sid.inpe.br/mtc-m19/2013/12.02.13.41-TDI

UMA FERRAMENTA PARA AVALIAÇÃO DE PLANOS DE VOO DE SATÉLITES USANDO MODELOS DE ESTADOS

André Aparecido de Souza Ivo

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Maurício Gonçalves Vieira Ferreira, aprovada em 13 de dezembro de 2013.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3FB2RJE>

INPE
São José dos Campos
2013

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

sid.inpe.br/mtc-m19/2013/12.02.13.41-TDI

UMA FERRAMENTA PARA AVALIAÇÃO DE PLANOS DE VOO DE SATÉLITES USANDO MODELOS DE ESTADOS

André Aparecido de Souza Ivo

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Maurício Gonçalves Vieira Ferreira, aprovada em 13 de dezembro de 2013.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3FB2RJE>

INPE
São José dos Campos
2013

Dados Internacionais de Catalogação na Publicação (CIP)

Ivo, André Aparecido de Souza.

Iv7u Uma ferramenta para avaliação de planos de voo de satélites usando modelos de estados / André Aparecido de Souza Ivo. – São José dos Campos : INPE, 2013.

xxii + 129 p. ; (sid.inpe.br/mtc-m19/2013/12.02.13.41-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2013.

Orientador : Dr. Maurício Gonçalves Vieira Ferreira.

1. avaliação. 2. teste. 3. planos de operação. 4. autômatos 5. inteligência artificial. I.Título.

CDU 004.383.4:629.783

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Mestre* em

Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais

Dra. Ana Maria Ambrosio

Presidente / INPE / São José dos Campos - SP

Dr. Mauricio Gonçalves Vieira Ferreira

Orientador(a) / INPE / SJCampos - SP

Dr. Nilson Sant'Anna

Membro da Banca / INPE / SJCampos - SP

Dr. Hassan Ahmad Sidaoui

Convidado(a) / MECTRON / SJCampos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

Aluno (a): *André Aparecido de Souza Ivo*

São José dos Campos, 13 de Dezembro de 2013

“Escolho meus amigos pela beleza, meus conhecidos pelo caráter e meus inimigos pela inteligência".

Oscar Wilde

Dedico este trabalho a meus avós (in Memoriam)
pela construção da minha família,
o meu maior tesouro e o alicerce do meu caráter.

AGRADECIMENTOS

É difícil agradecer todas as pessoas que de algum modo, nos momentos serenos e ou apreensivos, fizeram ou fazem parte da minha vida, por isso primeiramente agradeço a todos.

Dediquei este trabalho "*in Memoriam*" a meus avós, principalmente os maternos, Euclides e Geralda, e aproveito também para agradecê-los, estejam onde estiverem, pelo carinho e ensinamentos. Lembro-me do meu avô se arrumando pela manhã, após seu desjejum, para absorver um pouco de conhecimento ao ouvir notícias em seu velho rádio vermelho de pilhas. Dizia "*O conhecimento é a maior riqueza que um homem pode ter. Ninguém pode tirá-lo de você, por isso estude o máximo que puder meu filho!*". Também me recordo da minha avó, me amparando nos momentos de ansiedade, com sua sabedoria heroica me dizendo para ter paciência que a "*vitória é resultado da perseverança dos que lutam com dignidade*". Por esses e outros ensinamentos, além dos filhos maravilhosos que criaram - meus pais, é que os agradeço eternamente.

Agradeço a meus pais, Silvio e Dirce, por todas as dificuldades que enfrentamos e pela determinação e luta na minha formação, fazendo amparar os ensinamentos de meus avós.

A minha amiga e companheira, Sheila, pelo apoio e paciência durante todo o processo de desenvolvimento deste trabalho, principalmente nos momentos mais apreensivos.

Ao meu orientador, Prof. Dr. Mauricio G. V. Ferreira, pela orientação, amizade e determinação em seu trabalho.

Agradeço a todos amigos e as pessoas que me incentivaram e ajudaram a realizar este trabalho.

“Que todo o meu ser louve ao Senhor, e que eu não esqueça nenhuma das suas bênçãos!”

Salmos 103:2.

RESUMO

Os centros de controle de satélites operam diferentes tipos de satélites, famílias de satélites e missões espaciais. Normalmente, esta tarefa exige um intenso trabalho dos engenheiros e gerentes de missão devido aos diferentes conjuntos de comandos aplicados aos diferentes tipos de satélites. A intensa interversão humana é indesejável em atividades sensíveis como esta, uma vez que pode aumentar o risco operacional introduzido por fatores não controlados como: humor, horas de sono, nível de stress, etc. Portanto, pesquisas vem sendo realizadas com o objetivo de mitigar fatores não controlados associado à intervenção humana sobre a operação de controle de satélites, tais como: a pesquisa sobre o aumento da automação no âmbito das operações de satélite e pesquisas sobre procedimento para a validação de planos de operação de satélites. As técnicas de automação para operações de satélites têm a vantagem de eliminar fatores humano não controlado, retirando a intervenção humana do processo. Além disso, permite o aumento da operação de múltiplos satélites simultaneamente. No entanto, as técnicas de automação fornecem ao sistema de operação habilidades deliberativas, o que pode expor a missão espacial à situações - às vezes crítica - imprevistas, se não forem precisamente implementadas ou em caso de mau funcionamento dos equipamentos, por exemplo. Consequentemente, a comunidade de operações de satélite apresenta certa resistência na aplicação direta destas técnicas de automação. A validação dos planos de operação de satélite tem a vantagem de manter todo o controle operacional estratégico sob a intervenção humana por meio de um plano válido. Além disso, sua implantação requer o conhecimento de todos os estados possíveis e lógicas operacionais de satélite, que instrui os engenheiros e gerentes da missão a um aumento do controle operacional e sua segurança. No entanto, as pesquisas sobre a validação dos planos de operação de satélite ainda está no estado da arte. Portanto, este trabalho contribui para aprimorar a pesquisa sobre validação dos planos de operação de satélites, propondo uma nova ferramenta para avaliar os planos de operação. A ferramenta desenvolvida neste trabalho é, essencialmente, um ambiente computacional onde todos os comandos de operação de satélite, estados, e as respostas podem ser projetados e simulados independentemente da arquitetura do sistema de controle.

A TOOL FOR EVALUATION OF SATELLITE FLIGHT PLANS USING STATES MODELS

ABSTRACT

Satellite control centers operate different kind single satellites, families of satellites and missions. Usually, this task requires intense labor work from engineers and mission managers due to the different sets of commands applied to the different satellites. Intense human intervention is undesired in sensitive activities like these since it may raise the operational risk introduced by uncontrolled factor like: humor, sleeping hours, stress level, etc. Therefore, research has being carried out in order to mitigate the uncontrolled factor associated with the human intervention over satellite control operation, such as: the research on the increase of automation under satellite operations and the research on procedure for validating satellite operation plans. Automation techniques for satellite operations have the advantage of eliminate the uncontrolled human factor by simply cut the human intervention out. Also, it allows the increase the multiple simultaneous satellite operations.. However, automation techniques supply the operation system with deliberative skills, which can expose the space mission to unpredicted - sometimes critical - situations, if not precisely implemented or in case of malfunction of the equipments, for instance. Consequently, the satellite operations community is sometimes reluctant to the direct application of automation techniques. The validation of satellite operation plans has the advantage of keeping all the strategic operational control under the human intervention beneath a valid plan. Also, its implantation passes through having the knowledge of all the possible logical states of the satellite operational systems, which enlighten the engineers and mission managers increasing the operational control and its safety. However, research on validation of satellite operation plans is still in the early ages. Therefore, this work contributes to evolve the research on validation of satellite operation plans by proposing a new tool for evaluating the operation plans. The tool developed in this work is essentially a computer environment where all the satellite operation commands, states, and responses can be designed and simulated irrespective of the control system architecture.

LISTA DE FIGURAS

Pág.

Figura 1.1 - Missões do INPE propostas para o período 2011-2020 4
Figura 2.1 - Ciclo de operação de controle 9
Figura 2.2 - Classificação de atividades de controle 12
Figura 2.3 - Diagrama de transição de atividades de controle 13
Figura 2.4 - Diagrama de transição de modos de operação de um satélite 14
Figura 2.5 - Modelo conceitual de planejamento 18
Figura 3.1 - Verificador de plausibilidade 34
Figura 3.2 - Filtro de segurança 35
Figura 3.3 - Arquitetura de validação de plano de voo. 36
Figura 3.4 - Arquitetura para geração de diagnóstico. 37
Figura 4.1 - Exemplo de MEF 42
Figura 4.2 - Exemplo de *Statecharts* 47
Figura 5.1 - Diagrama da arquitetura proposta 52
Figura 5.2 - Diagrama de casos de uso - Modelo principal da ferramenta 55
Figura 5.3 - Exemplo de um DET para máquina de refrigerante 66
Figura 5.4 - Exemplo de um plano operacional válido 67
Figura 5.5 - Simulação da execução do plano válido 67
Figura 5.6 - Exemplo de um plano operacional inválido 68
Figura 5.7 - Simulação da execução do plano inválido 68
Figura 5.8 - Domínio Lógico e Domínio de Estados e Transições 69
Figura 5.9 - Esquema de geração de eventos 70
Figura 5.10 - Formato do plano de execução 71
Figura 5.11 - Diagrama de atividade de funcionamento da arquitetura 73
Figura 6.1 - Componentes do Atom SysVap 78
Figura 6.2 - IDE do Atom SysVap 79
Figura 6.3 - Atom SysVap - Ferramentas para interação com projeto 80
Figura 6.4 - Atom SysVap - Inspetor de objetos 80
Figura 6.5 - Atom SysVap - Área de trabalho 81
Figura 6.6 - Atom SysVap - Ações (DL) 82
Figura 6.7 - Atom SysVap - Inspetor de Objetos 83
Figura 6.8 - Atom SysVap - Opções de depuração 85
Figura 6.9 - Atom CORE- Exemplo de código para simulação 87
Figura 7.1 - Modelo de domínio para Câmera CCD 90
Figura 7.2 - Ação executada no estado "*picture*" 92
Figura 7.3 - Ação executada no estado "*alert*" 92
Figura 7.4 - Avaliação dos planos executados no domínio do sistema 93
Figura 7.5 - Plano para obter 2 fotos 94
Figura 7.6 - Avaliação realizada pelo Atom SysVap 94
Figura 7.7 - Diagrama do subsistema de suprimento de energia do satélite 95
Figura 7.8 - DET geral 98
Figura 7.9 - DL geral - Definição das variáveis 99

