial de cada variável modelada a partir da plataforma espacial.

Para a notação da abordagem híbrida descrita a seguir, baseou-se no formalismo de Geier e Bercher (2011). Trata-se do primeiro trabalho a definir formalmente a inserção

de tarefas no problema HTN sem depender de um método de decomposição. Segundo o autor, isso combina técnicas de HTN com planejamento de espaço de planos.

Na HARPIA, entretanto, esse conceito é explorado sob uma ótica diferente. Ele é utilizado de forma restrita durante o processo de replanejamento a partir de tarefas reparadoras (T_r). Isso permite estender a representação tradicional do planejamento hierárquico para lidar com situações não previstas pelo domínio.

Uma rede U_{sol} representa uma solução híbrida para o problema de planejamento de missão da HARPIA. Ela representa a saída do processo de planejamento automatizado, sendo composta por ações em série ($[a_1, t_1]$, $[a_2, t_2]$, $[a_3, t_3]$, ... $[a_n, t_n]$) com tempo de execução indicado por t.

O critério para que a rede U_{sol} seja uma solução é definido como segue:

1. Deve existir uma sequência de métodos de decomposição *m* (planejamento em HTN) e/ou a inserção de tarefas reparadoras (planejamento generativo) que refinam a rede inicial (U_i) em uma rede U_{sol}.
2. U_{sol} precisa ser executada a partir de S_i, onde:
   (a) Todas as ações são tarefas primitivas.
   (b) Todas as ações estão ordenadas e não precisam de um processo de linearização.

7.3. Visão geral da abordagem

Ao planejador da HARPIA deu-se o nome de *Liger*[38], em função de sua abordagem híbrida com propriedades de planejamento hierárquico e da abordagem generativa do planejamento clássico. O *Liger* tem como objetivo não ser intensivo em memória e processamento e, portanto, apto à execução em hardware qualificado para área espacial.

[38] Animal híbrido resultado de cruzamento entre um Leão e uma Tigresa.

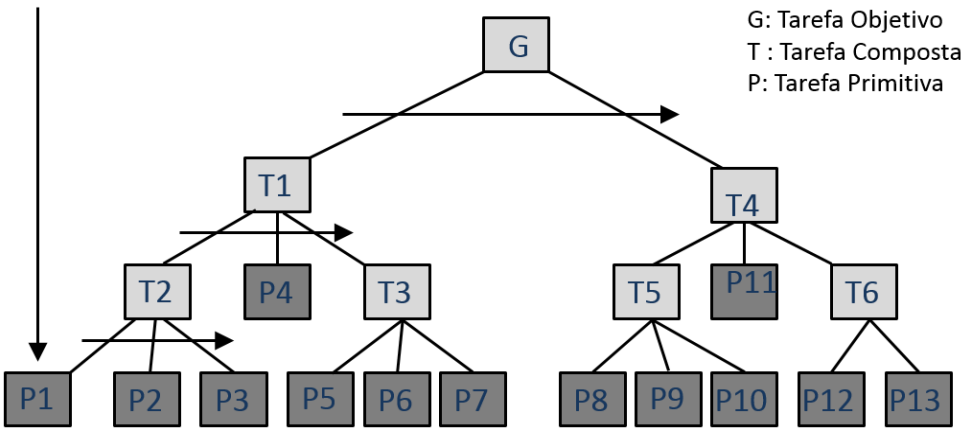
É baseado num planejador HTN de busca progressiva e com decomposição de ordem total das tarefas (*Total-order Forward Decomposition*). Considerando-se uma ávore de decomposição em HTN (vide a Figura 7.1), o nó raiz denota o objetivo e os nós folhas denotam as tarefas primitivas.

Para explorar as tarefas contidas na árvore, o *Liger* utiliza uma abordagem similar à busca em profundidade (*Depth-First Search* - DFS) cuja característica principal é explorar primeiro o maior nível de profundidade antes de retroceder e buscar novamente outro ramo de maior nível de profundidade, e assim sucessivamente.

No *Liger*, são utilizadas as variáveis de estados durante a fase de deliberação, tanto para a consulta das precondições, como para registro dos efeitos das ações que alteram estados e recursos do satélite.

Note-se que o objetivo dos planejadores baseados em *timelines* consiste em encontrar uma sequência de operações que leve as 'linhas do tempo' ao estado desejado usando CSP. Isso difere da proposta do Liger, na qual o estado das variáveis é fruto do desdobramento da solução hierárquica, não sendo o conceito de pontos no tempo o cerne da operação de planejamento.

Figura 7.1 – Mecanismo de inspeção de busca progressiva e de ordem total do Liger.



Fonte: Produção do autor.

O processo de planejamento da HARPIA é centralizado e sua abordagem se classifica como de domínio configurável, uma vez que o motor de busca é independente de domínio[39], mas os métodos HTN são de domínio específico. De acordo com Amigoni et al. (2010), isso traz em geral uma boa relação custo-benefício entre eficiência e generalidade, propriedades importantes para aplicações espaciais.

Para a decomposição das tarefas, utiliza-se a versão iterativa (não recursiva), a fim de evitar problemas de memória e complexidade que são potencialmente perigosos ao ambiente computacional embarcado. Para que a busca em profundidade proceda com as decomposições sem usar recursão, implementou-se um mecanismo de operação de uma pilha (*Last In, First Out* - LIFO).

Não se utilizou para isso bibliotecas prontas como a biblioteca *Stack* do *Standard Template Library* (STL) do C++.

Uma desvantagem da busca em profundidade é precisar explorar todo o domínio. No entanto, é importante notar que o algoritmo atravessa uma hierarquia de tarefas. Nela, o planejador não necessita considerar todos os operadores aplicáveis ao estado atual como no planejamento clássico, mas apenas aqueles que ocorrem nos métodos HTN e que são aplicáveis ao estado corrente.

Embora o *Liger* utilize uma abordagem híbrida, está mais próximo do espaço de estados, pois as ações aplicáveis da rede visam gerar um novo estado, ou refinar um estado em falha. Cada nó representa uma tarefa e um arco entre dois nós representa uma precedência temporal entre as tarefas. Cada nó folha, por sua vez, denota uma determinada configuração do satélite dada por uma atribuição às variáveis de estados.

Um nó do algoritmo *Liger* encapsula uma estrutura de dados que contém outras informações relevantes que são utilizadas durante a busca a partir de um mecanismo de manutenção de histórico das decisões.

[39] Apesar disso há espaço para uso de heurísticas, por exemplo, para seleção de ações de refinamento descritas a seguir.

Algumas delas estão relacionadas ao estado interno do planejador, que é muito útil para registrar pontos de retrocesso, tanto visando verificar métodos alternativos como para iniciar um processo de replanejamento. Isso difere do algoritmo DFS tradicional que usa tipicamente o procedimento de busca não informada.

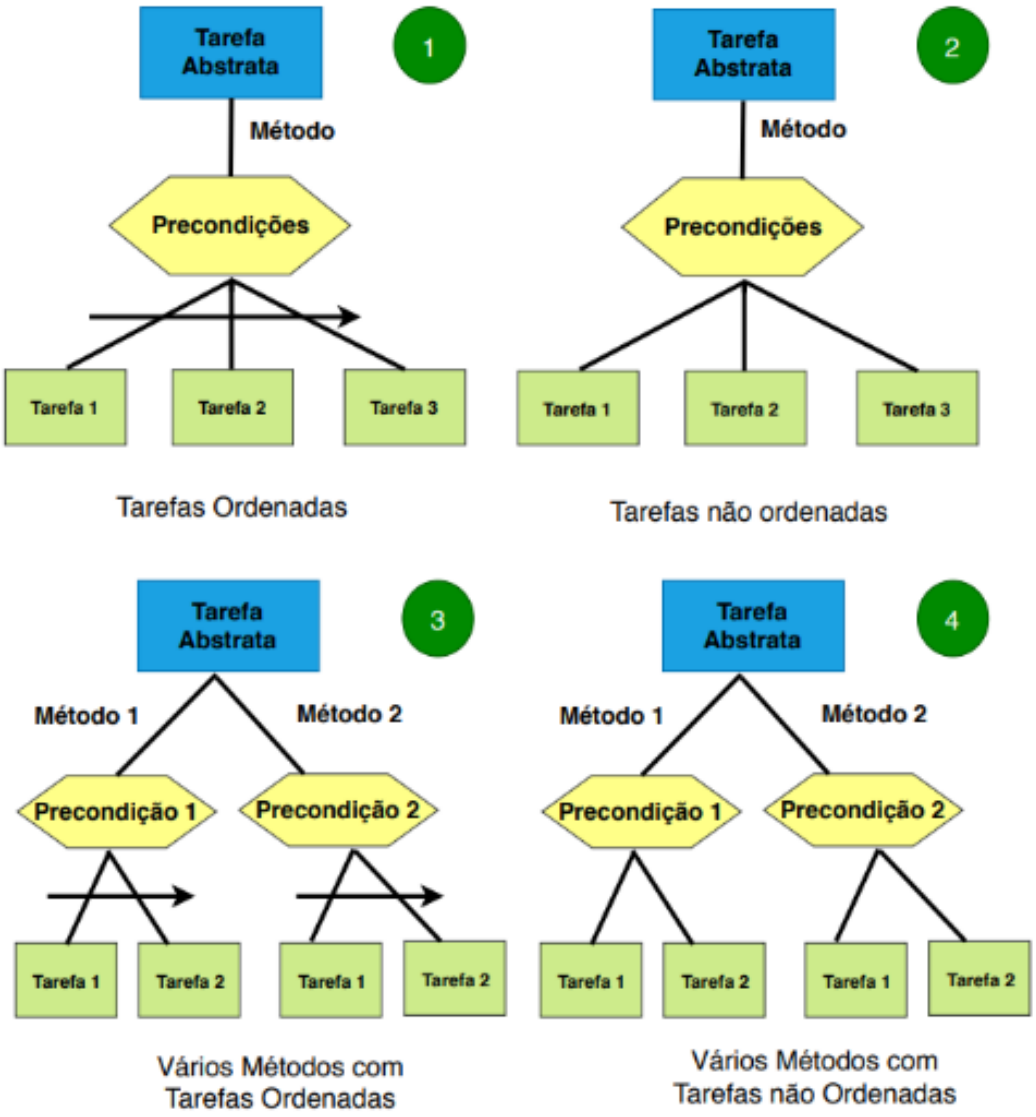
Da análise da Figura 7.2, têm-se quatro tipos de decomposição de tarefas em HTN: (1) decomposição única com subtarefas ordenadas e (2) idem ao primeiro, mas com tarefas não ordenadas; (3) múltiplas decomposições com subtarefas ordenadas e (4) idem ao terceiro, mas decompondo-se em tarefas não ordenadas.

O *Liger* comporta as decomposições ilustradas pelos números 1 e 3 dessa figura. As estruturas 2 e 4 configuram planejamento de ordem parcial, abordagem não adotada neste trabalho.

O algoritmo se classifica como um planejador HTN acíclico, visto que nenhuma tarefa composta alcança a si mesma a partir de uma decomposição.

Cabe destacar que a maioria dos planejadores HTN escolhe aleatoriamente possíveis métodos de decomposição. Embora isso seja de fácil adaptação na implementação do algoritmo, evita-se seu uso aqui pelo fato dela implicar em execuções diferentes para uma mesma entrada do problema. Os métodos no Liger são selecionados conforme a ordem especificada no domínio, garantindo um comportamento mais previsível do algoritmo. Além disso, limitou-se o uso de métodos alternativos às tarefas hiérarquicas de maior nível de abstração.

Figura 7.2 – Tipos de estruturas de decomposição em HTN.



Fonte: Produção do autor.

Assim como outros planejadores baseados em algoritmos de busca em profundidade, o *Liger* não produz necessariamente uma solução ótima, pois o algoritmo é interrompido quando um plano qualquer, que leve ao estado objetivo, é encontrado. No entanto, deve-se notar que se utiliza a regras de controle da decomposição de tarefa escrita por especialista que tipicamente garantem soluções de boa qualidade.

7.4. Representação temporal

Como visto no Capítulo 6, a linguagem estendida modela restrições temporais da operação dos equipamentos, mas não modela restrições relacionadas aos objetivos. Estas são herdadas pelas subtarefas durante a execução do algoritmo. Para a representação explícita do tempo, utiliza-se a abordagem de pontos no tempo, ao invés do Intervalo de Allen.

Uma vantagem da proposta de Allen é a representação das relações temporais qualitativas mais complexas entre as atividades do plano. A principal justificativa, segundo os autores (*e.g.*, MAYER et al., 2016) para empregar o Intervalo de Allen é alcançar maior flexibilidade para atuar em ambientes dinâmicos, onde certas condições do sistema não podem ser conhecidas em tempo de planejamento. No entanto, esse não determinismo não reflete a dinâmica real de operação de satélites. Observe a frase abaixo extraída de Ingrand e Ghallab (2017):

"*Um satélite autônomo tem um* ***ambiente estável*** *e* ***bastante previsível****. Possui um conjunto específico de tarefas, por exemplo, para usar seus dispositivos de imagem de forma eficiente, respondendo às demandas recebidas dos usuários, dada a cobertura de visibilidade esperada ou a presença de nuvens, e para gerenciar seus recursos de comunicação, processamento e energia.* ***Ele pode planejar e executar planos estáveis em horizontes longos***".

Após essa consideração, deve-se analisar outra solução utilizada pelos trabalhos e que está intimamente associada ao intervalo de Allen: a representação de restrições temporais por uma STN (vide Seção 3.5.3).

Embora uma HTN possa ser facilmente estendida para uma representação temporal das atividades em STN (vide AMIGONI et al., 2010), a HARPIA evita a sua adoção pelas seguintes razões:

- No domínio de satélites, usualmente, não há diversos caminhos temporais para se alcançar um mesmo objetivo; por exemplo, ao solicitar um imageamento ou

uma manobra orbital, não existem alternativas de caminhos temporais, mais curtos, que atendem ao mesmo objetivo;

- A manutenção e verificação de consistência de uma STN implica em uma sobrecarga computacional adicional ao algoritmo de planejamento;
- Conforme visto na Seção 3.5.3, a STN sozinha não resolve o problema de alocação de recursos.

Na solução de planejamento da HARPIA são modeladas as seguintes restrições temporais:

- Restrições de precedência previstas pela própria hierarquia da rede de tarefas;
- Restrições temporais absolutas advindas dos objetivos que são herdadas pelas subtarefas;
- Restrições temporais quantitativas intrínsecas à operação dos equipamentos.

7.5. Algoritmo de planejamento embarcado

Grosso modo, o algoritmo híbrido proposto nesta Tese tem três passos elementares: decomposição de métodos, inferência de estados e reparo por refinamento de estados. O primeiro é o processo de decomposição presente em todos os algoritmos HTN. O segundo e o terceiro são passos complementares entre si e diferenciais com relação a demais abordagens híbridas existentes, como será melhor discutido adiante.

A inferência de estados é utilizada para antecipar o comportamento das subtarefas, prevendo eventos para um horizonte de tempo, considerando também o cálculo de consumo de recursos. Isso, além de superar uma limitação das abordagens progressivas de ordem total que tipicamente não projetam estados futuros, unifica as etapas de planejamento e escalonamento das atividades.

Os momentos de inferência de estados são realizados em pontos específicos da rede de tarefas visando que o plano se prevaleça na sua forma hierárquica (não se permite inserir ações do modelo a partir de qualquer nó da rede), o que ajuda a manter a característica de ordem total do algoritmo.

A abordagem generativa utilizada permite adicionar tarefas abstratas ou primitivas ao plano para que uma solução viável seja alcançada para diferentes estados do satélite. Isso permite unificar planejamento e replanejamento usando o mesmo motor de busca baseado em HTN.

O Algoritmo 7.1 traz a lógica central do planejador da HARPIA. O plano solução é inicializado como vazio e as variáveis atualizadas pelas telemetrias correspondentes, conforme o modelo. Durante o decorrer do processo, T *e* S_i são atualizados pelo algoritmo para representar a rede de tarefas corrente que deve ser processada e o estado atual do satélite.

Em contraste ao planejamento clássico, não se define o estado meta. A descrição do objetivo está embutida de forma implícita a partir de uma rede que força os efeitos da última tarefa a ser executada como condição do estado objetivo do problema.

A lista de tarefas é iniciada na agenda de acordo com a temporização dos objetivos. O planejador, em princípio, pode deliberar sobre vários objetivos de uma só vez. O número de objetivos depende do horizonte temporal em que se deseja planejar (alguns minutos, uma órbita, quatro órbitas, etc.). Isso não é prefixado e pode ser ajustado conforme a necessidade de cada missão específica.

Algoritmo 7.1 – Algoritmo de Planejamento Liger

```
1 Procedimento HTN-Liger (s_o, T, Σ_HTN)
2 Entrada: Estado Inicial: s_o, Tarefas a ser processada na agenda: T, Domínio: Σ_HTN
3 Saída: Plano π
4 Início
5     π ← Ø;  // plano solução inicializa vazio
6     s ← s_o; // atualiza as variáveis de estados do modelo pelas telemetrias correspondentes
7     repita
8     início
9         T_top ← T; // tarefa a ser processada na agenda (no topo da pilha)
10        se T_top = Ø então retorne π;
11        s_t ← tempo inicial da T_top;
12        e_t ← tempo final da T_top;
13        g ← nó raiz da T_top;
14        c_t ← classe da T_top; // tarefa nominal ou reparadora
15        se T_top é uma ação primitiva então
16        início
17            T← T - T_top; // retira a tarefa do topo da agenda
18            se s satisfaz o conjunto de precondições de T_top então
19            s ← γ(s, T_top); // produz os efeitos da tarefa primitiva sobre as variáveis
20            π ← π + T_top; // insere T_top ao final do plano solução
21        fim
22        senão (T_top é uma tarefa composta)
23        início
24        s' ← s; // faz backup do estado do satélite
25        M ← método de T_top que satisfaça as precondições de s; // seleciona um método de Σ_HTN
26        T ← T - T_top; // retira tarefa da agenda
27        se T_top tem ações durativas então
28        início
29            T'_plist← SearchTasksTopDown(T_top , Σ_HTN, g); // encontra as subtarefas
30            Δ_t ← e_t - s_t;
31            // estratégia de comprometimento antecipado
32            ResourceStateInference(Δ_t, s ,T'_plist, g, Σ_HTN); //inferência p/ ações durativas que consomem recursos
33            se inferência falhou retorne (falha); // violação de recursos do satélite
34        fim
35        s ← γ(s, T'_plist); // inferência de estados, conforme os efeitos das tarefas {t_1, t_2, ... t_n} de T'_plist
36        se s satisfaz todas as precondições de T'_plist {t_1, t_2, ... t_n} então
37        DecomposeCompoundTask(s_t, e_t, c_t, T_top, g, s, Σ_HTN);
38        T ← T + subtarefas {t_1, t_2, ... t_n} de M inseridas em ordem total;
39        s ← s' ; //rollback dos estados do satélite
40        senão
41        // alcançou um estado não desejado, por exemplo, violação de alguma precondição de tarefa
42        StateRefinementRepair(s_t , e_t, T_top, g, s, Σ_HTN); // abordagem generativa
43        fim // tarefa composta
44 fim
45 fim
```

Fonte: Produção do autor.

Em função do conceito de pilha utilizado, as tarefas são removidas na ordem inversa na qual elas foram inseridas. Por exemplo, um método adiciona subtarefas que serão executadas antes que outras, já inseridas na agenda.

Para cada tarefa no topo da agenda (T_{top}), o algoritmo extrai as seguintes informações: tempo inicial e final, objetivo vinculado, e se é uma tarefa nominal (extraída da própria rede durante o processo de decomposição) ou uma tarefa de reparo (inserida pela etapa de refinamento).

Se a ação T_{top} for uma tarefa primitiva e satisfazer as condições para o estado atual do satélite, aplicam-se os efeitos nas variáveis de estados, adicionando-a no plano solução (π). Considerando o sucesso ou não desta operação, retira-se essa ação da agenda (linha 16). Sendo uma tarefa composta (linha 22), escolhe-se deterministicamente um método que atenda às precondições do estado atual (linha 25).

Se a tarefa composta tiver ações durativas, deve-se encontrar (*SearchTasksTopDown*) todas as tarefas primitivas (T'_{plist}) que estão abaixo na hierarquia de tarefas. Elas são utilizadas no processo de inferência para detectar alguma violação de recursos. Em seguida, efetua-se um processo de inferência de estados sobre essa mesma lista de ações primitivas, verificando-se as suas precondições: $S_1 \leftarrow \gamma(s_0, T_1)$, $S_2 \leftarrow \gamma(s_1, T_2)$ ... $S_n \leftarrow \gamma(s_{n-1}, T_m)$, onde *m* é o total de ações primitivas abaixo de T_{top} e S_n o estado final do satélite após a inferência.

Se ambos os processos de inferências foram bem-sucedidos, procede-se com a decomposição da tarefa (linha 37), adicionando as suas subtarefas (linha 38) na rede. Se o processo de inferência resultar em algum estado não desejado, violando, por exemplo, uma precondição de uma tarefa, deve-se iniciar um processo de reparo por refinamento (linha 42).

O algoritmo de planejamento continua de forma iterativa dentro de um laço de repetição até que não tenha mais tarefas a ser processada. O procedimento terá êxito se não restar tarefas na agenda e o plano final for composto somente por tarefas primitivas.

O *Liger* se beneficia da abordagem de busca progressiva, na qual o encadeamento de ações é alcançado impondo-se o ordenamento total das ações no plano. Isso garante que nenhuma ação atual a ser adicionada ao plano possa interferir com outras

condições ou efeitos de ações anteriores, nem mesmo que uma ação posterior possa interferir com as condições ou efeitos da ação atual.

Há uma prevalência dos planejadores HTN em segregar o processo de planejamento e escalonamento em duas etapas por simplificação. Isso se deve ao fato da modelagem em HTN carecer de mecanismos para representação conjunta das restrições temporais e de recursos (BIUNDO, 2002). Embora um estudo recente (QI et al., 2017) tenha sido realizado nesse sentido, ele foca em restrições temporais baseadas em intervalos e não considera a possibilidade de falhas, aproximando-se do modelo restrito do planejamento clássico. Uma contribuição do Liger é unir as etapas de P&S usando um planejador HTN, mas dispondo de técnicas de replanejamento de estados no mesmo algoritmo.

7.5.1. Estratégia de comprometimento

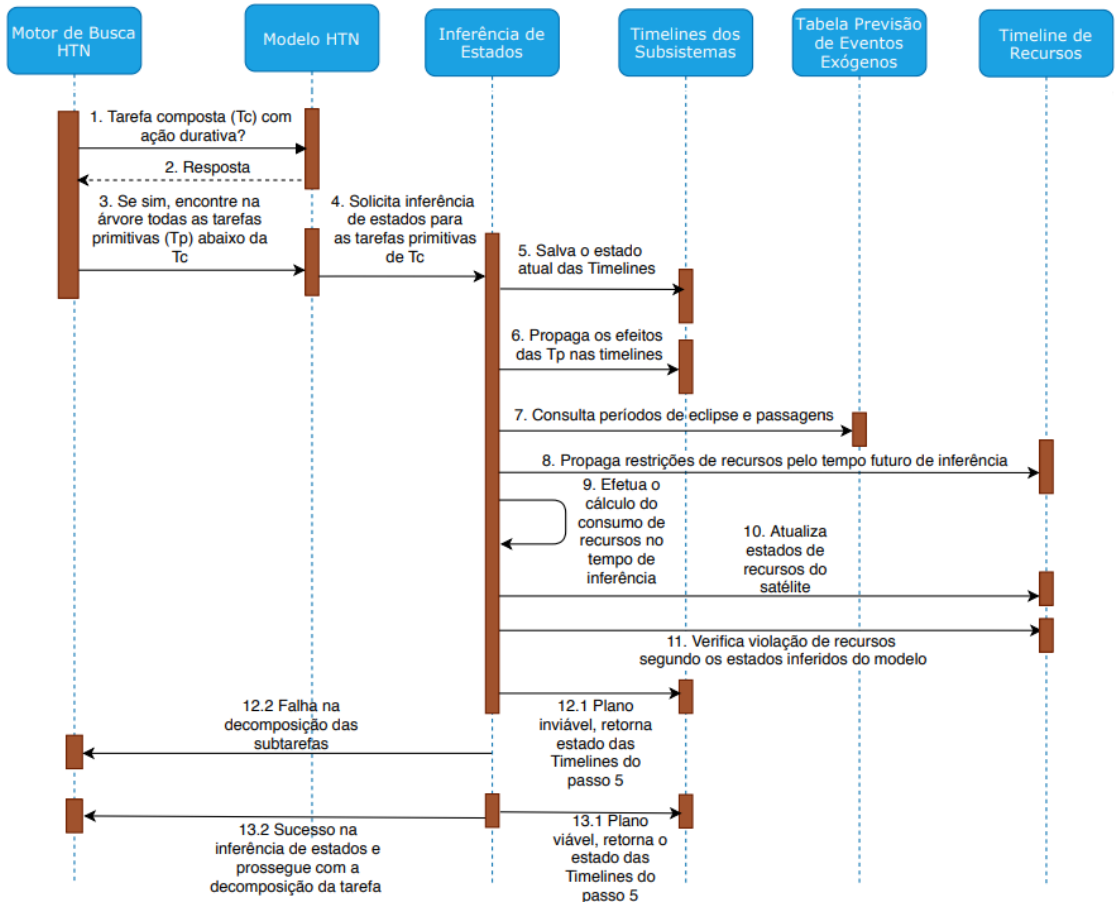
Embora a estratégia de comprometimento mínimo possa ser um recurso útil e frequentemente utilizado em algoritmos de ordem parcial, entende-se que ela é incompatível com a proposta de integrar planejamento e escalonamento num planejador HTN. Postergar decisões sobre restrições temporais e de recursos podem levar a estados incertos (*e.g.,* quantidade consumida de recursos) a depender da ramificação da árvore que o algoritmo HTN escolher.

Por essa razão, o *Liger* adota a estratégia de comprometimento antecipado, em que a qualquer instante é possível conhecer o estado atual dos recursos e dos estados dos equipamentos. Isso também evita a necessidade de dispor de um mecanismo para resolução de ameaças, além de simplificar o raciocínio sobre conflitos de recursos do satélite. Quando uma tarefa composta possui subtarefas com ações durativas que alteraram recursos, estes são alocados antecipadamente pelo motor de inferência, descrito a seguir.

7.5.2. Processo de inferência de estados

Para o processo de inferência de estados, utiliza-se a estratégia de comprometimento antecipado para alocação de recurso das atividades. Conforme a lógica apresentada pelo Diagrama de Sequência da Figura 7.3, o motor de busca verifica se a tarefa composta (T_c) é descrita no domínio como um ponto de inferência (vide Seção 6.4.1.1).

Figura 7.3 – Motor de inferência de estados para ações durativas.



Fonte: Produção do autor.

Nesse caso, executa-se um procedimento para encontrar todas as subtarefas primitivas (T_p) que estão abaixo de T_c na hierarquia. Por conseguinte, solicita-se um processo de inferência de estados para aquela lista de tarefas primitivas na mesma ordem em que foram encontradas pelo algoritmo de busca em profundidade. Armazena-se o estado atual do satélite, propagando-se os efeitos dessas subtarefas primitivas, bem como das restrições de recursos segundo o tempo futuro de

inferência. Nesse processo, o algoritmo consulta o período de eclipse e passagens referente àquela janela temporal, a fim de prever os eventos exógenos que possam alterar o comportamento da plataforma espacial. Neste ponto, também se deve verificar a possibilidade de violação de recursos relacionados a restrições temporais de eventos endógenos. No entanto, essas restrições não são propagadas às subtarefas durante o planejamento, conforme visto na Seção 5.5.2.

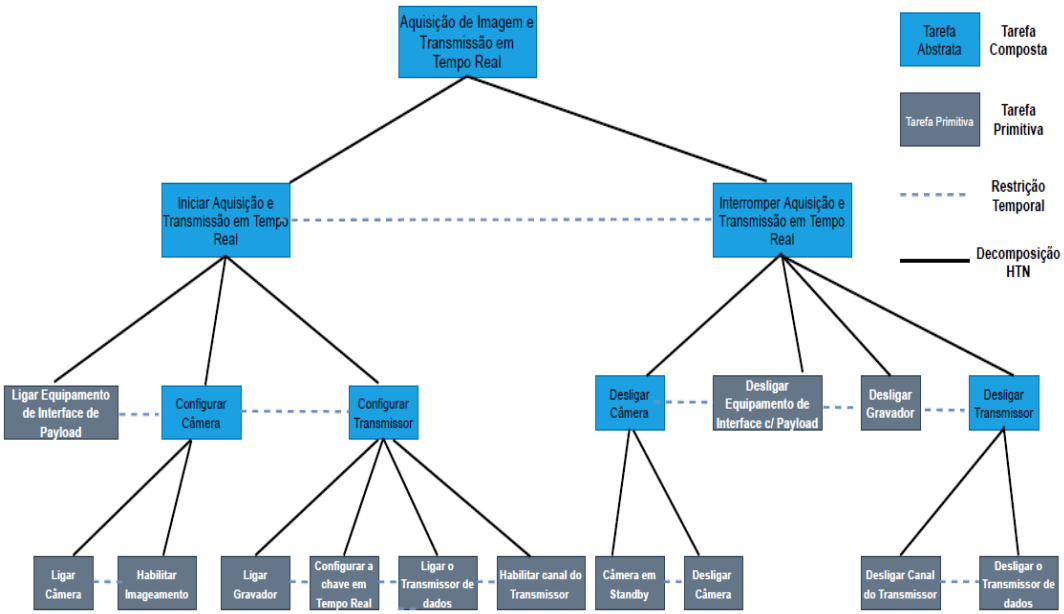
A partir disso, realiza-se o cálculo de consumo de recursos pelo tempo inferido, atualizando-se o perfil de recursos do satélite. Logo após, é verificado se houve violação de recursos. Em caso de sucesso, prossegue-se com a decomposição, do contrário, determina-se uma falha. Independentemente do resultado desta operação, os valores das variáveis são retornados ao estado inicial. Eles serão modificados após o algoritmo proceder com a decomposição e inserir a tarefa primitiva no plano. Note-se que não será preciso recalcular os consumos das atividades posteriormente quando o processo de decomposição de fato for realizado.

Cabe destacar que as tarefas abstratas indicadas para os 'momentos de inferências' pela HML são filhas do nó raiz e hierarquicamente superior às demais tarefas na árvore. A possibilidade de fazer a predição antecipada de estados a partir da tarefa de maior nível hierárquico decorre da característica de ordem total da HARPIA.

Isso também se torna viável graças a uma característica positiva da HTN, na qual é possível estabelecer níveis de abstração e relações temporais. Por exemplo, da análise da Figura 7.4, observa-se, que a rede de tarefas "Iniciar Aquisição de Imagem e Transmissão em Tempo Real" tem subtarefas agrupadas no mesmo nível de abstração[40] que possibilita a inferência antecipada de um conjunto de tarefas relacionadas sob o ponto de vista operacional.

[40] Embora esteja agrupada no mesmo nível de operação, não há impedimento para que as subtarefas tenham relações de precedências e restrições temporais entre si

Figura 7.4 – Exemplo de uma HTN para o domínio de um satélite.



Fonte: Produção do autor.

Em uma HTN, também é possível estabelecer níveis de relações temporais entre as ações, por exemplo, a tarefa "Iniciar Aquisição de Imagem e Transmissão em Tempo Real" apresenta uma relação temporal de precedência sobre “Interromper Aquisição de Imagem e Transmissão em Tempo Real”.

Caso seja preciso adaptar o objetivo para inserir novas atividades abstratas com outras precedências temporais, elas poderiam ser incluídas como tarefas abstratas filhas da rede inicial sem comprometer o modelo e o algoritmo de planejamento proposto por esta Tese.

Um exemplo poderia ser adicionar uma nova tarefa "*Realizar Manobra de Apontamento*[41]", para o objetivo desta figura, considerando uma missão de sensoriamento remoto ágil. Tal ação estaria modelada mais à esquerda e no mesmo nível hierárquico de que as demais tarefas citadas, cada qual com seu respectivo horizonte temporal, e pontos de inferências de estados.

[41] Para o alvo terrestre desejado.

7.6. Reparo e replanejamento

Um tópico de grande interesse atualmente é capacidade do sistema em reparar um plano. Segundo Ingrand e Ghallab (2017), revisão e reparo de um plano em situações de falhas são geralmente mais importantes na prática do que propriamente sintetizar novos planos a partir de uma agenda vazia.

Embora na literatura os termos 'reparo' e 'replanejamento' por vezes sejam tratados como conceitos equivalentes, a presente Tese segue a classificação conceitual de Holler et al. (2018):

- **Replanejamento**: O plano antigo é descartado e um novo plano é gerado a partir do estado atualizado do sistema que ocasionou a falha de execução. De forma mais específica, um novo plano é criado do zero.
- **Reparo**: O sistema modifica a parte não executada da solução original, a fim de lidar com mudanças de estados não previstas. Em outras palavras, adapta-se o plano original para o novo contexto percebido.

Na solução implementada pela HARPIA, as duas abordagens podem ser utilizadas em situações distintas:

- Durante a fase deliberativa, o algoritmo se depara com uma situação não prevista pelo domínio hierárquico, e aplica **reparo** no plano, modificando a solução original.
- Durante a fase de execução, a precondição de uma ação falha e a consecução do plano é interrompida. Uma solicitação de **replanejamento** é enviada à camada deliberativa. Nesse caso, um novo problema é encaminhado ao planejador com o estado atualizado do satélite.

Em ambos os casos, utiliza-se a mesma técnica de reparo na HARPIA. O algoritmo proposto lida com não determinismo do ambiente, porém a partir de ações determinísticas para os quais a técnica de reparo possa levar o satélite a satisfazer seus

objetivos com sucesso. A grande vantagem disso é não utilizar o planejamento condicional.

O plano nessa abordagem deve prever ramos de contingências para os diferentes estados possíveis, que o ambiente pode apresentar após executar uma ação. Segundo Lemai (2004), o planejamento condicional é considerado uma técnica custosa computacionalmente para ser executada a bordo de veículos espaciais. Especificar ações de contingência para um caso esperado fica mais próximo de programar uma aplicação do que um sistema de planejamento automatizado. A política de contingência também tem o contratempo de lidar apenas com o não determinismo limitado do ambiente.

7.7. Algoritmo de reparo por refinamento de estados

O algoritmo de reparo por refinamento de estados é centrado na abordagem generativa do planejamento clássico baseado em ações e tem função complementar ao processo de inferência de estados que integra o motor de busca HTN.

Utiliza-se uma heurística de distância ao objetivo como critério para a seleção das ações a partir da manutenção de históricos das decisões anteriores baseado no registro de falhas das precondições das ações e explorando-se a abordagem de ordem total do algoritmo.

A lógica de reparo de refinamento de estados da HARPIA está representada pelo Algoritmo 7.2. Seus principais parâmetros são: estado atual, tarefa composta em falha (T_c), domínio Σ_{HTN} e histórico *H* de execução anterior à falha.

Dentro de um laço de repetição, escolhe-se de forma determinística uma tarefa reparadora para T_c. Se for uma tarefa primitiva faz-se um *backup* do estado das variáveis para aplicar os efeitos desta tarefa reparadora.