Figura 7.10 - DET para o sistema satélite 99
Figura 7.11 - DL satelital - Definição das variáveis 100
Figura 7.12 - DL satelital - Definição dos eventos de transição..... 101
Figura 7.13 - DL satelital - Definição dos eventos dos estados..... 102
Figura 7.14 - Esquema da dinâmica de voo de um satélite..... 103
Figura 7.15 - DET para o ambiente 104
Figura 7.16 - DL ambiental - Definição dos eventos de transição 104
Figura 7.17 - DL ambiental - Definição do estado "*sun*" 105
Figura 7.18 - Plano 1 unificado do experimento 2 107
Figura 7.19 - Relatório da simulação do plano 1 unificado..... 108
Figura 7.20 - Plano 1 unificado do experimento 2 109
Figura 7.21 - Relatório da simulação do plano 2 unificado..... 110
Figura A.8.1 - Diagrama de Relacionamento de Classes..... 123
Figura A.8.2 - Descrição da Classe que representa o projeto..... 124
Figura A.8.3 - Descrição da Classe que representa a MEF 125
Figura A.8.4 - Descrição da Classe que representa o estado 126
Figura A.8.5 - Descrição da Classe que representa a transição 127
Figura A.8.6 - Descrição da Classe para os eventos das transições 128
Figura A.8.7 - Descrição da Classe que representa os tipos dos estados 129
Figura A.8.8 - Descrição da Classe que representa a lista de MEF 129

LISTA DE TABELAS

Pág.

Tabela 4.1 - Tabela de transição da MEF Figura 4.1 42
Tabela 6.1 - Atom SysVap - Objetos disponíveis no DL..... 83
Tabela 7.1 - Comandos de controle 106
Tabela 7.2 - Plano 1 do experimento 2 106
Tabela 7.3 - Eventos orbitais 1 do experimento 2 106
Tabela 7.4 - Plano 2 do experimento 2 109
Tabela 7.5 - Eventos orbitais 2 do experimento 2 109

LISTA DE SIGLAS E ABREVIATURAS

ABSTRIPS	*Abstraction-Based Stanford Research Institute Problem*
ADL	*Action Description Language*
AST	Astrofísica
AXLOG	*Société d'ingénierie experte en informatique industrielle technique et scientifique*
CBERS	*China-Brazil Earth-Resources Satellite* - Satélite Sino-Brasileiro de Recursos Terrestres
CCD	*Charge-Coupled Device*
CLE	Clima Espacial
DET	Domínio de Estados e Transições
DHS	*Data Handling System*
DL	Domínio Lógico
DOD	*Depth of Discharge*
EADS	*European Aeronautic Defence and Space Company*
ESA	*European Space Agency*
ESTEC	*European Space Research and Technology Centre*
HTN	*Hierarchical Task Network*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
INPE	Instituto Nacional de Pesquisas Espaciais
KBS	*Knowledge Based Systems*
LAAS-CNRS	*Laboratoire d'Architecture et d'Analyse des Systèmes - Centre National de la Recherche Scientifique*
LEO	*Low Earth Orbit*
MECB	Missão Espacial Completa Brasileira
MEF	Máquinas de Estados Finitos
MEFE	Máquinas de Estados Finitos Estendida
NASA	*National Aeronautics and Space Administration*
NOAH	*Nets of Action Hierarchies*

O-PLAN	*Open Planning Architecture*
PDDL	*Planning Domain Definition Language*
PMM	Plataforma Multimissão
POP	*Partial-Order Planning*
POV	Planos de Operações de Voo
SAR	*Synthetic Aperture Radar*
SPAAS	*Software Product Assurance for Autonomy on-board Spacecraft*
STRIPS	*Stanford Research Institute Problem*
SYSVAP	*System for Validation of finite Automatons and execution Plans*
UCPOP	*Universal Conditional Partial Order Planner*
XML	*eXtensible Markup Language*

SUMÁRIO

Pág.

1 INTRODUÇÃO 1
1.1. Motivação do trabalho de pesquisa 3
1.2. Objetivos do trabalho de pesquisa 5
1.3. Metodologia de pesquisa e organização da dissertação 6

2 PLANEJAMENTO 9
2.1. Planejamento de operações de controle de satélite 9
2.1.1. Atividades de controle e modos de operação de satélites 10
2.2. Conceitos de planejamento 16
2.2.1. Histórico 21
2.3. Análise de domínio 25
2.3.1. Conceitos de análise de domínio 25
2.4. Planos operacionais 27
2.5. Considerações 28

3 O ESTADO DA ARTE EM AVALIAÇÃO DE PLANOS 29
3.1. Avaliação 29
3.1.1. Qualidade dos planos 29
3.2. Segurança no planejamento da operação de sistemas espaciais 31
3.3. Uma estratégia para validação de planos operacionais 35
3.4. Considerações 38

4 REPRESENTAÇÃO DE MODELOS DE ESTADO 39
4.1. Máquinas de estados Finitos - MEF 41
4.1.1. Formalismo 42
4.1.2. Propriedades de MEFs 45
4.2. *Statecharts* 46
4.3. Considerações 48

5 ARQUITETURA PROPOSTA 51
5.1. Requisitos funcionais para a ferramenta de avaliação 52
5.2. Requisitos não funcionais para a ferramenta de avaliação 53
5.3. Casos de uso para a ferramenta de avaliação 54

5.3.1. Atores 54
5.3.2. Identificação dos casos de uso 54
5.4. Especificação da arquitetura 65
5.4.1. DET - Domínio de Estados e Transições 66
5.4.2. DL - Domínio lógico 69
5.4.3. Definição do Plano em formato XML 70
5.5. Diagrama de atividades 72
5.6. Considerações 74

6 A FERRAMENTA - ATOM SYSVAP 77
6.1. Atom IDE 78
6.2. Atom CORE 86
6.3. Considerações 87

7 ESTUDO DE CASO 89
7.1. Experimento 1 - Acionamento de câmera CCD 89
7.2. Experimento 2 - sistema de suprimento de energia 95
7.2.1. Modelo de domínio 97

8 CONCLUSÕES 111
8.1. Contribuições do trabalho 111
8.2. Trabalhos futuros 113

REFERÊNCIAS BIBLIOGRÁFICAS 115

APÊNDICE A - DIAGRAMA DE CLASSES 123

1 INTRODUÇÃO

Atualmente a operação de missões espaciais requer uma grande quantidade de recursos humanos e atividades o que eleva o custo da operação, constituindo uma das maiores preocupações dos profissionais que trabalham na operação de satélite.

Em especial, a atividade de operação de satélites, do programa espacial do INPE, necessita de intervenções manuais para produção de planos de voo para vários tipos e famílias de satélites. Essa intervenção pode levar a falhas na sequencia de comandos a serem enviados ao satélite, representando riscos para a missão.

O programa espacial do INPE possui um grande interesse relacionado à tarefa de controlar múltiplos satélites com soluções para redução de custos e aumento da confiabilidade nas atividades de controle dos satélites. Estas necessidades motivaram o desenvolvimento de sistemas para automação das atividades de controle de satélites no INPE, com o objetivo de atender a demanda crescente de lançamentos e o interesse de reduzir custos (SOUZA, 2011).

Visando contribuir neste tema, diversos trabalhos de pesquisa veem sendo elaborados com objetivos de automatizar o desenvolvimento ou de avaliar os planos de operações de voos (POV) de satélites, para evitar prejuízos financeiros causados por falhas, principalmente humanas.

Destacam-se trabalhos envolvendo o planejamento autônomo, uma das áreas da inteligência artificial (IA), que desenvolve o processo de definição de sequência de ações através de métodos computacionais para que sejam alcançados os objetivos estabelecidos (CARDOSO, 2006b). Neste tema, alguns trabalhos se destacam como os realizados no INPE: "*An Intelligent System For Generation Of Automatic Flight Operation Plans For The Satellite*

Control Activities At Inpe" (CARDOSO; FERREIRA; ORLANDO, 2006a), "Alocação dinâmica de recursos computacionais para experimentos científicos com replanejamento automatizado" (KUCINSKIS, 2007), e na NASA, "*Planning in Interplanetary Space: Theory and Pratice*" (JÓNSSON *et al.*, 2000).

O uso de técnicas de planejamento e escalonamento tem sido reconhecido como uma das mais promissoras soluções para o desenvolvimento de planos de voo de forma autônoma, no entanto o uso destas técnicas sofre grande resistência por conta da aplicação de técnicas não determinísticas (KUCINSKIS, 2007).

Neste cenário, surgem, na comunidade científica, trabalhos para avaliar planos, com o objetivo de conferir maior segurança ou mesmo para agregar maior confiabilidade a sistemas autônomos como, por exemplo, os trabalhos SPAAS (*Software Product Assurance for Autonomy on-board Spacecraft*) (BLANQUART *et al.*, 2004) e "Uma estratégia baseada em algoritmos de mineração de dados para validar plano de operação de voo a partir de predições de estados dos satélites do INPE" (SOUZA, 2011). O primeiro teve grande relevância na definição do padrão da ESA para desenvolvimento de sistemas espaciais, enquanto o segundo foi um dos primeiros trabalhos do INPE relacionado a validação de planos operacionais.

Considerando este contexto e seguindo os padrões definidos pela ESA no programa SPAAS (BLANQUART *et al.*, 2004), o presente trabalho tem como objetivo contribuir na área de avaliação de planos operacionais, através de técnicas determinísticas, concedendo maior segurança e confiabilidade além de possibilitar o uso de planejadores autônomos de forma mais segura.

A arquitetura proposta para ferramenta toma como base a técnica de modelagem baseada em modelos de estado, como por exemplo, as máquinas de estados finitos (MEF) e o StateCharts, para permitir que os planos sejam avaliados de forma determinística, permitindo a utilização em avaliação de

planos gerados por qualquer agente, seja humano ou computacional. Para validar essa arquitetura uma ferramenta foi construída e vários testes foram realizados.

1.1. Motivação do trabalho de pesquisa

Devido à importância dos satélites para incontáveis aplicações de interesse da sociedade brasileira, como o monitoramento da vegetação nativa, plantações, rios e clima, levantamento de recursos naturais e telecomunicações, entre outros, o Instituto Nacional de Pesquisas Espaciais (INPE) tem como objetivo principal o domínio de tecnologia espacial a partir do desenvolvimento e operação de satélites.

Atualmente o INPE tem dois satélites em operação: os Satélites de Coleta de Dados SCD1 e SCD2, que foram integralmente desenvolvidos pelo INPE, como parte da chamada Missão Espacial Completa Brasileira (MECB).

A Figura 1.1 a seguir sintetiza o conjunto de satélites e suas respectivas missões propostos até 2020, onde:

1. As missões científicas (CLE – Clima espacial e AST – Astrofísica) utilizarão plataformas na classe de 100 kg.

2. As missões baseadas na PMM estarão na classe de 500 kg.

Como se pode observar, esse plano de missões prevê mais de um lançamento de satélite por ano a partir de 2016. A partir de 2015, haverá anualmente pelo menos um lançamento de satélites baseados na PMM, além dos de pequeno porte (até 100 kg) e do Satélite SAR (INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE)).

A previsão do aumento do número de satélites a serem controlados e diminuição dos recursos financeiros destinados a operação de satélites no

INPE (CARDOSO; FERREIRA; ORLANDO, 2006a), levou ao desenvolvimento de sistemas para automação das atividades de controle de satélites capaz de gerar planos de operação de voo de forma automatizada por planejadores (CARDOSO; FERREIRA; ORLANDO, 2006a) e assim, reduzir custos relacionados às equipes de operação, além de trazer maior autonomia ao sistema espacial.

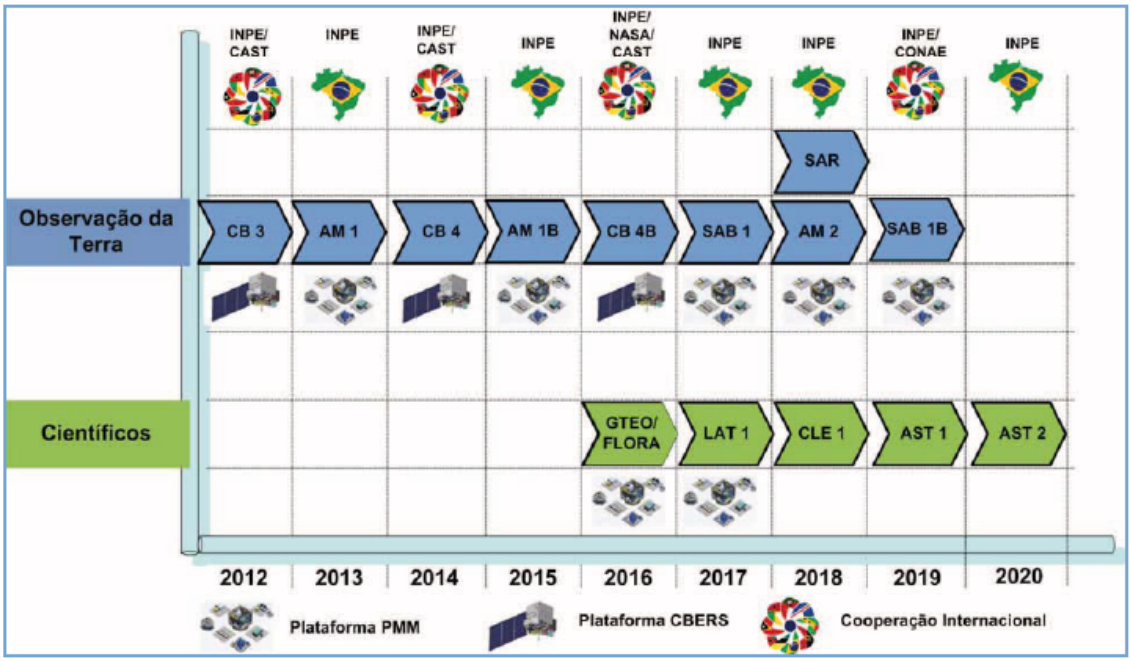

Figura 1.1 - Missões do INPE propostas para o período 2011-2020
Fonte: INPE (2011)

Entretanto, o aumento da autonomia expõe outro problema: a resistência de gerentes de missão e engenheiros, quanto ao uso de técnicas não determinísticas. (KUCINSKIS, 2007)

Esta resistência vem da crença de que todas as decisões tomadas devem ser absolutamente previsíveis. Saber exatamente como um sistema irá se comportar em cada uma das situações que lhe forem apresentadas aumenta a segurança com relação à sua robustez, mas reduz o número de situações às quais o sistema poderá gerar uma resposta adequada. (KUCINSKIS, 2007)

A resistência de gerentes de missão e engenheiros com relação ao aumento da autonomia é natural e até mesmo justificada, uma vez que o custo e o volume de trabalho de uma missão espacial são em geral considerados elevados para confiar ao sistema a tomada de mais decisões (KUCINSKIS, 2007).

A escolha deste trabalho de pesquisa foi motivada pela necessidade de conciliar a geração de planos de forma autônoma, concedendo maior autonomia ao sistema espacial, e a previsibilidade e confiabilidade requerida pelos gerentes de missão e engenheiros.

1.2. Objetivos do trabalho de pesquisa

Baseado na necessidade de geração de planos de forma autônoma e previsibilidade e confiabilidade requerida pelos gerentes de missão e engenheiros, este trabalho propõe o desenvolvimento de uma ferramenta para avaliação determinística dos planos, que deve servir de ferramenta de apoio à tomada de decisão nas atividades de avaliação de planos operacionais gerados por planejadores autônomos ou por especialistas.

Sendo os objetivos específicos do trabalho:

- Apresentar uma alternativa para avaliação de planos de operação de voo em especial os relacionados a missões espaciais;
- Permitir uma avaliação determinística dos planos;

Para atingir os objetivos apresentados foi desenvolvida uma ferramenta para validar a arquitetura proposta. Esta ferramenta é um *software* implementado em JAVA que permite a modelagem tanto do sistema espacial quanto do ambiente espacial, simulando o comportamento de um satélite, por exemplo. Esta ferramenta permite que os planos sejam executados e avaliados antes de

serem enviados para o sistema espacial. Com isto o trabalho apresenta os seguintes objetivos gerais:

- Propor uma ferramenta de apoio a modelagem alternativa à PDDL (*Planning Domain Definition Language*), uma linguagem para modelagem de sistemas reativos;

- Permitir que a ferramenta proposta seja integrada com outros sistemas para avaliação de planos em tempo real, o que deverá possibilitar o replanejamento através de técnicas de planejamento autônomo;

- Possibilitar novas pesquisas utilizando esta ferramenta alternativa de modelagem de sistemas reativos baseados em IA.

Por fim, este trabalho apresentará um experimento para avaliar o uso da ferramenta desenvolvida.

1.3. Metodologia de pesquisa e organização da dissertação

Na primeira etapa deste trabalho, a pesquisa concentra-se no levantamento da bibliografia existente sobre as soluções de automação que foram ou estão sendo adotadas pela comunidade espacial, o estado da arte na tecnologia de avaliação e validação de planos operacionais e um levantamento da situação atual das atividades de operação de satélites no INPE.

Em seguida, em uma fase preliminar, o trabalho concentra-se na implementação de protótipos para a avaliação das tecnologias de modelagem de sistema, cujos modelos pudessem ser determinísticos. Explorou-se o uso de Teoria dos conjuntos, Máquinas de Estados Finitos (MEF), Redes de Petri e Teoria de Filas, técnicas que possuem a propriedade de determinismo que indica se o modelo pode ou não ser determinístico. Ao final constatou-se que todas as técnicas poderiam ser utilizadas, no entanto, Redes de Petri e Teoria dos Conjuntos apresentam certa dificuldade na representação de paralelismo e

sincronização entre componentes (FRANCÊS *et al.*, 2001), o que pode ser claramente representado em Statecharts, uma extensão de Máquinas de Estados Finitas Estendidas (MEFE). (SILVA *et al.*, 2008)

A característica de paralelismo e sincronização é bastante importante para representação de sistemas, o que justifica a escolha pela técnica de Modelos de estados, em especial a representação em StateCharts, que possui grande disposição de literatura, além de possuir um formalismo matemático que apresenta dentre outras propriedades a de determinismo.