Algoritmo 7.2 – Reparo por Refinamento de Estados

```
1 Procedimento StateRefinementRepair (s, Tc, g, T, ΣHTN, H)
2 Entrada: Estado:s, Tc: tarefa composta em falha, g:objetivo, T:lista de tarefas, ΣHTN: Domínio, H: Histórico
3 Saída: Tr = Tarefa de reparo de estados para Tc
4 Início
5     Tr ← Ø;
6     st ← tempo inicial da tarefa Tc em falha;
7     et ← tempo final da tarefa Tc em falha;
8     repita
9     início
10        Tr ← escolha deterministicamente uma tarefa de refinamento para Tc
11        se não existem outras tarefas de reparo para Tc então
12         retorne (falha);
13        senão
14        início
15          se Tr for uma tarefa composta então
16          pare laço;
17          senão (tarefa reparadora é primitiva)
18          início
19          s' ← s; // faz backup das variáveis de estados do modelo
20          s ← γ(s, Ttop); // aplicam-se os efeitos da tarefa primitiva de refinamento
21          T'plist← SearchTasksTopDown (Tc , ΣHTN, g); // tarefas primitivas abaixo de Tc
22          s ← γ(s, T'plist); // conforme os efeitos das subtarefas primitivas {t1, t2, ... tn} de T'plist
23          H' ← novo histórico de execução dado pelo passo anterior;
24          CompareGoalDistance(s, H, H' ); // compara os dois históricos de execução
25          s ← s'; //rollback dos estados do satélite
26          se H ' de Tc ficou mais próxima do objetivo que H então
27          pare laço;
28          fim
29        fim
30    fim
31    PutTasksOnPlan (T, Tr , st , et , g); // adiciona tarefa de refinamento na agenda de tarefas (T)
32 fim
```

Fonte: Produção do autor.

Com estado atualizado, encontram-se todas as tarefas abaixo de Tc (linha 18). Em seguida, efetua-se o processo de inferência das ações primitivas (T'_{plist}), verificando-se as suas precondições, conforme: $s_1 \leftarrow \gamma(s_0, T_1)$, $S_2 \leftarrow \gamma(s_1, T_2)$ ... $S_n \leftarrow \gamma(s_{n-1}, T_m)$, sendo *m* o total de ações primitivas abaixo de T_c. Após essa execução, têm-se agora dois históricos de execução: um em falha antes de invocar a rotina de reparo e outro atualizado após a execução da tarefa reparadora.

Esses históricos são encaminhados a uma função que avalia se o novo histórico H' se aproximou mais do objetivo, considerando as precondições de falha de cada ação. Caso tenha resolvido alguma precondição aberta, adiciona-se a tarefa reparadora ao

plano, do contrário, ela não será considerada em uma primeira etapa do algoritmo de refinamento.

Isso é útil para tentar maximizar a estabilidade do plano, não incluindo ações que não colaboram para recuperação do estado do satélite para aquele objetivo em falha. Também é especialmente importante para o domínio de satélites que possuem ações finitas que delimitam a vida útil da missão.

O procedimento finaliza quando não há mais tarefas reparadoras para aquela tarefa em falha. Tendo encontrado uma ação reparadora ou não, o estado do satélite é retornado. Note-se que não há garantias que uma única tarefa de reparo poderá chegar ao objetivo.

Considere que para um dado objetivo, o domínio HTN escrito espere que o gravador esteja desligado, mas na realidade ele se encontra ligado e no modo gravação. Uma primeira tarefa de reparo pode retirá-lo do modo gravação, se aproximando mais do objetivo. Porém, ainda será preciso de outra subsequente para desligá-lo.

Nessa situação, o Algoritmo 7.1 irá mais uma vez falhar para a mesma tarefa Tc, porém agora mais próximo do objetivo. Novamente a rotina de refinamento de estados será acionada, a qual irá escolher outra tarefa. Em algum momento, a tarefa que desliga o gravador será inserida no plano, fazendo com que a tarefa em falha consiga ser executada. É importante notar que o processo é iterativo, as falhas podem ser corrigidas gradativamente por refinamento de estados, até que se encontre um plano de recuperação.

Uma vantagem da proposta da HARPIA é usar as mesmas tarefas do domínio como tarefas de reparo, sem requerer esforços adicionais para modelagem e outras técnicas de resolução de falhas específicas ao tipo de conflito encontrado[42]. Aliás, o Algoritmo 7.2 também permite inserir tarefas abstratas, isso dá flexibilidade ao algoritmo para refinar o estado do satélite usando métodos do próprio processo de planejamento,

[42] Isso é feito, por exemplo, pelo algoritmo de reparo iterativo usado pelo CASPER.

que já possui ações de mais alto nível que encapsulam uma atividade operacional do satélite já validada pelo especialista.

Para melhor compreender a expressividade que isso pode proporcionar, imagine que o subsistema de transmissão do satélite esteja ligado com canal de transmissão habilitado e para um determinado objetivo o mesmo deveria estar desligado.

Em vez de o algoritmo tentar encontrar um conjunto de ações primitivas que desligue o subsistema adequadamente (seja de forma exaustiva ou por heurísticas de resolução de falhas), é possível incluir no plano uma ação abstrata ("Desligar Transmissor de Dados"), que contém todo procedimento para efetuar esta operação.

Em princípio, qualquer tarefa abstrata independente de seu nível hierárquico pode ser adicionada no processo de reparo. Isso tende a aumentar o poder de reação do algoritmo a situações não previstas.

É importante observar que se a tarefa de refinamento for uma tarefa abstrata, não se faz necessário passar pelas etapas de inferência de estados e análise de estabilidade. Ela é imediatamente adicionada ao plano como uma tarefa de reparo (linha 13). A partir disso, o próprio algoritmo HTN se encarrega de proceder com sua validação, considerando o estado atual.

Como o algoritmo é de ordem total, as precondições das subtarefas verificam a aplicabilidade desta tarefa ao estado em falha, do contrário, ela será removida pelo próprio esquema de redução de tarefas.

Em certas situações, tarefas abstratas não conseguem refinar o estado do satélite. Isso ocorre, pois, as tarefas compostas conduzem um conjunto maior de atividades, que executam em geral ao menos duas ou três ações associadas. Há diversas situações, entretanto, que o objetivo é alcançado com o refinamento de uma única ação primitiva e por isso a importância das tarefas primitivas no processo.

A proposta da combinação de tarefas primitivas juntamente com ações abstratas se posiciona como um recurso interessante para aumentar a capacidade do algoritmo em

recuperar o estado do satélite, contornando uma das grandes limitações do planejamento em HTN.

A abordagem híbrida sobre a qual opera o algoritmo de refinamento também alivia uma das grandes críticas à representação em HTN: o alto custo para a descrição do domínio. Nela, o planejador deixa de depender exclusivamente de um conjunto de métodos para atender diferentes estados do satélite.

De forma sumária, o processo de refinamento do Liger tem quatro passos:

1. Seleção de Tarefas: Escolhe-se uma tarefa de reparo ao plano, executando-se os efeitos dela no ambiente; se for uma tarefa primitiva continuar com o passo seguinte, do contrário, ir ao Passo 4;

2. Inferência de Estados: encontram-se as subtarefas filhas da tarefa abstrata em falha, executam-se seus efeitos no ambiente e em seguida armazena-se o histórico dessa execução;

3. Análise de Distância ao Objetivo: compara-se o histórico de execução anterior com o atual; essa etapa é crucial para não inserir no plano ações não relevantes ou mesmo prejudiciais à solução;

4. Inserção da Tarefa: se a tarefa resolveu alguma precondição aberta do problema, adiciona-se ao plano (linha 28) como uma tarefa de refinamento; deve-se retornar o estado do satélite anterior à etapa de inferência para proceder com o algoritmo.

Se o processo descrito acima não conseguir obter um plano viável, procede-se em seguida com uma segunda heurística integrada ao algoritmo de refinamento. Basicamente, repete-se o mesmo procedimento apresentado, entretanto, dessa vez com um relaxamento na fase de estabilidade.

Os resultados obtidos neste trabalho (Capítulo 8) demonstram que isso é importante para manter o algoritmo mais responsivo a determinadas situações.

Por último, é relevante mencionar que o algoritmo é sistemático; ele não escolhe a mesma tarefa mais de uma vez dentro de cada etapa de refinamento. Isso garante que o algoritmo não atinja um nível de altura muito profundo durante o processo de replanejamento.

Se ao final restarem apenas tarefas primitivas incluídas via reparo na solução, significa que as ações de refinamento embora tenham conseguido se aproximar mais do objetivo, não foram suficientes para recuperar o estado do satélite. Nesse caso, devem-se retirar as tarefas remanescentes, configurando-se uma falha.

Após a etapa deliberativa, o algoritmo Liger é capaz de informar as seguintes informações: (i) objetivos realizados com sucesso; (ii) objetivos que falharam; (iii) tipo de falha: restrições de estados, recursos ou de eventos exógenos; (iv) tarefas primitivas que compõem o plano final e seus instantes de execução; (v) tarefas compostas que falharam durante a decomposição; (vi) tarefas de reparo inseridas com sucesso no plano; (vii) total de consumo de recursos, e (viii) quantidade de nós visitados durante a busca.

7.8. Análise comparativa

A Tabela 7.2 traz uma análise comparativa do *Liger* em relação aos sistemas de planejamento voltados a aplicações embarcadas da área espacial.

Tabela 7.2 – Comparativo e estratégias de planejamento.

Trabalho	Planejamento	Estratégia	Linguagem	RT	TL	MPU	SPI
Muscettola et al. (1998)	Timeline (CSP) Temporal	Busca em Profundidade	PDDL	IA	Sim	Não	Não
Knight et al. (2001)	Timeline (CSP) Temporal	Reparo Iterativo	Baseada na AML	IA	Sim	Não	Não
Kucinskis e Ferreira (2013)	Baseado em CSP (Timeline)	Similar ao Reparo Iterativo	ISISml	TP	Sim	Sim	Não
Liger	HTN e Planejamento Generativo	Busca Progressiva com Decomposição de Ordem Total	HML+ISISml	TP	Não	Sim	Não

RT = Representação Temporal, TL = *Timelines,* MPU = Modelo de Planejamento Unificado, SPI = Sistema de Planejamento Independente de domínio, IA = Intervalo de Allen, TP = *Time Points,* AML = ASPEN *Modeling Language,* HML = *Hierarchical Modeling Language,* ISIS = *Internal State Inference Service.*

Fonte: Produção do autor.

Uma grande vantagem dos planejadores CSP é que uma instância de planejamento é sempre convertida em um problema de satisfação de restrições de estados do satélite. Se a intenção é planejar ou replanejar, a maneira de resolução do problema é a mesma. À exceção da possibilidade de construção de heurísticas, não existe uma solução específica para lidar com as técnicas de reparo. Por outro lado, é justamente nesse aspecto que tem seu maior contraponto.

Os resultados relatados de desempenho dos sistemas, quando executados em hardware espacial, são da ordem de dezenas de minutos para reparar um plano e em alguns cenários até horas.

Isso ocorre porque o planejador precisa satisfazer restrições adicionais dentro do mesmo problema, incrementando-se sensivelmente o tempo de execução. Isso configura planejamento em tempo “quase” real e inviabiliza atender a demandas percebidas em órbita que necessitam de reações imediatas.

Uma contribuição importante do algoritmo proposto da HARPIA é manter o tempo de execução da ordem de milissegundos mesmo para os casos de reparos ao plano. Para

lidar com estados não previstos, a HARPIA propõe uma estratégia híbrida (integrada ao algoritmo de planejamento) cujas vantagens estão descritas a seguir.

- O planejamento generativo do algoritmo em HTN proposto não visa substituir a solução hierárquica embutida no domínio. Em outras palavras, uma solução via método de decomposição sempre estará contida na solução híbrida ($S_{HTN} \subseteq S_{HYB}$) encontrada pelo Liger. Isso traz mais confiabilidade, mantendo-se sempre a descrição operacional do especialista na solução;
- O planejamento generativo é acionado em pontos específicos da HTN devido à estratégia de comprometimento antecipado. Isso faz com que a abordagem se aproxime mais do planejamento hierárquico do que a do planejamento generativo, mantendo-se a boa performance do planejador HTN guiado pelos métodos de decomposição;
- A carga computacional adicional da técnica de reparo incorporada no algoritmo Liger só é executada nos casos em que os métodos HTN se depararem com situações não previstas; isso mantém constante o tempo de execução nas situações nominais de operação;
- Soluções híbridas existentes fora do domínio de satélites utilizam HTN e POCL (vide GEREVINI et al., 2008; BECHON et al., 2014; DVORAK et al., 2014). O processo para resolução de falhas de precondições abertas e vínculos causais degradam a performance do algoritmo.

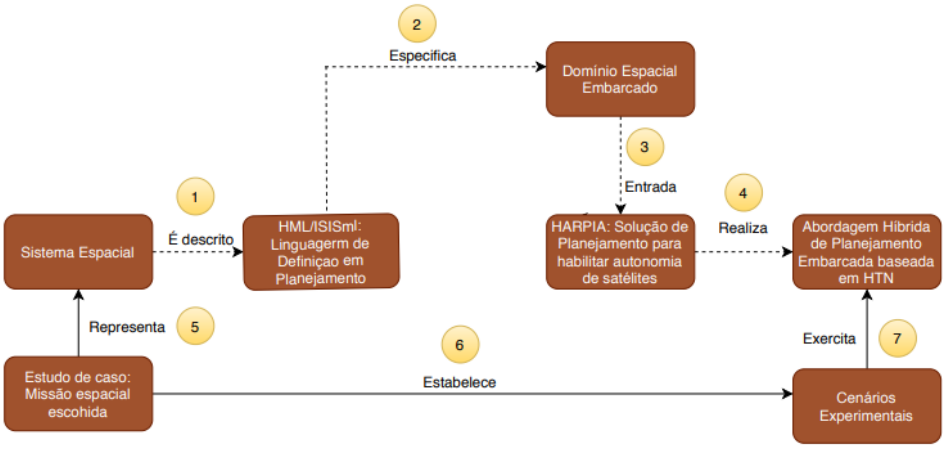
Uma desvantagem da abordagem proposta frente ao algoritmo de reparo iterativo é que este tem maior capacidade para lidar com falhas relacionadas a restrições temporais devido à sua capacidade de mover temporalmente as ações. No entanto, é importante observar que tipicamente a grande preocupação na operação de um satélite está com seus estados, sendo o tempo das ações uma consequência do processo.

8 ESTUDO DE CASO E CENÁRIOS EXPERIMENTAIS

Este capítulo apresenta a descrição de estudo de caso para a HARPIA e definição de cenários experimentais. Observe o diagrama da Figura 8.1 (mais especificamente as setas dois, cinco, seis e sete) para melhor compreensão de como eles se relacionam no escopo desta Tese.

O objetivo deste capítulo é exercitar o funcionamento da abordagem de planejamento, considerando-se o desdobramento das seguintes etapas: especificação do problema na linguagem estendida, instância do modelo embarcado construído, carregamento do plano, verificação e execução das ações.

Figura 8.1 – Proposta do estudo de caso no contexto dos objetivos desta Tese.



Fonte: Produção do autor.

A seguir, apresenta-se a definição do ambiente embarcado de operação da arquitetura, missão alvo para modelagem e cenários relacionados ao estudo de caso da solução híbrida de planejamento proposta.

8.1. Ambiente computacional embarcado

Trabalhos que não consideraram nos estudos preliminares um ambiente embarcado compatível com o ambiente espacial acabaram realizando validações experimentais em ambientes computacionais cujas soluções são tipicamente comerciais.

Atualmente o INPE conta com um projeto de Pesquisa e Desenvolvimento intitulado Computador Avançado (COMAV), sob responsabilidade do grupo de Supervisão de Bordo (SUBORD) da Divisão de Eletrônica Espacial e Computação (DEC), pertencente à Coordenação Geral de Engenharia, Tecnologia e Ciência Espaciais.

Trata-se de um computador de bordo para satélites cuja finalidade essencial é unificar as funções de Supervisão de Bordo (OBDH) e de Controle de Atitude e Órbita (*Attitude and Orbit Control,* AOC), usualmente realizadas por dois ou mais computadores em outras missões do INPE, de acordo com Arias et al. (2008).

Apenas como exemplo, a arquitetura do OBDH do satélite *China-Brazil Earth Resources Satellite* (CBERS) do INPE, que pode ser vista com mais detalhes em Wang et al. (2017), além de não possuir essas funcionalidades unificadas em um só computador, apresenta uma arquitetura roteada baseada no paradigma mestre-escravo.

O COMAV é baseado em um processador de aplicação espacial ERC32 (arquitetura SPARC V7 de 32 bits) resistente à radiação, cuja versão de baixo consumo elétrico pode operar a uma frequência de até 15 MHz, com 4 MBytes de memória RAM; utiliza-se o sistema operacional de tempo real *Real-Time Executive for Multiprocessor Systems* (RTEMS).

Além de modelos de engenharia desse computador, o SUBORD possui um ambiente de desenvolvimento de software com um simulador do processador ERC32, projetado pela ESA, intitulado SPARC *Instruction Simulator* (SIS), que foi utilizado no desenvolvimento deste trabalho.

Outra motivação para adotá-lo é sua compatibilidade com a PMM do INPE. Essa plataforma é uma estrutura de uso geral criada pelo Instituto para a fabricação de satélites na classe de 500 quilos de massa total cuja órbita está na faixa de 600 a 1000 quilômetros. Pode ser empregada por diferentes tipos de satélites, como missões de sensoriamento remoto, científicas e outras que dispõem de tais características. O programa Amazônia do INPE é uma missão de sensoriamento remoto, a primeira a adotar a plataforma PMM.

8.2. Missão espacial do estudo de caso

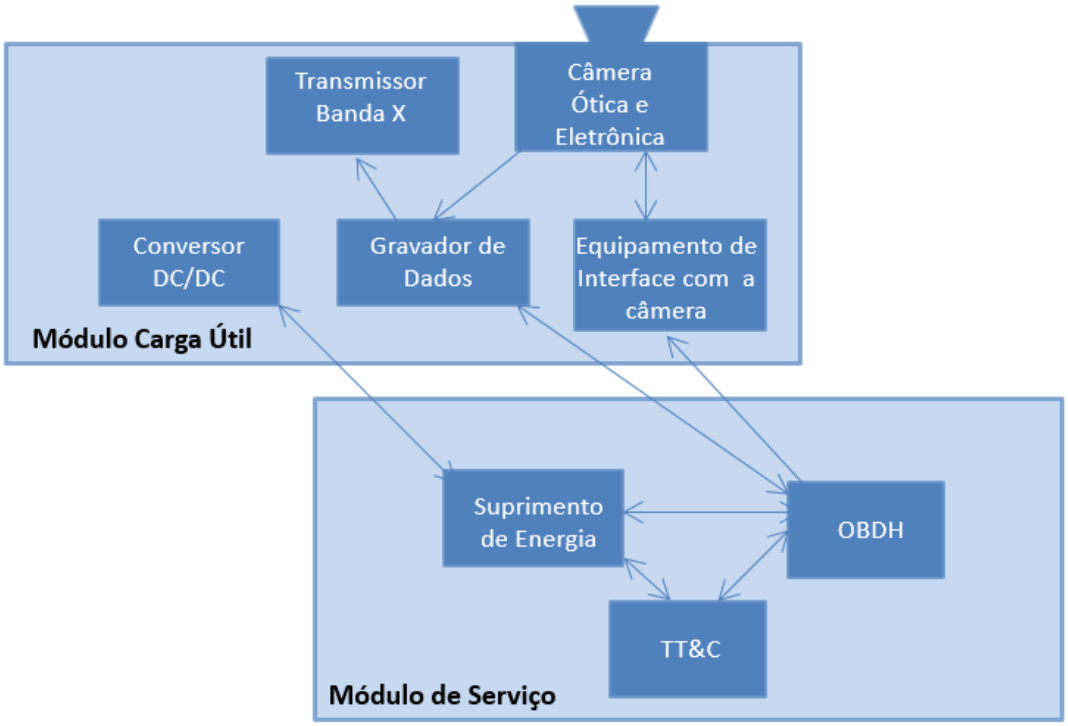
Como os principais satélites do INPE são de Observação da Terra, é natural que se pense em uma missão desse segmento. Para o estudo de caso, escolheu-se a missão Amazonia-1 (Figura 8.2) baseada na PMM, cujo objetivo é realizar a observação da Terra especialmente para melhorar a capacidade de monitoramento do desmatamento na região Amazônica.

Figura 8.2 – Perspectiva do satélite Amazonia-1.

Fonte: INPE (2020).

O satélite está previsto para ser lançado e injetado em uma órbita polar heliossíncrona a uma altitude aproximada de 760 quilômetros. Não diferente de outros satélites de Sensoriamento Remoto que se situam nessa altitude, seu período orbital é em torno de 100 minutos, sendo o tempo em eclipse próximo a um terço do período total da órbita. A Figura 8.3, traz uma representação simplificada deste satélite, mostrando os principais componentes referentes aos módulos de carga útil e serviço.

Figura 8.3 – Representação simplificada do módulo carga útil e da plataforma de serviço do Amazonia-1.



Fonte: Produção do autor.

A seguir descrevem-se sumariamente os elementos principais que compõem a carga útil da missão.

A carga útil do satélite é composta por uma câmera denominada *Wide Field Imager* (WFI) do tipo *Charged-Coupled Device* (CCD). Trata-se de um imageador óptico de varredura cujo campo de visada é capaz de observar uma faixa terrestre de 866 quilômetros com resolução espacial de 64 metros na direção do Nadir (INPE, 2013).

A comunicação entre computador de bordo e câmera é intermediada por um equipamento de interface denominado *Remote Terminal Unit* (RTU).

No sistema de controle há um receptor de *Global Positioning System* (GPS) que determina a posição exata do satélite no momento em que uma área da superfície terrestre é imageada pela câmera WFI da missão.

Há também um gravador de dados digital (do inglês: *Digital Data Recorder - DDR*) responsável por armazenar em bordo as imagens fora de visada e um Transmissor de Dados (do inglês: *Data Transmitter* - DT) que transmite as imagens em Banda X para a estação terrena durante uma passagem. Ele pode transmitir uma imagem adquirida em tempo real ou armazenada no gravador de bordo.

Da plataforma de serviço destacam-se o subsistema de suprimento de energia (*Power Conditioning and Distribution Unit* - PCDU), o subsistema de Controle de Atitude e Órbita e Supervisão de Bordo (ACDH) e o subsistema de Telemetria e Telecomando (TT&C) que detém o *transponder* para comunicação com a estação terrena, conforme a Figura 8.3.

8.3. Considerações sobre a modelagem do Amazonia-1

Para a modelagem do estudo de caso, estudaram-se os documentos técnicos da missão Amazonia-1, a fim de compreender o modo de operação e funcionamento dos equipamentos e subsistemas. Foram modelados os elementos do sistema espacial que determinam o comportamento da operação da carga útil desta missão.

A seguir, faz-se uma descrição sumária de cada elemento, tomando-se como base aspectos relevantes para a sua modelagem, tais como: modos de operação, restrições de transição de estados e consumo de recursos.

- **Unidade Terminal Remota (RTU):** A unidade de interface com a carga útil não dispõe de modos de operação e seu funcionamento se resume a estar ligado ou desligado. Para comandar a câmera, é necessário primeiro ligar a RTU. Todos os comandos passam pela unidade de interface antes de chegar à câmera.
- **Câmera (WFI):** A WFI tem quatro modos operacionais: *power-off*, *stand-by*, *imaging* e *calibration*. No modo *standby*, ela está apenas energizada. Os demais modos são para aquisição nominal de imagens e para fins de calibração. Para comandar o modo calibração, a câmera deve estar em imageamento.
- **Gravador de Dados (DDR):** Embora o DDR tenha mais modos de operação, ele pôde ser abstraído a partir de cinco modos, mas sem perder suas propriedades

funcionais: *power-off*, *stand-by*, *recording*, *playback* e *erase.* Para fazer gravação de imagens fora de visada, o DDR deve estar em *recording*, enquanto que para o *download* de dados, deve estar em *playback*. O gravador entra no modo *erase,* ao receber um comando para apagar o arquivo. O DDR possui ainda um modo denominado '*starting-up*' (não modelado) que corresponde a um estado intermediário entre a inicialização do equipamento e o modo *standby.* Também existe uma chave eletrônica interna denominada *Digital Signal Switch* (DSS) que informa ao transmissor se as imagens são provenientes da câmera (*real-time*) ou do próprio gravador (*playback*). Para operação do gravador, a câmera precisa estar ligada devido a uma restrição de projeto[43].

- **Transmissor de Dados (DT):** O transmissor tem três modos de operação: *power-off*, *standby* e nominal. Para transmissão de imagens, ele deve estar no modo nominal, o qual é alcançado a partir do modo *stand-by*. O DDR deve estar ligado para o DT realizar a transmissão de imagens, independentemente se os dados advêm da câmera ou do gravador.
- **Unidade de Condicionamento e Distribuição de Potência (PCDU)**: faz parte do subsistema de suprimento de energia do satélite. Esta unidade é responsável por fornecer e condicionar energia aos equipamentos da plataforma de serviço e da carga útil.

De forma geral, os equipamentos se encontram energizados e aptos a receber comandos de operação no modo *stand-by*, sendo este o primeiro modo após a sua inicialização. Com relação aos recursos do satélite, foram considerados para este estudo de caso dois recursos:

Memória: A memória do gravador é um recurso consumível. A taxa de dados da câmera é da ordem de 51 megabits por segundo que são consumíveis em aproximadamente 52 minutos de imageamento para os 160 *gigabits* disponíveis. O nível de memória não deve atingir sua capacidade máxima em nenhum cenário de

[43] O hardware do gravador utiliza um *clock* da câmera.

planejamento. Ao final do *playback*, os dados podem ser apagados e a capacidade do gravador restaurada.

Energia: A energia é um recurso renovável e limitada pela sua capacidade. Nesse estudo de caso, estabeleceu-se um nível máximo para a operação dos equipamentos da carga útil, considerando-se: 200 W para o período de eclipse e 250 W durante o período solar da órbita. Isso impossibilita que a câmera e o transmissor sejam ligados ao mesmo tempo pelo planejador quando a bateria do satélite não estiver sendo carregada.

A Tabela 8.1 traz o consumo de cada subsistema modelado considerando seus diferentes modos de operação. Para casos em que o equipamento possui mais de um consumo para o mesmo modo de operação (*e.g.*, *heaters* ligados consomem mais energia), considerou-se o de maior valor.

Tabela 8.1 – Consumo dos equipamentos modelados para o estudo de caso.

Elemento	Modo	Consumo
RTU	*Power-on*	10 W
Câmera	*Stand-by*	30 W
	Imaging	81 W
	Calibration	85 W
Transmissor	*Stand-by*	107 W
	Nominal	121 W
Gravador	*Stand-by*	23 W
	Record	28 W
	Playback	28 W

Fonte: Produção do autor.

Com relação às restrições operacionais entre as transições de modos dos equipamentos, foram modeladas duas restrições temporais, conforme a Tabela 8.2. Note-se que a restrição do gravador é intrínseca ao funcionamento do próprio hardware, ao passo que a do transmissor é uma restrição operacional. Apesar disso, ambos são modelados da mesma forma, como será visto adiante na Seção 8.5.3.

Tabela 8.2 – Restrições temporais na operação dos equipamentos.

Subsistema	Restrição temporal
Gravador de Dados	Após ligado, o gravador leva 10 segundos antes de entrar efetivamente no modo '*stand-by*'
Transmissor de Dados	Uma vez que o transmissor de dados entrar no modo *stand-by*, deve-se esperar 240 segundos (tempo de pré-aquecimento do TWT necessário para sua operação nominal) antes de comandá-lo para o modo nominal

Fonte: Produção do autor.

8.4. Objetivos do estudo de caso

O estudo de caso deste trabalho teve dois propósitos. O primeiro é descrever o domínio da missão Amazonia-1 utilizando-se a linguagem estendida apresentada nesta Tese. O segundo é avaliar o comportamento da HARPIA em cenários experimentais de planejamento. Para o experimento, considerou-se a fase de operação de rotina do satélite. Trata-se da maior fase do ciclo de vida operacional de uma missão de Sensoriamento Remoto.

8.5. Estudo de caso da linguagem estendida

A linguagem de definição (HML/ISISml) pode ser explorada usando seus diferentes recursos de planejamento em IA. Neste estudo de caso, considerou-se sua descrição para o planejamento proposto pela solução híbrida desta Tese cuja abordagem é de ordem total.

8.5.1. Especificação hierárquica

Como visto no Capítulo 6, a descrição hierárquica é definida pelos objetivos, tarefas abstratas e primitivas, métodos e suas decomposições. A seguir, apresenta-se a modelagem construída para este estudo, considerando os três elementos dessa descrição.

Os objetivos na HARPIA são restritos e conhecidos antecipadamente em tempo de descrição de domínio. Para este estudo de caso, foram definidos seis objetivos que estão relacionados com os usuários finais da missão:

- Aquisição e transmissão de imagem em tempo real;
- Aquisição e gravação de imagem em bordo;
- Reprodução das imagens gravadas em bordo;
- Aquisição e transmissão em tempo real de imagem no modo calibração;
- Aquisição e gravação de imagem no modo calibração; e
- Aquisição, gravação e transmissão de imagem em tempo real.

Os objetivos foram modelados para descrever tipicamente um par de tarefas, onde a primeira inicializa as atividades daquele objetivo e a segunda finaliza o processo, retornando o satélite na sua configuração inicial[44], conforme mostra a Figura 8.4. Cada objetivo tem a sua própria rede e formas de decomposição.

Um objetivo pode ter diferentes níveis de abstração em HTN. Para melhor compreensão disso, imagine um objetivo "Aquisição de Imagem" descrito em alto nível decomposto pelas subtarefas: (i) "*Aquisição de Imagem e Transmissão em Tempo Real*" ou (ii) "*Aquisição de Imagem e Gravação em bordo*". A depender se o satélite está em visada ou não, o planejador pode decidir se irá transmitir ou gravar. Isso também poderia ser modelado pela HML graças à sua capacidade em descrever precondições em tarefas abstratas.

[44] Para este estudo, ao final da realização dos objetivos os equipamentos da carga útil são desativados (exceto aqueles que já estavam ligados e não foram utilizados naquele objetivo específico). Se um novo objetivo estiver na fila, cabe a esta rede de tarefas ativar novamente os equipamentos.

Figura 8.4 – Redes inicias de tarefas (Amazonia-1).

```
<hierarchical_description>
    <goaltask name = "AcquisitionRealTime" id = "41" methods = "ordered">
        <method name = "AcquisitionRealTime_method" >
            <subtasks order = "ordered"  >
                <task  name  = "StartAcquisitionRealTime"   id = "20"   type = "compound"/>
                <task  name  = "StopAcquisitionRealTime"    id = "21"   type = "compound"/>
            </subtasks>
        </method>
      </goaltask>
    <goaltask name = "AcquisitionRercording" id = "42" methods = "ordered">
        <method name = "AcquisitionRercording_method" >
            <subtasks  order = "ordered"  >
                <task  name  = "StartAcquisitionRecording"  id = "26"   type = "compound"/>
                <task  name  = "StopAcquisitionRecording"   id = "27"   type = "compound"/>
            </subtasks>
        </method>
    </goaltask>
    <goaltask name = "DownloadStoredImage" id = "43" methods = "ordered">
        <method name = "DownloadStoredImage_method" >
            <subtasks  order = "ordered"  >
                <task  name  = "StartDownloadStoredImage"   id = "30"   type = "compound"/>
                <task  name  = "StopDownloadStoredImage"    id = "31"   type = "compound"/>
            </subtasks>
        </method>
    </goaltask>
    <goaltask name = "CameraCalibrationRecording" id = "44" methods = "ordered">
        <method name = "CameraCalibrationRecording_method">
            <subtasks order = "ordered"  >
                <task  name  = "StartCameraCalibrationRecord" id = "34"    type = "compound"/>
                <task  name  = "StopCameraCalibrationRecord"  id = "35"    type = "compound"/>
            </subtasks>
        </method>
    </goaltask>
    <goaltask name = "CameraCalibrationRealTime" id = "45">
        <method name = "CameraCalibrationRealTime_method" methods = "ordered">
            <subtasks order = "ordered"  >
                <task  name  = "StartCalibrationRealtime"   id = "36"    type = "compound"/>
                <task  name  = "StopCalibrationRealtime"    id = "37"    type = "compound"/>
            </subtasks>
        </method>
    </goaltask>
    <goaltask name = "AcquisitionRercordingRealTime" id = "46" methods = "ordered">
        <method name = "AcquisitionRercordingRealTime_method" >
            <subtasks order = "ordered"  >
                <task  name  = "StartRealTimeRecording"     id = "39"    type = "compound"/>
                <task  name  = "StopRealTimeRecording"      id = "40"    type = "compound"/>
            </subtasks>
        </method>
    </goaltask>
    <goalroottasks>
        <task  name  = "AcquisitionRealTime"                id = "41"  />
        <task  name  = "AcquisitionRercording"              id = "42"  />
        <task  name  = "DownloadStoredImage"                id = "43"  />
        <task  name  = "CameraCalibrationRecording"         id = "44"  />
        <task  name  = "CameraCalibrationRealTime"          id = "45"  />
        <task  name  = "AcquisitionRercordingRealTime"      id = "46"  />
    </goalroottasks>
</hierarchical_description>
```

Fonte: Produção do autor.

Uma vez definidos os objetivos da missão, devem-se especificar as decomposições da rede a partir das tarefas abstratas. O plano de voo derivado de cada objetivo deve ser projetado de forma hierárquica nesta etapa. Para sua a construção, consultaram-se documentos técnicos dos equipamentos e manuais de usuário da missão. Um exemplo de uma rede vinculada ao objetivo “Aquisição de Imagem e Transmissão em Tempo Real” do modelo Amazonia-1 pode ser visualizado na Figura 7.4.

A Tabela 8.3 define o domínio hierárquico do estudo de caso. Ela traz a descrição textual das tarefas abstratas, precondições e suas formas de decomposição. Optou-se por modelar nas tarefas abstratas precondições relacionadas exclusivamente a eventos exógenos. Isso, além de antecipar falhas que estão fora do controle do planejador, pode facilitar estratégias de replanejamento. Os números entre chaves correspondem aos IDs de suas subtarefas.

Tabela 8.3 – Decomposição hierárquica das tarefas compostas do estudo de caso.

ID	Descrição da Tarefa Abstrata	Decomposição	Precondição
20	Iniciar aquisição e transmissão de imagem em tempo real	{A22, A23}	- Período solar da órbita - ACE no modo missão - Satélite em modo de rotina - Em visibilidade com a estação - Alvo Terrestre livre de nuvens
21	Interromper aquisição e transmissão de imagem em tempo real	{A24, P12, P10, A25}	--
22	Configurar câmera para imageamento	{P11, P7, P1}	--
23	Configurar transmissor para transmissão em tempo real	{P9, P7, P13, P15}	--
24	Interromper imageamento e desligar a câmera	{P2, P8}	--
25	Desabilitar transmissão e desligar transmissor	{P16, P14}	--
26	Iniciar aquisição e gravação de imagem em bordo	{A22, A28}	- Período solar da órbita - ACE no modo missão - Satélite em modo de rotina - Alvo Terrestre livre de nuvens
27	Interromper aquisição e gravação de imagem	{A29, A24, P12}	--
28	Configurar o gravador para o modo gravação	{P9, P3}	--
29	Interromper gravação e desligar o gravador	{P4, P10}	--
30	Iniciar *playback* de imagens	{P11, P7, A32, A38}	- Satélite em modo de rotina - Em visibilidade com a estação
31	Interromper *playback* de imagens	{A33, P8, P12, A25}	--
32	Ligar e configurar gravador em modo *playback*	{P9, P5}	--
33	Interromper *playback* e desligar gravador	{P6, P19, P10}	--
34	Iniciar aquisição e gravação de imagens no modo calibração	{A22, P17, A28}	- ACE no modo missão - Satélite em modo de rotina
35	Interromper aquisição e gravação de imagens no modo calibração	{A29, P18, A24, P12}	--
36	Iniciar aquisição e transmissão de imagens no modo calibração	{A22, P17, A23}	- ACE no modo missão - Satélite em modo de rotina - Em visibilidade com a estação

continua

Tabela 8.3 – Conclusão.