Alguns trabalhos sobre MEF podem ser citados, como, por exemplo, algoritmos de busca e minimização de máquinas: *"Nondeterministic state machines in protocol conformance testing"* (PETRENKO *et al.*, 1993), "*ConData: a Tool for Automating Specification-based Test Case Generation for Communication Systems"* (MARTINS; SABIÃO; AMBROSIO, 1999), "*Plavis/FSM: an environment to integrate FSM-based testing tools*" (SIMÃO *et al.*, 2005) e *"Generation Reduced Test for FSMs with Extra States"* (SIMÃO; PETRENKO; YEVTUSHENKO, 2009).

Após a escolha da técnica de Modelo de estados o trabalho seguiu para o desenvolvimento da arquitetura e a implementação da ferramenta e desenvolvimento de um experimento hipotético e controlado para validar a arquitetura. Para tal o trabalho foi organizado da seguinte maneira:

- CAPÍTULO 2 - PLANEJAMENTO : neste capítulo são apresentados os conceitos sobre planejamento autônomo, uma das área da inteligência artificial.

- CAPÍTULO 3 - O ESTADO DA ARTE EM AVALIAÇÃO: neste capítulo são apresentados os conceitos sobre a avaliação de planos operacionais em missões espaciais.

- CAPÍTULO 4 - REPRESENTAÇÃO DE MODELOS DE ESTADO: neste capítulo são apresentados os conceitos sobre máquinas de estados finitos.

- CAPÍTULO 5 - ARQUITETURA PROPOSTA: neste capítulo é apresentada a arquitetura proposta para avaliação de planos operacionais para missões espaciais.

- CAPÍTULO 6 - A FERRAMENTA - ATOM SYSVAP: neste capítulo é apresentado o Atom, ferramenta implementada para validação da arquitetura proposta no capítulo 5.

- CAPÍTULO 7 - ESTUDO DE CASO: neste capítulo é apresentado um experimento hipotético e controlado para auxiliar a validação da ferramenta e da arquitetura proposta.

- CAPÍTULO 8 - CONCLUSÕES: neste capítulo são apresentadas as conclusões finais deste trabalho.

2 PLANEJAMENTO

Com o objetivo de analisar o atual cenário da área de planejamento automático, neste capítulo são descritas brevemente a aplicação da técnica no controle de satélite e as principais áreas e técnicas associadas ao processo de modelagem de sistemas, principalmente os de planejamento automático. Além disso, são apresentadas alguns conceitos sobre representação do conhecimento, como: Análise de Domínio, Modelos de Domínios.

2.1. Planejamento de operações de controle de satélite

As atividades envolvidas na operação de controle de satélites incluem dinâmica de voo, planejamento de operações e execução de procedimentos operacionais. Estas atividades são realizadas periodicamente, independentemente para cada satélite, conforme exibido na Figura 2.1. (TOMINAGA, 2010)

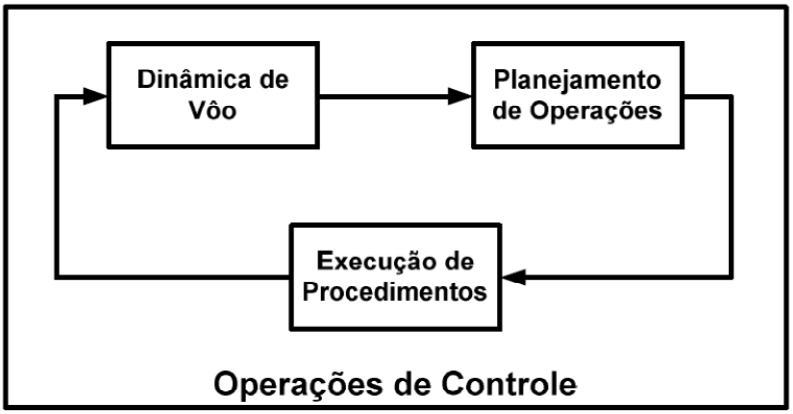

Figura 2.1 - Ciclo de operação de controle
Fonte: Tominaga (2010)

A dinâmica de voo é responsável pela determinação e propagação de informações referentes a posicionamento orbital e apontamento de atitude de

satélites. Ela utiliza medidas de rastreio e calibração, obtidas por meio de execução de procedimentos em passagem, para determinação e propagação de parâmetros orbitais e de atitude. (TOMINAGA, 2010)

O planejamento de operações agenda tarefas que devem ser desempenhadas para possibilitar a execução de operações de missão para cada satélite. Este processo é realizado em cima de previsões de contato entre o satélite e antenas de controle, fornecidos pela dinâmica de voo. (TOMINAGA, 2010)

A execução de procedimentos consiste na realização, em tempo real, de tarefas agendadas conforme o planejamento de operações. É através desta execução, que, por exemplo, equipamentos de carga-útil dos satélites são ligados para a obtenção de dados da missão, e medidas de posição e apontamento são obtidas para serem processadas pela dinâmica de voo. (TOMINAGA, 2010)

A geração de planos de operações em voo constitui o objetivo principal do planejamento de operações. A fim de se realizar o planejamento de operações, é necessário conhecer com precisão a posição e a trajetória do satélite. Isto é realizado pela dinâmica de voo, a partir de medidas de rastreio. A partir destas informações, é possível calcular os ângulos de apontamento de antenas de controle. (TOMINAGA, 2010)

Por definição um plano de operação de voo é uma sequencia finita de ações linearmente ordenada e temporizada. (TOMINAGA, 2010) (GHALLAB; NAU; TRAVERSO, 2004)

2.1.1. Atividades de controle e modos de operação de satélites

Atividades de controle de satélites são específicas para cada satélite e mudam, mesmo para a execução de uma mesma operação de controle, conforme o

modo de operação em que se encontra o satélite. Elas se dividem primeiramente entre atividades planejadas e atividades emergenciais. Atividades planejadas englobam tarefas executadas segundo diretrizes estabelecidas por requisitos de operações. Já atividades emergenciais incluem tarefas cujas execuções não são planejadas a priori, que devem ser executadas conforme a necessidade. (TOMINAGA, 2010)

Entre as atividades planejadas, destacam-se aquelas que são executadas periodicamente, repetindo um padrão cíclico num curto intervalo de tempo. Estas atividades são denominadas atividades rotineiras. Em condições normais de operação, a maior parte das atividades de controle de satélites é composta por atividades rotineiras. (TOMINAGA, 2010)

Atividades planejadas especiais são elaboradas por projetistas de satélite ou especialistas de sistema, a partir de análise de estado corrente do satélite. Estas atividades têm como finalidade ou manter parâmetros controláveis dentro da faixa nominal de operação, ou levar o satélite de um modo degradado para um modo de operação normal. A execução bem-sucedida de atividades especiais leva o satélite a um modo de operação em rotina, regida por atividades rotineiras. Já uma execução malsucedida pode levar o satélite a um modo de operação emergencial. (TOMINAGA, 2010)

Atividades emergenciais são causadas por alguma anomalia, sendo normalmente resultantes de uma ou mais falhas. Procedimentos de atividades emergenciais são elaborados e documentados por projetistas de equipamentos embarcados com base em previsões de falhas. Estas atividades têm como finalidade conter uma maior degradação do satélite e levá-lo rapidamente a um modo seguro de operação. A execução de atividades emergenciais leva invariavelmente à elaboração de atividades planejadas especiais. (TOMINAGA, 2010)

Para uma facilitar a compreensão da classificação das atividades de controle, a Figura 2.2 mostra como estas atividades se subdividem. (TOMINAGA, 2010)

Figura 2.2 - Classificação de atividades de controle
Fonte: Tominaga (2010)

A Figura 2.3 explicita ações e eventos inerentes à execução de determinadas atividades de controle e aquelas que disparam transições. Tais ações e eventos caracterizam os modos de operação de um satélite. (TOMINAGA, 2010)

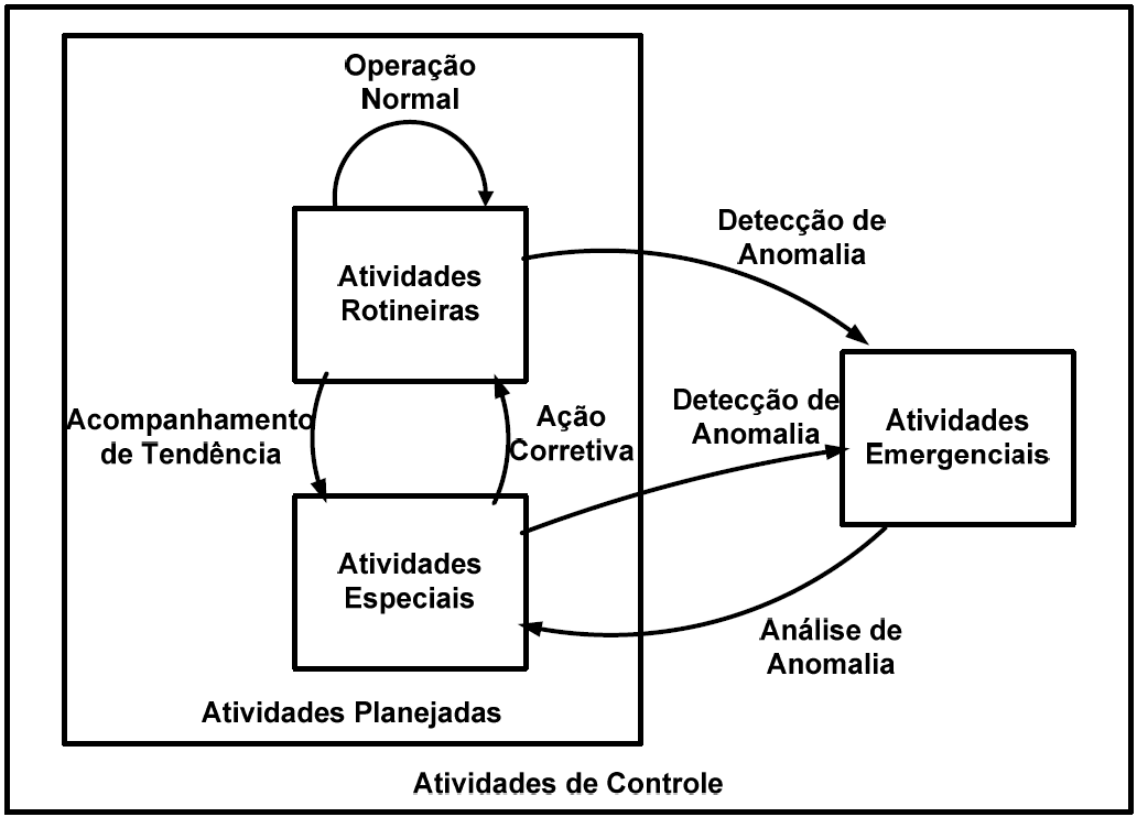

Figura 2.3 - Diagrama de transição de atividades de controle

Fonte: Tominaga (2010)