ID	Descrição da Tarefa Abstrata	Decomposição	Precondição
37	Interromper aquisição e transmissão de imagens no modo calibração	{P18, A24, P12, P10, A25}	--
38	Habilitar canal de transmissão e ligar transmissor	{P13, P15}	--
39	Iniciar aquisição, transmissão e gravação de imagens	{A22, A28, A38}	- Período solar da órbita - ACE no modo missão - Satélite em modo de rotina - Em visibilidade com a estação - Alvo Terrestre livre de nuvens
40	Interromper aquisição, transmissão e gravação de imagens	{A24, P12, A29, A25}	--

A= Tarefa Abstrata, P = Tarefa Primitiva.
Fonte: Produção do autor.

A partir desta tabela, foram especificadas as decomposições em HML. Por questões de espaço, são apresentadas algumas das decomposições do domínio construído (Ações 32, 33 e 34 da Tabela 8.3), conforme a Figura 8.5. Idealmente, a solução hierárquica deve ser modelada usando uma ferramenta gráfica e posteriormente exportada no formato XML esperado pela linguagem estendida. Neste trabalho, o XML foi codificado manualmente.

Figura 8.5 – Tarefas abstratas e métodos de decomposição na linguagem estendida (Amazonia-1).

```
<hierarchical_description>
    <compoundtask name = "StartDownloadStoredImage" id= "32" inference_moment = "true" methods = "ordered">
        <method name = "StartDownloadStoredImage_method">
            <precondition>
                <precond constraints = "DT.CommunicatingWithGround == withcommuication"/>
                <precond constraints = "OBDH.satelliteMode == routine"/>
            </precondition>
            <subtasks order = "ordered">
                <task  name  = "SwitchOnRtu"           id = "11"   type = "primitive"/>
                <task  name  = "SwitchOnCamera"        id = "7"    type = "primitive"/>
                <task  name  = "TurnOnDt"              id = "40"   type = "compound"/>
                <task  name  = "ConfigureDdrPlayback"  id = "34"   type = "compound"/>
            </subtasks>
        </method>
    </compoundtask>

    <compoundtask name = "StopDownloadStoredImage" id= "33" inference_moment = "true" methods = "ordered">
        <method name = "StopDownloadStoredImage_method">
            <subtasks  order = "ordered">
                <task  name  = "SetDtWfiChannelTurnOff" id = "16"    type = "primitive"/>
                <task  name  = "StopDdrPlayback"        id = "35"    type = "compound"/>
            </subtasks>
        </method>
    </compoundtask>

    <compoundtask name = "ConfigureDdrPlayback" id= "34" inference_moment = "false" methods = "ordered">
        <method name = "ConfigureDdrPlayback_method">
            <subtasks order = "ordered">
                <task  name  = "SwitchOnDdr"              id = "9"    type = "primitive"/>
                <task  name  = "SetChannelCameraPlayback" id = "18"   type = "primitive"/>
                <task  name  = "StartPlaybackDdr"         id = "5"    type = "primitive"/>
            </subtasks>
        </method>
    </compoundtask>

    <compoundtask name = "StopDdrPlayback" id= "35" inference_moment = "false" methods = "ordered">
        <method name = "StopDdrPlayback_method">
            <subtasks order = "ordered"  >
                <task  name  = "StopPlaybackDdr"         id = "6"    type = "primitive"/>
                <task  name  = "SwitchOffDdr"            id = "10"   type = "primitive"/>
            </subtasks>
        </method>
    </compoundtask>
</hierarchical_description>
```

Fonte: Produção do autor.

As tarefas abstratas deste estudo se reduzem com apenas um método de decomposição devido à característica do próprio domínio Amazonia-1. Isto também é um dos preceitos da solução de planejamento proposta em não depender de métodos alternativos para atender a diferentes estados do satélite.

Observe-se que as tarefas compostas vão se desdobrando em um menor nível de abstração. Algumas aparecem em diferentes objetivos, como a "*TurnOffCamera*", - presente em todas as redes. Isso evidencia que a solução hierárquica descrita pelo especialista exerce papel importante em termos de reusabilidade, eficiência do planejador, e, no caso da HARPIA, na capacidade de replanejar.

Com a hierarquia definida, o próximo passo é especificar as tarefas primitivas que representam os elementos básicos do modelo. Na descrição estrutural elas devem ser apenas listadas, conforme pode ser visto na Figura 8.6. Observe que sob a ótica da 'Descrição Estática' não se estabelece vínculos das ações primitivas com os elementos modelados, apenas com a própria rede de tarefas.

Figura 8.6 – Lista de tarefas primitivas na linguagem estendida (Amazonia-1).

```
<hierarchical_description>
    <primitivetasks>
        <task  name  = "StartImagingCamera"          id = "1"  />
        <task  name  = "StandbyCamera"               id = "2"  />
        <task  name  = "StartRecordDdr"              id = "3"  />
        <task  name  = "StopRecordDdr"               id = "4"  />
        <task  name  = "StartPlaybackDdr"            id = "5"  />
        <task  name  = "StopPlaybackDdr"             id = "6"  />
        <task  name  = "SwitchOnCamera"              id = "7"  />
        <task  name  = "SwitchOffCamera"             id = "8"  />
        <task  name  = "SwitchOnDdr"                 id = "9"  />
        <task  name  = "SwitchOffDdr"                id = "10" />
        <task  name  = "SwitchOnRtu"                 id = "11" />
        <task  name  = "SwtichOffRtu"                id = "12" />
        <task  name  = "SwitchOnDt"                  id = "13" />
        <task  name  = "SwtichOffDt"                 id = "14" />
        <task  name  = "SetDtWfiChannelTurnOn"       id = "15" />
        <task  name  = "SetDtWfiChannelTurnOff"      id = "16" />
        <task  name  = "StartCalibrationCamera"      id = "17" />
        <task  name  = "StopCalibrationCamera"       id = "18" />
        <task  name  = "EraseFileDdr"                id = "19" />
    </primitivetasks>
</hierarchical description>
```

Fonte: Produção do autor.

O Apêndice A.1 deste documento traz o Domínio Hierárquico completo do modelo Amazonia-1 construído.

8.5.2. Especificação estrutural

Pela Descrição Estrutural da ISISml, é preciso identificar o domínio de operação das variáveis de estados e os elementos do sistema espacial. Cada elemento modelado possui uma ou mais variável e recursos associados. Antes de apresentar esta descrição na linguagem estendida, é conveniente mostrar uma especificação textual, como mostra a Tabela 8.4.

Tabela 8.4 – Especificação dos elementos e domínios do estudo de caso (Amazonia-1).

Elemento	Timeline	Tipo	Domínio
WFI	timeline_wfi_mode	V	*poweroff, standby, imaging, calibration*
WFI	timeline_cloudy_image	E	*cloudy_image, cloudless_image*
AOCS[45]	timeline_aocs_mode	V	*standby, survival, safehold, propulsion, mission*
DDR	timeline_ddr_mode	V	*poweroff, standby, record, playback, erase*
DDR	timeline_state_dss_mode	V	*dss_channel_real_time, dss_channel_playback*
DDR	timeline_memory_usage	R	*{0,160}*
DT	timeline_dt_mode	V	*poweroff, standby_wfi_channel_off, nominal_wfi_channel_on*
DT	timeline_ground _communication	E	*with_ground_communication, without_ground_communication*
OBDH	timeline_satellite_mode	V	*launch, emergency, degraded, routine, oribt_maneuver*
PCDU	timeline_wfi_line	V	*line_off, line_on*
PCDU	timeline_ddr_line	V	*line_off, line_on*
PCDU	timeline_rtu_line	V	*line_off, line_on*
PCDU	timeline_dt_line	V	*line_off, line_on*
PCDU	timeline_period_orbit	E	*eclipse, sun_light*
PCDU	timeline_power_total_ consumption	R	{0,200} p/ eclipse e {0,250} p/ sunlight

E = Timelines de **V**ariáveis de Estados; C = Timelines para controle de **E**ventos Exógenos; R = Timelines que controlam **R**ecursos.

Fonte: Produção do autor.

Como pode ser visto na Tabela 8.4, a maioria das variáveis definidas controlam recursos e modos de operação dos subsistemas. Suas considerações de modelagem para o satélite Amazonia-1 foram apresentadas na Seção 8.3.

Além de estados e recursos que gerenciam os elementos modelados, especificaram-se as seguintes variáveis para controlar eventos exógenos advindos do ambiente espacial:

- **period_orbit:** controlar períodos de órbita; impedir, por exemplo, que uma solicitação de imageamento seja realizada no período de eclipse.

[45] *Attitude and Orbit Control System* - AOCS

- **ground_communication:** controlar período de visibilidade do satélite; impedir, por exemplo, que uma solicitação de *Download* de dados seja realizada fora de visada;
- **cloudy_image:** impedir que uma solicitação de imageamento seja realizada quando o alvo terrestre estiver totalmente encoberto por nuvens.

Para melhor gerenciar os recursos a bordo e de forma a controlar adequadamente os objetivos segundo o modo do satélite e do controle de atitude, foram previstas as seguintes variáveis:

- **satellite_mode:** impedir a operação da carga útil quando o satélite não estiver no modo de rotina. Os demais modos do satélite não preveem a operação da carga útil;
- **aocs_mode:** impedir que uma solicitação de imageamento seja realizada quando o controle de atitude do satélite não estiver em '*Mission*', no qual a precisão de apontamento é melhor. No modo sobrevivência, o satélite está apontado para o sol e nos demais a acurácia do controle de atitude é relaxada, não fazendo sentido obter uma imagem distorcida da Terra.

Outro ponto importante da representação estrutural prevista pela ISISml é identificar os recursos modelados, elementos e quem são seus consumidores. Neste estudo do Amazonia-1, o consumidor da memória do gravador é a câmera óptica e os consumidores da energia da PCDU são os elementos: WFI, RTU, DT e DDR.

A Figura 8.7 traz um exemplo concreto de especificação para o elemento PCDU. Trata-se de um trecho reduzido da descrição estrutural.

Figura 8.7 – Exemplo da especificação de elemento e variáveis de na linguagem estendida (Amazonia-1).

```
<structural_description>
    <elements>
        <element name = "PCDU">
            <timeline name = "OrbitPeriod"          type = "OrbitPeriods"   hkparameterid = "11" />
            <timeline name = "WfiPowerLineStatus"   type = "OnOff"          hkparameterid = "12" />
            <timeline name = "DdrPowerLineStatus"   type = "OnOff"          hkparameterid = "13" />
            <timeline name = "DtPowerLineStatus"    type = "OnOff"          hkparameterid = "14" />
            <timeline name = "RtuPowerLineStatus"   type = "OnOff"          hkparameterid = "15" />

            <resource name = "Power" capacityMax = "250" access = "shareable" property = "discrete" category =
                "reusable">
                <conditional_constraint condition = "PCDU.OrbitPeriod = eclipse"    capacityMax = "200" />
                <consumer name = "WFI"  initialtotalconsumptionhkparameterid = "16"/>
                <consumer name = "DDR"  initialtotalconsumptionhkparameterid = "17"/>
                <consumer name = "DT"   initialtotalconsumptionhkparameterid = "18"/>
                <consumer name = "RTU"  initialtotalconsumptionhkparameterid = "19"/>
            </resource>
        </element>
    </elements>
</structural_description>
```

Fonte: Produção do autor.

A Figura 8.8 traz a definição dos valores de domínios modelados para o Amazonia-1. Eles são essencialmente relacionados a modos de operação dos equipamentos, subsistemas e do próprio satélite, além de abstrações lógicas de controle do ambiente espacial.

Figura 8.8 – Exemplo da especificação de domínios na linguagem estendida (Amazonia-1).

```
<?xml version="1.0"?>
<structural_description>
    <domains>
        <domain name = "WfiModes">
            <value>wfi_power_off</value>
            <value>wfi_stand_by</value>
            <value>wfi_imaging</value>
            <value>wfi_calibration</value>
        </domain>
        <domain name = "DdrModes">
            <value>ddr_power_off</value>
            <value>ddr_stand_by</value>
            <value>ddr_record</value>
            <value>ddr_playback</value>
            <value>ddr_erase</value>
        </domain>
        <domain name = "DssStatus">
            <value>dss_channel_real_time</value>
            <value>dss_channel_playback</value>
        </domain>
        <domain name = "DtModes">
            <value>dt_power_off</value>
            <value>dt_standby_wfi_channel_off</value>
            <value>dt_nominal_wfi_channel_on</value>
        </domain>
    </domains>
</structural_description>
<stretches omitted .../>
```

Fonte: Produção do autor.

Para encerrar a Descrição Estática, sumariza-se na Tabela 8.5 os itens modelados para o modelo do Amazonia-1. A relação entre domínio e *timelines* não é “um para um”, pois algumas *timelines* têm os mesmos domínios, enquanto outras estão associadas a intervalos numéricos, declarados na própria especificação do recurso - vide, por exemplo, a linha 11 da Figura 8.7. Note-se que cada timeline em ISISml representa uma variável de estado ou de recurso do satélite.

Tabela 8.5 – Itens modelados da Representação Estática para o estudo de caso (Amazonia-1).

Item Modelado	Número
Objetivos	6
Tarefas Primitivas	19
Tarefas Abstratas	21
Elementos	6
Domínios	11
Variáveis de Estados	10
Recursos	2

Fonte: Produção do autor.

O Apêndice A.2 deste documento traz o Domínio Estrutural completo para o modelo Amazonia-1 construído.

8.5.3. Especificação comportamental

Com a representação estrutural definida, especificam-se na descrição comportamental as precondições e efeitos das ações primitivas (Tabela 8.6) e os eventos modelados (Tabela 8.7) para o ambiente de planejamento.

As precondições das ações foram modeladas neste estudo segundo os seguintes critérios: (i) diagrama de transição de estados dos equipamentos, (ii) efetividade do planejador e (iii) aspectos de segurança operacional.

O item (ii) pode influenciar não só no processo de planejamento, como na etapa de reparo da solução proposta. Desta forma, algumas precondições foram ajustadas, - desde que não invalidasse a operação real do equipamento -, avaliando-se o comportamento da HARPIA de acordo com os cenários experimentais executados.

Com relação ao item (iii), a falta de restrições permitiria ao algoritmo produzir ações não recomendáveis. Considere o ato de desligar a linha de potência da câmera no modo imageamento, sem antes colocá-la em *standby.* Embora permitido pela eletrônica da câmera*,* essa não é exatamente uma operação padrão.

No modelo construído, consideraram-se tais tipos de restrições, evitando-se produzir ações operacionalmente permitidas, mas que são potencialmente nocivas à saúde dos equipamentos. Veja como exemplo as precondições da tarefa de número oito da Tabela 8.6.

O Apêndice B deste documento traz todas as ações e eventos do modelo Amazonia-1 codificados na HARPIA.

Tabela 8.6 – Precondições e efeitos das ações primitivas do estudo de caso (Amazonia-1).

Elemento	**ID**	**Descrição da Ação Primitiva**	**Precondição (Descrição Informal)**	**Efeitos**
WFI	1	Colocar a câmera em *imaging* para aquisição de imagens	- Linha[46] da RTU ligada - Linha da WFI ligada - WFI no modo *standby*	WFI.mode PCDU.power
WFI	2	Colocar a câmera em *standby*	- Linha da RTU ligada - Linha da WFI ligada - WFI não deve estar no modo *power off*	WFI.mode PCDU.power
WFI	17	Iniciar aquisição de imagem no modo calibração	- Linha da RTU ligada - Linha da WFI ligada - WFI no modo *imaging*	WFI.mode WFI.power
WFI	18	Desligar aquisição de imagem no modo calibração	- Linha da RTU ligada - Linha da WFI ligada - WFI em modo calibração	WFI.mode WFI.power
DDR	3	Iniciar gravação de imagem	- DDR em modo *standby*	DDR.mode PCDU.power DDR.memory
DDR	4	Interromper gravação de imagem	- DDR em modo *record*	DDR.memory DDR.mode PCDU.power
DDR	5	Iniciar reprodução (*Playback)* dos dados	- DDR em modo standby - WFI em modo standby	PCDU.power DDR.mode DDR.dssState

continua

[46] Nesta tabela, “Linha” se refere à linha de alimentação elétrica controlada pelo subsistema de fornecimento de energia do satélite.

Tabela 8.6 – Conclusão.

Elemento	ID	Descrição da Ação Primitiva	Precondição (Descrição Informal)	Efeitos
DDR	6	Interromper *Playback* dos dados do Gravador	- DDR em modo *playback*	DDR.mode DDR.memory DDR.dssState
DDR	19	Apagar dados do DDR	- O DDR deve estar em standby	DDR.mode DDR.memory
DT	15	Habilitar o canal da WFI[47] no Transmissor	- Linha do DT ligada	DT.mode PCDU.power
DT	16	Desabilitar o canal da WFI no Transmissor	- Transmissor em modo standby	DT.mode PCDU.power
PCDU	7	Ligar a linha de alimentação elétrica da Câmera	- WFI não pode estar em modo *imaging* - WFI não pode estar em modo *calibration*	WFI.mode PCDU.power PCDU. wfiLine
PCDU	8	Desligar a linha de alimentação elétrica da Câmera	- WFI não pode estar no modo *imaging* - WFI não pode estar em modo *calibration*	WFI.mode PCDU.power PCDU.wfiLine
PCDU	9	Ligar a linha de alimentação elétrica do Gravador	- Linha do DDR não pode estar ligada	PCDU.ddrLine DDR.mode PCDU.power DDR.memory DDR.dssState
PCDU	10	Desligar a linha de alimentação elétrica do Gravador	- DDR não pode estar em modo *record* - DDR não pode estar em modo *playback* - DDR não pode estar em modo *erase* - DDR não pode estar em modo *power off*	PCDU.power DDR.mode PCDU.ddrLine
PCDU	11	Ligar a linha da RTU	- Linha da RTU desligada	PCDU.power PCDU.rtuLine
PCDU	12	Desligar a linha da RTU	- Linha da RTU ligada	PCDU.power PCDU.rtuLine
PCDU	13	Ligar a linha de alimentação elétrica do transmissor de dados	- Transmissor de dados (DT) não pode estar no modo nominal	PCDU.dtLine PCDU.power DT.mode
PCDU	14	Desligar a linha de alimentação elétrica do transmissor de dados	- Transmissor de dados (DT) não pode estar no modo nominal (operacionalmente deve estar em standby)	PCDU.dtLine DT.mode PCDU.power

Fonte: Produção do autor.

[47] Embora o nome faça alusão à WFI, essa ação habilita o canal de transmissão, seja os dados advindos da câmera (em tempo real) ou do gravador (reprodução).

Para a descrição dos efeitos das ações, baseou-se no consumo dos recursos apresentados na Tabela 8.1. As restrições temporais dos eventos endógenos foram modeladas, conforme disposto na Tabela 8.2. Neste estudo de caso, não foram descritas tarefas elementares previstas pela linguagem estendida.

Tabela 8.7 – Eventos modelados pela HML/ISISml para o estudo de caso (Amazonia-1).

Evento	Categoria	Tipo	Afeta
Comunicação com a estação	Exógeno	Nominal	Timelines
Período da Órbita	Exógeno	Nominal	Timelines
Visibilidade do Alvo Terrestre[48]	Exógeno	Ambiental	Timelines
Gravador apto à operação	Endógeno	Temporal	Restrição Temporal
Transmissor apto à Transmissão de Dados	Endógeno	Temporal	Restrição Temporal

Fonte: Produção do autor.

É importante frisar que o modelo criado é próximo ao comportamento real do satélite. Isso proporciona cenários experimentais bastantes realísticos. Os trabalhos que utilizaram HTN com domínio voltado a satélites de sensoriamento remoto (vide AMIGONI et al., 2010; ZHANG et al., 2006) se basearam em modelos bastante simplificados, distanciando-se de aplicações reais.

8.6. Cenários experimentais de planejamento automatizado

Esta seção traz os cenários experimentais de planejamento executados. Ela tem como objetivo analisar o desempenho e a capacidade de resposta da HARPIA, embarcada em hardware equivalente ao da missão real. Para facilitar a apresentação dos resultados, destacam-se a seguir dados comuns utilizados pelos testes. A Tabela 8.8 enumera os objetivos da missão espacial deste estudo.

[48] Não faz parte do escopo deste trabalho detectar nuvens nas imagens, apenas considerar essa informação no problema de planejamento. Pela HARPIA, essa detecção poderia ser feita pelo "Gerenciador de Resposta à Ciência", que não foi desenvolvido, conforme destacado no Capítulo 5.

Tabela 8.8 – Enumeração dos objetivos no âmbito da missão espacial considerando-se o modelo construído (Amazonia-1).

Número	Objetivo
1	Aquisição e transmissão de imagem em tempo real
2	Aquisição e gravação de imagem em bordo
3	Reprodução das imagens gravadas em bordo
4	Aquisição e transmissão em tempo real de imagem no modo calibração
5	Aquisição e gravação de imagem no modo calibração
6	Aquisição, gravação e transmissão de imagem em tempo real

Fonte: Produção do autor.

A Tabela C.1 do Apêndice C deste documento traz um conjunto de estados que foram selecionados para realização de testes mais extensivos. São simulações de telemetria do software de voo que representam cenários com os quais o planejador poderia se deparar durante o processo de planejamento.

Foram previstos inclusive estados operacionalmente inválidos. Consideraram-se válidos aqueles estados em que a carga útil está efetivamente realizando algum dos seis objetivos da missão, ou quando todos os equipamentos se encontram desligados[49] ou em *standby*.

Cenários com ruídos nos sensores não foram considerados, pois isso configura planejamento em ambiente parcialmente observável, que está fora do escopo desta Tese. Um exemplo de ruído seria a linha de potência da câmera estar desligada, mas a telemetria de seu modo indica que a mesma se encontra em estado de calibração.

Há dois propósitos principais para se considerar diferentes estados operacionais nos cenários experimentais:

I. Considerar múltiplos estados iniciais do problema, isso força a HARPIA a buscar diferentes caminhos para encontrar uma solução do problema, mesmo quando o satélite se encontra em uma situação não nominal de operação da carga útil.

[49] Como existe uma preocupação constante em poupar energia do satélite, a operação padrão (sem uso do *payload*) é manter os equipamentos da carga útil desligados ou em *standby*.

II. Simular estados após a constatação de uma eventual falha. Se uma ação deixa de executar corretamente seus efeitos no ambiente, a execução do plano deve falhar nas precondições das ações subsequentes. Tal situação implica em um estado operacionalmente inválido ou diverso daquele previsto pelo planejador, o que implica em uma solicitação de replanejamento.

Destaca-se que problemas que foram produzidos de forma aleatória se basearam nas configurações de estados previstas pela Tabela C.1. Há outros experimentos discutidos adiante que também utilizam alguma configuração desta tabela para uma análise mais refinada de algum caso particular.

8.7. Sumário dos cenários de operação autônoma

Esta seção sumariza os cenários de operação autônoma definidos para a análise do comportamento da HARPIA. Os cenários previstos são:

- Cenário I (Equipamentos Desligados);
- Cenário II (Diferentes Objetivos);
- Cenário III (Replanejamento com Novos Objetivos);
- Cenário IV (Análise da Capacidade de Resposta);
- Cenário V (Análise de Tempo Médio de Execução);
- Cenário VI (Análise de Cenário Nominal e com Refinamento);
- Cenário VII (Falha em Restrições de Eventos Exógenos);
- Cenário VIII (Simulação de Falha na Execução do Plano).

Uma vez apresentados os cenários em alto nível, a Tabela 8.9 caracteriza o problema sobre o qual os três primeiros cenários foram realizados. Em nenhum deles há previsão de nuvens sobre o alvo, e assume-se que o satélite está em modo de rotina com o AOCS em modo missão.

Tabela 8.9 – Descrição dos cenários experimentais I, II e II.

Cenário	Descrição do Cenário	Novos Objetivos	Número de Objetivos	Estado Inicial
I	Agenda vazia, previsão de uma passagem, e um objetivo a atingir	Não há	1	Equipamentos desligados e sem consumo inicial de memória do DDR
II	Agenda vazia, previsão de três passagens; objetivos a cumprir: {1, 2, 5 e 3}	Não há	4	Equipamentos desligados e consumo inicial da memória do DDR com 5 minutos de gravação
III	Agenda com o objetivo de aquisição de imagens no modo calibração e transmissão em tempo real com previsão de duas passagens	(i) aquisição e gravação de imagem (ii) transmissão da imagem gravada	1 preexistente + 2 novos	Equipamentos ligados e no modo *standby* (Cenário 5 da Tabela C.1) e sem consumo inicial de memória DDR

Fonte: Produção do autor.

Os demais cenários (IV-VIII) objetivam analisar outros aspectos da solução de planejamento, que não demandam o mesmo nível de informação apresentado na Tabela 8.9. O propósito de cada um é detalhado na sua própria apresentação.

Com relação aos resultados dos experimentos discutidos adiante, eles foram obtidos no ambiente computacional descrito na Seção 8.1, e os tempos de execução pelo simulador SIS do processador ERC32. A codificação da HARPIA foi realizada nesse ambiente com o compilador C++, mas prevalecendo-se os recursos da linguagem C.

Destaca-se que os dados contidos nas tabelas e gráficos das próximas seções foram extraídos usando-se o modo *verbose* da HARPIA. Trata-se de um modo cujo console da aplicação traz informações relevantes ao processo de planejamento.

8.7.1. Cenário I (equipamentos desligados)

A Tabela 8.10 traz os resultados obtidos para o Cenário I. Trata-se do cenário mais simplificado para uma operação autônoma, sem objetivos preexistentes na agenda, apenas um objetivo no problema com estado inicial dos equipamentos desligados.

As linhas desta tabela se referem à execução individual de cada objetivo. Conforme os resultados obtidos, a solução de planejamento responde a este problema em menos de cinquenta milissegundos, considerando-se a soma das duas camadas da HARPIA.

Note-se que a camada reativa cumpre com sua proposta e apresenta um tempo de resposta mais célere. Como esperado, a maior quantidade de nós visitados pelo algoritmo implica em um maior tempo de execução.

Tabela 8.10 – Resultados de desempenho da HARPIA para o Cenário I.

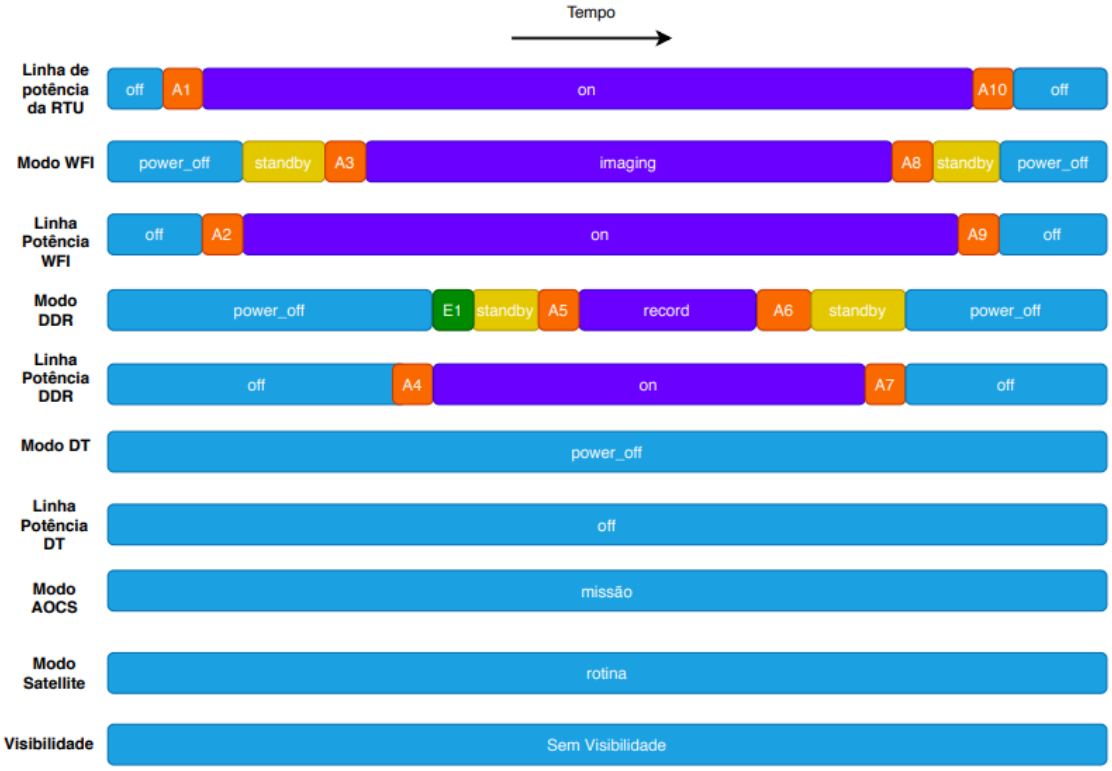
Objetivo	Nós visitados	Tempo Total	Tempo Deliberativa	Tempo Reativa	Total de Ações
1	60	40 ms	37 ms	3 ms	12
2	52	36 ms	33 ms	3 ms	10
3	64	43 ms	39 ms	4 ms	13
4	60	41 ms	37 ms	4 ms	12
5	68	45 ms	41 ms	4 ms	14
6	72	49 ms	46 ms	3 ms	14

Fonte: Produção do autor.

A Figura 8.9 mostra os efeitos das ações sobre as variáveis de estados no horizonte temporal do Objetivo 2. Suprimiu-se a variável que controla se a chave interna do DDR está em modo *playback* ou tempo real por ela não ter relevância para o cumprimento deste objetivo.

Adicionalmente, a Tabela 8.11 dispõe do plano de ações encontrado pela HARPIA. Neste objetivo, efetua-se a gravação de imagens fora de visada por quatorze minutos h_t [1120, 1960], cujo horizonte de planejamento não dispõe de previsão de passagens e a órbita encontra-se em seu período solar.

Figura 8.9 – Variáveis de estados no domínio do tempo para o Objetivo 2 do Cenário I.



Fonte: Produção do autor.

Note-se a presença de um evento temporal endógeno produzido pela camada reativa, a qual sinaliza 10 segundos após o DDR ter sido ligado a sua entrada no modo *standby*.

Tabela 8.11 – Solução encontrada pela HARPIA para Cenário I e Objetivo 2.

Ação/Evento	Tipo	Descrição	T
[A1]	Ação	PCDU:SwitchOnRtu	[1120]
[A2]	Ação	PCDU: SwitchOnWfi	[1122]
[A3]	Ação	WFI: StartImagingWfi	[1123]
[A4]	Ação	PCDU:SwitchOnDdr	[1125]
[E1]	Evento Temporal Endógeno	DDR Ready Standby	[1135]
[A5]	Ação	DDR: StartRecordDdr	[1136]
[A6]	Ação	DDR:StopRecordDdr	[1960]
[A7]	Ação	PCDU:SwitchOffDdr	[1961]
[A8]	Ação	WFI: StandbyWfi	[1963]
[A9]	Ação	PCDU: SwitchOffWfi	[1964]
[A10]	Ação	PCDU: SwtichOffRtu	[1965]

Fonte: Produção do autor.

8.7.2. Cenário II (diferentes objetivos)

O Cenário II contempla diferentes objetivos e é mais próximo de um cenário real de operação autônoma. Ele realiza um imageamento em tempo real para, em seguida, efetuar duas gravações fora de visada, sendo uma delas em modo calibração e, por último, reproduz todas as imagens gravadas a bordo para a estação terrena, inclusive as que já estavam inicialmente gravadas na memória do DDR.

Os horizontes de cada objetivo contidos neste cenário são:

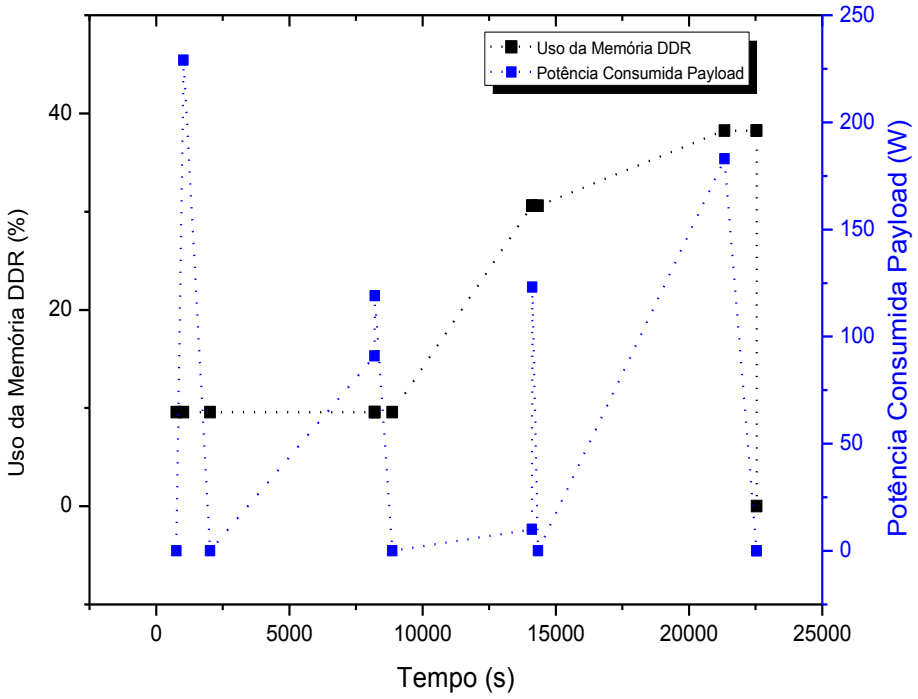
- Objetivo nº 1 e h_t[760, 2020], sendo 17 minutos de transmissão de tempo real;
- Objetivo nº 2 e h_t[8200, 8860], totalizando 11 minutos de gravação;
- Objetivo nº 5 e h_t[14100, 14340], totalizando 4 minutos de gravação em modo calibração;
- Objetivo nº 3 e h_t[21080, 22520], sendo 20 minutos de *playback*.

A Figura 8.10 traz um gráfico em função do tempo de eixo y duplo, cada qual mostrando o perfil de uso da memória do gravador e a potência consumida pela carga útil. Os picos de consumo de energia representam os objetivos realizados neste cenário.

O primeiro e o último objetivo têm os maiores picos de energia em razão de precisarem manter o transmissor ligado, com o canal de transmissão habilitado. O terceiro objetivo gasta mais potência que o segundo em virtude de o modo calibração consumir mais que o modo imageamento.

O uso da memória se mantém constante até a execução dos dois objetivos intermediários que fazem a gravação em bordo à taxa de dados de 51 MB/s. Após o *playback*, os dados do gravador são apagados, restaurando-se a capacidade da memória de 160 Gb.

Figura 8.10 – Uso de memória e energia em função do tempo para o Cenário II.



Fonte: Produção do autor.

Foram executadas duas versões adicionais ao Cenário II original. Na versão A, uma configuração inicial da carga útil é produzida de forma randômica, enquanto na versão B, uma configuração também aleatória é injetada entre cada objetivo. Para geração de estados aleatórios, ambas as versões se basearam nos cenários do Apêndice C.

À luz dos dados da Tabela 8.12, pode-se verificar que as Versões A e B tiveram aumento no tempo de planejamento devido à necessidade de refinar o estado do satélite. Apesar disso, o tempo de execução permaneceu da ordem de milissegundos, mantendo-se a capacidade de execução em tempo real.

Tabela 8.12 – Dados de desempenho da HARPIA para o Cenário II em três versões.

Versões	Objetivos	Consumo Inicial (DDR)	Nós Visitados	Tempo Total	Ações do Plano	Reparo no Plano
Original	{1, 2, 5 e 3}	15300 MB	236	153 ms	47	Não
A	{1, 2, 5 e 3}	15300 MB	458	233 ms	55	Sim
B	{1, 2, 5 e 3}	15300 MB	916	412 ms	67	Sim

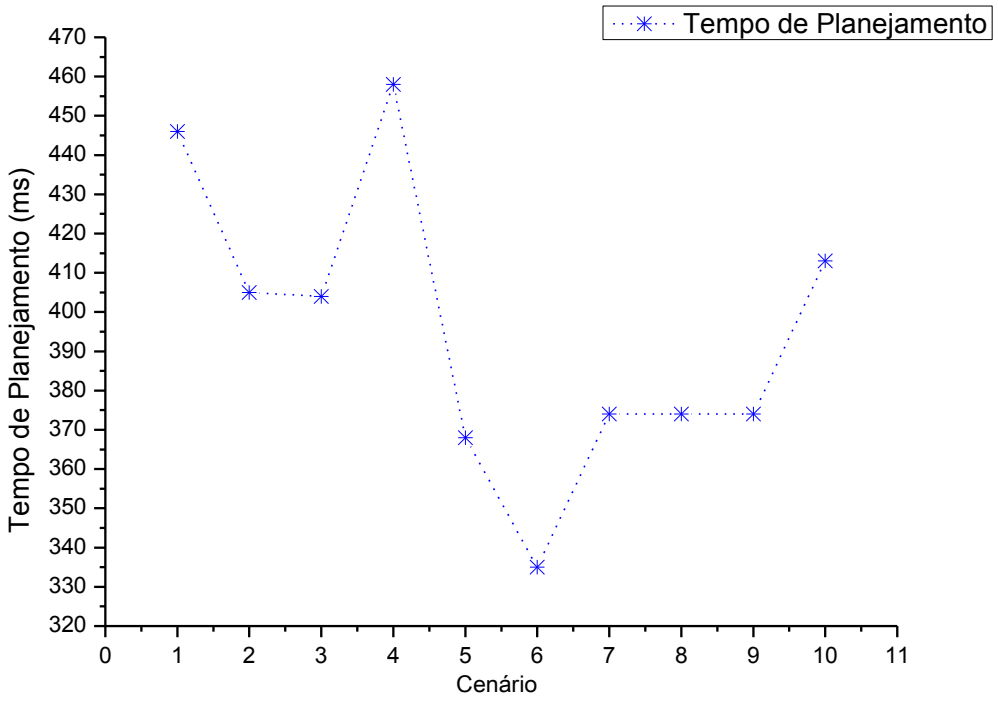
Fonte: Produção do autor.

Apresentam-se abaixo exemplos de outras informações providas pela HARPIA ao final do processo planejamento para a Versão B.

- Total de tarefas abstratas que falharam durante o processo de decomposição: 40;
- Total de tarefas reparadoras que falharam durante o processo de refinamento: 21;
- Total de tarefas abstratas analisadas durante o processo de planejamento: 91;
- Objetivos realizados com sucesso: 4;

De forma a realçar que o tempo de planejamento depende da configuração inicial de estados do problema, a Figura 8.11 ilustra tempos de planejamento da HARPIA para diferentes cenários da Versão B do Cenário II. Sobre os resultados da Tabela 8.12, note-se que foi mostrado apenas um exemplo de execução, considerando-se um cenário escolhido aleatoriamente.

Figura 8.11 – Tempo de Planejamento da HARPIA para dez configurações da Versão III do Cenário II.



Fonte: Produção do autor.

8.7.3. Cenário III (replanejamento com novos objetivos)

Outro cenário relevante para uma aplicação espacial automatizada é verificar o seu comportamento quando um ou mais objetivos são solicitados, modificando-se uma agenda preexistente. Para simular esse tipo de replanejamento, selecionou-se o Cenário III da Tabela 8.9.

Os horizontes de cada objetivo {4, 2 e 3} contidos neste cenário são os seguintes: h_t[640, 1540], h_t[4200, 4800] e h_t[10100, 10700]. A Figura 8.12 traz um trecho da codificação da HARPIA que adiciona os objetivos antes do planejador *Liger* fazer a busca pelo plano. A inicialização do modelo pelo índice 4 corresponde à linha 5 da Tabela C.1 cujos equipamentos encontram-se ligados.

Figura 8.12 – Inicialização do modelo e adição dos objetivos na HARPIA.

```
void Scenarios::ExperimentalScenarioIII(Stack &s)
{
    ModelInitialization::InitialStateTimelinesById(4);

    // a ordem precisa ser invertida por utilizarmos a busca em profundidade usando o princípio de uma pilha
    GoalsManager::AddNewGoalOnCurrentPlan(s, DownloadStoredImage,       10100, 10700);
    GoalsManager::AddNewGoalOnCurrentPlan(s, AcquisitionImageRecording, 4200,  4800);
    GoalsManager::AddNewGoalOnCurrentPlan(s, CameraCalibrationRealtime, 640,   1540);

    LigerHtnPlanner::SeekPlan(s);
}
```

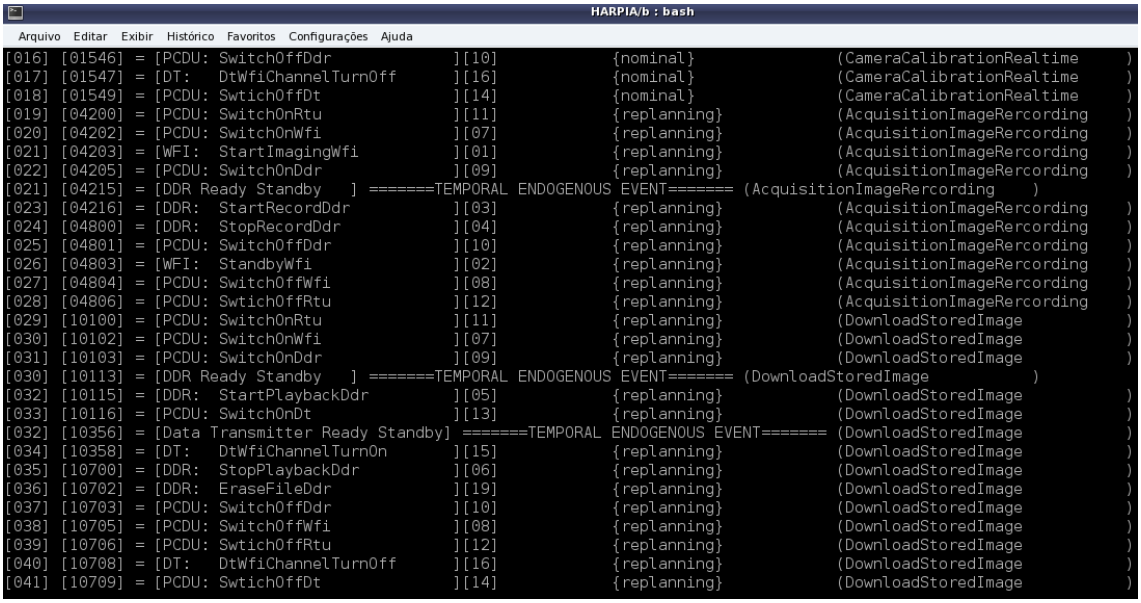
Fonte: Produção do autor.

O cenário sem replanejamento gastou 120 ms e encontrou dezoito ações no plano final. O experimento completo com replanejamento levou 195 ms e o total de 41 ações. A HARPIA identificou picos de energia de 183, 119 e 233 W da carga útil e 30600 MB de memória do DDR.

A Figura 8.13 traz parte do plano encontrado pela HARPIA. A primeira coluna diz respeito à ordem da ação na solução, a segunda o instante de execução, e a quarta o *id* da tarefa primitiva (ver Tabela 8.6 da Seção 8.5.3).

Por meio desta figura, pode-se observar a atuação do Gerenciador de Atividades (*e.g.,* T = 04215), que atendeu às restrições temporais impostas pela ocorrência dos eventos endógenos.

Figura 8.13 – Console da HARPIA no modo *verbose* com a reprodução de parte do plano encontrado para o Cenário III.



Fonte: Produção do autor.

Demais análises realizadas demonstraram que outras variações de cenários com novos objetivos para replanejamento na HARPIA não diferem muito do tempo de execução do experimento apresentado, o que abrevia os cenários aqui descritos.

Na HARPIA, as ações primitivas do problema original retornam ao nível de objetivos e um novo plano é produzido pelo processo de replanejamento, sem considerar efetivamente as ações primitivas dispostas no plano inicial.

Observe que o cenário de replanejamento (considerando novos objetivos a cumprir) não tem forçosamente a necessidade de refinar o estado do satélite. Isso depende da configuração inicial do problema recebido.

Isso decorre em razão da HTN poder representar várias redes iniciais com espaços de buscas independentes, cada qual retratando um objetivo do problema. No cenário de replanejamento, objetivos adicionados correspondem a novas redes, que devem ser inseridas na agenda de acordo com seus intervalos temporais.

Tal característica permite ao planejador lidar melhor com novos objetivos inseridos em uma agenda preexistente. Isso resulta em um menor impacto em termos de tempo de

execução se comparado a outras abordagens de planejamento. Na prática, isso é muito importante, pois o cenário de replanejamento com novos objetivos requer tipicamente respostas rápidas.

8.7.4. Cenário IV (análise da capacidade de resposta)

Análises da capacidade de resposta realizadas nos cenários anteriores (Seções 8.7.2 8.7.3) foram baseadas em estados gerados aleatoriamente, mas realizados de forma pontual, visando aferir basicamente o plano resultante e o tempo de execução.

Para verificar a taxa de sucesso[50] de um planejador, é preciso considerar uma série de execuções mais extensa, modificando-se as configurações de estados e, sobretudo, aumentando-se o tamanho da entrada do problema com mais objetivos.

Com essa finalidade, realizaram-se experimentos cujas entradas dos problemas têm os seguintes tamanhos: 50, 60, 70, 80, 90 e 100 objetivos, variando-se aleatoriamente o tipo do objetivo e os estados para cada cenário executado.

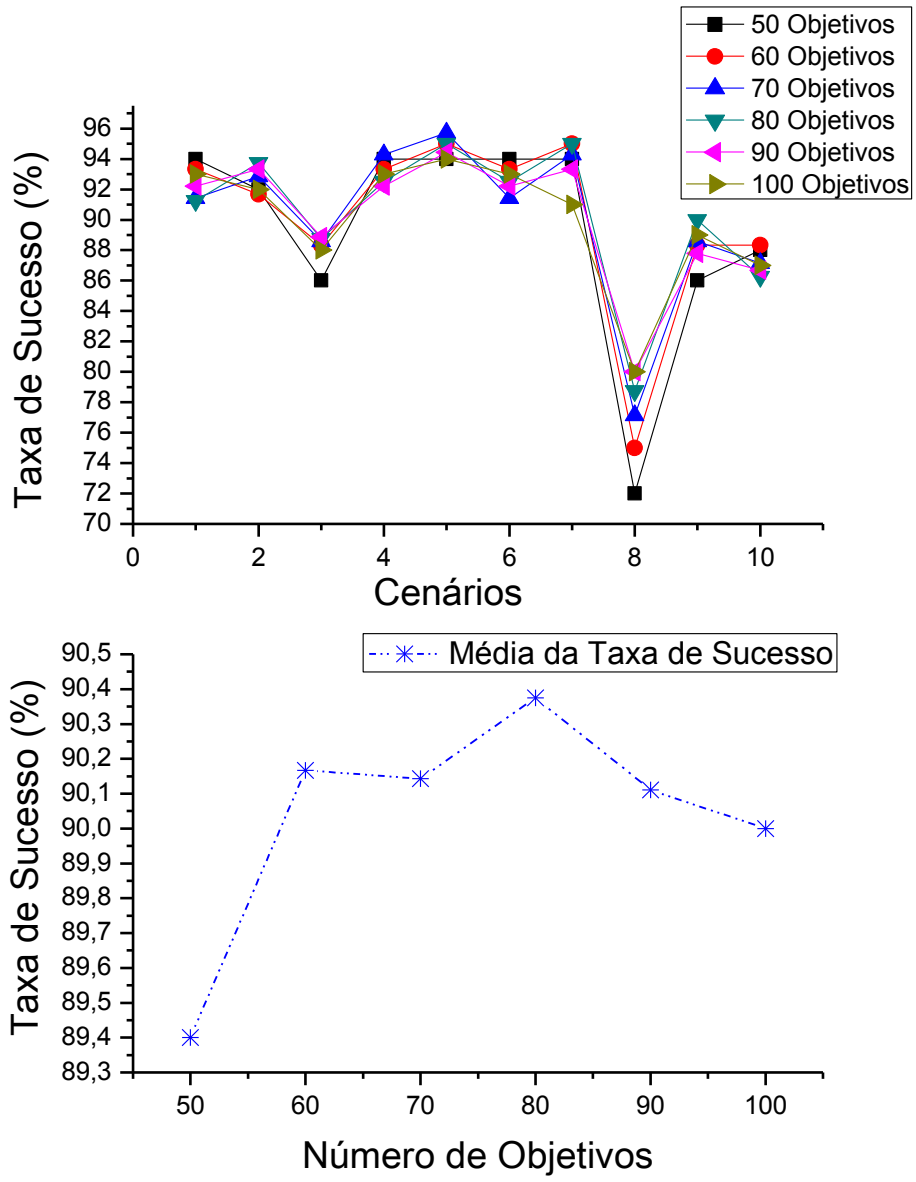
Como se pode observar pelo gráfico superior da Figura 8.14 foram executados dez cenários para cada tamanho definido. Cada ponto do gráfico representa a taxa de sucesso do problema naquele cenário realizado.

O gráfico inferior da Figura 8.14 representa a média dos resultados obtidos no gráfico superior, considerando-se os dez cenários. Neste ensaio, o percentual médio final de sucesso para os problemas gerados foi de 90,03%.

Trata-se de um resultado bastante satisfatório, uma vez que não se utiliza métodos alternativos, apenas a abordagem híbrida baseada em rede de tarefas, que realiza a estratégia de refinamento discutida no Capítulo 7.

[50] A taxa de sucesso corresponde à relação: Nº de Problemas Solucionados / Nº de Problemas Executados.

Figura 8.14 – Taxas de sucesso da HARPIA segundo diferentes tamanhos de problemas.



Fonte: Produção do autor.

Com intuito de melhorar esse percentual, a solução híbrida propõe uma heurística adicional que fora descrita na Seção 7.7. Nela, o processo de refinamento é composto por uma segunda etapa, a qual realiza um relaxamento da estabilidade do plano.

Grosso modo, em algumas situações o planejador não consegue encontrar uma solução para a heurística inicial de aproximação do objetivo, mesmo utilizando-se da estratégia de combinar tarefas primitivas com tarefas abstratas.

O segundo passo do algoritmo de refinamento permite inserir uma tarefa primitiva reparadora (desde que seja executável a partir do ponto de inserção) mesmo que ela sozinha não colabore diretamente para resolver alguma precondição aberta do problema.

A combinação das duas etapas permite gerar planos estáveis e aumentar a capacidade de respostas para algumas situações específicas, como, por exemplo, quando o planejador se depara com um estado operacionalmente inválido.

Repetiram-se os testes da Figura 8.14, inclusive com outros cenários mais extensivos (vide a Seção 8.7.5), e a taxa de sucesso alcançou 100% dos cenários executados.

É importante observar que a heurística adicional não substitui o algoritmo de refinamento, ela o complementa. Se executada de forma isolada, ela não somente não alcançaria este resultado, como os planos gerados poderiam ser insatisfatórios em termos qualidade e estabilidade do plano.

Uma análise interessante é particularizar um exemplo em que a HARPIA não obtém sucesso sem empregar a heurística integrada.

O cenário escolhido corresponde à linha 30 da Tabela C.1 para o objetivo *Playback*. Seu estado inicial é a câmera em imageamento, DDR em gravação, chave do DSS em tempo real, transmissor desligado e RTU desligada.

Um dos motivos que dificulta o processo de refinamento é o fato de a RTU estar desligada, sendo que a câmera se encontra em imageamento. Operacionalmente não faz sentido deixar a RTU desligada nessa situação.

A Tabela 8.13 traz o plano gerado pela HARPIA. Observe que as ações de refinamento foram totalmente pertinentes para o estado inicial recebido: ligando-se a RTU e colocando-se a câmera em *standby*, retirando-se o DDR do modo gravação, e ao final desligando-se os equipamentos envolvidos.

Tabela 8.13 – Solução de refinamento encontrada pela HARPIA utilizando a heurística integrada.

Tempo	Ação/Evento	Tipo	Descrição
[2000]	[A1]	Ação de Refinamento	PCDU:SwitchOnRtu
[2003]	[A2]	Ação de Refinamento	DDR:StopRecordDdr
[2004]	[A3]	Ação de Refinamento	PCDU:SwitchOffDdr
[2006]	[A4]	Ação de Refinamento	WFI:StandbyWfi
[2007]	[A5]	Ação de Refinamento	PCDU:SwitchOffWfi
[2009]	[A6]	Ação de Refinamento	PCDU:SwtichOffRtu
[2010]	[A7]	Ação	PCDU:SwitchOnRtu
[2012]	[A8]	Ação	PCDU:SwitchOnWfi
[2013]	[A9]	Ação	PCDU:SwitchOnDdr
[2023]	[E1]	Evento Endógeno Temporal	DDR Ready Standby
[2025]	[A10]	Ação	DDR: StartPlaybackDdr
[2026]	[A11]	Ação	PCDU:SwitchOnDt
[2266]	[E2]	Evento Endógeno Temporal	Data Transmitter Ready Standby
[2268]	[A12]	Ação	DT:DtWfiChannelTurnOn
[2270]	[A13]	Ação	DDR:StopPlaybackDdr
[2272]	[A14]	Ação	DDR:EraseFileDdr
[2273]	[A15]	Ação	PCDU:SwitchOffDdr
[2275]	[A16]	Ação	PCDU:SwitchOffWfi
[2276]	[A17]	Ação	PCDU:SwtichOffRtu
[2278]	[A18]	Ação	DT:DtWfiChannelTurnOff
[2279]	[A19]	Ação	PCDU: SwtichOffDt

Fonte: Produção do autor.

As ações de refinamento incluídas com sucesso foram: tarefa primitiva 11 (*SwitchOnRtu*) e tarefa abstrata 27 (*StopAcquisitionRecording*), vide a Tabela 8.3 da Seção 8.5.1 para verificar as subtarefas.

Note-se que quando uma tarefa composta é inserida pelo algoritmo de refinamento, ela ainda pode se desdobrar em outras tarefas abstratas, como é o caso da Tarefa 27. Cabe ao motor de busca HTN da HARPIA reduzi-las aos níveis inferiores, procedendo-se com o mecanismo de inferência de estados para validar sua decomposição.

Outros testes mais extensivos analisados de forma empírica indicam que a heurística proposta traz planos com ações de refinamento de boa qualidade, mesmo com certo relaxamento da estabilidade.

Isso se justifica pela característica de ordem total do algoritmo da HARPIA e pelo seu funcionamento integrado com o algoritmo de refinamento que se baseia no mecanismo antecipado de inferência de estados.

Para encerrar este experimento, a Tabela 8.14 mostra dez problemas selecionados, considerando-se cenários previstos pelo Apêndice C.

A finalidade é observar o comportamento da HARPIA em cenários operacionais diversificados, onde o planejador precisa lidar com situações não previstas pelo domínio.

Ressalta-se que "Ações de Refinamento" da Tabela 8.14 correspondem às ações inseridas pela abordagem generativa do planejamento clássico e não via métodos HTN, cuja estratégia fora descrita no Capítulo 7.

Tabela 8.14 – Planos obtidos pela HARPIA para cenários de refinamento de estados.

Ordem dos Experimentos	Objetivo	Cenário Inicial	Ações de Refinamento	Total de Ações
1	4	23	7	19
2	3	19	4	17
3	3	14	6	19
4	1	40	6	18
5	6	36	7	21
6	2	35	6	16
7	1	24	6	18
8	3	15	5	18
9	6	13	7	21
10	1	2	0	12

Fonte: Produção do autor.

Considerando-se uma média, 70% das ações do plano final deste experimento estão presentes na rede hierárquica inicial. Em nenhum dos cenários analisados, uma ação de reparo incluída no plano final era dispensável ou irrelevante ao processo de refinamento. Isso corrobora que as análises empíricas realizadas estavam corretas quanto à qualidade e estabilidade do plano gerado.

8.7.5. Cenário V (análise de tempo médio de execução)

Como este trabalho propõe uma extensão ao planejamento HTN para lidar com situações não previstas, é preciso verificar como a solução se comporta em termos de tempo de execução quando o planejador se depara com cenários não previstos no domínio hierárquico.

Para fazer uma análise mais representativa, foram gerados quinhentos problemas de forma randômica com diferentes números de objetivos. Uma vez que o tempo de resposta pode ser diferente para cada execução, cada linha da Tabela 8.15 representa o tempo médio calculado para cem cenários realizados naquela configuração.

Tabela 8.15 – Tempo médio de planejamento de problemas gerados aleatoriamente.

Número de Objetivos no Problema	Tempo Médio de Execução (s)
1	0,105
2	0,207
3	0,309
4	0,416
5	0,518

Fonte: Produção do autor.

A fim de completar os resultados da Seção 8.7.4, a taxa de sucesso também foi mensurada neste experimento. Para os 500 problemas gerados da Tabela 8.15, 100% dos objetivos foram realizados, ratificando-se os dados apresentados anteriormente.

Com relação ao tempo médio de planejamento deste experimento, pode-se observar que se obteve uma resposta para até cinco objetivos no problema entre 100 e 500 milissegundos.

Como o número de atividades de um horizonte (uma órbita, cinco órbitas, um dia, etc.) pode variar significativamente conforme a demanda da missão, uma análise mais conveniente é considerar o pior cenário.

Isto é, aquele que realiza o máximo de objetivos possíveis para um dado horizonte temporal, considerando-se as limitações de recursos a bordo e restrições de visibilidade com a estação de controle da missão espacial.

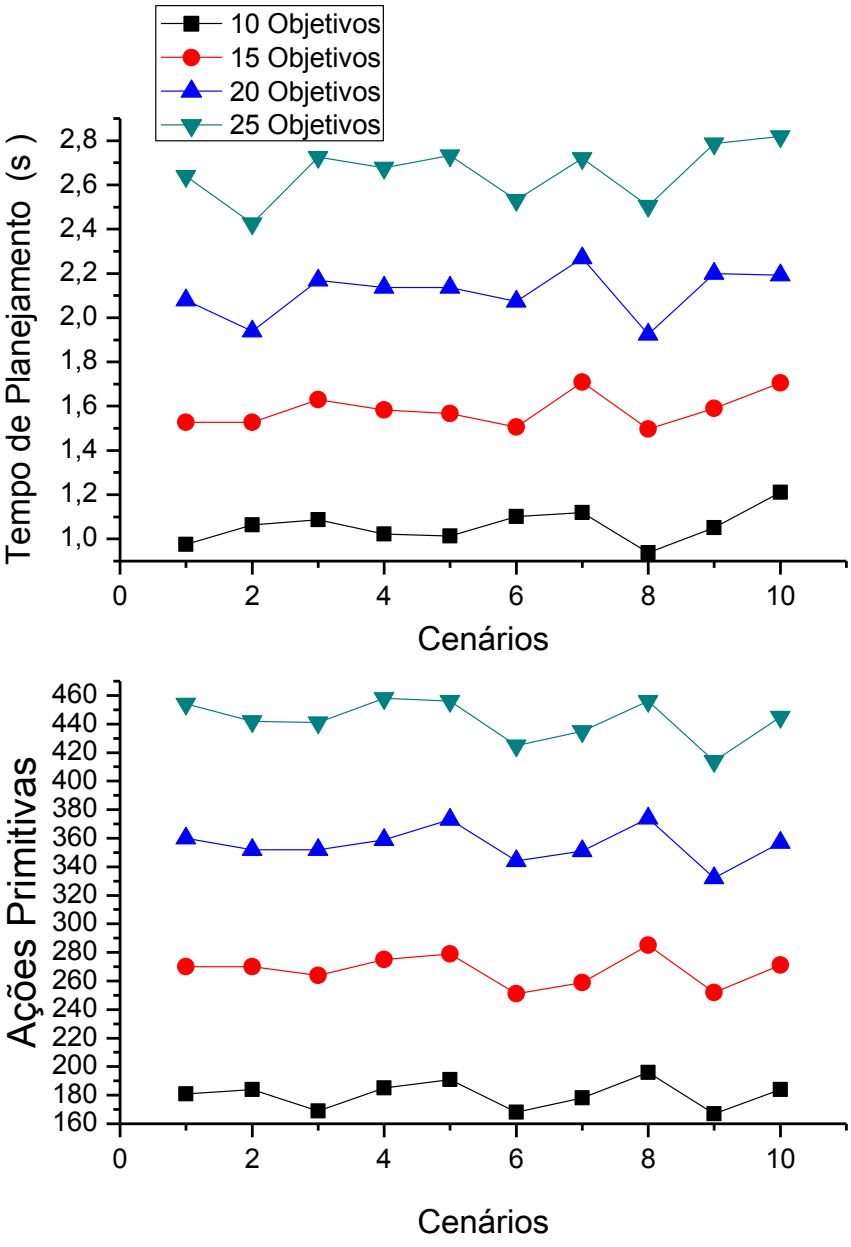
No pior cenário, a HARPIA produz um plano de voo em aproximadamente meio segundo, estimando-se em cinco objetivos o número máximo de atividades suportadas por um satélite de sensoriamento remoto, similar ao Amazonia-1, para o horizonte de uma órbita.

Vale lembrar que esse número maximizado de atividades dificilmente será exercitado em um cenário real de operação autônoma, considerando-se o curto espaço de tempo de uma órbita.

Note-se que para o cenário nominal (não considerado na Tabela 8.15), dependendo do conjunto de objetivos formulado, a HARPIA alcança um plano em menos de 200 ms para os mesmos cinco objetivos contidos no problema.

De forma a verificar a escalabilidade da HARPIA, a Figura 8.15 traz um experimento com tamanhos maiores no problema, considerando-se dez cenários que definem aleatoriamente estados e objetivos. Cada curva identificada por uma cor representa o resultado obtido, segundo os números de objetivos contidos naquele problema.

Figura 8.15 – Análise da escalabilidade da HARPIA conforme o número de objetivos contidos no problema.



Fonte: Produção do Autor.

Os gráficos da Figura 8.15 retratam os resultados obtidos exatamente dos mesmos cenários realizados. O primeiro ilustra o tempo de planejamento necessário, enquanto o segundo traz o número de ações primitivas que compõem o plano final.

Pode-se observar pelo gráfico superior que o aumento do número de objetivos no problema fez o tempo de execução de planejamento crescer de forma linear, mostrando a solução híbrida ser escalável.

Para os cenários executados, calcularam-se a variância[51] e o desvio padrão, que são apresentados na Tabela 8.16. O desvio padrão baixo obtido realça que a HARPIA tem boa previsibilidade do tempo de planejamento, independentemente do cenário de execução do problema; uma propriedade crucial para sistemas de tempo real, como satélites.

Tabela 8.16 – Variância e desvio padrão para os diferentes tempos de execução do experimento realizado.

Número de Objetivos	**10**	**15**	**20**	**25**
Variância (σ^2)	0,00545 s	0,00527 s	0,0111 s	0,0150 s
Desvio Padrão (σ)	73,84 ms	72,59 ms	105,24 ms	122,78 ms

Fonte: Produção do autor.

Pensando em missões espaciais autônomas que desejarem produzir planos de voo para longos horizontes temporais, realizaram-se experimentos com 50 e 200 objetivos. Os resultados da HARPIA foram respectivamente:

- No melhor cenário em aproximadamente 2 e 8 segundos;
- No pior cenário em aproximadamente 5 e 20 segundos;
- A taxa de sucesso foi de 100%.

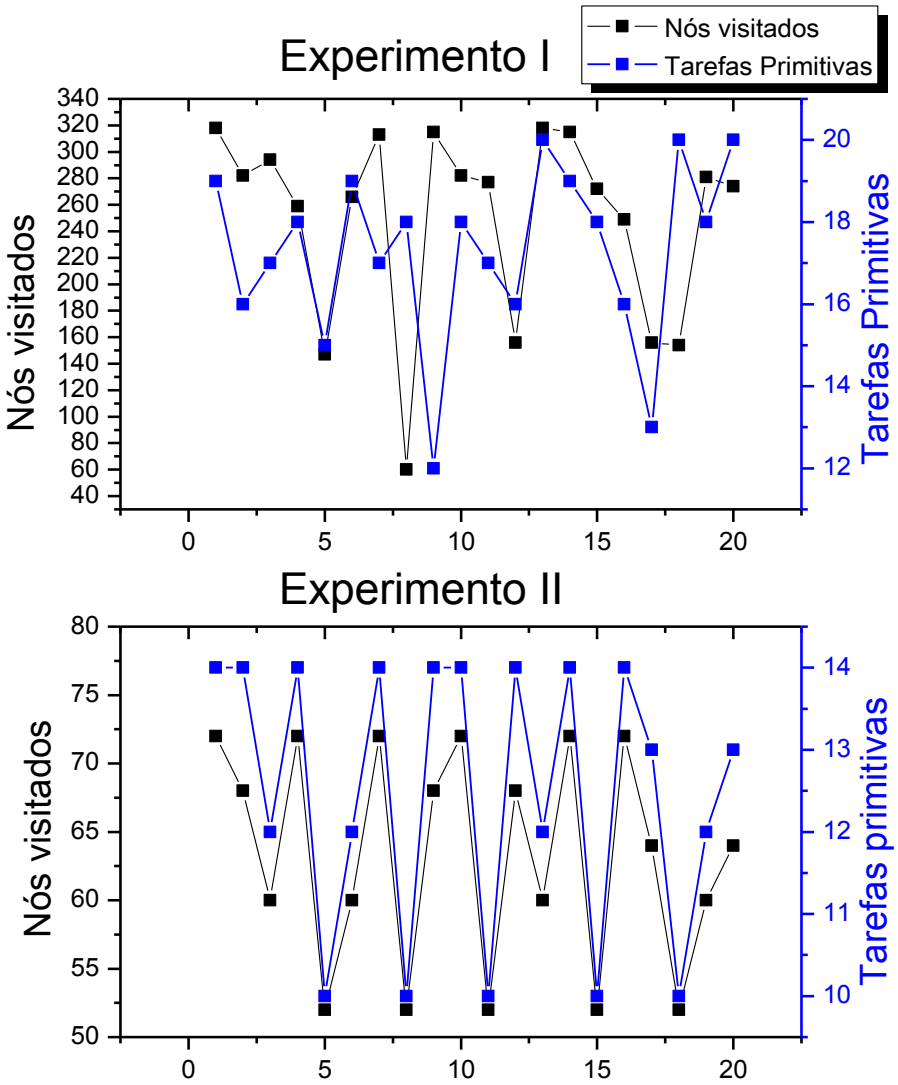
Os dados de desempenhos são promissores, considerando-se o tamanho da entrada do problema.

[51] Dado um conjunto de dados, a variância (σ^2) é uma medida de dispersão que mostra o quão distante cada valor desse conjunto está do valor médio.

8.7.6.Cenário VI (análise de cenário nominal e com refinamento)

Nesta seção mostra-se um breve experimento apresentado pelos gráficos da Figura 8.16, que relaciona o número de nós visitados e o total de ações contidas no plano solução. O objetivo é confrontar o comportamento da solução de planejamento, considerando-se cenários nominais e aqueles que requerem refinamento de estados.

Figura 8.16 – Experimento de vinte cenários considerando nós visitados *versus* ações primitivas.



Fonte: Produção do autor.

Cada ponto do gráfico representa um problema contendo um objetivo escolhido aleatoriamente. A diferença entre eles é que o Experimento I tem estados iniciais

imprevisíveis e o Experimento II lida com situações previstas no domínio a partir dos métodos HTN.

Observa-se que para cada objetivo com cenários de reparo a solução da HARPIA visitou de 200 a 320 nós, enquanto que no segundo de 60 a 75. Isso dá uma noção da sobrecarga de processamento necessária para alcançar o refinamento de estados.

Outra análise interessante é que no gráfico inferior a quantidade de nós visitados acompanha o número de ações primitivas do plano final. Já no gráfico superior isso nem sempre ocorre devido à necessidade de refinar o plano na busca por uma solução.

8.7.7. Cenário VII (falha em restrições de eventos exógenos)

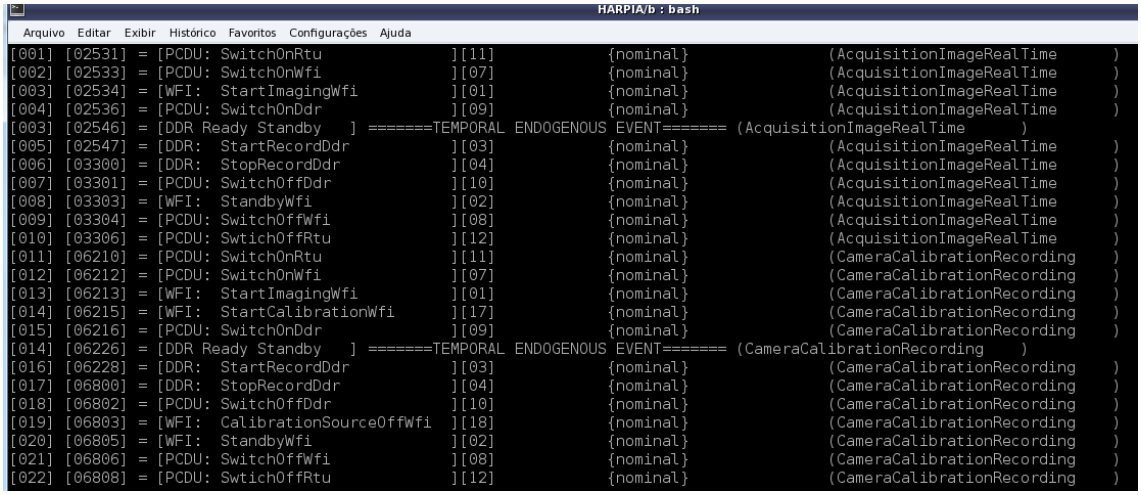
Todos os cenários executados anteriormente assumem que os eventos exógenos modelados estavam conforme o objetivo realizado esperava. A principal motivação de modelá-los é manter a segurança operacional do satélite e poupar recursos a bordo, não acionando, por exemplo, um transmissor fora de visibilidade.