Cada satélite possui seus próprios modos de operação, diferenciados conforme especificidades de cada missão. Porém, de forma geral, todos os modos de operação podem-se enquadrar dentro de quatro categorias básicas. A Figura 2.4 resume o relacionamento entre as atividades de controle e os modos de operação a que se aplicam. (TOMINAGA, 2010)

No modo de operação normal, tem-se uma situação estável, dentro do qual o satélite é operado plenamente conforme as especificações de projeto da missão. Neste modo, executam-se operações de controle que são compostos majoritariamente por atividades rotineiras. Exemplos incluem programação e execução de medidas de rastreio, recepção de telemetrias, e envio de

telecomandos para ativação e desativação de carga-útil de acordo com visibilidade de estações terrenas de recepção de dados. Atividades especiais são também ocasionalmente executadas, conforme os valores de parâmetros sob monitoração aproximam-se de seus valores limite. Exemplos incluem programação e execução de correção de tempo de computador embarcado, e manobras de correção orbital ou de atitude. (TOMINAGA, 2010)

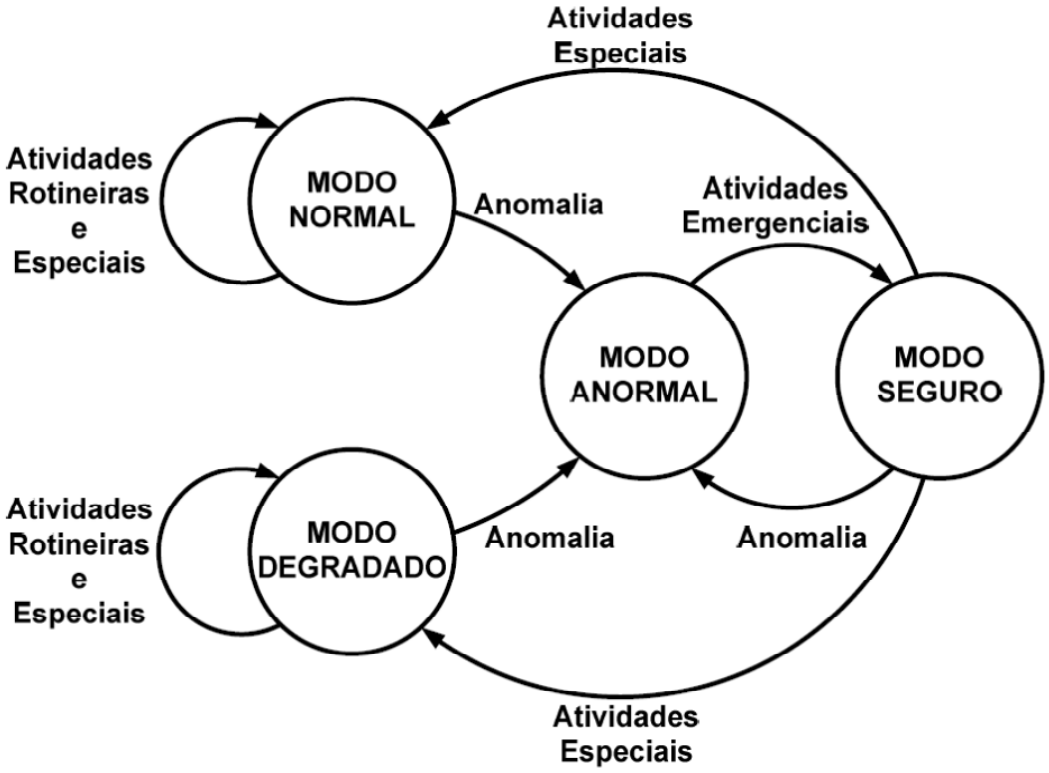

Figura 2.4 - Diagrama de transição de modos de operação de um satélite
Fonte: Tominaga (2010)

O modo de operação anormal é uma situação instável, causada por uma ou mais anomalias em bordo. A entrada e permanência neste modo constituem uma situação indesejável que, caso não seja corrigida em tempo, pode pôr em risco a saúde do satélite. Portanto, assim que a execução de procedimentos de

monitoração de telemetria identifica a entrada do satélite neste modo, atividades emergenciais devem ser disparadas para levar o satélite a um modo de operação seguro. (TOMINAGA, 2010)

O modo de operação seguro consiste em uma situação estável alcançada após a execução de atividades emergenciais. Neste modo, o satélite é mantido vivo, porém inoperante. Em outras palavras, somente são mantidos ligados equipamentos embarcados com funções de manutenção de plataforma em vôo. Equipamentos de carga-útil, para desempenho de funções de missão, são mantidos desligados. Enquanto o satélite encontra-se neste modo, realizam-se análises para se buscar as causas das anomalias que trouxeram o satélite para o modo anormal, e buscam-se soluções para levá-lo de volta ao modo normal. Ou então, caso o retorno ao modo normal seja comprovadamente impossível, tenta-se encontrar um modo degradado em que o satélite possa operar. (TOMINAGA, 2010)

O modo de operação degradado é outra situação estável, em que as cargas-úteis do satélite são operadas, porém sem satisfazer inteiramente as especificações de projeto da missão. O satélite entra neste modo após uma ou mais anomalias terem causado dano permanente em um ou mais equipamentos em bordo. Neste modo, atividades de controle são desempenhadas de forma semelhante ao que ocorre no modo normal. Porém, atualizam-se atividades de controle para refletir as alterações dos requisitos de missão. Por exemplo, a perda de um equipamento de carga-útil sem redundância leva necessariamente ao cancelamento de operações que dependem dele. Dependendo do caso, uma falha deste tipo pode ocasionar um uso mais agressivo das cargas-úteis restantes, devido ao acúmulo de energia excedente em bordo, ou na tentativa de tirar máximo proveito da vida remanescente do satélite antes que se alcance seu fim. (TOMINAGA, 2010)

Acordos de cooperação internacionais estão sendo firmados para novos satélites, de forma a dividir tanto custos de desenvolvimento quanto retornos de investimento. Centros de missão procuram atender a demandas de um número crescente de usuários, ao passo que usos dos equipamentos embarcados devem ser evitados. Novas tecnologias adicionam cada vez mais recursos aos equipamentos de carga-útil, o que contribui para o aumento de complexidade em suas operações. Os requisitos operacionais de satélites evidenciam a complexidade crescente das novas missões, comparadas a seus antecessores. (TOMINAGA, 2010)

Para atender essa demanda, planejadores de operações rotineiras, que utilizam técnicas de planejamento autônomo e inteligência artificial, vem sendo desenvolvidos. (BIANCHO *et al.*, 2006) (CARDOSO, 2006b)

O próximo tópico apresenta as principais técnicas sobre planejamento autônomo e quais são suas características.

2.2. Conceitos de planejamento

Planejar é o processo abstrato e deliberativo de escolha e organização de ações antecipando seus efeitos esperados. Esse processo tem como missão atingir, da melhor forma possível, os objetivos pré-estabelecidos, que também são definidos como problemas de planejamento. (VAQUERO, 2007)

O planejamento autônomo, uma das áreas da inteligência artificial (IA), desenvolve o processo de definição de sequencia de ações para que sejam alcançados os objetivos estabelecidos de um determinado problema. Dado uma situação inicial, um conjunto de ações e uma situação final desejada, a tarefa de planejamento consiste em determinar uma sequência de ações que solucione o problema (CANTONI, 2010).

A motivação para os avanços nas pesquisas de planejamento automático se dá por conta da atividade de sistemas em cenários que necessitam de habilidades autônomas (CANTONI, 2010), como planejamento de trajetória e movimentação de sistemas automáticos móveis; planejamento de percepção envolvendo ações de sensoriamento para captação de informação do ambiente; planejamento de navegação que combina sensoriamento e definições de trajetórias; planejamento de manipulação relacionado com movimentação de objetos como, por exemplo, montagem de peças, organização de contêineres, entre outros (GHALLAB; NAU; TRAVERSO, 2004).

Um sistema planejador típico possui as seguintes características (GHALLAB; NAU; TRAVERSO, 2004):

- um modelo do ambiente com representações de seus estados;
- um conjunto de ações aplicáveis ao modelo (cada uma possuindo suas pré-condições para execução e a descrição de seus efeitos);
- restrições;
- recursos disponíveis;
- os objetos;
- os objetivos a serem atingidos (eventualmente com critérios de otimização);
- estado inicial de um problema de planejamento.

O plano é gerado a partir de um processo de planejamento que se baseia no estado inicial, e passa a selecionar ações, dentre um conjunto especificado no domínio, levando em consideração suas precondições, para atingir o objetivo

estabelecido. O conjunto das ações executadas com sucesso é chamado de “plano”. (GHALLAB; NAU; TRAVERSO, 2004)

Pode ser observado na Figura 2.5 um modelo conceitual de planejamento. Observa-se que a especificação do problema e o conhecimento sobre os domínios fornecidos ao planejador são fundamentais para o processo de busca da solução. (GHALLAB; NAU; TRAVERSO, 2004)

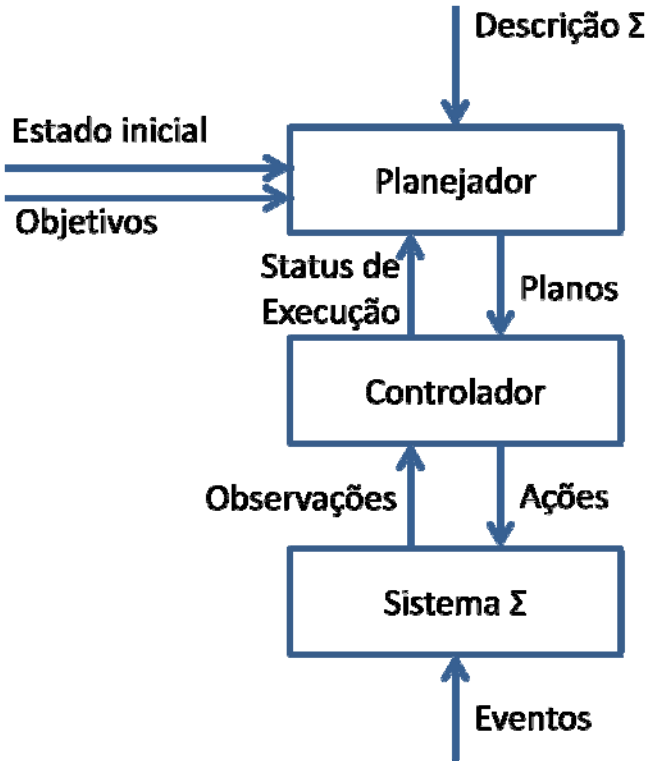

Figura 2.5 - Modelo conceitual de planejamento
Fonte: Ghallab, Nau e Traverso (2004)

Os três principais elementos que, tradicionalmente, fazem parte de um sistema de planejamento (GHALLAB; NAU; TRAVERSO, 2004):

- o Planejador que recebe como entrada a descrição de um sistema estado-transição Σ, uma situação inicial, e os objetivos para sintetizar um plano e fornecê-lo ao controlador;

- o Controlador recebe o estado atual do sistema real Σ através de observações e providencia a ação de acordo com o plano definido pelo Planejador.

- O Sistema estado-transição Σ (Sistema Σ) a ser controlado evolui de acordo com as ações e eventos por ele recebidos (um sistema estado-transição pode ser representado por um grafo direcionado onde os nós são estados e os arcos são ações ou eventos).

A diferença entre ações e eventos está em como o planejador pode controlá-las. Ações são transições que são controladas pelo executor do plano. Eventos são transições que são contingentes, isto é, ao invés de serem controladas pelos executores do plano, elas correspondem a uma dinâmica externa ao sistema estado-transição Σ (GHALLAB; NAU; TRAVERSO, 2004).

O Planejamento Automático aborda os diversos aspectos proposto no modelo conceitual, porém nem todos os conceitos são triviais de serem tratados. Por este motivo, por muito tempo esse modelo precisou ser restringido para que fosse possível desenvolver planejadores capazes de apresentar soluções satisfatórias. Tais limitações (simplificações e suposições no modelo conceitual) representavam um conjunto (ou classe) bem definido de problemas. Este conjunto de limitações caracteriza o chamado **Planejamento Clássico** (GHALLAB; NAU; TRAVERSO, 2004).

Essas simplificações e suposições são descritas a seguir (GHALLAB; NAU; TRAVERSO, 2004):

- A0: **Σ Finito**: O sistema estado-transição Σ possui um conjunto limitado de estados;

- A1: **Σ Totalmente Observável**: O sistema Σ é totalmente observável, isto é, tem-se total conhecimento do estado de Σ;

- A2: **Σ Determinístico**: O sistema Σ é determinístico;

- A3: **Σ Estático**: O sistema Σ é estático, isto é, o conjunto de Eventos exógenos E é vazio;

- A4: **Objetivos restritos**: Os planejadores lidam apenas com objetivos restritos que são especificados explicitamente no estado objetivo (*goal state*). Os objetivos estendidos, tais como, estados a serem evitados, restrições na trajetória da solução ou funções de otimização, não são permitidos;

- A5: **Planos Sequenciais**: Um plano-solução de um problema de planejamento é uma sequencia finita de ações linearmente ordenada;

- A6: **Tempo Implícito**: Ações e eventos não possuem duração. Elas são transições de estado instantâneas;

- A7. **Planejamento Offline**: O planejador não percebe alterações do sistema Σ enquanto está planejando. Ele planeja apenas com as condições iniciais e os objetivos, independentemente da dinâmica que ocorre em Σ. O planejador não recebe o status de execução.

As limitações e suposições tornaram viável o processo de busca por um caminho no grafo de estado, pois se trata de um problema bem resolvido na Ciências da Computação. Todavia, mesmo no domínio de planejamento mais simples, o grafo de Σ pode ser tão grande que a busca por uma solução ser extremamente complexa. Consequentemente, o uso de técnicas e heurísticas

de IA tornam menos onerosa a tarefa de busca por uma solução (GHALLAB; NAU; TRAVERSO, 2004).

2.2.1. Histórico

John McCarthy, um dos pioneiros e fundadores da IA, publicou o primeiro artigo que propõe o uso da lógica para representação de conhecimento em um computador (MCCARTHY, 1968). Neste artigo, McCarthy formulou o que é considerado o primeiro problema de planejamento da história (LIFSCHITZ *et al.*, 2000).

Um dos principais motivos do interesse pela área de planejamento é o fato dela combinar conceitos de duas grandes áreas da IA: busca e lógica. Dessa maneira, um planejador pode ser visto tanto como um algoritmo que busca por uma solução quanto um algoritmo que, de maneira construtiva, prova a existência (ou não) de uma solução (RUSSELL; NORVIG, 2003).

A área de planejamento autônomo prevê uma separação entre a descrição do problema de planejamento e o algoritmo utilizado para resolvê-lo: a representação e a resolução. A representação faz uso de linguagens padronizadas para descrição do problema, enquanto a resolução implementa algum tipo de algoritmo ou lógica computacional para resolver o problema descrito pela representação (CANTONI, 2010).

No final da década de cinquenta, um programa de computador denominado General Problem Solver (NEWELL; SHAW; SIMON, 1959) foi o responsável por introduzir o conceito de separação entre a representação e a resolução de problemas. A estrutura geral deste programa foi a base para o desenvolvimento do *Stanford Research Institute Problem Solver* (STRIPS), que implementa a

linguagem utilizada para representação do problema e o algoritmo que o resolve. (FIKES; NILSSON, 1971).

O planejamento autônomo toma como base três tipos de planejamento: planejamento clássico, planejamento hierárquico, *hierarchical task network* (HTN) e planejamento temporal e com recursos. O planejamento clássico: prevê um ambiente completamente observável, determinístico, estático, finito e discreto (CANTONI, 2010).

Planejamento hierárquico, (HTN) é baseado em decomposição hierárquica de tarefas. Este tipo de planejamento é uma generalização do planejamento clássico e inclui, além das ações regulares (ações primitivas), um conjunto de tarefas ou ações de alto nível (NAU *et al.*, 2001). Como no planejamento clássico, o estado do domínio é alterado pela execução das ações primitivas, entretanto, para alcançar estas ações é necessário decompor, de maneira recursiva, tarefas em subtarefas até que o planejador encontre as ações primitivas (CANTONI, 2010).

O planejamento temporal e com recursos: consideram as dimensões de tempo e recursos para modelar problemas do mundo real. Neste planejamento as ações podem assumir duração, o que permite explorar a concorrência e o paralelismo nas ações. Já os recursos são utilizados para expressar quantidades numéricas de um determinado item modelado. Por meio dessa característica é possível representar, por exemplo, a carga total da bateria de um satélite (CANTONI, 2010).

Diante de algumas limitações que STRIPS apresentava surgiram alguns planejadores como ABSTRIPS (SACERDOTI, 1974), NOAH (*Nets of Action Hierarchies*, um dos primeiros a trabalhar com conceito de planejamento hierárquico utilizando tarefas hierárquicas) (SACERDOTI, 1977) e NONLIN (TATE, 1977) almejando obter resultados mais satisfatórios.

A representação de domínios, principalmente a representação de ações, ganhou um grande avanço em 1989 com a introdução da linguagem ADL (PEDNAULT, 1989) (uma combinação das representações do STRIPS e Cálculo de Situações). Esta linguagem realizava uma extensão na descrição de ações, incorporando, por exemplo, efeitos condicionais (when..do, if..then) e quantificadores universais (forall) podendo assim descrever efeitos dependentes de contexto. Estas extensões permitiram uma maior flexibilidade e poder de descrição dos efeitos de uma ação em domínios já um pouco mais complexos (GHALLAB; NAU; TRAVERSO, 2004).

Em meados dos anos 90 surgiram novos processo através do qual os planejadores resolviam os problemas, e também por diversos estudos realizados sobre linguagens de representação e técnicas alternativas ao STRIPS. Destacam-se, os planejadores (GHALLAB; NAU; TRAVERSO, 2004).