Como o planejador não tem controle sobre os eventos exógenos, eles foram considerados nas precondições das tarefas abstratas de maior nível hierárquico, ao invés de descrevê-los nas precondições de ações primitivas. Como dito antes, isso visa antecipar falhas que não têm uma solução, ao menos pela técnica de refinamento.

A seguir, particulariza-se um exemplo de como uma falha poderia ser resolvida usando-se um método HTN alternativo. O plano de voo original tinha dois objetivos, o primeiro era realizar um imageamento em tempo real h_t[2530,3300], entretanto, o satélite estava fora de comunicação, violando-se um evento exógeno do domínio.

Para fins de testes, escreveu-se um segundo método que tenta alternativamente fazer a gravação da imagem a bordo ao invés de transmiti-la. O plano alcançado pela HARPIA está representado na Figura 8.17.

Figura 8.17 – Plano com método HTN alternativo a uma restrição de um evento exógeno de visibilidade com a estação.



Fonte: Produção do autor.

As ações de 1 a 10 correspondem à gravação de imagens a bordo. Note-se que o primeiro objetivo continua sendo virtualmente o objetivo original de transmissão em tempo real e que não houve atuação do algoritmo de refinamento, uma vez que métodos alternativos fazem parte do processo nominal de decomposição de tarefas.

Este exemplo tem a finalidade apenas de mostrar a robustez da solução e a versatilidade da representação por HTN. Na verdade, espera-se que a HARPIA entregue ao planejador um plano livre de conflitos relacionados a eventos exógenos, conforme descrito no Capítulo 5.

Por fim, destaca-se que o novo método HTN escrito não visa prever um estado diferente do satélite, mas sim buscar caminhos alternativos para superar uma restrição referente a um evento exógeno modelado.

8.7.8. Cenário VIII (simulação de falha na execução do plano)

Até agora se mostraram casos de cenários de reparo na fase de planejamento. Isto é, a necessidade de reparar um plano antes de ele ser executado pelo computador de bordo na camada de aplicação.

A pesquisa em falhas na execução é uma temática pouco explorada pelos trabalhos de planejamento em IA, que tipicamente se restringem à fase de deliberação. No entanto, há sinais de mudança em direção a pesquisas aplicadas.

Nessa linha e buscando o planejamento contínuo, a HARPIA propõe um Gerenciador de Atividades, que identifica falhas nos efeitos das ações primitivas ou nas precondições de ações subsequentes do plano.

Para identificar uma falha nos efeitos, o Gerenciador de Atividades analisa o estado do satélite após a execução de uma ação primitiva conforme os estados esperados (vide Tabela 8.17).

Tabela 8.17 – Verificação de estados realizada pelo Gerenciador de Atividades da HARPIA.

Tarefa Primitiva	**Elemento**	**Verificação Após Execução**
StartRecord	DDR	*ddr_mode = ddr_record*
StopRecord	DDR	*ddr_mode = ddr_standby*
StartPlayback	DDR	*ddr_mode = ddr_playback,* *dssState = dss_channel_playback,*
StopPlayback	DDR	*ddr_mode = ddr_stand_by* *dssState = dss_channel_realtime*
EraseFile	DDR	*ddr_mode = ddr_stand_by*
SwitchOnDtLine	PCDU	*dtLine = line_on* *dt_mode = dt_standby_wfi_channel_off*
SwitchOffDtLine	PCDU	*dtLine = line_off* *dt_mode = dt_power_off*
SwitchOnDcDcRtuLine	PCDU	*dcdcRtuLine = line_on*
SwitchOffDcDcRtuLine	PCDU	*dcdcRtuLine = line_off*
SwitchOnWfiSpeLine	PCDU	*wfi_mode = wfi_standby, wfiLine = line_on*
SwitchOffWfiSpeLine	PCDU	*wfi_mode = wfi_power_off, wfiLine = line_off*
SwitchOnDdrLine	PCDU	*dssState = realtime,* *ddrLine = line_on, ddr_mode = ddr_standby*
SwitchOffDdrLine	PCDU	*ddr_mode = ddr_power_off, ddrLine = line_off*
StartImagingWfi	WFI	*wfi_mode = wfi_imaging*
StandbyWfi	WFI	*wfi_mode = wfi_standby*
StartCalibrationWfi	WFI	*wfi_mode = wfi_calibration*
StopCalibrationWfi	WFI	*wfi_mode = wfi_imaging*
SetTurnOnWfiChannel	DT	*dt_mode = dt_nominal_wfi_channel_on*
SetTurnOffWfiChannel	DT	*dt_mode = dt_standby_wfi_channel_off*

Fonte: Produção do autor.

A Figura 8.18 traz o trecho de codificação da HARPIA que trata a verificação e execução das ações primitivas do presente estudo de caso.

Figura 8.18 – Trecho da codificação da HARPIA para verificação (à esquerda) e execução (à direita) de uma ação primitiva do modelo Amazonia-1.

```
bool (*CheckPerformActionTask[number_primitive_actions])()=
{
    NULL,
    &CheckPrimitiveActions::WFIStartImaging,
    &CheckPrimitiveActions::WFIStandby,
    &CheckPrimitiveActions::DDRStartRecord,
    &CheckPrimitiveActions::DDRStopRecord,
    &CheckPrimitiveActions::DDRStartPlayback,
    &CheckPrimitiveActions::DDRStopPlayback,
    &CheckPrimitiveActions::PCDUSwitchOnWfiSpeLine,
    &CheckPrimitiveActions::PCDUSwitchOffWfiSpeLine,
    &CheckPrimitiveActions::PCDUSwitchOnDdrLine,
    &CheckPrimitiveActions::PCDUSwitchOffDdrLine,
    &CheckPrimitiveActions::PCDUSwitchOnDcDcRtuLine,
    &CheckPrimitiveActions::PCDUSwitchOffDcDcRtuLine,
    &CheckPrimitiveActions::PCDUSwitchOnDtLine,
    &CheckPrimitiveActions::PCDUSwitchOffDtLine,
    &CheckPrimitiveActions::DTSetTurnOnWfiChannel,
    &CheckPrimitiveActions::DTSetTurnOffWfiChannel,
    &CheckPrimitiveActions::WFIStartCalibration,
    &CheckPrimitiveActions::WFIStopCalibration,
    &CheckPrimitiveActions::DDREraseFile
};
```

```
// lista de ponteiro de funcao contendo as ações primitivas (HTN)
Action (*PerformPrimitiveActionTask[number_primitive_actions])()=
{
    NULL,
    &WFI::StartImagingWfi,
    &WFI::StandbyWfi,
    &DDR::StartRecord,
    &DDR::StopRecord,
    &DDR::StartPlayback,
    &DDR::StopPlayback,
    &PCDU::SwitchOnWfiSpeLine,
    &PCDU::SwitchOffWfiSpeLine,
    &PCDU::SwitchOnDdrLine,
    &PCDU::SwitchOffDdrLine,
    &PCDU::SwitchOnDcDcRtuLine,
    &PCDU::SwitchOffDcDcRtuLine,
    &PCDU::SwitchOnDtLine,
    &PCDU::SwitchOffDtLine,
    &DT::SetTurnOnWfiChannel,
    &DT::SetTurnOffWfiChannel,
    &WFI::StartCalibrationWfi,
    &WFI::StopCalibrationWfi,
    &DDR::EraseFile
};
```

Fonte: Produção do autor.

A estratégia para a recuperação na fase de execução da HARPIA é a seguinte:

- Durante a execução, o Gerenciador de Atividades monitora se houve alguma falha nos efeitos de uma ação primitiva ou de alguma precondição;
- Identifica-se a ação primitiva em falha e seu instante de execução;
- Verifica-se à qual tarefa abstrata de maior nível hierárquico pertence a ação primitiva em falha, considerando-se seu intervalo temporal;
- Interrompe-se a execução do plano, repassando-se tais informações à camada deliberativa.

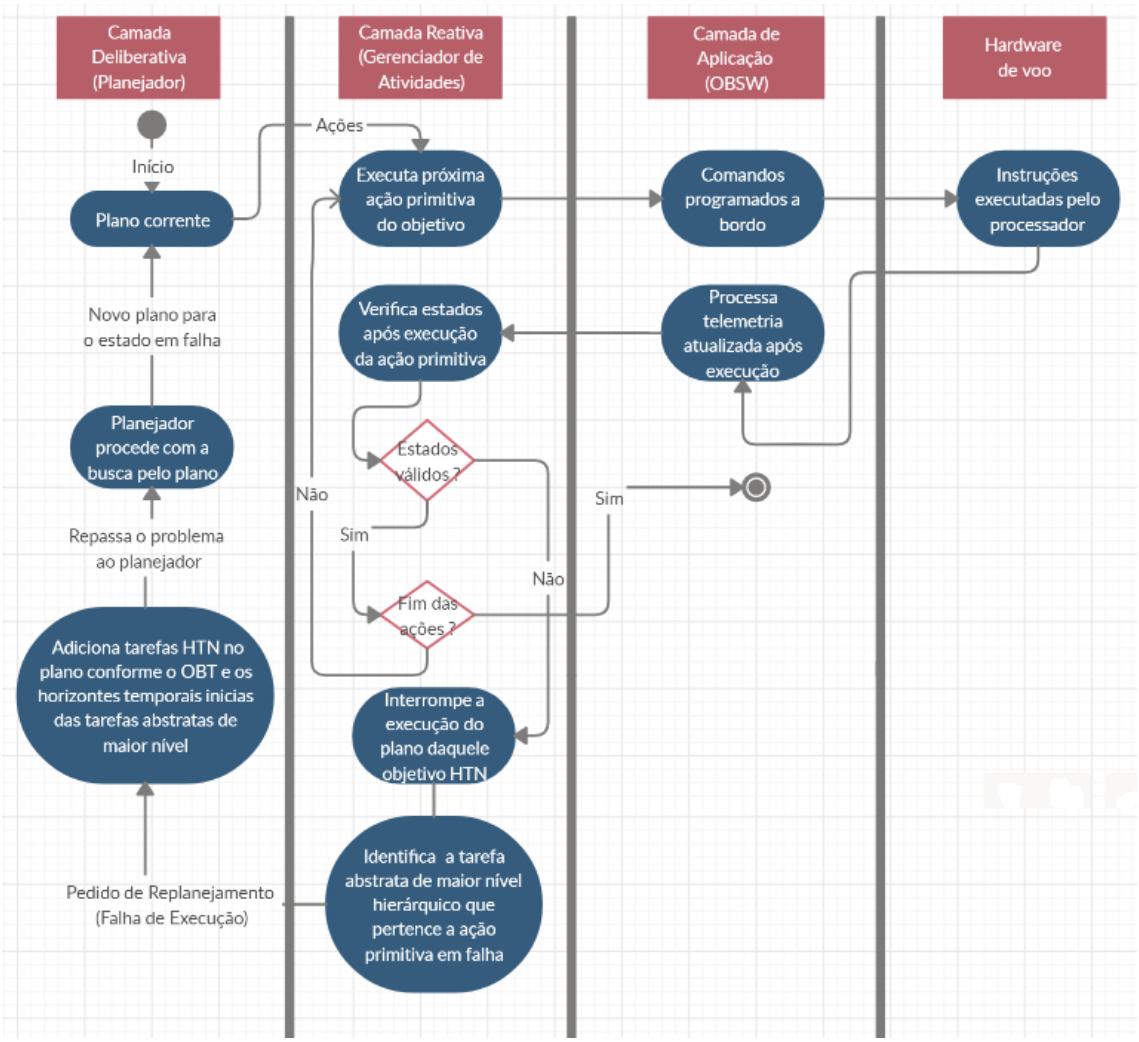
De posse dessas informações, a camada deliberativa procede da seguinte forma:

- Tarefas abstratas cujo intervalo temporal é anterior ao OBT atual são desconsideradas, pois elas já foram executadas pelo computador de bordo;
- Nesse caso, o Liger não realizará mais um objetivo, e sim uma tarefa abstrata com intervalos de execução atualizados;
- Efetua o replanejamento e encaminha o plano à camada reativa;

- A camada deliberativa não analisa falha dupla para um mesmo objetivo já replanejado.

A Figura 8.19 traz um diagrama de atividades com uma síntese da estratégia de monitoração e recuperação da HARPIA para falhas de execução.

Figura 8.19 – Diagrama de atividades representando a estratégia de monitoração e recuperação de falhas de execução da HARPIA.



Fonte: Produção do autor.

Para melhor compreensão da estratégia proposta, pode-se considerar um objetivo representado em HTN, que tenha três níveis de tarefas abstratas (A_1, A_2 e A_3), cada uma contendo um intervalo temporal *h*. Uma faz o ajuste do espelho da câmera h_1

[t_1,t_2], a segunda h_2 [t_3,t_4] adquire a imagem, e a última finaliza o objetivo h[t_5], desligando a carga útil.

Nessa situação, há três possibilidades de replanejamento de execução na HARPIA:

- Falha em A_1, o objetivo é replanejado com o estado em falha, e atualizando o valor t_1 para o OBT atual, após a tentativa de execução;
- Falha em A_2, nessa situação o Liger não trabalha mais no nível de objetivo. São adicionadas as tarefas abstratas A_1 e A_2 considerando o estado em falha, e atualizando o valor t_3 para o OBT atual, após a tentativa de execução.
- Falha em A_3, nessa situação o planejador não trabalha mais no nível de objetivo. É adicionada a tarefa abstrata A_3, considerando o estado em falha recebido e atualizando o valor t_5 para o OBT atual após a falha.

Isso significa que dependendo da ação em falha e do instante temporal, o replanejamento na HARPIA executa apenas parte da rede de tarefas hierárquicas inicial.

Assumindo para o exemplo acima, que uma falha decorreu de uma ação primitiva a_7 cujo pai de maior nível hierárquico é a ação abstrata A_2.

Para o processo de replanejamento da HARPIA não importa exatamente em qual subtarefa de A_2 ocorreu a falha, um novo problema é gerado do zero, considerando na agenda as tarefas abstratas A_2 e A_3 que são filhas do objetivo em falha.

Isso é possível, uma vez que o efeito das ações já executadas no satélite estará implícito no estado em falha recebido da camada reativa.

Como na HARPIA cada tarefa de maior nível hierárquico (A_n) é um momento de inferência de estados, o processo de replanejamento ocorre nesse ponto e possíveis falhas são corrigidas pela abordagem proposta.

À exceção das particularidades discutidas acima, que são realizadas antes de iniciar a busca pelo plano, o processo de replanejamento proposto é análogo às falhas percebidas em tempo de planejamento discutidas anteriormente.

Uma vez que no modelo Amazonia-1 existe dois níveis de abstração para cada objetivo relacionado (ver exemplo da Figura 7.4 da Seção 7.5.2), qualquer falha de execução ocorrida numa ação primitiva filha da subrede do primeiro nível implica em um processo de replanejamento análogo à fase deliberativa.

Nesse caso, o replanejamento volta a ser realizado no nível de objetivo, considerando-se a rede inicial completa do problema, mas com os tempos atualizados. Por essa razão, as simulações de falhas em execução aqui descritas são concentradas no segundo nível de abstração, conforme apresenta a Tabela 8.18.

Como se pode perceber, utilizou-se como rede inicial uma subrede do problema original: a tarefa composta em falha (*e.g., StopAcquisitionRealTime*).

Tabela 8.18 – Descrição dos cenários de simulação de falhas na execução do plano do estudo de caso.

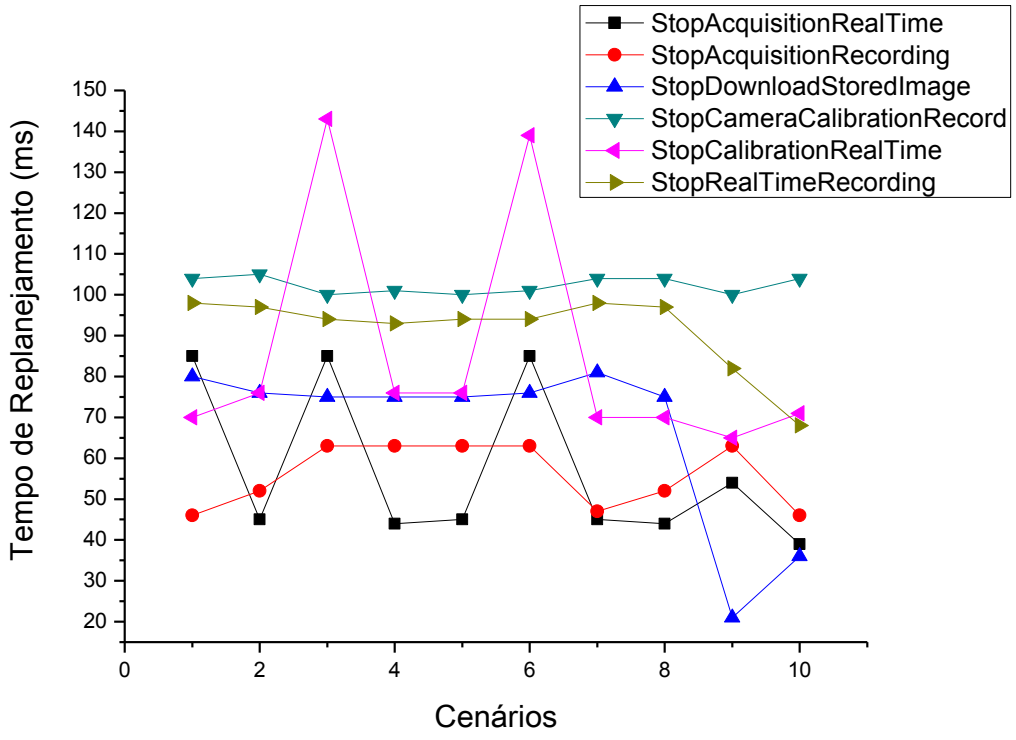
Objetivo	**Ação Primitiva em Falha**
Falha na execução do Objetivo I	Ação primitiva '*n*' da subrede da Tarefa Abstrata "*StopAcquisitionRealTime*" {21}
Falha na execução do Objetivo II	Ação primitiva *'n'* da subrede da Tarefa Abstrata "*StopAcquisitionRecording*" {27}
Falha na execução do Objetivo III	Ação primitiva '*n*' da subrede da Tarefa Abstrata *StopDownloadStoredImage*{31}
Falha na execução do Objetivo IV	Ação primitiva *'n'* da subrede da Tarefa Abstrata *StopCameraCalibrationRecord* {35}
Falha na execução do Objetivo V	Ação primitiva '*n*' da subrede da Tarefa Abstrata *StopCalibrationRealTime*{37}
Falha na execução do Objetivo VI	Ação primitiva '*n*' da subrede da Tarefa Abstrata *StopRealTimeRecording* {40}

Fonte: Produção do autor.

Com base nos cenários descritos no Apêndice C, executaram-se dez cenários aleatórios, aferindo-se o tempo necessário para o replanejamento na HARPIA. O resultado está ilustrado pelo gráfico da Figura 8.20.

Embora tenha apresentado uma oscilação nos tempos de replanejamento um pouco maior se comparado à fase deliberativa[52], a HARPIA se mostra bastante responsiva para falhas de execução, obtendo-se tempos menores que cento e cinquenta milissegundos no pior cenário deste experimento.

Figura 8.20 – Tempo de replanejamento da HARPIA para simulação de falhas ocorridas na execução.



Fonte: Produção do autor.

O desvio padrão calculado para os cenários da Figura 8.20 foram respectivamente (iniciando de cima para baixo a partir da curva de cor preta da legenda): 18,59 ms; 7,38 ms; 19,65 ms; 1,95 ms; 27,91 ms; e 10,75 ms.

[52] Considerando-se as devidas proporções para os tamanhos dos problemas aqui comparados.

Como se pode verificar pelo próprio gráfico, a curva de cor cinza (*StopCameraCalibrationRecord*) apresentou o menor desvio padrão: 1,95 ms e a de cor rosa o maior: 27,91 ms.

Tais resultados mantêm a previsibilidade de execução do algoritmo e sua capacidade de planejamento em tempo real.

De forma a se calcular a média do tempo de planejamento e da quantidade de nós visitados nas falhas de execução, realizaram-se mais quatro séries de experimentos para todos os quarentas cenários previstos pelo Apêndice C, perfazendo-se duzentos e quarenta execuções.

O resultado obtido é exposto na Tabela 8.19. Em todos eles, obteve-se um plano, o que mostra a excelente capacidade de recuperação, também alcançada nos cenários anteriores relativos à fase deliberativa.

Tabela 8.19 – Tempo médio de replanejamento da HARPIA no âmbito de falhas de execução.

Objetivo	Rede Inicial	Tempo Médio de Execução (s)	Média de Nós Visitados
1	*StopAcquisitionRealTime*	0,059	160,15
2	*StopAcquisitionRecording*	0,043	88,275
3	*StopDownloadStoredImage*	0,070	162,675
4	*StopCameraCalibrationRecord*	0,068	155,975
5	*StopCalibrationRealTime*	0,085	220,475
6	*StopRealTimeRecording*	0,090	257,65

Fonte: Produção do autor.

Após a análise do tempo médio, particulariza-se um exemplo de um cenário em falha para o Objetivo 5. A Tabela 8.20 apresenta as ações primitivas programadas no plano de operação e suas temporizações, antes da ocorrência da falha de execução.

Durante a fase de monitoração e controle das ações, o Gerenciador de Atividades constata uma falha ao verificar por telemetria que a ação A11 (destacada em vermelho na Tabela 8.20) não executou seus efeitos esperados.

Isso em função de uma eventual falha (*e.g.*, na comunicação do barramento 1553 entre OBDH e PCDU), que não permitiu desligar corretamente o equipamento de interface da carga útil previsto pela ação A11.

Tabela 8.20 – Objetivo 5 com falha de execução na ação A11.

T	Ação	Descrição
[02001]	[A1]	PCDU:SwitchOnRtu
[02003]	[A2]	PCDU:SwitchOnWfi
[02004]	[A3]	WFI:StartImagingWfi
[02006]	[A4]	WFI:StartCalibrationWfi
[02007]	[A5]	PCDU:SwitchOnDdr
[02017]	[A6]	PCDU:SwitchOnDt
[02260]	[A7]	DT:DtWfiChannelTurnOn
[02720]	[A8]	WFI: CalibrationSourceOffWfi
[02721]	[A9]	WFI: StandbyWfi
[02723]	[A10]	PCDU:SwitchOffWfi
[02724]	[A11]	PCDU:SwtichOffRtu
[02726]	[A12]	PCDU:SwitchOffDdr
[02727]	[A13]	DT:DtWfiChannelTurnOff
[02729]	[A14]	PCDU: SwtichOffDt

Fonte: Produção do autor.

Em decorrência disso, a telemetria da linha de potência da RTU se manteve em "*line_on*". Logo após, o Gerenciador de Atividades verifica que o plano não está apto a proceder com sua consecução, interrompendo-o após a execução de A11. Isso faz disparar uma falha e um pedido de replanejamento é encaminhado à camada deliberativa.

Após a interrupção, o estado em falha da carga útil ficou da seguinte forma: *{wfi_power_off}, {ddr_standby}, {dss_real_time}, {dt_channel_on}, {wfi_line_off}, {ddr_line_on}, {rtu_line_on} e {dt_line_on}*.

Com o estado recebido, o planejador da HARPIA faz o replanejamento agora para o tempo atual do computador t_i[7730] (o instante anterior era conforme destacado em amarelo na Tabela 8.20 indicado por A8). O plano resultante da recuperação consta na Tabela 8.21.

Antes de encaminhar ao software de voo, o Gerenciador de Atividades verifica a ocorrência de eventos temporais endógenos do plano replanejado, de modo análogo ao processo nominal de planejamento.

Tabela 8.21 – Plano de recuperação após falha de execução.

T	Ação	Descrição	Ação
[07731]	[A1]	PCDU: SwitchOnWfi	Ação de Refinamento
[07733]	[A2]	WFI: StartImagingWfi	Ação de Refinamento
[07734]	[A3]	WFI: StartCalibrationWfi	Ação de Refinamento
[07736]	[A4]	WFI: CalibrationSourceOffWfi	Ação de Replanejamento
[07737]	[A5]	WFI: StandbyWfi	Ação de Replanejamento
[07739]	[A6]	PCDU: SwitchOffWfi	Ação de Replanejamento
[07740]	[A7]	PCDU: SwtichOffRtu	Ação de Replanejamento
[07742]	[A8]	PCDU: SwitchOffDdr	Ação de Replanejamento
[07743]	[A9]	DT:DtWfiChannelTurnOff	Ação de Replanejamento
[07745]	[A10]	PCDU: SwtichOffDt	Ação de Replanejamento

Fonte: Produção do autor.

Observe pelo plano resultante que a HARPIA acionou novamente os equipamentos, inclusive aqueles que já haviam sido desligados corretamente antes da falha. Isso decorre pela filosofia de replanejamento empregada, na qual um novo plano é gerado do zero, mas sempre consistente com o estado em falha recebido.

Com relação à taxa de sucesso, sem utilizar a heurística adicional, o percentual alcançado para as falhas de execução é na média de 55,4%, considerando-se os cenários do Apêndice C. Trata-se de um resultado significativamente inferior se comparado à fase deliberativa (90,03%).

Por outro lado, com a heurística integrada obteve-se o percentual de 100% para os mesmos cenários realizados. Isso demonstra a relevância da integração das heurísticas propostas nesta Tese no sentido de aumentar a capacidade de resposta do satélite.

Assim como na etapa deliberativa, em nenhum dos cenários analisados uma ação de refinamento incluída no plano final era dispensável, ou irrelevante ao processo de replanejamento aqui descrito.

Um estudo de escalabilidade para falhas de execução não se fez necessário, pois os problemas de replanejamento são resolvidos na HARPIA considerando-se um objetivo por vez, conforme descrito anteriormente.

Por fim, destaca-se que todos os dados apresentados neste capítulo acerca de reparo e replanejamento, sejam no âmbito de falhas em tempo de planejamento ou de execução, foram obtidos por um único motor de busca HTN. Sendo mais específico, a partir da solução híbrida de planejamento embarcado que a HARPIA realiza.

8.8. Comparação de desempenho com demais abordagens

O desempenho é sempre um fator primordial para trabalhos de planejamento ainda mais se tratando de uma aplicação espacial de tempo real. A própria NASA em seus artigos apresenta o algoritmo de reparo iterativo (CASPER) como uma grande evolução frente ao seu antecessor PS-RAX em termos de desempenho.

Considerando-se que pesquisas aplicadas possuem modelos voltados a uma aplicação, não se podem comparar igualmente os resultados de cada planejador (exceto os trabalhos do próprio INPE).

Apesar disso, é bastante representativa a análise da ordem de grandeza dos resultados (milissegundos, segundos, minutos, etc.) como forma de mensurar o potencial de resposta da aplicação e de consolidar os resultados da literatura.

Isso é factível, pois os modelos, embora diferentes, lidam com os mesmos aspectos, considerando-se o domínio de satélite: componente temporal, recursos a bordo, e necessidade de replanejar.

8.8.1. Análise comparativa

Uma análise comparativa mais geral das soluções de trabalhos relacionados foi apresentada na Seção 7.8. Nesta seção mostra-se uma compilação dos dados de desempenho dos estudos que tiveram seus resultados divulgados.

Os tempos de planejamento devem ser analisados à luz dos dados expostos na Tabela 8.22. A Tabela 8.23 traz o ambiente computacional e a alocação do processador em que estes dados de desempenho foram aferidos.

Como visto, o tempo de execução depende dentre outros fatores das características do problema, como tamanho do horizonte temporal, tipo de solicitação (criar um novo plano ou requisitar mudança em um plano preexistente), do grau de impacto das mudanças no plano original, e do estado dos equipamentos.

Tabela 8.22 – Comparação de dados de desempenho das soluções de planejamento embarcado em satélites.

Nome	Domínio de Aplicação	Tempo de Planejamento	Algoritmo de Busca	Referência do Desempenho
PS-RAX (RAX)	Científica de Espaço Profundo	De **30 minutos a 4 horas** dependendo do tamanho do problema	CSP (Busca em profundidade com retrocesso)	Muscettola et al. (2002)
CASPER (ASE)	Observação da Terra	Da **ordem** de **dezenas** de **minutos**	CSP (Busca local)	Sherwood et al. (2004)
LetMeDo (GOESA)	Observação da Terra	Para novos planos, ou modificações de baixo impacto: 1 a 3 minutos. Pior caso: **25 minutos**	CSP (Busca local)	Kucinskis (2012)
Liger (HARPIA)	Observação da Terra	**Da ordem de milissegundos** mesmo no pior caso, ver Tabela 8.25	HTN (Abordagem híbrida com busca progressiva e de ordem total)	-

Fonte: Produção do autor.

Nota-se pela Tabela 8.22 que o Liger/HARPIA supera outras soluções de planejamento embarcado, permitindo que o satélite produza planejamento automatizado em tempo real, característica importante e não alcançada pelas demais abordagens.

Tabela 8.23 – Ambiente computacional espacial das soluções de planejamento embarcado.

Nome	Processador Espacial	Uso da CPU	Memória RAM	RTOS[53]
PS-RAX (RAX)	PowerPC RAD6000 25 MHz (3 a 35 MIPS)	50%	32 MB	VxWorks
CASPER (ASE)	Mongoose-V R3000 12 MHz 4 MIPS[54]	100%	40 MB para o CASPER	VxWorks
LetMeDo (GOESA)	ERC32 15 MHz 5.8 MIPS	100%	4 MB	RTEMS
Liger (HARPIA)	ERC32 15 MHz 5.8 MIPS	100%	4 MB	RTEMS

Fonte: Produção do autor.

Em algoritmos CSP de busca local, a adição de uma nova atividade pode impactar sensivelmente no tempo de execução, impedindo respostas de tempo real. A principal vantagem do uso de CSP é geralmente ter flexibilidade para resolver diferentes tipos de problemas e sua boa capacidade de replanejamento.

Nessa abordagem, entretanto, não se tem previsibilidade do tempo de execução e a garantia de que uma solução de fato será encontrada.

Infelizmente os trabalhos supramencionados não divulgaram análises da capacidade de resposta para diferentes configurações de estados, como realizado nesta Tese, a qual apresentou resultados bastante promissores para as simulações realizadas, conforme sintetiza a Tabela 8.24.

Uma vez que a capacidade de resposta é uma das principais deficiências do planejamento HTN, pode-se demonstrar pelos resultados obtidos que a solução híbrida proposta nesta Tese é bastante eficiente.

[53] RTOS = *Real-time Operating System.*
[54] MIPS = *Millions of Instructions Per Second*.

Tabela 8.24 – Sumário da taxa de sucesso da HARPIA para as simulações realizadas.

Tipo de Planejamento	Taxa de Sucesso Heurística Nominal	Taxa de Sucesso Heurística Integrada
Planejamento e Replanejamento (Camada Deliberativa)	90,03%	100%
Replanejamento de Execução (Camada Reativa)	55,4%	100%

Fonte: Produção do autor.

Os resultados sumarizados na Tabela 8.25 permitem afirmar que a HARPIA é computacionalmente leve e, portanto, cumpre com um dos objetivos deste trabalho. Isso possibilita sua integração a um software de voo de satélites, cuja solução consome geralmente parte considerável da capacidade de processamento da CPU.

Ressalta-se que existem processadores espaciais com maior poder computacional[55] do que o ERC32 utilizado. Isso traz margem quanto aos dados de desempenho apresentados.

Tabela 8.25 – Síntese dos tempos de planejamento da HARPIA para as simulações realizadas.

Descrição do Cenário	Tempo Médio	Pior Caso
Um objetivo, independente do estado inicial	~ 100 ms	~120 ms
Até cinco objetivos com replanejamento, independente do estado inicial	< 500 ms	~ 630 ms
Replanejamento decorrente de falha na execução do plano	~ 75 ms	~140 ms

Fonte: Produção do autor.

[55] Como exemplo, cita-se a família de processadores LEON.

9 CONCLUSÃO

O domínio de satélites é certamente um tópico de grande relevância na área de planejamento automatizado. Os experimentos em voo realizados pela NASA em busca da maior autonomia a bordo de veículos espaciais são frequentemente citados pelos diversos artigos da comunidade de planejamento, como exemplos expressivos de aplicações práticas realizadas nesta área.

Apesar da evidente relevância deste tema, o número de trabalhos de planejamento em IA que se propõe a ser embarcados em satélites, seja por simulações via estudo de caso ou por experimentos com validação em voo, ainda é bastante reduzido e com poucos resultados alcançados se comparado a outros domínios.

Até hoje os trabalhos da área de autonomia a bordo, inclusive os do INPE, utilizaram planejamento baseado em CSP de busca local, com heurística de reparo iterativo cujos resultados de desempenho não conseguiram reprogramar as atividades operacionais do satélite em tempo real.

Um dos desafios é que os processadores espaciais passam por um rigoroso processo de qualificação, a fim de permitir menor consumo de energia e maior resistência às altas taxas de radiação espacial. Isso pode levar anos para ser concluído, de forma que os processadores utilizados em missões espaciais são muito defasados em relação aos modelos mais recentes do mercado, com capacidade computacional ordens de grandeza inferior.

Buscando contribuir com o estado da arte, um planejador híbrido e modelo embarcado foram propostos, implementados e testados neste trabalho de Doutorado como elementos de uma arquitetura de planejamento HTN organizada em camadas, visando à capacidade de revisão de planos em tempo real. Tal solução é computacionalmente leve e adequada ao hardware no qual se pretende executar.

De forma a intercalar as fases de deliberação e execução do plano e buscando o planejamento contínuo da aplicação, propôs-se uma camada intermediária, que

coordena a monitoração e controle das ações durante a execução do plano a partir do software de voo.

A solução centraliza planejamento e replanejamento em um único motor de busca HTN baseado no algoritmo de refinamento de estados proposto. Independente de qual operação autônoma é solicitada (planejamento ou falha de execução), o mecanismo de busca é o mesmo.

Isso revela uma propriedade importante da HARPIA (sobretudo para aplicações práticas), uma vez que ela não depende de outras abordagens específicas voltadas a reparo do plano.

Os experimentos realizados em cenários realísticos de planejamento automatizado, usando-se um ambiente computacional representativo da área espacial evidenciaram que a solução proposta é escalável e com boa taxa de sucesso na resolução dos problemas independente do estado inicial dos equipamentos.

Os resultados de desempenho se mostraram bastante promissores frente ao estado da arte do domínio embarcado de satélites. Ressalta-se que, no melhor juízo deste autor, não se conhece outro estudo aplicado ao domínio de satélites, que utilizando processadores espaciais, tenha conseguido atingir o potencial de desempenho apresentado pelos resultados desta Tese.

O modelo criado do Amazonia-1 para o estudo de caso também é bastante representativo e próximo do comportamento real do satélite.

Isso realça que a solução da HARPIA consegue lidar com as severas restrições de recursos computacionais de hardware espacialmente qualificado e com as estritas restrições de tempo real impostas por um software de voo de satélites.

Considera-se que a proposta desta Tese tem condição de evoluir para uso em uma missão futura de validação científica e tecnológica voltada à operação automatizada de um satélite cujo controle é baseado em objetivos.