- O-PLAN (CURRIE; TATE; BRIDGE, 1991);
- SNLP (MCALLESTER; ROSENBLITT, 1991);
- POP (BARRETT *et al.*, 1993);
- UCPOP (PENBERTHY; WELD, 1992).

A história do Planejamento Automático ganhou um grande estimulo em meados de 1995 quando Avrim Blum apresentou o planejador GRAPHPLAN(BLUM; FURST, 1995). Este tinha uma abordagem diferente para o método de extração de planos, pois era baseado em grafos.

A representação e modelagem dos domínios é um dos aspectos mais importantes em Planejamento Automático. Todos os sistemas/domínios de planejamento, e seus respectivos problemas, devem ser modelados e representados utilizando alguma linguagem para que sejam interpretados pelo planejador. O Planejamento Clássico possui três principais formas de

representar os domínios de planejamento, são elas: Teoria de Conjuntos, Variáveis de Estados e a mais utilizada representação, chamada Representação Clássica uma das mais utilizadas na área espacial (GHALLAB; NAU; TRAVERSO, 2004).

Com o objetivo de comparar os planejadores existentes e incentivar ainda mais a pesquisa nesta área da IA, em 1998 surge a linguagem de definição de domínios de planejamento chamada PDDL – *Planning Domain Definition Language,* utilizada para representação e modelagem de domínios (MCDERMOTT *et al.*, 1998).

Em 2003 surge o conceito de escalonamento (*Scheduling*), que ganha mais espaço com a evolução da PDDL para a versão 2.1, capaz de expressar ações com duração de tempo (FOX; LONG, 2003).

A Engenharia do Conhecimento estuda os princípios e métodos para o desenvolvimento e construção de Sistemas Baseados em Conhecimento (*Knowledge Based Systems - KBS*) (STUDER; BENJAMINS; FENSEL, 1998).

A Engenharia do Conhecimento para Planejamento Automático e também para Escalonamento estuda processos que envolvem a aquisição, modelagem, validação, verificação, manutenção dos modelos de domínios, bem como a seleção e integração de técnicas de planejamento (um planejador) para raciocinar sobre o modelo especificamente para sistemas de planejamento automático. Alguns dos principais objetivos desta área estão relacionados com a exploração de técnicas e métodos provenientes das áreas da engenharia de aquisição, de requisitos, do conhecimento (grande área) e de *software* de modo a criar modelos de domínios satisfatórios e integrá-las às técnicas de planejamento desenvolvidas (MCCLUSKEY *et al.*, 2000).

A seguir é apresentada uma breve descrição dos conceitos sobre a atividade de análise do domínio do problema.

2.3. Análise de domínio

A análise de domínio começou a ser considerada importante pela comunidade de engenharia de *software* a partir da necessidade de se diminuir o custo desproporcional da manutenção de *software*, resultante da introdução de mudanças arbitrárias (ARANGO, 1994).

Se considerarmos, por exemplo, o domínio aeroespacial, objetos típicos são voos, missões espaciais, base de lançamento, operadores, gerente de missão, engenheiros e satélites por exemplo. Operações e ações incluem escalonamento de missões, operação de satélites entre outros. Uma linguagem específica de domínio pode ser criada para representar estes objetos, operações e relações, que pode, posteriormente, ser utilizada para descrever outros sistemas neste domínio.

Intuitivamente, esta atividade pode ser considerada equivalente à análise de requisitos convencional, no entanto atuando em um meta-nível e, portanto, ao invés de explorar requisitos de uma aplicação específica, os requisitos explorados dizem respeito a uma família de aplicações de uma determinada área (ARANGO; PRIETO-DÍAZ, 1994).

2.3.1. Conceitos de análise de domínio

a) **Domínio do problema**: O domínio do problema representa um conjunto de itens de informação presentes em certo contexto do mundo real, inter-relacionados de forma bastante coesa, e que desperta o interesse de uma certa comunidade. Esta definição cobre duas perspectivas (ARANGO, 1994):

I. Domínio do problema como um conjunto de problemas relacionados, o que aproxima a análise de domínio da teoria de problemas;

II. Domínio do problema como uma taxonomia de componentes que torna explícita as partes comuns de aplicações presentes e futuras identificadas como similares.

b) Modelo do Domínio: Pode ser descrito como um sistema formal de termos, relações entre termos, regras de composição de termos, regras para raciocínio usando estes termos e regras para mapeamento de itens do domínio do problema para expressões neste modelo e vice-versa. O modelo do domínio define entidades, operações, eventos e relações que abstraem similaridades e regularidades em um determinado domínio, formando uma arquitetura de componentes comuns às aplicações analisadas e criando modelos que tornam possível identificar, explicar e prever fatos difíceis de serem observados diretamente. Depois de pronto, este modelo deve servir como uma fonte unificada de referência, quando ambiguidades surgirem em discussões sobre este domínio, e como um repositório de conhecimento comum, auxiliando de forma direta a comunicação, o aprendizado e reuso em um nível mais alto de abstração (ARANGO, 1994).

c) Análise e Modelagem do Domínio: Um conjunto de atividades cujo propósito é reduzir a complexidade da nossa percepção de um determinado domínio, impondo uma coerente organização aos dados adquiridos através de experimentos, busca de especialistas e engenharia reversa de sistemas existentes (ARANGO, 1994)

O sucesso do desenvolvimento de *software* com reutilização é obtido através da análise de domínio com seu foco abrangente de promover o reuso de itens mais abstratos do que somente código. Entre estes itens podem ser citados

como exemplos arquiteturas de *software*, soluções de projeto, modelos de requisitos e, idealmente, a representação do conhecimento (NEIGHBORS, 1980).

2.4. Planos operacionais

O produto de um planejador é um plano de, operação (GHALLAB; NAU; TRAVERSO, 2004) que por definição é uma sequencia finita de ações linearmente ordenada e temporizada.

Os planos operacionais tradicionalmente são baseados no estado inicial do sistema e no objetivo final ao qual se deseja levar o sistema, no entanto a natureza complexa dos sistemas reativos torna a busca pela solução bastante onerosa, dispensando um alto poder de processamento (KUCINSKIS, 2007).

Por este motivo é que tradicionalmente os planejadores modernos utilizam algoritmos e heurísticas não determinísticas, baseadas em IA, para que o resultado possa ser encontrado em menor tempo possível. (KUCINSKIS, 2007)

Desta forma os planejadores podem ser classificados em (MENEZES, 2010) (PINHEIRO, 2012):

- **Determinísticos:** aquele no qual para um mesmo conjunto de entrada, o resultado sempre será o mesmo conjunto de saída, podendo-se prever sem nenhum risco ou erro.

- **Não determinísticos:** aquele no qual para um mesmo conjunto de entrada, o resultado poderá ser um conjunto de saída diferente para cada execução, dependendo da heurística ou técnica utilizada para busca de resultados.

Como regra geral, ambos os tipos de planejadores entregam um plano que pode atingir o objetivo final, no entanto nos planejadores não determinísticos não há como prever o plano operacional gerado.

2.5. Considerações

Apresentou-se nesse capítulo, o estudo teórico realizado para o desenvolvimento da arquitetura de avaliação proposta neste trabalho. O estudo de planejamento automático possibilitou uma visão geral da área, além de permitir a identificação das características internas de domínios e problemas de planejamento tratadas na literatura. O conhecimento técnico e os conceitos de análise de domínio possibilitou formar uma boa base quanto a conceitos inerentes a modelagem e representação de conhecimento.

No próximo capítulo apresentam-se alguns trabalhos relacionados a avaliação de planos operacionais, que de alguma forma se assemelham com este trabalho.

3 O ESTADO DA ARTE EM AVALIAÇÃO DE PLANOS

Com o objetivo de analisar o atual cenário da área de avaliação de planos operacionais, neste capítulo são descritas brevemente as principais técnicas associadas ao processo de garantia da qualidade de planos.

3.1. Avaliação

O trabalho assume avaliação como sendo o processo de determinar a valia ou o valor de; estimar, a importância, o risco; fazer a apreciação; ajuizar, com base em critérios definidos.

O processo de ajuizar sobre um determinado plano nos permite definir padrões qualitativos no processo de avaliação. A seguir são apresentados alguns conceitos sobre qualidade de planos gerados por planejadores.

3.1.1. Qualidade dos planos

A geração de planos operacionais com qualidade é um elemento fundamental na transformação dos planejadores de ferramentas de pesquisa em aplicações do mundo real. No entanto, a grande maioria dos trabalhos realizados sobre planejamento autônomo tem tido como foco apresentar novas formas de aprendizados para melhorar a eficiência do planejamento (PÉREZ; CARBONELL, 1994).

Desta forma a maioria dos trabalhos sobre o aprendizado no contexto dos sistemas planejadores concentrou-se em dois tipos de objetivo de aprendizagem (PÉREZ, 1994):

- aprendizado orientado por metas do domínio, quando o conhecimento do domínio é incompleto ou impreciso;

- aprendizado orientado pela eficiência das metas do planejamento;

Uma grande quantidade de pesquisas na área de planejamento e aprendizado tem se dedicado a ciência do planejamento autônomo, conhecido também como "speedup learning" (MINTON *et al.*, 1989) (CHISTOPHER; ZUCKERMAN, 1998) (BENSON; NILSSON, 1996) (ESTLIN; MOONEY, 1996) (KAMBHAMPATI; KATUKAM; QU, 1996) (LUGER, 2004).

Em 1994 foi apresentado um novo objetivo de aprendizagem para melhorar a qualidade dos planos produzidos por planejadores. O domínio do planejador foi estendido para apoiar uma função de avaliação da qualidade para geração de planos. Como regra geral os planejadores passam a utilizar métricas como custo de execução do plano ou eficiência do plano quando executado, ao invés de métricas como a eficiência na busca da solução do plano (PÉREZ; CARBONELL, 1994).