Conclui-se que o paradigma híbrido permite acrescentar ganhos de desempenho computacional usando as informações hierárquicas do domínio e, ao mesmo tempo, ser mais responsivo a situações imprevistas, através da abordagem generativa, se mostrando uma estratégia bastante interessante principalmente para aplicações de tempo real, como satélites em órbita.

O fato de a solução proposta conseguir refinar o estado do satélite sem depender exclusivamente de métodos de decomposição alternativos também traz como benefício direto a redução do esforço de engenharia do conhecimento para a construção do domínio, umas das desvantagens imputadas à representação em HTN.

9.1. Contribuições da Tese

A presente Tese tem três contribuições principais ao estado da arte, as quais estão elencadas a seguir.

I. **Uma solução híbrida de planejamento embarcado baseada em rede de tarefas hierárquicas e inferência de estados** para aumentar a capacidade de resposta de satélites;
II. **Uma arquitetura de planejamento embarcado organizada em camadas hierárquicas** que atende à solução proposta e à capacidade de monitoração das atividades durante a execução do plano; e
III. **Uma linguagem estendida para representação em HTN**, que unifica as etapas de planejamento, escalonamento e execução das atividades.

As três principais contribuições desta Tese são melhor destacadas a seguir.

I. Solução Híbrida de Planejamento Embarcado Baseada em HTN

Esta contribuição se dá por uma nova abordagem híbrida de planejamento embarcada centrada nas regras de controle da decomposição hierárquica de tarefas aliada à abordagem generativa do planejamento clássico.

Tal proposta somada a outras heurísticas que integram o algoritmo de refinamento transforma os planos hierárquicos do planejamento HTN tradicional em planos

parcialmente hierárquicos, aumentando substancialmente a capacidade de resposta do satélite a eventos não previstos e ao mesmo tempo reduzindo o custo de descrição do domínio.

Em relação a outras abordagens híbridas que aliam planejamento HTN com a abordagem generativa, a proposta deste trabalho se diferencia por não utilizar o planejamento no espaço de planos. As ações de refinamento são selecionadas em pontos específicos da rede de tarefas a partir de heurísticas de seleção. Isso mantém a busca progressiva e faz com que os estados evoluam sempre de forma determinística, trazendo os seguintes benefícios:

- O algoritmo se mantém de ordem total usando a abordagem de espaço de estados, diferentemente das demais abordagens. Isso traz aumento de desempenho por não precisar de um solucionador de conflitos, além de evitar a necessidade de um executor que faça a linearização do plano. Em termos práticos, isso alcança respectivamente maior capacidade de resposta em tempo real do satélite e compatibilidade com um OBSW tradicional.
- Um dos principais argumentos da abordagem de ordem parcial, é que esta seria mais adequada a ambientes dinâmicos, onde certas informações são conhecidas apenas em tempo de execução. Como dito antes (Seção 7.4), este não é o caso de satélites em órbita que operam em ambientes previsíveis e bem definidos.

II. Arquitetura de Planejamento Automatizado a Bordo de Satélites

A HARPIA tem componentes que permitem unificar planejamento e escalonamento usando um planejador hierárquico com a possibilidade de replanejar um plano por falhas ou desvios identificados em órbita. Em termos práticos, isso torna o sistema mais responsivo, uma das principais deficiências do planejamento em HTN quando aplicado fora do planejamento clássico.

A proposta deste trabalho também se diferencia de demais abordagens que realizam monitoração e controle de atividades por unificar o processo de planejamento e

execução a partir de uma representação hierárquica centralizada, com diferentes níveis de abstração das atividades operacionais. Isso evita a necessidade de uma linguagem intermediária para execução do plano, como proposto por abordagens similares (RAX e ASE).

III. Extensão da Linguagem ISISml

A linguagem estendida prevê recursos que permitem codificar procedimentos operacionais hierárquicos escritos por especialista, integrando as atividades de planejamento e execução a partir do padrão PUS da ECSS.

Também propõe uma maneira prática e simples de descrever o comportamento do sistema espacial e, ao mesmo tempo, que atende às demandas atuais de satélites com autonomia a bordo: maior abstração, desempenho e modularidade do modelo.

Comparando com demais linguagens existentes, destaca-se:

O domínio espacial até hoje foi modelado por abordagens ricas em representações temporais voltadas a planejadores CSP que carregam as seguintes deficiências: tem baixa capacidade de abstração do domínio e não é um modelo escalável e de planejamento adaptável.

Em princípio, os contrapontos acima são superáveis pela proposta da extensão hierárquica à ISISml.

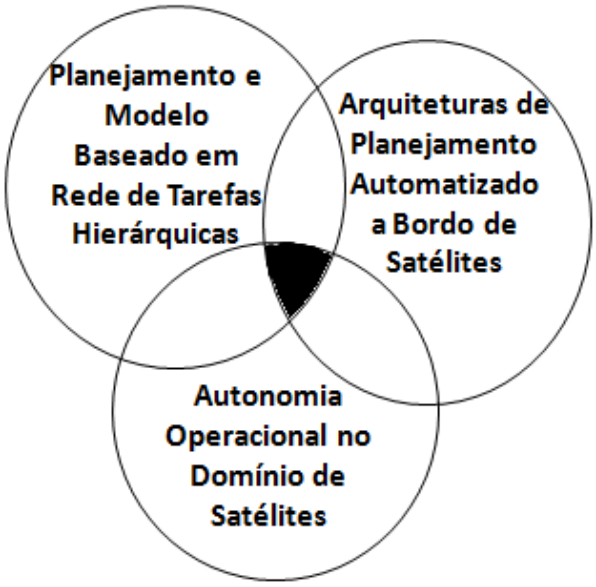
Ao estender a forma de consumir e produzir recursos da ISISml para o domínio baseado em rede de tarefas, traz-se uma nova forma de representação de consumo para planejadores hierárquicos que unifica planejamento e escalonamento, facilitando o processo de inferência de estados para problemas que são dependentes de recursos, como o domínio de satélites e outras aplicações de engenharia.

Isso alia características positivas de planejadores temporais (para a qual foi desenvolvida originalmente a ISISml), mas usando planejadores HTN, como o proposto nesta Tese.

9.1.1. Áreas do conhecimento das contribuições

A Figura 9.1 ilustra as áreas de conhecimento que estão contidas na contribuição desta Tese delimitada pela intersecção dos Diagramas de Venn. Note-se que a contribuição contempla áreas oriundas da pesquisa acadêmica (planejamento em IA) e aplicada (área espacial).

Figura 9.1 – Diagramas de Venn com as áreas de conhecimento relacionadas à contribuição desta Tese.



Fonte: Produção do autor.

Uma vez apresentado isso, correlacionam-se a partir da Tabela 9.1 as áreas do conhecimento apresentadas nos diagramas com as contribuições desta Tese.

Tabela 9.1 – Áreas do conhecimento relacionado às contribuições da Tese.

Identificação da Área	Área de Conhecimento	Contribuição da Tese
I	Planejamento e Modelo Baseado em Rede de Tarefas Hierárquicas	A solução híbrida de planejamento embarcado e a linguagem estendida para representação em HTN
II	Autonomia Operacional no Domínio de Satélites	A capacidade de revisão de planos em tempo real das atividades do satélite a partir da solução híbrida de planejamento proposta
III	Arquitetura de Sistemas de Planejamento Automatizado a Bordo de Satélites	A solução da HARPIA a partir de uma pesquisa aplicada a um software de voo de satélites

Fonte: Produção do autor.

9.1.2. Contribuições para o INPE

As missões do INPE atualmente dependem de telecomandos temporizados, planejados com antecedência, para operar o satélite em intervalos fora de visibilidade cujo percentual pode perfazer até 90% do tempo total de operação da missão.

Tal forma de controle não permite aos satélites do INPE reagirem a eventos que não podem ser previstos com antecedência em solo.

Com o objetivo de melhorar essa forma de operação, se propôs neste trabalho uma solução de planejamento automatizado com alta capacidade de resposta, visando à maior autonomia a bordo de satélites.

Podem-se destacar as seguintes contribuições para o INPE derivadas desta Tese:

a) A possibilidade de reprogramar um plano de voo em tempo real quando da ocorrência de falhas na sua execução;
b) A maior capacidade de resposta do satélite a eventos imprevisíveis identificados em órbita, como um evento ambiental ou científico de interesse da missão;
c) A potencial redução de custo de operação em solo; e

d) A capacidade de um satélite operar melhor em missões espaciais com pouco tempo de visibilidade.

Os itens a) e b) têm o potencial de diminuir o tempo de reação do satélite, provendo uma resposta adequada ao evento percebido em órbita. Isso, além de aumentar o retorno dos dados de missão aos usuários finais, maximiza o uso da plataforma espacial.

Com relação à maior capacidade de resposta, citam-se dois exemplos concretos de limitações enfrentadas pelas missões atuais do INPE que poderiam ser superadas pela abordagem proposta:

- Se uma ação operacional da carga útil falhar (*e.g.,* a câmera não entrou no modo previsto), todo o objetivo relacionado é perdido.
- Se um novo evento é identificado a bordo (*e.g.*, evento ambiental de interesse), não há tempo hábil para replanejar as atividades devido à operação ser realizada em malha aberta.

Nas missões atuais do INPE, a restrição temporal imposta pela comunicação solo-bordo somada ao planejamento de operação realizado em solo com antecedência impede a reprogramação das atividades em tempo real.

O contexto acima evidencia que este trabalho traz avanços para o INPE, alavancando estudos em direção à maior autonomia de satélites do instituto.

Por fim, cita-se como contribuição complementar deste trabalho o extenso estudo acerca do estado da arte de autonomia operacional e técnicas de planejamento automatizado para o domínio embarcado espacial cuja temática foi pouco explorada pela literatura atual. Considera-se que esta Tese proporciona maior compreensão acerca deste assunto, ajudando a direcionar pesquisas futuras do próprio instituto.

9.2. Publicações

Listam-se a seguir as publicações derivadas desta Tese que foram publicadas e/ou apresentadas em eventos da comunidade acadêmica.

- CIVIDANES, F. S.; FERREIRA, M. G. V.; KUCINSKIS, F. N. On-board Automated Mission Planning for Spacecraft Autonomy: A Survey. *IEEE Latin America Transactions*, v. 17, p. 884-896, 2019.
- CIVIDANES, F. S.; FERREIRA, M. G. V.; KUCINSKIS, F. N. Um planejador embarcado em satélites autônomos baseado em rede de tarefas hierárquicas e inferência de estados. In: ENCONTRO LATINO AMERICANO DE PÓS-GRADUAÇÃO (EPG), 2019, São José dos Campos. XIX Encontro Latino Americano de Pós-Graduação (XIX EPG). SAO JOSE DOS CAMPOS: UNIVAP, 2019.
- CIVIDANES, F. S.; FERREIRA, M. G. V.; KUCINSKIS, F. N. Uma estratégia híbrida de planejamento embarcado de missão para aumentar a capacidade de resposta de um satélite com autonomia operacional em bordo. In: WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS (WETE), 2019, SÃO JOSE DOS CAMPOS. X WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS. SÃO JOSÉ DOS CAMPOS: INPE, 2019.
- CIVIDANES, F. S.; FERREIRA, M. G. V.; KUCINSKIS, F. N. Uma Arquitetura Hierárquica em Camadas para Habilitar o Planejamento Embarcado em Software de Voo de Satélites Baseada em Rede de Tarefas Hierárquicas. In: WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS (WETE), 2020, SÃO JOSE DOS CAMPOS. XI WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS. SÃO JOSÉ DOS CAMPOS: INPE, 2020.

Além disso, foi submetido um artigo derivado da tese à *IEEE Aerospace and Electronic Systems Magazine* (Fator de Impacto Scopus = 2.370 e JCR = 1.539) contendo os resultados do estudo de caso.

Cabe mencionar que a partir dos artigos publicados, foram recebidos alguns convites de atuação na comunidade acadêmica, os quais estão listados a seguir.

- Convite para atuar como revisor do periódico IEEE *Systems Journals* (Qualis = A2 e FI = 3.987), especificamente, em artigos da área de autonomia e planejamento;

- Convite para ser membro do comitê organizador do congresso internacional: "ICNPAA 2020 *world congress* - 13th: *International Conference on Mathematical Problems in Engineering, Aerospace and Sciences*" na Universidade Técnica Tcheca em Praga, República Tcheca;
- Convite para publicação do artigo apresentado no X WETE como capítulo de livro na editora Atena;
- Convite para publicações em outras revistas internacionais (*e.g.*, *Journal of Electrical Engineering*) e para atuar como revisor de outros periódicos de menor fator de impacto que se mostraram interessadas nos temas publicados pelo INPE.

9.3. Trabalhos futuros

A seguir são enumeradas ideias para possíveis trabalhos futuros que podem ser entendidas como uma complementação ou continuação deste trabalho. A HARPIA juntamente com o modelo construído poderiam ser utilizados como estudo de caso para os temas nº 2 e 6.

1. **Replanejamento por falhas de restrições temporais**: A abordagem de refinamento de estados baseada em HTN proposta nesta Tese lida com falhas de ações, precondições e efeitos sob o ponto de vista de estados do satélite, mas não permite replanejar um plano por falha de restrições temporais; este é um problema que poderia ser estudado como continuação dessa pesquisa. No algoritmo de reparo iterativo, por exemplo, isso é possível pela sua capacidade de deslocar as ações do plano no domínio do tempo. No entanto, cabe destacar que a grande preocupação na operação de um satélite geralmente está no seu estado, sendo a componente temporal uma consequência do processo.
2. **Verificação e testes**: Métodos formais, como a verificação por modelos (*Model Checking*), podem oferecer uma cobertura exaustiva ou mensurável de teste, o que traz uma maior confiança no modelo de planejamento embarcado construído, vide Smith et al. (2005) para um exemplo de pesquisa neste assunto. Trata-se de um tema relevante para área espacial, considerando-se a

importância de se produzir planos automatizados confiáveis. Este é um tópico praticamente não explorado na área espacial.

3. **Modelos HTN com aprendizado de máquina**: O mecanismo de decomposição de métodos HTN desta Tese poderia ser estendido para dispor de técnicas de autoaprendizagem. Um exemplo de pesquisa nessa linha pode ser visto em Zhuo et al. (2014). Isso permitiria construir o domínio hierárquico de forma automatizada, economizando esforço na sua descrição. A dificuldade nesse caso consiste em manter o nível de confiabilidade esperado por missões espaciais. Um artigo de revisão sobre aprendizado de máquina aplicado a modelos e domínios de planejamento pode ser visto em Jiménez et al. (2012);
4. **Planejamento em ambiente parcialmente observável**: Embora alguns planos possam ser produzidos na presença de ruídos, a HARPIA não dispõe de técnicas próprias para lidar com ruídos nos sensores do satélite. Isso permitiria encontrar um plano mesmo quando parte da telemetria do satélite apresente inconsistências ou esteja indisponível. Ressalva-se, no entanto, que o domínio de satélites não constitui exatamente um problema típico de um ambiente parcialmente observável. Seu projeto de hardware é construído com redundâncias e interfaces robustas. Se uma falha nos sensores se persistir é provável que se tenha comprometido partes vitais à operação nominal da plataforma espacial;
5. **Satélites autônomos colaborativos**: Esta Tese trata a solução de planejamento embarcado a partir de um satélite autônomo. A HARPIA poderia ser estendida para estudos voltados a sistemas colaborativos, onde se constitui uma constelação de satélites autônomos a partir de um veículo espacial líder que coordena a missão, distribuindo objetivos entre as plataformas espaciais. Cita-se como exemplo de aplicação, uma constelação de satélites de sensoriamento remoto voltada ao controle de eventos ambientais na superfície Terrestre. A missão FireBird (LENZEN et al., 2014) da DLR busca essa linha de autonomia. O segmento espacial é composto por dois satélites que monitoram incêndios, no

entanto, eles não utilizam planejamento embarcado baseado em objetivos, conforme dito na Seção 4.3;

6. **Determinador embarcado de objetivos**: Para uma operação completamente autônoma é preciso que o segmento espacial possua algoritmos inteligentes para detectar fenômenos de interesse a partir da análise de dados da carga útil, conforme conceitualmente proposto pela HARPIA, mas não implementado. Isso permitiria gerar novos objetivos ao planejador embarcado em tempo real. Um estudo poderia ser feito, por exemplo, aplicado a missões de sensoriamento remoto para embarcar algoritmos de reconhecimento de padrões, de forma a detectar eventos de interesse da missão a partir de imagens de satélites. Um tipo de evento de alta relevância ao Brasil é a detecção de incêndios, já realizada e monitorada pelo INPE no segmento solo. Ao final da Seção 2.3.2 encontram-se referências a trabalhos inseridos nesta linha de pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

AI-CHANG, M.; BRESINA, J.; CHAREST, L.; CHASE, A.; HSU, J. C. J.; JONSSON, A. MAPGEN: mixed-initiative planning and scheduling for the Mars exploration rover mission. **IEEE Intelligent Systems**, v. 19, n. 1, p. 8-12, 2004.

ALTINOK, A.; THOMPSON, D. R.; BORNSTEIN, B.; CHIEN, S. A.; DOUBLEDAY, J.; BELLARDO, J. Real-time orbital image analysis using decision forests, with a deployment onboard the IPEX spacecraft**. Journal of Field Robotics***,* v. 33, p. 187-204, 2015.

AMIGONI, F.; GUALANDI, S.; MENOTTI, D.; SANGIOVANNI, G.A multiagent architecture for controlling the Palamede satellite. **Web Intelligence and Agent Systems: An International Journal**, v. 8, n. 3, p. 269-289, 2010.

ARIAS, R.; KUCINSKIS, F. N.; ALONSO, J. D. Lessons learned from an onboard ECSS PUS object-oriented implementation. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 10., 2008, Heidelberg, Alemanha. **Proceedings...** Heidelberg: AIAA, 2008.

ASCHWANDEN, P.; BASKARAN, V.; BERNARDINI, S.; FRY, C.; MORENO, M.; MUSCETTOLA, N.; PLAUNT, C.; RIJSMAN, D.; TOMPKINS, P. Model-unified planning and execution for distributed autonomous system control. In: WORKSHOP ON SPACECRAFT AUTONOMY: USING AI TO EXPAND HUMAN SPACE EXPLORATION, AIAA FALL SYMPOSIUM, 2006, Washington DC, USA. **Proceedings...** Washington DC: AIAA, 2006.

AXMANN, R. **Interactive acquisition scheduling for low earth orbiting satellites**. 2010. 112 p. Tese (Doutorado em Engenharia Mecânica) - Universidade Técnica de Munique, Munique, 2010.

BARREIRO, J.; BOYCE, M.; DO, M.; FRANK, J.; IATAURO, M.; KICHKAYLO, T.; MORRIS, P.; ONG, J.; REMOLINA, E.; SMITH, T.; SMITH. D. EUROPA: a platform for AI planning, scheduling, constraint programming, and optimization. In: INTERNATIONAL COMPETITION ON KNOWLEDGE ENGINEERING FOR PLANNING AND SCHEDULING (ICKEPS), 4., 2012, Atibaia, SP, Brasil. **Proceedings...** Atibaia: AAAI, 2012.

BEAUMET, G. **Planification continue pour la conduite d'un satellite d'observation agile autonome.** 2008. 139 p. Tese (Doutorado em Sistemas Embarcados) - Instituto Superior de Aeronáutica e Espaço, Universidade de Toulouse, Toulouse, 2008.

BEAUMET, G.; VERFAILLIE, G.; CHARMEAU M. C. Feasibility of autonomous decision making on board an agile earth-observing satellite. **Computational Intelligence,** v. 27, n.1, p. 123-139, 2011.

BECHON, P.; BARBIER, M.; INFANTES, G.; LESIRE, C.; Vidal, V. HiPOP: hierarchical partial-order planning. In: STARTING AI RESEARCHER SYMPOSIUM (STAIRS), 7., 2014, Prague, Czech Republic. **Proceedings...** Prague: IOS Press, 2014.

BENSALEM, S.; HAVELUND K.; ORLANDINI, A. Verification and validation meet planning and scheduling. **International Journal on Software Tools for Technology Transfer**, v.16, n.1, p. 1–12, 2014.

BERNARD, D. E.; DORAIS, G. A.; FRY, C.; GAMBLE, Jr.; E. B.; KANFESKY, B.; KURIEN, J.; MILLAR, W.; MUSCETTOLA, N.; NAYAK, P. P.; PELL, B.; RAJAN, K.; ROUQUETTE, N.; SMITH, B.;WILLIAMS, B. C. Design of the Remote Agent experiment for spacecraft autonomy. In: IEEE AEROSPACE CONFERENCE, 1997, Aspen, CO, USA. **Proceedings...** Aspen: IEEE, 1998.

BOZZANO, M; CIMATTI, A.; GUIOTTO, A.; MARELLI, A.; ROVERI, M.; TCHALTSEV, A.; YUSHTEIN, Y. On-board autonomy via symbolic model-based reasoning. In: ESA WORKSHOP ON ADVANCED SPACE TECHNOLOGIES FOR ROBOTICS AND AUTOMATION (ASTRA), 10., 2008, Noordwijk, Holanda. **Proceedings...** Noordwijk: ESA, 2008.

BOZZANO, M.; CIMATTI, A.; ROVERI, M.; TCHALTSEV, A. A comprehensive approach to on-board autonomy verification and validation. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 22., 2011, Barcelona, Catalonia, Spain. **Proceedings...** Barcelona: AIAA, 2011.

BRAT, G.; JONSSON, A. Challenges in verification and validation of autonomous systems for space exploration. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 5., 2005, Montreal, Quebec, Canada. **Proceedings...** Montreal: IEEE, 2005.

BYLANDER, T. The computational complexity of propositional STRIPS planning. **Artificial Intelligence,** v. 69, n.1-2, p. 165–204, 1994.

BU, H.; ZHANG J.; LUO, Y. Constraint satisfaction and optimization for space station short-term mission planning based on an iterative conflict-repair method. **Journal of Engineering Optimization**, v. 48, 2016.

BUTLER, R.; PENNOTTI, M. The evolution of software and its impact on complex system design in robotic spacecraft embedded systems. **Procedia Computer Science**, v. 16, p. 747–756, 2013.

CAMERON, G. E.; MARSHALL, M. H. Exploring the practical limits of operations autonomy. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 5., 1998, Tóquio, Japão. **Proceedings...** Tóquio: AIAA, 1998.

CAMPOS, D. R. C. **Reparo de plano por refinamento reverso**. 2008. 88 p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, 2008.

CANTONI, L. F. A. **Avaliação do uso da linguagem PDDL no planejamento de missões para robôs aéreos**. 2008. 128 p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, 2010.

CARDOSO, R. C. **A decentralised online multi-agent planning framework for multi-agent systems**. 2018. 150 p. Tese (Doutorado em Ciência da Computação) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2018.

CASTANO, R.; ESTLIN, T.; ANDERSON, R.C.; GAINES, D.M.; CASTANO, A.; BORNSTEIN, B.; CHOUINARD, C.; JUDD, M. Oasis: Onboard autonomous science investigation system for opportunistic rover science. **Journal of Field Robotics**, v. 24, n. 5, p. 379–397, 2007.

CASTANO, A.; BIESIADECKI, J.; NEAKRASE, L.; WHELLEY, P.; GREELEY, R.; LEMMON, M.; CASTANO, R.; CHIEN, S. Automatic detection of dust devils and clouds at Mars. **Machine Vision and Applications**, v. 19, n. 5-6, p. 467-482, 2008.

CEBALLOS, A.; BENSALEM, S; CESTA, A.; DE SILVA, L.; FRATINI, S.; INGRAND, F.; OCON, J.; ORLANDINI, A.; RAJAN, K; RASCONI, R.; VAN WINNENDAEL, R. A goal-oriented autonomous controller for space exploration. In: SYMPOSIUM ON ADVANCED SPACE TECHNOLOGIES IN ROBOTICS AND AUTOMATION (ASTRA), 11., 2011, Noordwijk, Netherlands. **Proceedings...** Noordwijk: ESA, 2011.

CESTA, A.; CORTELLESSA, G.; FRATINI, S.; ODDI, A. Developing an end-to-end planning application from a timeline representation framework. In: CONFERENCE ON INNOVATIVE APPLICATIONS OF ARTIFICIAL INTELLIGENCE (IAAI-09). 21., 2009, Pasadena, California, USA. **Proceedings...** Pasadena: MIT, 2009.

CESTA, A.; OCON, J.; RASCONI, R.; MONTERO, A. M. S. Injecting on-board autonomy in a multi-agent system for space service providing. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL ENGINEERING AND OTHER APPLICATIONS OF APPLIED INTELLIGENT SYSTEMS (IEA/AIE), 23*.,* 2010*,* Berlin, Heidelberg. **Proceedings...** Berlin: SPRINGER, 2010.

CHIEN, S.; SMITH, B.; RABIDEAU, G.; MUSCETTOLA, N.; RAJAN, K. Automated planning and scheduling for goal based autonomous spacecraft. **IEEE Intelligent Systems,** v. 13, n. 5, p. 50-55, 1998.

CHIEN, S.; KNIGHT, R.; STECHERT, A.; SHERWOOD, R.; RABIDEAU, G. Using iterative repair to increase the responsiveness of planning and scheduling for autonomous spacecraft. In: WORKSHOP ON SCHEDULING AND PLANNING MEET REAL-TIME MONITORING IN A DYNAMIC AND UNCERTAIN WORLD (IJCAI), 1999, Estocolmo, Suécia. **Proceedings...** Estocolmo: Morgan Kaufman, 1999a.

CHIEN, S.; RABIDEAU, G.; WILLIS, J.; MANN, T. Automating planning and scheduling of shuttle payload operations. **Artificial Intelligence Journal Special Issue on Applications**, v. 114, n. 1-2, p. 239-255, 1999.

CHIEN, S.; SHERWOOD, R.; TRAN, D.; CASTANO, R.; CICHY, B.; Using autonomy flight software to improve science return on Earth Observing One (EO-1). **Journal of Aerospace Computing, Information and Communication**, v. 2, 2005.

CHIEN, S.; SILVERMAN, D.; DAVIES, A.; MANDL, D. Onboard science processing concepts for the HyspIRI mission. **IEEE Intelligent Systems**, v. 24, n. 6, p. 12-19, 2009.

CHIEN, S.; TRAN, D.; RABIDEAU, G.; SCHAFFER, S.; MANDL, D.; FRYE, S. Improving the operations of the Earth Observing One mission via automated mission planning. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 11., 2010, Huntsville, AL, USA. **Proceedings**... Huntsville: AIAA, 2010.

CHIEN, S.; JOHNSTON, M.; FRANK, J.; GIULIANO, M.; KAVELAARS, A.; LENZEN, C.; POLICELLA, N. A generalized timeline representation, services, and interface for automating space mission operations. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 11., 2012, Estocolmo, Suécia. **Proceedings...** Estocolmo: AIAA, 2012.

CHIEN, S.; MORRIS, R. Space applications of articial intelligence. **AI Magazine**, v. 35, n. 4, 2014.

CIVIDANES, F.; FERREIRA, M. G. V.; F. KUCINSKIS. On-board automated mission planning for spacecraft autonomy: a survey. **IEEE Latin America Transactions**, v. 17, n. 06, p. 884-896, 2019.

CODETTA-RAITERI, D.; PORTINALE, L. Dynamic bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 45, n. 1, p. 13-24, 2015.

COLES, A.; COLES, A.; FOX, M.; LONG, D. Forward-chaining partial-order planning. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS), 20., 2010, Toronto, Canada. **Proceedings...** Canada: AIAA, 2010.

D'ANGELO, G.; TIPALDI, M.; GLIELMO, L.; RAMPONE, S. Spacecraft autonomy modeled via Markov decision process and associative rule-based machine learning. In: INTERNATIONAL IEEE WORKSHOP ON METROLOGY FOR AEROSPACE (MetroAeroSpace), 2017, Pádua, Itália. **Proceedings**... Padua: IEEE, p. 324-329. 2017.

MURRAY, R. M.; DAY, J.; INGHAM, M. D.; REDER, L.; WILLIAMS, B. C. **Engineering resilient space systems final report**. 2013. Disponível em: <https://www.kiss.caltech.edu/final_reports/Systems_final_report.pdf>. Acesso em: 01 jun. 2020.

DECHTER, R.; MEIRI, I.; PEARL, J. Temporal constraint networks. **Artificial Intelligence**, v. 49, n. 1–3, p. 61–95, 1991.

DIAS, M.; LEMAI S.; MUSCETTOLA N. A real-time rover executive based on model-based reactive planning. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, ROBOTICS AND AUTOMATION IN SPACE (iSAIRAS), 7., 2003, Nara, Japão. **Proceedings...** Nara: JAXA, 2003.

DVORAK, D.; AMADOR A.; STARBIRD, T. Comparison of goal-based operations and command sequencing**.** In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 10., 2008, Heidelberg, Alemanha. **Proceedings...** Heidelberg: AIAA, 2008.

DVORAK, D.; MICHEL D.; INGHAM, J.; MORRIS, R. Goal-based operations: an overview. **Journal of Aerospace Computing, Information, and Communication**, v. 6, 2009.

DVORAK, F.; BIT-MONNOT, A.; INGRAND, F.; GHALLAB, M.; A flexible ANML actor and planner in robotics. In: WORSHOP AT INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS), 24., 2014, Portsmouth, NH, USA. **Proceedings...** Portsmouth: AAAI, 2014a. p. 40–49.

DVORAK, F.; BARTÁK, R.; BIT-MONNOT, A.; INGRAND, F.; GHALLAB, M. In: IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE (ICTAI), 2014, Limassol, Chipre. **Proceedings...** Limassol: IEEE, 2014. p. 115-121.

EDDY, D.; KOCHENDERFER, M. J. **Satellite image tasking under orbit prediction uncertainty**. 2019. Disponível em: <https://arxiv.org/pdf/1905.01417.pdf>. Acesso em: 01 jun. 2020.

EICKHOFF, J. **Onboard computers, onboard software and satellite operations**. 1. ed. Heidelberg, Alemanha: Springer-Verlag Heidelberg, 2012. 282 p. ISBN 978-3-642-25170-2.

EROL, K.; HENDLER, J.; D. S. NAU. Complexity results for HTN planning. **Annals of Mathematics and Artificial Intelligence**, v.18 p. 69–93, 1994.

ESTEVE, M.; KATOEN, J.; NGUYEN, V.; POSTMA, B.; YUSHTEIN, Y. Formal correctness, safety, dependability, and performance analysis of a satellite. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE). 34., 2012, Zurique, Suíça. **Proceedings**... Zurique: IEEE, 2012. p. 1022–1031.

ESTLIN, T.; VOLPE, R.; NESNAS, I.; MUTZ, D.; FISHER, F.; ENGELHARDT, B.; CHIEN, S. Decision-making in a robotic architecture for autonomy. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, ROBOTICS AND AUTOMATION IN SPACE (i-SAIRAS/01), Montreal, Canada. **Proceedings...** Montreal, 2001.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Space engineering -** space segment operability. Noordwijk, Holanda: ECSS, 2008. (ECSS-E-ST-70-11C).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS**)**. **Space engineering -** software engineering. Noordwijk, Holand: ECSS, 2009. (ECSS-E-ST-40C).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS**)**. **Space engineering** - spacecraft on-board control procedures**.** Noordwijk, Holanda: ECSS, 2010. (ECSS-E-ST-70-01C).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Ground systems and operations** – telemetry and telecommand packet utilization. Noordwijk, Holanda: ECSS, 2016. (ECSS-E-70-41C).

FINZI, A.; INGRAND F.; MUSCETTOLA, N. Model-based executive control through reactive planning for autonomous rovers. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS). 2004, Sendai, Japão. **Proceedings**... Sendai: IEEE, 2004. p. 879-884.

FIKES, R. E.; NILSSON, N. J. STRIPS: A new aproach to the application of the theorem proving to problem solving. **Artificial Intelligence**, v. 2, n. 3-5, p. 189-208, 1971.

FOX, M.; LONG, D. PDDL2.1: An extension to PDDL for expressing temporal planning domains. **Journal of Artificial Intelligence Research**, v. 20, p. 61–124, 2003.

FRANK, J.; JONSSON, A, K. Constraint-based attribute and interval planning. **Journal of Constraints Special Issue on Constraints and Planning**, v.8, n. 4, 2003.

FRANCIS, R.; ESTLIN, T.; GAINES, D.; BORNSTEIN, B.; SCHAFFER, S.; VERMA, V; ANDERSON, R.; BURL, M.; CHU, S.; CASTAÑO, R. AEGIS autonomous targeting for the Curiosity rover's ChemCam instrument. In: IEEE APPLIED IMAGERY PATTERN RECOGNITION WORKSHOP (AIPR), 2015, Washington, DC, USA. **Proceedings...** Washington, DC: IEEE, 2015. p. 1-5.

FRATINI, S.; PECORA, S.; CESTA, A. Unifying planning and scheduling as timelines in a component- based perspective. **Archives of Control Sciences**, v. 18, n. 2, p. 231-271, 2008.

FRATINI, S.; CESTA, A. The APSI Framework: A platform for timeline synthesis. In: WORKSHOPS ON PLANNING AND SCHEDULING WITH TIMELINES (ICAPS), 22., 2012, Atibaia, São Paulo, Brasil. **Proceedings...** Atibaia: AAAI, 2012, p. 8-15.

FRATINI, S.; NICOLA, P.; DONATI, A. Pattern-Based Modeling for Timeline Planning in Space Domains. **Acta Futura,** v. 9, p. 73-82, 2014.

FROST, C. **Challenges and opportunities for autonomous systems in space**. 2010. Disponível em: < https://www.nap.edu/read/13043/chapter/17>. Acesso em: 22 fev. 2017.

FUKUNAGA, A.; RABIDEAU, G.; CHIEN, S.; YAN, D. Towards an application framework for automated planning and scheduling. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE ROBOTICS AND AUTOMATION IN SPACE (I-SAIRAS), 4., 1997, Tóquio, Japão. **Proceedings...** Tóquio: NASDA, 1997.

GAO, Y.; POLICELLA, N.; KIRCHNER, F. Computational intelligence for space systems and operations. **IEEE Computational Intelligence Magazine**, v. 8, n.4, p. 10-63, 2013.

GARRIDO A.; BARBER F. Integrating planning and scheduling. **Applied Artificial Intelligence**, v. 15, n.5, p. 471–491, 2001.

GEIER, T.; BERCHER, P. On the decidability of HTN planning with task insertion. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI11), 22., 2011, Barcelona, Catalonia, Spain. **Proceedings...** Barcelona: AAAI, 2011, v. 3, p. 1955-1961.

GEORGIEVSKI, I.; AIELLO, M. HTN planning: Overview, comparison, and beyond. **Artificial Intelligence**, v. 222, p. 124-156, 2015.

GEREVINI, A.; KUTER, U.; NAU, D.; SAETTI, A.; WAISBROT, N. Combining domain in dependent planning and HTN planning: The Duet planner. In: EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (ECAI), 18., 2008, Patras, Grécia. **Proceedings...** Cambridge: ACM, 2008, p. 573-577.

GHALLAB M.; LARUELLE, H. Representation and control in IxTeT, a temporal planner. In INTERNATIONAL CONFERENCE ON AI PLANNING SYSTEMS (AIPS), 2, 1994, Chicago, Illinois, USA. **Proceedings...** Chicago: AAAI, 1994. p. 61–67.

GHALLAB, M.; NAU, D.; TRAVERSO, P.; editors. **Automated planning - theory and practice**. Elsevier, San Francisco, 2004. 635 p. ISBN 1-55860-856-7.

GHALLAB, M.; NAU, D.; TRAVERSO, P. The actor's view of automated planning and acting: A position paper. **Artificial Intelligence**, v. 208, p. 1-17, 2014.

GHALLAB, M.; NAU, D.; TRAVERSO, P.; editors. **Automated planning and acting**. Cambridge University Press, Inglaterra, 2016. 368 p. ISBN-10 (1107037271).

GOETZ, K.; HUBER, F.; SCHOENERMARK, M. VIMOS - autonomous image analysis on board of BIROS. In: SYMPOSIUM ON AUTOMATIC CONTROL IN AEROSPACE, 19., 2013, Wurzburg, Alemanha. **Proceedings…** Wurzburg: IFAC, v. 46, 2013.

GREY, S. **Distributed agents for autonomous spacecraft**. 2012. 241 p. Tese (Doutorado em Filosofia, Escola de Engenharia Faculdade de Ciências e Engenharia) - Universidade de Glasgow, Reino Unido, 2012.

HE, Y.; WANG, Y.; CHEN, Y.; XING, L. Auto mission planning system design for imaging satellites and its applications in environmental field. **Polish Maritime Research**, v. 23, 2016.

HOLLER, D.; BERCHER, P.; BEHNKE, G.; BIUNDO, S. HTN plan repair using unmodified planning systems. In: WORKSHOP ON HIERARCHICAL PLANNING IN INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS), 28., 2018, Delft, Holanda. **Proceedings…** Delft: AAAI, 2018. p. 26–30.

INDRA, A.; AGRAWAL, V. K.; SARMA, V. V. S. Stratified agent architecture for on-board mission planning and execution for an autonomous spacecraft. In: IEEE REGION 10 CONFERENCE (TENCON), 2008, Hyderabad, India. **Proceedings...** Hyderabad: IEEE, 2008.

INGRAND, F.; GHALLAB, M. Deliberation for autonomous robots: A Survey. **Artificial Intelligence**, v. 247, p. 10-44, 2017.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **China Brazil earth resources satellite CBERS 3 & 4**. São José dos Campos-SP, Brasil, 2013. HANDBOOK OF THE WFI-FM3 SUBSYSTEM (Equatorial Sistemas S.A/RBS-HBK-W010/00).

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Página da missão Amazonia-1 do INPE**. 2020. Disponível em: <http://www3.inpe.br/amazonia-1/>. Acesso em: 07 fev. 2020.

JIANG X.; XU R. A constraint-programmed planner for deep space exploration problems with table constraints. **IEEE Access**, v. 5, p. 17258-17270, 2017.

JIANG, X.; CUI, P.; XU, R.; GAO A.; ZHU, S. An action guided constraint satisfaction technique for planning problem. In: INTERNATIONAL CONFERENCE ON COGNITIVE INFORMATICS & COGNITIVE COMPUTING (ICCI*CC), 15., 2016, Palo Alto, CA, USA. **Proceedings...** Palo: IEEE, 2016. p. 167-173.

JIMÉNEZ, S.; DE LA ROSA, T.; FERNÁNDEZ, S.; FERNÁNDEZ, F.; BORRAJO, D. A review of machine learning for automated planning. **Knowledge Engineering Review**, v. 27, n. 4, p. 433–467, 2012.

JONSSON, A.; MORRIS R.; PEDERSEN, L. Autonomy in space: current capabilities and future challenges. **AI Magazine**, v. 27, n. 4, 2007.

KAELBLING, L. P.; LITTMAN, M. L.; CASSANDRA, A. R. Planning and acting in partially observable stochastic domains, **Artificial Intelligence**, v. 101, n.1-2, p. 99-134, 1998.

KNIGHT, R.; RABIDEAU, G.; CHIEN, S.; ENGELHARDT, B.; SHERWOOD, R. CASPER: space exploration through continuous planning. **IEEE Intelligent Systems**, v. 16, n. 4, p. 70-75, 2001.

KNIGHT, R.; CHOUINARD, C.; JONES, G.; TRAN, D. Leveraging multiple artificial intelligence techniques to improve the responsiveness in operations planning: Aspen for orbital express. **AI Magazine**, v. 35, n. 4, p. 26–36, 2014.

KOLCIO, K.; BREGER, L.; ZETOCHA, P. Model-based fault management for spacecraft autonomy. In: IEEE AEROSPACE CONFERENCE, 2014, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2014. p. 1–14.

KUCINSKIS, F. N. **Alocação dinâmica de recursos computacionais para experimentos científicos com replanejamento automatizado a bordo de satélites**. 2007. 165 p. (INPE-14798-TDI/1241). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São Jose dos Campos, 2007. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m17@80/2007/04.23.11.42>. Acesso em: 22 fev. 2017.

KUCINSKIS, F. N.; FERREIRA, M. G. V. Taking the ECSS autonomy concepts one step further. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS 2010), 11., 2010, Huntsville, USA. **Proceedings...** 2010. DVD. Disponível em: <http://www.spaceops2010.org/>. Acesso em: 07 dez. 2016.

KUCINSKIS, F. N. Validation of reasoning and decision-making of a satellite autonomous on-board software. In: LATIN-AMERICAN SYMPOSYUM ON DEPENDABLE COMPUTING, (LADC), 5., 2011, São José dos Campos, Brasil. **Proceedings…** São José dos Campos: IEEE, 2011.

KUCINSKIS, N. F. **Uma arquitetura de software embarcado no segmento espacial para habilitar a operação de missões baseada em objetivos**. 2012. 192 p. Tese (Doutorado em Engenharia e Gerenciamento de Sistemas Espaciais) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012. Disponível em: <http://urlib.net/8JMKD3MGP7W/3BEQBSP>. Acesso em: 04 abr. 2017.

KUCINSKIS, N. F.; FERREIRA, M. G. V. On-board satellite software architecture for the goal-based Brazilian mission operations. **IEEE Aerospace and Electronic Systems Magazine**, v. 28, n.8, p. 32-45, 2013.

KULU, E. **Nanosatellite and cubesat database**. 2019. Disponível em: <http://www.nanosats.eu/>. Acesso em: 08 dez. 2019.

KUTER, U.; NAU, D. Forward-chaining planning in nondeterministic domains. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE, 19., 2004, San jose, California, USA. **Proceedings...** SAN JOSE: AAAI, 2004, p. 513–518.

LEMAI, S. **IXTET-EXEC: planning, plan repair and execution control with time and resource management**. 2004. 204 p. Tese (Doutorado em Ciência da Computação e Telecomunicações - Laboratório de Análise e Arquitetura de Sistemas) - Instituto Politécnico Nacional de Toulouse, Toulouse, 2004.

LEMAI, S.; OLIVE, X.; CHARMEAU, M. C. Decisional architecture for autonomous space systems. In: ESA WORKSHOP ON ADVANCED SPACE TECHNOLOGIES FOR ROBOTICS AND AUTOMATION (ASTRA), 2006, Noordwijk, Holanda. **Proceedings...** Noordwijk: European Space Agency, 2006.

LENZEN, C.; WOERLE, M. T.; GÖTTFERT, T.; MROWKA F.; WICKLER, M. Onboard planning and scheduling autonomy within the scope of the FireBird mission. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 13., 2014, Pasadena, CA, USA. **Proceedings...** Pasadena: AIAA, 2014.

LONG, J.; LI, C.; ZHU, L.; CHEN, S.; LIU, J. An efficient task autonomous planning method for small satellites. **EURASIP Journal on Image and Video Processing**, v. 1, n. 29, 2018.

FOX, M.; LONG, D. The 3rd international planning competition: results and analysis. **Journal of Artificial Intelligence Research,** v. 20, p.1-59, 2003.

MAYER, M.C.; ORLANDINI, A.; UMBRICO, A. Planning and execution with flexible timelines: a formal account. **Acta Informatica**, v.53 n. 6–8, p. 649-680, 2016.

MCGANN, C.; PY, F.; RAJAN, K.; OLAYA, A. Integrated planning and execution for robotic exploration. In: WORKSHOP ON HYBRID CONTROL OF AUTONOMOUS SYSTEM AT INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI-09), 2009, Pasadena, California, USA. **Proceedings...** Pasadena: AAAI, 2009.

MINTON, S.; BRESINA, J.; DRUMMOND, M. Total-order and partial-order planning: a comparative analysis. **Journal of Artificial Intelligence Research**, v. 2, p. 227–262, 1994.

MUÑOZ, P.; CESTA, A.; ORLANDINI, A.; R-MORENO, M. D. First steps on an on-ground autonomy test environment. In: INTERNATIONAL CONFERENCE ON SPACE MISSION CHALLENGES FOR INFORMATION TECHNOLOGY (SMC-IT), 2014, Laurel, MD, USA. **Proceedings...** Laruel: IEEE, 2014. p. 30-37.

MUÑOZ, P.; R-MORENO, D.; BARRERO, D. F.; ROPERO, F. MoBAr: A hierarchical action-oriented autonomous control architecture. **Journal of Intelligent and Robotic Systems**, v. 94, p. 1-16, 2018.

MUSCETTOLA, N. HSTS: integrated planning and scheduling**.** In: FOX, M.; ZWEBEN, M. **Intelligent scheduling**, Morgan Kaufman, São Francisco, CA, 1994. p. 169-212.

MUSCETTOLA, N.; NAYAK, P.; PELL, B.; WILLIAMS, B. C. Remote Agent: to boldly go where no AI system has gone before. **Artificial Intelligence**, v. 103, n. 1, p. 5-47, 1998.

MUSCETTOLA, N.; DORAIS, G. A.; FRY, C.; LEVINSON, R.; PLAUNT, C. IDEA: planning at the core of autonomous reactive agents. In: INTERNATIONAL CONFERENCE ON AI

PLANNING AND SCHEDULING (AIPS), 6., 2002, Toulouse, França. **Proceedings...** Toulouse: AIAA, 2002. p. 58-60.

NARDONE, V; SANTONE, A; TIPALDI, M; GLIELMO, L. Probabilistic model checking applied to autonomous spacecraft reconfiguration. In: IEEE METROLOGY FOR AEROSPACE (MetroAeroSpace), 2016, Florence, Itália. **Proceedings...** Florence: IEEE, 2016. p. 556–560.

NARDONE V.; SANTONE, A.; TIPALDI, M.; LIUZZA, D.; GLIELMO, L. Model checking techniques applied to satellite operational mode management. **IEEE Systems Journal**, v. 13, n.1, p. 1-12, 2019.

NAU, D. S.; AU, T. C.; ILGHAMI, O.; KUTER, U.; MURDOCK, J. W.; WU, D.; YAMAN, F. SHOP2: an HTN planning system. **Journal of Artificial Intelligence Research**, v. 20, p. 379–404, 2003.

NAU, D. S. Current trends in automated planning. **AI Magazine**, v. 28, n. 4, p. 43–58, 2007.

NESNAS, I. A. D.; WRIGHT, A.; BAJRACHARYA, M.; SIMMONS, R.; ESTLIN, T.; KIM, W. S. CLARATy: an architecture for reusable robotic software. In: SPIE AEROSENSE CONFERENCE, 2003, Orlando, FL, USA. **Proceedings...** Orlando: SPIE, 2003.

NOGUEIRA, T.; FRATINI, S.; SCHILLING, K. Autonomously controlling flexible timelines: From domain-independent planning to robust execution. In: IEEE AEROSPACE CONFERENCE, 2017, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2017. p. 1-15.

OLIVE, X. FDI(R) for satellites: How to deal with high availability and robustness in the space domain? **International Journal of Applied Mathematics and Computer Science**, v. 22, n. 1, p. 99–107, 2012.

PAN, L. **Autonomous rock science analysis system for planetary exploration**. 2016. Tese (Doutorado em Filosofia - Departamento de Ciência da Computação) - Universidade Aberystwyth, Inglaterra, 2016.

PEDNAULT, E. P. D. ADL: exploring the middle ground between strips and the situation calculus. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR89), 1., 1989, San Mateo, CA, USA. **Proceedings...** San Mateo: Morgan Kaufmann, 1989. p. 324-332.

PENBERTHY, J. S.; WELD, D. S. UCPOP: A sound, complete, partial order planner for ADL In: CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING (KR92), 3., 1992, Cambridge, Massachusetts, USA. **Proceedings...** Cambridge: ACM, 1992. p. 103–114.

PENG, S.; CHEN, H.; DU, C.; LI, J.; JING, N. Onboard observation task planning for an autonomous earth observation satellite using long short-term memory. **IEEE Access,** v. 6, p. 65118-65119, 2018.

PEREIRA, L. S. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. 2007. 148 p. Tese (Doutorado em Ciência da Computação) - Instituto de Matemática e Estatística da Universidade de São Paulo, 2007.

PUTERMAN, M. L. **Markov decision processes: discrete stochastic dynamic programming**. New York, NY, USA: Wiley, 1994. 672 p. ISBN 9780471619772.

QI, C.; WANG, D.; MUNOZ-AVILA, H.; ZHAO, P.; WANG, H. Hierarchical task network planning with resources and temporal constraints. **Knowledge-Based Systems**, v. 15, n.1, p. 17-32, 2017.

RAJAN, K.; BERNARD, D.; DORAIS, G.; GAMBLE, E.; KANEFSKY, B.; KURIEN, J.; MILLAR,W.; MUSCETTOLA, N.; NAYAK, P.; ROUQUETTE,N.; SMITH, B.; TAYLOR,W.; TUNG, Y. Remote Agent: an autonomous control system for the new millennium. In: EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (ECAI), 14., 2000, Berlin, Alemanha. **Proceedings...** Berlin, 2000.

ROUFF, C. Autonomous and agent technology in future space missions. In: IEEE AEROSPACE CONFERENCE, 2003, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2003.

RUI, X.; PING-YUAN C.; XIAO-FEI, X. Design for autonomous mission planning system. **Aircraft Engineer and Aerospace Technology: An International Journal**, v. 75, n. 4, p. 365–71, 2003.

RUI, X.; PINGYUAN, C.; XU, X. Realization of multi-agent planning system for autonomous spacecraft. **Advances in Engineering Software**, v. 36, n. 4, p. 266-272, 2005.

RUSSELL, S.; NORVIG, P. **Inteligência artificial**. Tradução da 2. ed. São Paulo: Editora Campus, 2004. 1040 p. ISBN 8535211772.

SAPENA, O.; ONAINDIA, E.; TORREÑO, A. FLAP: applying least-commitment in forward-chaining planning. **AI Communications**, v. 28, n.1, p. 5–20, 2015.

SHE, Y.; LI, S.; ZHAO, Y. Onboard mission planning for agile satellite using modified mixed-integer linear programming. **Aerospace Science and Technology**, v. 72, p. 204-216, 2018.

SHERWOOD, R.; CHIEN, S.; TRAN, D.; CICHY, B.; CASTAÑO, R.; DAVIES, A.; RABIDEAU, G. Operating the autonomous sciencecraft experiment. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 8., 2004, Montréal, Canadá. **Proceedings...** Montreal: AIAA, 2004.

SIMMONS, R. G.; YOUNES, H. L. VHPOP: versatile heuristic partial order planner. **Journal on Artificial Intelligence Research,** v. 20, p. 405–430, 2003.

SMITH, D. E.; FRANK, J.; JONSSON, A. K. Bridging the gap between planning and scheduling**. Knowledge Engineering Review**, v. 1, n. 15, p. 47-83, 2000.

SMITH, M. H.; HOLZMANN, G. J.; CUCULLU, G. C.; SMITH, B. D. Model checking autonomous planners: even the best laid plans must be verified. In: IEEE AEROSPACE CONFERENCE, 2005, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2005. p. 1–11.

SMITH, D.; FRANK, J.; CUSHING, W. The ANML language. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS), 2008, Sydney, Australia. **Proceedings...** Sydney: AAAI, 2008.

TATE, A. Generating project networks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI-77), 5., 1977, Cambridge, Massachusetts, USA. **Proceedings...** Cambridge, 1977. p. 888–893,

TESTON, F.; CREASEY, R.; BERMYN, J.; MELLAB, K. PROBA: ESA's autonomy and technology demonstration mission. In: INTERNATIONAL ASTRONAUTICAL CONGRESS (IAC), 48., 1997, Turin, Italy. **Proceedings**...Turin: AIAA, 1997.

THUMMALA, B. **Spacecraft autonomy**. 2003. 94 p. Dissertação (Mestrado em Astronáutica e Engenharia Espacial) - Universidade de Cranfield, Inglaterra, 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.2777&rep=rep1&type=pdf>. Acesso em: 01 mar. 2020.

TIPALDI, M.; GLIELMO, L. A Markovian based approach for autonomous space systems. In: IEEE INTERNATIONAL WORKSHOP ON METROLOGY FOR AEROSPACE, 2., 2015, Benevento, Itália. **Proceedings...** Benevento: IEEE, 2015. p. 426-430.

TIPALDI, M.; BRUENJES, B. Survey on fault detection, isolation, and recovery strategies in the space domain. **Journal of Aerospace Information Systems**, v. 12, n. 2, p. 235–256, 2015.

TIPALDI, M.; GLIELMO, L. A survey on model-based mission planning and execution for autonomous spacecraft. **IEEE Systems Journal**, v. 12, n. 4, p. 3893-3905, 2018.

TIPALDI, M.; LEGENDRE, C.; KOOPMANN, O.; FERRAGUTO, M.; WENKER, R.; D'ANGELO, G. Development strategies for the satellite flight software on-board Meteosat Third Generation. **Acta Astronautica**, v. 145, p. 482-491, 2018.

TOMINAGA, J. **Simulador de satélites para verificação de planos de operações em voo**. 2010. 174 p. (sid.inpe.br/mtc-m19@80/2010/05.24.18.55-TDI). Dissertação

(Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2010. Disponível em: <http://urlib.net/8JMKD3MGP7W/37HL3J8>. Acesso em: 22 fev. 2016.

TRAN, D.; CHIEN, S.; RABIDEAU, G.; CICHY, B. Flight software, issues in onboard automated planning - lessons learned on EO-1. In: INTERNATIONAL WORKSHOP ON PLANNING AND SCHEDULING FOR SPACE (IWPSS). 2004, Darmstadt, Alemanha. **Proceedings...** Darmstadt: STCI, 2004.

TRUSZKOWSKI, W.; HALLOCK, H.; ROUFF, C.; KARLIN, J.; RASH, J.; HINCHEY, M.; STERRITT, R. **Autonomous and autonomic systems: with applications to NASA intelligent spacecraft operations and exploration systems**. 1. ed. New York, NY, USA: Springer-Verlag New York, Inc. 2009. 306 p. ISBN-13 (978-1846282324).

VAN DER HA, J. Trends in cost-effective mission operations. **Acta Astronautica**, v. 52, n. 2, p. 337–342, 2003.

VASSEV, E.; HINCHEY, M. On requirements for space missions. In: INTERNATIONAL SYMPOSIUM ON OBJECT/COMPONENT/SERVICE-ORIENTED REAL-TIME DISTRIBUTED COMPUTING (ISORC), 16., 2013, Paderborn, Alemanha. **Proceedings...** Paderborn: IEEE, 2013. p. 1-10.

VERMA, V.; JONSSON, A.; SIMMONS, R.; ESTLIN, T.; LEVINSON, R. Survey of command execution systems for NASA robots and spacecrafts. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS), 2005, Monterey, CA, USA. **Proceedings...** Monterey: AAAI, 2005. p. 40–49.

VICTORIA, J. M. D.; POLICELLA, N.; YANG, G.; STRYK, O. V. Design concepts for a new temporal planning paradigm. In: WORKSHOP ON PLANNING AND SCHEDULING WITH TIMELINES ON INTERNATIONAL COMPETITION ON KNOWLEDGE ENGINEERING FOR PLANNING AND SCHEDULING (ICAPS), 4., 2012, Atibaia, SP, Brasil. **Proceedings...** Atibaia: AAAI, 2012.

VILAIN, M. B.; KAUTZ, H. Constraint propagation algorithms for temporal reasoning. In: AIAA CONFERENCE, 5., 1986, Philadelphia, PA. **Proceedings…** Philadelphia: AIAA, 1986. p. 377-382.

VOLPE, R.; NESNAS, I.; ESTLIN, T.; MUTZ, D.; PETRAS, R.; DAS, H. The CLARAty architecture for robotic autonomy. In: IEEE AEROSPACE CONFERENCE, 2001, Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2001.

WANDER, A.; FÖRSTNER, R. Innovative fault detection, isolation and recovery on-board spacecraft: study and implementation using cognitive automation. In: CONFERENCE ON CONTROL AND FAULT-TOLERANT SYSTEMS (SysTol) 2., 2013, Nice, França. **Proceedings...** Nice: IEEE, 2013. p. 336-341.

WANG, W.; WANG, X.; HAN, B.; ZHANG, Y.; LEI, Y. Design of OBDH subsystem for remote sensing satellite based on onboard route architecture. In: INTERNATIONAL CONFERENCE ON MECHANICAL ELECTRONIC AND INFORMATION TECHNOLOGY ENGINEERING (ICMITE), 3., 2017, Chengdu, China. **Proceedings...** Chengdu: TIB, 2017.

WERTZ, J. R.; REINERT, R. P. Mission characterization. In: LARSON, W. J.; WERTZ, J. R. (eds.). **Space mission analysis and design**. 3. ed. Torrance, CA, USA: Microcosm, Inc. and Kluwer Academic Publishers, 1999. 969 p. ISBN-13 (978-1881883104).

WILLE, B.; WORLE M, T.; LEZEN C. VAMOS – verification of autonomous mission planning on-board a spacecraft. In: SYMPOSIUM ON AUTOMATIC CONTROL IN AEROSPACE, 19., 2013, Wurzburg, Alemanha. **Proceedings…** Wurzburg: IFAC, 2013, v. 46, p. 1-576.

WILKINS, D. E.; MYERS, K. L.; LOWRANCE, J. D.; WESLEY, L. P. Planning and reacting in uncertain and dynamic environments. **Journal of Experimental Theoretical Artificial Intelligence**, v. 7, n. 1, p. 121–152, 1995.

WOJTKOWIAK, H.; BALAGURIN, O.; FELLINGER, G.; KAYAL, H. ASAP: autonomy through on-board planning. In: INTERNATIONAL CONFERENCE ON RECENT ADVANCES IN SPACE TECHNOLOGIES (RAST), 6., 2013, Istanbul, Turquia. **Proceedings...** Istanbul: IEEE, 2013, p. 377-381.

WOODS, M. J.; BALDWIN, L.; WILSON, G.; HALL, S.; PIDGEON, A.; LONG, D.; FOX, M.; AYLETT, R.; VITULI, R. MMOPS: assessing the impact of on-board autonomy for planetary exploration missions. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 9., 2006, Roma, Italia. **Proceedings...** Roma: AIAA, 2006.

ZHANG, Z.; HE, R.; TAN, Y. An HTN-based mission planning model and algorithm for autonomous spacecraft. In: INTERNATIONAL SYMPOSIUM ON SYSTEMS AND CONTROL IN AEROSPACE AND ASTRONAUTICS (ISSCAA), 1., 2006, Harbin, China. **Proceedings...** Harbin: IEEE, 2006, p. 264-267.

ZHELTOUKHOV, A. A.; STANKEVICH, L. A. A survey of control architectures for autonomous mobile robots. In: IEEE CONFERENCE OF RUSSIAN YOUNG RESEARCHERS IN ELECTRICAL AND ELECTRONIC ENGINEERING (EIConRus), 2017, St. Petersburg, Russia. **Proceedings...** St. Petersburg: IEEE, 2017. p. 1094-1099.

ZHUO, H.; MUÑOZ-AVILA, H.; YANG, Q. Learning hierarchical task network domains from partially observed plan traces. **Artificial Intelligence**, v. 212, p. 134-157, 2014.

APÊNDICE A – O DOMÍNIO DO ESTUDO DE CASO EM ISISml

Este Apêndice traz o domínio estático do estudo de caso. Ele é formado pelo domínio hierárquico e estrutural, ambos definidos em XML pela linguagem estendida ISISml.

A.1 O domínio hierárquico em ISISml

Abaixo, mostra-se o domínio hierárquico completo para o modelo Amazonia-1 do estudo de caso na linguagem estendida.

Domínio Hierárquico

```
<?xml version="1.0"?>
<hierarchical_description>
  <tasks>
    <compoundtask name = "StartAcquisitionRealtime" id= "20" inference_moment = "true" methods = "ordered">
      <method name = "StartAcquisitionRealtime_method">
        <precondition>
          <precond constraints = "PCDU.OrbitPeriod == sunlight"/>
          <precond constraints = "AOCS.mode == mission"/>
          <precond constraints = "DT.CommunicatingWithGround == withcommuication"/>
          <precond constraints = "OBDH.satelliteMode == routine"/>
          <precond constraints = "WFI.imageCloudy == cloudless"/>
        </precondition>
        <subtasks order  = "ordered">
          <task  name  = "PrepareConfigCamera"          id = "22"  type = "compound"/>
          <task  name  = "SetDataTransmitterRealTime"   id = "23"  type = "compound"/>
        </subtasks>
      </method>
    </compoundtask>
    <compoundtask name = "StopAcquisitionRealTime" id  = "21" inference_moment = "true" methods = "ordered">
      <method  name = "StopAcquisitionRealTime_method">
        <subtasks order  = "ordered">
          <task  name  = "TurnOffCamera"        id = "24"   type = "compound"/>
          <task  name  = "SwitchOffRtu"         id = "12"   type = "primitive"/>
          <task  name  = "SwitchOffDdr"         id = "10"   type = "primitive"/>
          <task  name  = "TurnOffDtRealTime"    id = "25"   type = "compound"/>
        </subtasks>
      </method>
    </compoundtask>
    <compoundtask name = "PrepareConfigCamera" id = "22" inference_moment = "false" methods= "ordered">
      <method  name = "PrepareConfigCamera_method">
        <subtasks order  = "ordered">
          <task  name  = "SwitchOnRtu"          id = "11"   type = "primitive"/>
          <task  name  = "SwitchOnCamera"       id = "7"    type = "primitive"/>
          <task  name  = "StartImagingCamera"   id = "1"    type = "primitive"/>
        </subtasks>
      </method>
    </compoundtask>
    <compoundtask name = "SetDataTransmitterRealTime" id = "23" inference_moment = "false" methods= "ordered">
      <method name = "SetDataTransmitterRealTime_method">
        <subtasks order = "ordered">
          <task  name  = "SwitchOnDdr"                  id = "9"   type = "primitive"/>
          <task  name  = "SwitchOnDt"                   id = "13" type = "primitive"/>
          <task  name  = "SetDtWfiChannelTurnOn"        id = "15"  type = "primitive"/>
        </subtasks>
      </method>
    </compoundtask>
    <compoundtask name = "TurnOffCamera" id = "24" inference_moment = "false" methods= "ordered">
      <method name = "TurnOffCamera_method">
        <subtasks  order = "ordered">
          <task  name  = "StandbyCamera"     id = "2"   type = "primitive"/>
          <task  name  = "SwitchOffCamera"   id = "8"   type = "primitive"/>
        </subtasks>
      </method>
    </compoundtask>
    <compoundtask name = "TurnOffDtRealTime" id = "25" inference_moment = "false" methods= "ordered">
      <method name = "TurnOffDtRealTime_method">
        <subtasks  order = "ordered">
          <task  name  = "SetDtWfiChannelTurnOff"  id = "16"   type = "primitive"/>
          <task  name  = "SwtichOffDt"             id = "14"   type = "primitive"/>
        </subtasks>
```

```
    </method>
  </compoundtask>
  <compoundtask name = "StartAcquisitionRecording" id = "26" inference_moment = "true" methods = "ordered">
    <method name = "StartAcquisitionRecording_method">
      <precondition>
        <precond constraints = "PCDU.OrbitPeriod == sunlight"/>
        <precond constraints = "AOCS.mode == mission"/>
        <precond constraints = "OBDH.satelliteMode == routine"/>
        <precond constraints = "WFI.imageCloudy == cloudless"/>
      </precondition>
      <subtasks order = "ordered">
        <task  name  = "PrepareConfigCamera"      id = "22"   type = "compound"/>
        <task  name  = "ConfigureRecordDdr"       id = "28"   type = "compound"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "StopAcquisitionRecording" id= "27" inference_moment = "true" methods = "ordered">
    <method name = "StopAcquisitionRecording_method">
      <subtasks order = "ordered">
        <task  name  = "TurnOffDdr"       id = "29"  type = "compound"/>
        <task  name  = "TurnOffCamera"  id = "24"  type = "compound"/>
        <task  name  = "SwtichOffRtu"     id = "12"  type = "primitive"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "ConfigureRecordDdr" id= "28" inference_moment = "false" methods = "ordered">
    <method name = "ConfigureRecordDdr_method">
      <subtasks order = "ordered">
        <task  name  = "SwitchOnDdr"              id = "9"    type = "primitive"/>
        <task  name  = "StartRecordDdr"           id = "3"    type = "primitive"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "TurnOffDdr" id= "29" inference_moment = "false" methods = "ordered">
    <method name = "TurnOffDdr_method">
      <subtasks order = "ordered">
        <task  name  = "StopRecordDdr"            id = "4"  type = "primitive"/>
        <task  name  = "SwitchOffDdr"             id = "10" type = "primitive"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "StartDownloadStoredImage" id = "30" inference_moment = "true" methods = "ordered">
    <method name = "StartDownloadStoredImage_method">
      <precondition>
        <precond constraints = "DT.CommunicatingWithGround == withcommuication"/>
        <precond constraints = "OBDH.satelliteMode == routine"/>
      </precondition>
      <subtasks order = "ordered">
        <task  name  = "SwitchOnRtu"              id = "11"   type = "primitive"/>
        <task  name  = "SwitchOnCamera"           id = "7"    type = "primitive"/>
        <task  name  = "ConfigureDdrPlayback"     id = "32"   type = "compound"/>
        <task  name  = "TurnOnDt"                 id = "38"   type = "compound"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "StopDownloadStoredImage" id= "31" inference_moment = "true" methods = "ordered">
    <method name = "StopDownloadStoredImage_method">
      <subtasks  order = "ordered"  >
        <task  name  = "SetDtWfiChannelTurnOff"  id = "16"    type = "primitive"/>
        <task  name  = "StopDdrPlayback"           id = "33"    type = "compound"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "ConfigureDdrPlayback" id= "32" inference_moment = "false" methods = "ordered">
    <method name = "ConfigureDdrPlayback_method">
      <subtasks order = "ordered">
        <task  name  = "SwitchOnDdr"              id = "9"    type = "primitive"/>
        <task  name  = "StartPlaybackDdr"         id = "5"    type = "primitive"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "StopDdrPlayback" id= "33" inference_moment = "false" methods = "ordered">
    <method name = "StopDdrPlayback_method"                                    >
      <subtasks order = "ordered">
        <task  name  = "StopPlaybackDdr"          id = "6"    type = "primitive"/>
        <task  name  = "EraseFileDdr"             id = "19"   type = "primitive"/>
        <task  name  = "SwitchOffDdr"             id = "10"   type = "primitive"/>
      </subtasks>
    </method>
  </compoundtask>
  <compoundtask name = "StartCameraCalibrationRecord" id= "34" inference_moment = "true" methods = "ordered">
    <method name = "StartCameraCalibrationRecord_method">
      <precondition>
        <precond constraints = "AOCS.mode == mission"/>
```

```
            <precond constraints = "OBDH.satelliteMode == routine"/>
          </precondition>
          <subtasks  order = "ordered">
            <task  name  = "PrepareConfigCamera"     id = "22"  type = "compound" />
            <task  name  = "StartCalibrationCamera"  id = "17"  type = "primitive"/>
            <task  name  = "ConfigureRecordDdr"      id = "28"  type = "compound"/>
          </subtasks>
        </method>
      </compoundtask>
      <compoundtask name = "StopCameraCalibrationRecord"       id = "35" inference_moment = "true" methods = "ordered">
        <method name = "StopCameraCalibrationRecord_method">
          <subtasks order = "ordered">
            <task  name  = "TurnOffDdr"                id = "29"  type = "compound"/>
            <task  name  = "StopCalibrationCamera"     id = "18"  type = "primitive"/>
            <task  name  = "TurnOffCamera"             id = "24"  type = "compound"/>
            <task  name  = "SwitchOffRtu"              id = "12"  type = "primitive"/>
          </subtasks>
        </method>
      </compoundtask>
      <compoundtask name = "StartCalibrationRealtime" id = "36" inference_moment = "true" methods = "ordered">
        <method name = "StartCalibrationRealtime_method">
          <precondition>
            <precond constraints = "DT.CommunicatingWithGround == withcommuication"/>
            <precond constraints = "OBDH.satelliteMode == routine"/>
            <precond constraints = "AOCS.mode == mission"/>
          </precondition>
          <subtasks order  = "ordered">
            <task  name  = "PrepareConfigCamera"          id = "22"  type = "compound" />
            <task  name  = "StartCalibrationCamera"       id = "17"  type = "primitive" />
            <task  name  = "SetDataTransmitterRealTime"   id = "23"  type = "compound"/>
          </subtasks>
        </method>
      </compoundtask>
      <compoundtask name = "StopCalibrationRealtime" id = "37" inference_moment = "true" methods = "ordered">
        <method name = "StopCalibrationRealtime_method"          >
          <subtasks order = "ordered">
            <task  name  = "StopCalibrationCamera"    id = "18"  type = "primitive"/>
            <task  name  = "TurnOffCamera"            id = "24"  type = "compound"/>
            <task  name  = "SwtichOffRtu"             id = "12"  type = "primitive"/>
            <task  name  = "SwitchOffDdr"             id = "10"  type = "primitive"/>
            <task  name  = "TurnOffDtRealTime"        id = "25"  type = "compound"/>
          </subtasks>
        </method>
      </compoundtask>
      <compoundtask name = "TurnOnDt" id = "38" inference_moment = "false" methods = "ordered">
        <method name = "TurnOnDt_method">
          <subtasks order = "ordered">
            <task  name  = "SwitchOnDt"               id = "13"   type = "primitive"/>
            <task  name  = "DtWfiChannelTurnOn"       id = "15"   type = "primitive"/>
          </subtasks>
        </method>
      </compoundtask>
      <compoundtask name = "StartRealTimeRecording" id = "39" inference_moment = "true" methods = "ordered">
        <method name = "StartRealTimeRecording_method">
          <precondition>
            <precond constraints = "PCDU.OrbitPeriod == sunlight"/>
            <precond constraints = "AOCS.mode == mission"/>
            <precond constraints = "DT.CommunicatingWithGround == withcommuication"/>
            <precond constraints = "OBDH.satelliteMode == routine"/>
            <precond constraints = "WFI.imageCloudy == cloudless"/>
          </precondition>
          <subtasks  order = "ordered"  >
            <task  name  = "PrepareConfigCamera"      id = "22"   type = "compound"/>
            <task  name  = "ConfigureRecordDdr"       id = "28"   type = "compound" />
            <task  name  = "TurnOnDt"                 id = "38"   type = "primitive" />
          </subtasks>
        </method>
      </compoundtask>
      <compoundtask name = "StopRealTimeRecording" id = "40" inference_moment = "true" methods = "ordered">
        <method name = "StopRealTimeRecording_method">
          <subtasks order  = "ordered">
            <task  name  = "TurnOffCamera"            id = "24"   type = "compound"/>
            <task name   = "SwtichOffRtu"             id = "12"   type = "primitive"/>
            <task  name  = "TurnOffDdr"               id = "29"   type = "compound"/>
            <task  name  = "TurnOffDtRealtime"        id = "25"   type = "compound"/>
          </subtasks>
        </method>
      </compoundtask>
      <!-- metodos de tarefas objetivos-->
      <goaltask name = "AcquisitionRealTime" id = "41" methods = "ordered">
        <method name = "AcquisitionRealTime_method">
          <subtasks order = "ordered">
            <task  name  = "StartAcquisitionRealTime"  id = "20"   type = "compound"/>
            <task  name  = "StopAcquisitionRealTime"   id = "21"   type = "compound"/>
```

```
        </subtasks>
      </method>
    </goaltask>
    <goaltask name = "AcquisitionRercording" id = "42" methods = "ordered">
      <method name = "AcquisitionRercording_method">
        <subtasks  order = "ordered"  >
          <task  name  = "StartAcquisitionRecording"   id = "26"   type = "compound"/>
          <task  name  = "StopAcquisitionRecording"   id = "27"   type = "compound"/>
        </subtasks>
      </method>
    </goaltask>
    <goaltask name = "DownloadStoredImage" id = "43" methods = "ordered">
      <method name = "DownloadStoredImage_method">
        <subtasks  order = "ordered"  >
          <task  name  = "StartDownloadStoredImage"    id = "30"   type = "compound"/>
          <task  name  = "StopDownloadStoredImage"     id = "31"   type = "compound"/>
        </subtasks>
      </method>
    </goaltask>
    <goaltask name = "CameraCalibrationRecording" id = "44" methods = "ordered">
      <method name = "CameraCalibrationRecording_method">
        <subtasks order = "ordered">
          <task  name  = "StartCameraCalibrationRecord"  id = "34"    type = "compound"/>
          <task  name  = "StopCameraCalibrationRecord"  id = "35"    type = "compound"/>
        </subtasks>
      </method>
    </goaltask>
    <goaltask name = "CameraCalibrationRealTime" id = "45">
      <method name = "CameraCalibrationRealTime_method" methods = "ordered">
        <subtasks order = "ordered">
          <task  name  = "StartCalibrationRealtime"  id = "36"    type = "compound"/>
          <task  name  = "StopCalibrationRealtime"  id = "37"    type = "compound"/>
        </subtasks>
      </method>
    </goaltask>
    <goaltask name = "AcquisitionRercordingRealTime" id = "46" methods = "ordered">
      <method name = "AcquisitionRercordingRealTime_method">
        <subtasks order = "ordered">
          <task  name  = "StartRealTimeRecording"   id = "39"    type = "compound"/>
          <task  name  = "StopRealTimeRecording"   id = "40"    type = "compound"/>
        </subtasks>
      </method>
    </goaltask>
    <compoundtasks_list>
      <task  name  = "StartAcquisitionRealTime"       id = "20"/>
      <task  name  = "StopAcquisitionRealTime"        id = "21"/>
      <task  name  = "PrepareConfigCamera"            id = "22"/>
      <task  name  = "SetDataTransmitterRealTime"  id = "23"/>
      <task  name  = "TurnOffCamera"                  id = "24"/>
      <task  name  = "TurnOffDtRealtime"              id = "25"/>
      <task  name  = "StartAcquisitionRecording"      id = "26"/>
      <task  name  = "StopAcquisitionRecording"       id = "27"/>
      <task  name  = "ConfigureRecordDdr"             id = "28"/>
      <task  name  = "TurnOffDdr"                     id = "29"/>
      <task  name  = "StartDownloadStoredImage"    id = "30"/>
      <task  name  = "StopDownloadStoredImage"     id = "31"/>
      <task  name  = "ConfigureDdrPlayback"           id = "32"/>
      <task  name  = "StopDdrPlayback"                id = "33"/>
      <task  name  = "StartCameraCalibrationRecord"id = "34"/>
      <task  name  = "StopCameraCalibrationRecord" id = "35"/>
      <task  name  = "StartCalibrationRealTime"       id = "36"/>
      <task  name  = "StopCalibrationRealTime"        id = "37"/>
      <task  name  = "TurnOnDt"                       id = "38"/>
      <task  name  = "StartRealTimeRecording"         id = "39"/>
      <task  name  = "StopRealTimeRecording"          id = "40"/>
    </compoundtasks_list>
    <goaltasks_list>
      <task  name  = "AcquisitionRealTime"            id = "41"/>
      <task  name  = "AcquisitionRercording"          id = "42"/>
      <task  name  = "DownloadStoredImage"            id = "43"/>
      <task  name  = "CameraCalibrationRecording"  id = "44"/>
      <task  name  = "CameraCalibrationRealTime"   id = "45"/>
      <task  name ="AcquisitionRercordingRealTime"id = "46"/>
    </goaltasks_list>
    <primitivetasks_list>
      <task  name  = "StartImagingCamera"             id = "1"/>
      <task  name  = "StandbyCamera"                  id = "2"/>
      <task  name  = "StartRecordDdr"                 id = "3"/>
      <task  name  = "StopRecordDdr"                  id = "4"/>
      <task  name  = "StartPlaybackDdr"               id = "5"/>
      <task  name  = "StopPlaybackDdr"                id = "6"/>
      <task  name  = "SwitchOnCamera"                 id = "7"/>
      <task  name  = "SwitchOffCamera"                id = "8"/>
      <task  name  = "SwitchOnDdr"                    id = "9"/>
```

```
        <task  name  = "SwitchOffDdr"                id = "10"/>
        <task  name  = "SwitchOnRtu"                 id = "11"/>
        <task  name  = "SwtichOffRtu"                id = "12"/>
        <task  name  = "SwitchOnDt"                  id = "13"/>
        <task  name  = "SwtichOffDt"                 id = "14"/>
        <task  name  = "SetDtWfiChannelTurnOn"       id = "15"/>
        <task  name  = "SetDtWfiChannelTurnOff"      id = "16"/>
        <task  name  = "StartCalibrationCamera"      id = "17"/>
        <task  name  = "StopCalibrationCamera"       id = "18"/>
        <task  name  = "EraseFileDdr"                id = "19"/>
      </primitivetasks_list>
    </tasks>
</hierarchical_description>
```

A.2 O domínio estrutural em ISISml

A seguir, mostra-se o domínio estrutural completo para o modelo Amazonia-1 do estudo de caso na linguagem estendida.

Domínio Estrutural

```
<?xml version="1.0"?>
<structural_description>
  <domains>
    <domain name = "WfiModes">
      <value>wfi_power_off</value>
      <value>wfi_stand_by</value>
      <value>wfi_imaging</value>
      <value>wfi_calibration</value>
    </domain>
    <domain name = "DdrModes">
      <value>ddr_power_off</value>
      <value>ddr_stand_by</value>
      <value>ddr_record</value>
      <value>ddr_playback</value>
      <value>ddr_erase</value>
    </domain>
    <domain name = "DssStatus">
      <value>dss_channel_real_time</value>
      <value>dss_channel_playback</value>
    </domain>
    <domain name = "DtModes">
      <value>dt_power_off</value>
      <value>dt_standby_wfi_channel_off</value>
      <value>dt_nominal_wfi_channel_on</value>
    </domain>
    <domain name = "OrbitPeriods">
      <value>sunlight</value>
      <value>eclipse</value>
    </domain>
    <domain name = "OnOff">
      <value>line_off</value>
      <value>line_on</value>
    </domain>
    <domain name = "AocsMode">
      <value>aocs_standby</value>
      <value>aocs_survival</value>
      <value>aocs_safe_hold</value>
      <value>aocs_propulsion</value>
      <value>aocs_mission</value>
    </domain>
    <domain name = "GroundCommunication">
      <value>with_ground_communication_</value>
      <value>without_ground_communication</value>
    </domain>
    <domain name = "targetVisibility">
      <value>cloudy_image</value>
      <value>cloudless_image</value>
    </domain>
    <domain name = "SatelliteMode">
      <value>satellite_launch</value>
      <value>satellite_emergency</value>
      <value>satellite_degraded</value>
      <value>satellite_routine</value>
      <value>satellite_oribt_maneuver</value>
```

```
    </domain>
  </domains>
  <elements>
    <element name = "WFI">
      <timeline name = "Mode"           type = "WfiModes" hkparameterid = "1" />
      <timeline name = "CloudyImage" type = "targetVisibility" hkparameterid = "2" />
    </element>
    <element name = "DT">
      <timeline name = "Mode" type = "DtModes" hkparameterid = "3" />
      <timeline name = "CommunicatingWithGround" type = "bool" hkparameterid = "4" />
    </element>
    <element name = "DDR">
      <timeline name = "Mode" type = "DdrModes" hkparameterid = "5" />
      <timeline name = "DssState" type = "DssStatus" hkparameterid = "6" />
      <resource name ="Memory"capacityMax ="160000" capacityMin ="0" access="exclusive" property="continuous"
                                                                    category = "consumable" >
        <consumer name = "WFI" initialtotalconsumptionhkid = "7" initialrateconsumptionhkid = "8"/>
      </resource>
    </element>
    <element name = "AOCS">
      <timeline name = "Mode" type = "AocsMode" hkparameterid = "9" />
    </element>
    <element name = "OBDH">
      <timeline name = "Mode" type = "SatelliteMode" hkparameterid = "10"/>
    </element>
    <element name = "PCDU">
      <timeline name = "OrbitPeriod"   type = "OrbitPeriods" hkparameterid = "11"/>
      <timeline name = "WfiPowerLineStatus"        type = "OnOff"              hkparameterid = "12"/>
      <timeline name = "DdrPowerLineStatus"        type = "OnOff"              hkparameterid = "13"/>
      <timeline name = "DtPowerLineStatus"         type = "OnOff"              hkparameterid = "14"/>
      <timeline name = "RtuPowerLineStatus"        type = "OnOff"              hkparameterid = "15"/>
      <resource name = "Power" capacityMax ="250 access ="shareable" property ="discrete"category ="reusable">
        <conditional_constraint condition = "PCDU.OrbitPeriod = eclipse"    capacityMax = "200"/>
        <consumer name = "WFI"       initialtotalconsumptionhkid = "16"/>
        <consumer name = "DDR"       initialtotalconsumptionhkid = "17"/>
        <consumer name = "DT"        initialtotalconsumptionhkid = "18"/>
        <consumer name = "RTU"       initialtotalconsumptionhkid = "19"/>
      </resource>
    </element>
  </elements>
</structural_description>
```

APÊNDICE B – O MODELO EMBARCADO DO ESTUDO DE CASO

Este Apêndice traz o modelo embarcado codificado na HARPIA relativo ao estudo de caso do satélite Amazonia-1.

B.1 Ações do modelo Amazonia-1

A seguir as ações primitivas do modelo embarcado. Elas estão organizadas por elementos e em arquivos separados.

Arquivo WFI.cpp

```
Action WFI::StartImagingWfi()
{
	Action actionStatus;
	actionStatus.MarkAsSuccessfullyPerformed();

	_VERBOSE_ printk("\r\n[Preconditions] Start Imaging WFI \r\n");

	// para enviar um comando a WFI é preciso que a RTU esteja ligada
	if (!(PCDU::dcdcRtuLine.GetCurrentStatus() == line_on))
	{
		_VERBOSE_ printk("***Failure: action_wfi_imaging_precond_timeline_rtu_line ***");
		actionStatus.MarkAsInvalidPreCond(action_wfi_imaging_precond_timeline_rtu_line);
		return actionStatus;
	}
	// A câmera precisa estar em modo standby
	if (!(WFI::mode.GetCurrentStatus() == wfi_stand_by))
	{
		_VERBOSE_ printk("***Failure: action_wfi_imaging_precond_timeline_wfi_mode ***");
		actionStatus.MarkAsInvalidPreCond(action_wfi_imaging_precond_timeline_wfi_mode);
		return actionStatus;
	}
	// A power line da câmera WFI precisa estar ligada para colocá-la em modo imaging
	if (!(PCDU::wfiSpeLine.GetCurrentStatus() == line_on))
	{
		_VERBOSE_ printk("***Failure: action_wfi_imaging_precond_timeline_wfi_line ***");
		actionStatus.MarkAsInvalidPreCond(action_wfi_imaging_precond_timeline_wfi_line);
		return actionStatus;
	}

	_VERBOSE_ printk("\r\n[Effects] Start Imaging WFI \r\n");
	WFI::mode.SetCurrentStatus(wfi_imaging);
	PCDU::Power.Consume(element_wfi, 81, resource_power);

	return actionStatus;
}

Action WFI::StandbyWfi()
{
	Action actionStatus;
	actionStatus.MarkAsSuccessfullyPerformed();

	_VERBOSE_ printk("\r\n[Preconditions] Stand-by WFI mode \r\n");
	// a RTU precisa estar ligada para que a WFI possa receber comandos
	if (!(PCDU::dcdcRtuLine.GetCurrentStatus() == line_on))
	{
		_VERBOSE_ printk("***Failure: action_wfi_standby_predcond_timeline_rtu_line ***");
		actionStatus.MarkAsInvalidPreCond(action_wfi_standby_predcond_timeline_rtu_line);
		return actionStatus;
```

```
        }
        // a linha da WFI precisa estar ligada para opera-la
        if (!(PCDU::wfiSpeLine.GetCurrentStatus() == line_on))
        {
            _VERBOSE_ printk("***Failure: action_wfi_standby_predcond_timeline_wfi_line ***");
            actionStatus.MarkAsInvalidPreCond(action_wfi_standby_predcond_timeline_wfi_line);
            return actionStatus;
        }
        // Para que esse comando seja aceito é preciso que a câmera esteja no modo imaging/calibration ou no
       // próprio modo standby (ou não estar no modo power_off)
        if ((WFI::mode.GetCurrentStatus() == wfi_power_off))
        {
            _VERBOSE_ printk("***Failure: action_wfi_standby_predcond_timeline_wfi_mode ***");
            actionStatus.MarkAsInvalidPreCond(action_wfi_standby_predcond_timeline_wfi_mode);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] Stand-by WFI mode \r\n");
        WFI::mode.SetCurrentStatus(wfi_stand_by);
        PCDU::Power.Consume(element_wfi, 30, resource_power);

        return actionStatus;
}

Action WFI::StartCalibrationWfi()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Start Calibration WFI \r\n");

        if (!(PCDU::dcdcRtuLine.GetCurrentStatus() == line_on))
        {
           _VERBOSE_ printk("***Failure: action_wfi_calibration_precond_timeline_rtu_line  ***");
           actionStatus.MarkAsInvalidPreCond(action_wfi_calibration_precond_timeline_rtu_line);
           return actionStatus;
        }
        // a power line da câmera precisa estar ligada para colocá-la em modo imaging
        if (!(PCDU::wfiSpeLine.GetCurrentStatus() == line_on))
        {
            _VERBOSE_ printk("***Failure: action_wfi_calibration_precond_timeline_wfi_line***");
            actionStatus.MarkAsInvalidPreCond(action_wfi_calibration_precond_timeline_wfi_line);
            return actionStatus;
        }
        // a câmera precisa estar em modo standby
        if (!(WFI::mode.GetCurrentStatus() == wfi_imaging))
        {
            _VERBOSE_ printk("***Failure: action_wfi_calibration_precond_timeline_wfi_mode ***");
            actionStatus.MarkAsInvalidPreCond(action_wfi_calibration_precond_timeline_wfi_mode);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] Start Calibration WFI \r\n");
        WFI::mode.SetCurrentStatus(wfi_calibration);
        PCDU::Power.Consume(element_wfi, 85, resource_power);
        return actionStatus;
}

Action WFI::StopCalibrationWfi()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Stop Calibration WFI  \r\n");

        // A RTU precisa estar ligada para que a WFI possa receber comandos
        if (!(PCDU::dcdcRtuLine.GetCurrentStatus() == line_on))
        {
```

```
                _VERBOSE_ printk("***Failure: action_wfi_cabliration_off_precond_timeline_rtu_line ***");
                actionStatus.MarkAsInvalidPreCond(action_wfi_cabliration_off_precond_timeline_rtu_line);
                return actionStatus;
            }
            // A linha da WFI precisa estar ligada
            if (!(PCDU::wfiSpeLine.GetCurrentStatus() == line_on))
            {
                _VERBOSE_ printk("***Failure: action_wfi_cabliration_off_precond_timeline_wfi_line ***");
                actionStatus.MarkAsInvalidPreCond(action_wfi_cabliration_off_precond_timeline_wfi_line);
                return actionStatus;
            }
            // A câmera precisa estar em modo calibração para que esse comando seja executado com sucesso
            if (!(WFI::mode.GetCurrentStatus() == wfi_calibration))
            {
                _VERBOSE_ printk("***Failure: action_wfi_cabliration_off_precond_timeline_wfi_mode***");
                actionStatus.MarkAsInvalidPreCond(action_wfi_cabliration_off_precond_timeline_wfi_mode);
                return actionStatus;
            }

            _VERBOSE_ printk("\r\n[Effects] Stop WFI mode \r\n");
            WFI::mode.SetCurrentStatus(wfi_imaging);
            PCDU::Power.Consume(element_wfi, 30, resource_power);
            return actionStatus;
}
```

Arquivo PCDU.cpp

```
Action PCDU::SwitchOnDdrLine()
{
            Action actionStatus;
            actionStatus.MarkAsSuccessfullyPerformed();

            _VERBOSE_ printk("\r\n[Preconditions] Switch on DDR Line [PCDU] \r\n");

            if (PCDU::ddrLine.GetCurrentStatus() == line_on)
            {
                _VERBOSE_ printk("***Failure: action_pcdu_switch_on_ddr_precond_timeline_ddr_line ***");
                actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_on_ddr_precond_timeline_ddr_line);
                return actionStatus;
            }

            _VERBOSE_ printk("\r\n[Effects] Switch on DDR Line  [PCDU]  \r\n");
            PCDU::ddrLine.SetCurrentStatus(line_on);
            DDR::mode.SetCurrentStatus(ddr_stand_by);
            DDR::dssState.SetCurrentStatus(dss_channel_real_time);
            PCDU::Power.Consume(element_ddr, 23, resource_power);
            DDR::MemoryDDR.Consume(element_ddr, 0, resource_ddr_memory);

            return actionStatus;
}

Action PCDU::SwitchOnDcDcRtuLine()
{
            Action actionStatus;
            actionStatus.MarkAsSuccessfullyPerformed();

            _VERBOSE_ printk("\r\n[Preconditions] Switch on DCDC RTU Line [PCDU] \r\n");

            if (!(PCDU::dcdcRtuLine.GetCurrentStatus() == line_off))
            {
                _VERBOSE_ printk("***Failure: action_pcdu_switch_on_rtu_timeline_line_off ***");
                actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_on_rtu_timeline_line_off);
                return actionStatus;
```

```
        }
        _VERBOSE_ printk("\r\n[Effects] Switch on DCDC RTU Line [PCDU]   \r\n");
        PCDU::dcdcRtuLine.SetCurrentStatus(line_on);
        PCDU::Power.Consume(element_rtu, 10, resource_power);

        return actionStatus;

}

Action PCDU::SwitchOnDtLine() // DT = Transmissor de Dados da Banda X
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Switch on Data Transmitter [PCDU]  \r\n");

        if (DT::mode.GetCurrentStatus() == dt_nominal_wfi_channel_on)
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_on_dt_precond_timeline_dt_mode ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_on_dt_precond_timeline_dt_mode);
            return actionStatus;
        }
        _VERBOSE_ printk("\r\n[Effects] Switch on Data Transmitter [PCDU]  \r\n");
        PCDU::dtLine.SetCurrentStatus(line_on);
        DT::mode.SetCurrentStatus(dt_standby_wfi_channel_off);
        PCDU::Power.Consume(element_dt, 15, resource_power);

        return actionStatus;
}

Action PCDU::SwitchOnWfiSpeLine()
{

        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Switch on WFI Line [PCDU]  \r\n");

        if (WFI::mode.GetCurrentStatus() == wfi_imaging)
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_on_wfi_precond_timeline_wfi_mode_01***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_on_wfi_precond_timeline_wfi_mode_01);
            return actionStatus;
        }
        if (WFI::mode.GetCurrentStatus() == wfi_calibration)
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_on_wfi_precond_timeline_wfi_mode_02 ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_on_wfi_precond_timeline_wfi_mode_02);
            return actionStatus;
        }
        _VERBOSE_ printk("\r\n[Effects] Switch on WFI Line [PCDU]  \r\n");
        PCDU::wfiSpeLine.SetCurrentStatus(line_on);
        WFI::mode.SetCurrentStatus(wfi_stand_by);
        PCDU::Power.Consume(element_wfi, 30, resource_power);

        return actionStatus;
}

Action PCDU::SwitchOffWfiSpeLine()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Switch off WFI Line [PCDU]   \r\n");

        // Embora por hardware seja possível desligar a câmera em qualquer modo de operação,
        // recomenda-se que esteja em modo standby para desligá-la.
        if (WFI::mode.GetCurrentStatus() == wfi_imaging)
```

```
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_wfi_precond_timeline_wfi_mode_01***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_wfi_precond_timeline_wfi_mode_01);
            return actionStatus;
        }
        if (WFI::mode.GetCurrentStatus() == wfi_calibration)
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_wfi_precond_timeline_wfi_mode_02 ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_wfi_precond_timeline_wfi_mode_02);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] Switch off WFI Line [PCDU]  \r\n");
        PCDU::wfiSpeLine.SetCurrentStatus(line_off);
        WFI::mode.SetCurrentStatus(wfi_power_off);
        PCDU::Power.Consume(element_wfi , 0 , resource_power);

        return actionStatus;
}

Action PCDU::SwitchOffDcDcRtuLine()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Switch off DCDC RTU Line [PCDU]  \r\n");

        if (!(PCDU::dcdcRtuLine.GetCurrentStatus() == line_on))
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_rtu_timeline_line_on ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_rtu_timeline_line_on);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] Switch off DCDC RTU Line  [PCDU]  \r\n");
        PCDU::dcdcRtuLine.SetCurrentStatus(line_off);
        PCDU::Power.Consume(element_rtu, 0 , resource_power);

        return actionStatus;
}

Action PCDU::SwitchOffDdrLine()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Switch off DDR Line [PCDU]  \r\n");

        // não se deve colocar o DDR em power off se tivermos em gravação ou playback
        if ((DDR::mode.GetCurrentStatus() == ddr_record))
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_01 ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_01);
            return actionStatus;
        }

        if ((DDR::mode.GetCurrentStatus() == ddr_playback))
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_02***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_02);
            return actionStatus;
        }
        if ((DDR::mode.GetCurrentStatus() == ddr_erase))
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_03***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_03);
            return actionStatus;
        }
```

```
        if ((DDR::mode.GetCurrentStatus() == ddr_power_off))
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_04 ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_ddr_precond_timeline_ddr_mode_04);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] Switch off DDR Line [PCDU]  \r\n");
        PCDU::ddrLine.SetCurrentStatus(line_off);
        DDR::mode.SetCurrentStatus(ddr_power_off);
        PCDU::Power.Consume(element_ddr, 0 , resource_power);
        DDR::MemoryDDR.Consume(element_ddr, 0, resource_ddr_memory);

        return actionStatus;
}


Action PCDU::SwitchOffDtLine() // DT = Transmissor de Dados da Banda X
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Switch off Data Transmitter X [PCDU]  \r\n");

        // embora seja possível desligar o DT em qualquer modo, operacionalmente deve-se estar em standby
        if ((DT::mode.GetCurrentStatus() == dt_nominal_wfi_channel_on))
        {
            _VERBOSE_ printk("***Failure: action_pcdu_switch_off_dt_precond_timeline_dt_line ***");
            actionStatus.MarkAsInvalidPreCond(action_pcdu_switch_off_dt_precond_timeline_dt_line);
            return actionStatus;
        }
        _VERBOSE_ printk("\r\n[Effects] Switch off Data Transmitter X [PCDU]   \r\n");

        PCDU::dtLine.SetCurrentStatus(line_off);
        DT::mode.SetCurrentStatus(dt_power_off);
        PCDU::Power.Consume(element_dt, 0 , resource_power);

        return actionStatus;
}
```

Arquivo DDR.cpp

```
Action DDR::StartRecord()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] StartRecord DDR \r\n");
        if (!(DDR::mode.GetCurrentStatus() == ddr_stand_by))
        {
            _VERBOSE_ printk("***Failure: action_ddr_start_recording_precond_timeline_ddr_mode ***");
            actionStatus.MarkAsInvalidPreCond(action_ddr_start_recording_precond_timeline_ddr_mode);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] StartRecord DDR \r\n");

        DDR::mode.SetCurrentStatus(ddr_record);
        PCDU::Power.Consume(element_ddr, 28, resource_power);
        DDR::MemoryDDR.Consume(element_ddr, 51, resource_ddr_memory); // 51 Mbps (taxa de dados de consumo).

        return actionStatus;
}
```

```
Action DDR::StopRecord()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] Stop DDR \r\n");
        if (!(DDR::mode.GetCurrentStatus() == ddr_record))
        {
            _VERBOSE_ printk("****Failure: action_ddr_stop_recording_precond_timeline_ddr_mode****");
            actionStatus.MarkAsInvalidPreCond(action_ddr_stop_recording_precond_timeline_ddr_mode);
            return actionStatus;
        }
        _VERBOSE_ printk("\r\n[Effects] Stop Ddr \r\n");
        DDR::mode.SetCurrentStatus(ddr_stand_by);
        PCDU::Power.Consume(element_ddr, 23, resource_power);
        DDR::MemoryDDR.Consume(element_ddr, 0, resource_ddr_memory);

        return actionStatus;
}

Action DDR::StartPlayback()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] StartPlayback Ddr \r\n");

        if (! (DDR::mode.GetCurrentStatus() == ddr_stand_by)) // o DDR utiliza um clock da WFI
        {
            _VERBOSE_ printk("****Failure: action_ddr_start_playback_precond_timeline_ddr_mode ****");
            actionStatus.MarkAsInvalidPreCond(action_ddr_start_playback_precond_timeline_ddr_mode);
            return actionStatus;
        }
        if (!(WFI::mode.GetCurrentStatus() == wfi_stand_by))
        {
            _VERBOSE_ printk("***Failure: action_ddr_start_playback_precond_timeline_wfi_mode***");
            actionStatus.MarkAsInvalidPreCond(action_ddr_start_playback_precond_timeline_wfi_mode);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] Start Playback DDR \r\n");
        DDR::mode.SetCurrentStatus(ddr_playback);
        DDR::dssState.SetCurrentStatus(dss_channel_playback);
        PCDU::Power.Consume(element_ddr, 28, resource_power);

        return actionStatus;
}

Action DDR::StopPlayback()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();
        _VERBOSE_ printk("\r\n[Preconditions] StopPlayback Ddr \r\n");

        if (!(DDR::mode.GetCurrentStatus() == ddr_playback))
        {
            _VERBOSE_ printk("***Failure: action_ddr_stop_playback_precond_timeline_ddr_mode ***");
            actionStatus.MarkAsInvalidPreCond(action_ddr_stop_playback_precond_timeline_ddr_mode);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] StopPlayback DDR \r\n");
        DDR::mode.SetCurrentStatus(ddr_stand_by);
        DDR::dssState.SetCurrentStatus(dss_channel_real_time);
        PCDU::Power.Consume(element_ddr, 23, resource_power);

        return actionStatus;
```

```
}

Action DDR::EraseFile()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] EraseFile DDR \r\n");

        if (!(DDR::mode.GetCurrentStatus() == ddr_stand_by))
        {
            _VERBOSE_ printk("***Failure: action_ddr_erase_file_precond_timeline_ddr_mode ***");
            actionStatus.MarkAsInvalidPreCond(action_ddr_erase_file_precond_timeline_ddr_mode);
            return actionStatus;
        }
        _VERBOSE_ printk("\r\n[Effects] EraseFile DDR \r\n");
        DDR::mode.SetCurrentStatus(ddr_erase);
        DDR::MemoryUsage.SetCurrentStatus(0);
        DDR::mode.SetCurrentStatus(ddr_stand_by);
        PCDU::Power.Consume(element_ddr, 23, resource_power);

        return actionStatus;
}
```

Arquivo DT.cpp

```
Action DT::SetTurnOnWfiChannel()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_  printk("\r\n[Preconditions] DT::SetTurnOnWfiChannel \r\n");

        if (!(PCDU::dtLine.GetCurrentStatus() == line_on))
        {
             _VERBOSE_ printk("***Failure: action_dt_enable_wfi_channel_precond_timeline_dt_line ***");
             actionStatus.MarkAsInvalidPreCond(action_dt_enable_wfi_channel_precond_timeline_dt_line);
             return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] DT::SetTurnOnWfiChannel \r\n");
        DT::mode.SetCurrentStatus(dt_nominal_wfi_channel_on);
        PCDU::Power.Consume(element_dt, 115, resource_power);

        return actionStatus;
}

Action DT::SetTurnOffWfiChannel()
{
        Action actionStatus;
        actionStatus.MarkAsSuccessfullyPerformed();

        _VERBOSE_ printk("\r\n[Preconditions] DT::SetTurnOffWfiChannel \r\n");

        if (!(DT::mode.GetCurrentStatus() == dt_nominal_wfi_channel_on))
        {
            _VERBOSE_  printk("***Failure: action_dt_disable_wfi_channel_precond_timeline_dt_line ***");
            actionStatus.MarkAsInvalidPreCond(action_dt_disable_wfi_channel_precond_timeline_dt_line);
            return actionStatus;
        }

        _VERBOSE_ printk("\r\n[Effects] DT::SetTurnOffWfiChannel \r\n");
        DT::mode.SetCurrentStatus(dt_standby_wfi_channel_off);
        PCDU::Power.Consume(element_dt, 15, resource_power);
```

```
        return actionStatus;

}
```

B.2 Eventos do modelo Amazonia-1

A seguir os eventos do modelo embarcado. Eles estão agrupados no mesmo arquivo "*Events*" e correspondem aos eventos exógenos e endógenos temporais do estudo de caso.

Events.cpp

```
TemporalEndogenousEvent Event::ReadyDdr()
{
        TemporalEndogenousEvent temporal(event_ready_ddr);
        temporal.SetTimeConstraint(10); // o tempo para o DDR ficar disponível após ligá-lo é de 10 segundos
        return temporal;
}

TemporalEndogenousEvent Event::ReadyDataTransmitter()
{
        TemporalEndogenousEvent temporal(event_ready_data_transmitter);

                                        // após ligar o equipamento são necessários 240 segundos
        temporal.SetTimeConstraint(240); // antes de habilitar o canal de transmissão (pré-aquecimento do subsistema)
}

ExogenousEvent Event::EnterEclipse()
{
        ExogenousEvent exogenous(event_enter_eclipse);
        PCDU::orbitPeriod.SetCurrentStatus(eclipse);
        return exogenous;
}

ExogenousEvent Event::ExitEclipse()
{
        ExogenousEvent exogenous(event_exit_elcipse);
        PCDU::orbitPeriod.SetCurrentStatus(sunlight);
        return exogenous;
}

ExogenousEvent Event::StartCommunication()
{
        ExogenousEvent exogenous(event_start_communication);
        DT::communicatingWithGround.SetCurrentStatus(true);
        return exogenous;
}

ExogenousEvent Event::FinishCommunication()
{
        ExogenousEvent exogenous(event_finish_communication);
        DT::communicatingWithGround.SetCurrentStatus(false);
        return exogenous;
}
```

APÊNDICE C – SIMULAÇÕES DE TELEMETRIA PARA OS CENÁRIOS EXPERIMENTAIS

Este Apêndice traz simulações de telemetria do software de voo utilizadas para refletir diferentes estados iniciais da carga útil. As configurações de estados da Tabela C.1 foram utilizados nos experimentos da Seção 8.6, sobretudo, para geração de problemas aleatórios dos experimentos.

A coluna 'ID' diz respeito ao índice de identificação dos quarentas cenários estabelecidos. Os símbolos 'V' e 'I' correspondem respectivamente a estados válidos e inválidos da carga útil.

Tabela C.1 – Conjunto de estados dos equipamentos da carga útil do Amazonia-1 selecionados para os cenários experimentais.

Config.	**Modo dos Equipamentos**				**Linha de Potência dos Equipamentos**			
ID	**WFI**	**DDR**	**DT**	**DSS_CH**	**WFI**	**DDR**	**RTU**	**DT**
1 (V)	standby	power-off	power-off	real-time	on	off	on	off
2 (V)	power-off	power-off	standby-choff	real-time	off	off	off	on
3 (V)	power-off	standby	power-off	real-time	off	on	off	off
4 (V)	power-off	standby	standby-choff	real-time	off	on	on	on
5 (V)	standby	standby	standby-choff	real-time	on	on	on	on
6 (I)	standby	playback	power-off	playback	on	on	on	off
7 (V)	standby	power-off	standby-choff	real-time	on	off	on	on
8 (V)	power-off	power-off	standby-choff	real-time	off	off	on	on
9 (V)	standby	playback	nominal-chon	playback	on	on	on	on
10 (I)	standby	power-off	nominal-chon	real-time	on	off	on	on
11 (I)	imaging	power-off	power-off	real-time	on	off	on	off
12 (V)	imaging	record	power-off	real-time	on	on	on	off
13 (V)	imaging	playback	nominal-chon	playback	on	on	on	on
14 (V)	calibration	record	standby-choff	real-time	on	on	on	on
15 (I)	standby	record	power-off	real-time	on	on	on	off
16 (I)	imaging	playback	standby-choff	playback	on	on	on	on
17 (I)	calibration	standby	standby-choff	real-time	on	on	on	on
18 (I)	calibration	power-off	power-off	real-time	on	off	on	off

continua

Tabela C.1 – Conclusão.

Config.	Modo dos Equipamentos				Linha de Potência dos Equipamentos			
ID	**WFI**	**DDR**	**DT**	**DSS_CH**	**WFI**	**DDR**	**RTU**	**DT**
19 (I)	imaging	standby	standby-choff	real-time	on	on	on	on
20 (I)	standby	record	standby-choff	real-time	on	on	on	on
21 (I)	standby	record	standby-choff	real-time	on	on	off	on
22 (I)	imaging	playback	nominal-chon	playback	on	on	off	on
23 (I)	imaging	record	nominal-chon	real-time	on	on	off	on
24 (I)	calibration	record	power-off	real-time	on	on	off	off
25 (I)	imaging	power-off	power-off	real-time	on	off	off	off
26 (I)	calibration	standby	standby-choff	real-time	on	on	off	on
27 (I)	standby	standby	power-off	real-time	on	on	off	off
28 (I)	standby	record	power-off	real-time	on	on	off	off
29 (I)	imaging	standby	nominal-chon	real-time	on	on	off	on
30 (I)	imaging	record	standby-choff	real-time	on	on	off	on
31 (V)	imaging	standby	nominal-chon	real-time	on	on	on	on
32 (V)	imaging	record	standby-choff	real-time	on	on	on	on
33 (V)	calibration	record	power-off	real-time	on	on	on	off
34 (V)	calibration	standby	nominal-chon	real-time	on	on	on	on
35 (V)	imaging	record	nominal-chon	real-time	on	on	on	on
36 (V)	calibration	record	nominal-chon	real-time	on	on	on	on
37 (I)	calibration	record	nominal-chon	real-time	on	on	off	on
38 (I)	calibration	record	standby-choff	real-time	on	on	off	on
39 (I)	imaging	playback	power-off	playback	on	on	on	off
40 (I)	calibration	playback	power-off	playback	on	on	on	off

Fonte: Produção do autor.