Métricas de qualidade de planos (PÉREZ; CARBONELL, 1994):

- **Custo de execução**: Alguns fatores que afetam o custo de execução de um plano podem ser calculados pela soma de todas as etapas ou operadores no plano, que é definido como $C_{total} = \Sigma c_i$ onde C_{total} é igual ao custo total de execução do plano e c_i é igual ao custo para cada operador. O custo de cada operador c_i pode ser definido como: o tempo de execução, o custo do recurso usado passo a passo, ou simplesmente 1 se a medida de custo é simplesmente o comprimento do plano ou o numero total de ações.

- **Robustez do plano** ou habilidade em responder bem sob condições de mudanças ou incertezas.

- **Satisfação dos objetivos do usuário:** Outros fatores que captam a satisfação do cliente (por exemplo, a precisão do resultado, ou o conforto que ele proporciona para o usuário). Estes, em alguns casos, são difíceis de quantificar.

A busca por planos em domínios é computacionalmente muito caro. Simon (1981) introduziu a ideia de "satisfação" argumentando que um agente racional nem sempre têm os recursos para determinar que a ação é ótima, e em vez disso deve tentar apenas fazer o suficiente. (SIMON, 1996)

Utilizar as métricas de qualidade para conduzir os planejadores a produzirem planos com qualidade pode atender a condição de "satisfação" proposta por Simon (1981). (PÉREZ; CARBONELL, 1994).

Apesar dos trabalhos propostos apresentarem ganhos com relação a qualidade dos planos operacionais, eles são totalmente dependente do planejador, já que o aumento da qualidade está ligada ao processo de geração do plano (PÉREZ; CARBONELL, 1994) (ESTLIN; MOONEY, 1996), (FUENTETAJA; BORRAJO, 2006) (GIUNCHIGLIA; MARATEA, 2009).

3.2. Segurança no planejamento da operação de sistemas espaciais

Nos projetos espaciais desenvolvidos pelas principais agências internacionais têm se utilizado ferramentas baseadas em técnicas de inteligência artificial para o controle de operações de missão espacial. Essas ferramentas vêm sofrendo melhorias contínuas decorrentes da própria experiência ou como resultado de restrições impostas aos novos conceitos de planejamento de missão. (SOUZA, 2011).

O objetivo principal dessas ferramentas é a obtenção de um plano de execução que contemple as operações rotineiras e de contingências, um conjunto de

operações de rotina que busca balancear requisitos e recursos, escalonando um grande número de atividades de curto, médio e longo prazo (RABENAU; DENIS; JAYARAMAN, 2002) (RABENAU; PESCHKE, 2004).

De acordo com os órgãos de pesquisas das principais agências espaciais internacionais é inegável que o desenvolvimento de sistemas autônomos tem sido a chave para solucionar os problemas relacionados às aplicações que envolvem exploração espacial, tais como: as grandes distâncias, os limites de comunicação, bem como os altos custos de operação. No entanto, o risco e o custo dessas missões espaciais levam a certa resistência para empregar uma tecnologia nova, complexa e de difícil entendimento. Em função dessa resistência existe forte coesão dos grupos para que o processo de desenvolvimento demonstre prover a segurança e confiança exigidas para este tipo de aplicação (BLANQUART *et al.*, 2004).

Com crescente demanda em agregar mais segurança e confiabilidade nos sistemas espaciais, surge uma iniciativa da ESA em coordenar um projeto com o objetivo de pesquisar medidas para garantia da qualidade dos produtos de *software* que utilizem técnicas autônomas, o SPAAS (*Software Product Assurance for Autonomy on-board Spacecraft*) (SOUZA; FERREIRA; SILVA, 2007).

Esse projeto resultou de uma parceria entre os seguintes centros de pesquisa e tecnologia europeus: *European Aeronautic Defence and Space company* (EADS ASTRIUM), *Laboratoire d'Architecture et d'Analyse des Systèmes* (LAAS-CNRS), *European Space Research and Technology Centre* (ESTEC) e o *Société d'ingénierie experte en informatique industrielle technique et scientifique* (AXLOG Ingénierie) (BLANQUART *et al.*, 2004).

O objetivo do projeto consistiu em investigar as medidas para garantia do produto de *software* que utiliza técnicas avançadas em inteligência artificial na geração de autonomia.

Durante o desenvolvimento do projeto SPAAS, diversos métodos relacionados à segurança foram analisados, sendo recomendadas normas e padronizações. Os integrantes do projeto desenvolveram ainda componentes de *software*, um verificador de plausibilidade (*plausibility checker*) e filtro de segurança (*safety bag*), relacionados à segurança para tratamento de situações imprevistas provenientes desta maior autonomia nas operações de controle em veículos espaciais. As principais recomendações foram (BLANQUART *et al.*, 2004):

- Uso de testes exaustivos de simulação;
- Uso de técnicas de segurança monitoradas, tais como: um supervisor ou validador.

O componente de *software* desenvolvido no SPAAS para o tratamento de situações imprevistas, chamado de *plausibility checker* tem como função suportar e complementar a validação das ações de um *software* de solo, especialmente no controle dos procedimentos de bordo gerados de forma automática antes de serem enviados para execução real. A arquitetura geral e a forma como atua esse componente é apresentada na Figura 3.1 (BLANQUART *et al.*, 2004).

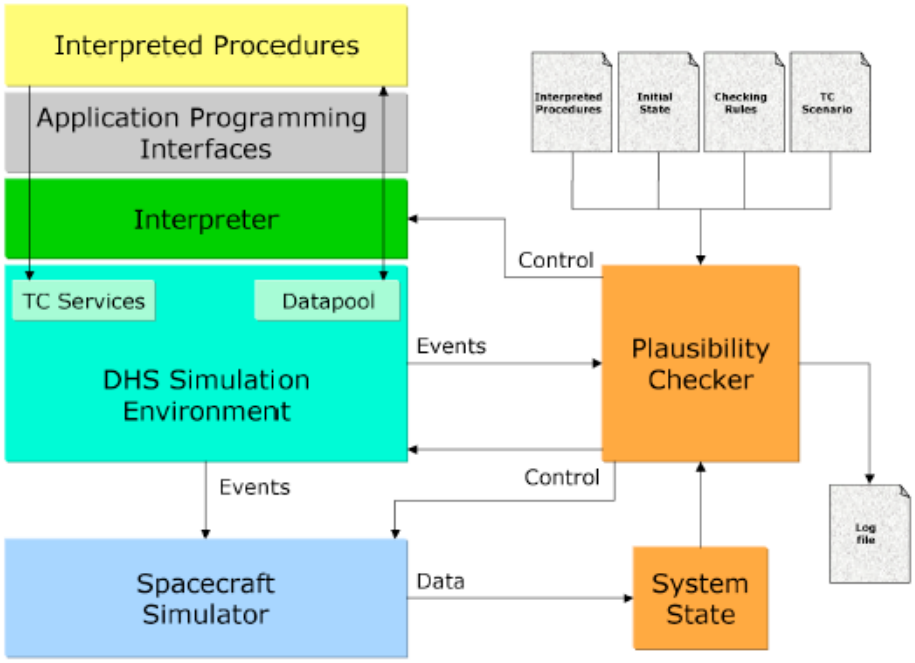

Figura 3.1 - Verificador de plausibilidade

Fonte: Blanquart *et al.* (2004)

O Verificador de plausibilidade controla os comandos a serem enviados para o simulador da espaçonave ou para o simulador *Data Handling System* (DHS), baseando-se nas informações recebidas do próprio simulador da espaçonave (*Spacecraft Simulator*) sobre o estado simulado da espaçonave. Para realizar o controle dos comandos, o componente Verificador de plausibilidade também recebe de arquivos de dados, as informações relativas às operações de solo, gerando um arquivo de registro de eventos.

O Filtro de segurança é um componente de *software* para monitorar em tempo real os estados e ações futuras do sistema. Ele é responsável por apoiar a decisão do computador de bordo em executar o plano de voo com base nas condições atuais. A arquitetura do filtro de segurança pode ser observada na Figura 3.2 (BLANQUART *et al.*, 2004).

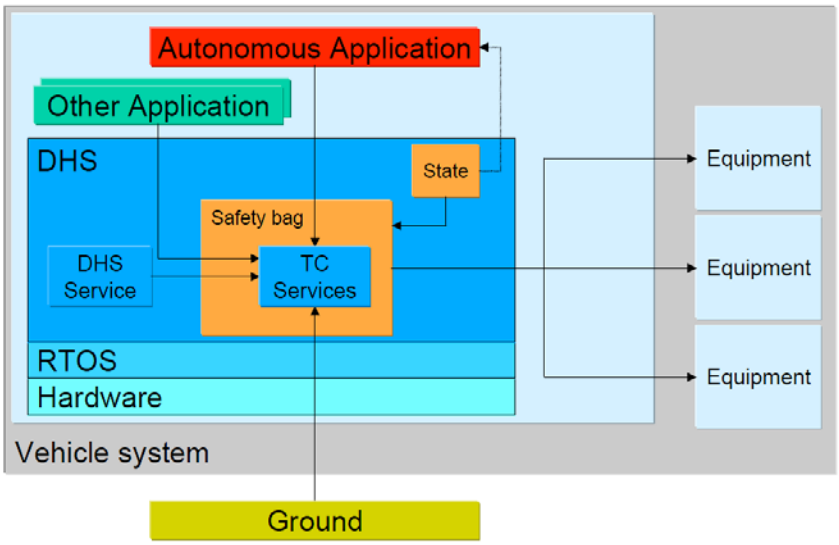

Figura 3.2 - Filtro de segurança

Fonte: Blanquart *et al.* (2004)

3.3. Uma estratégia para validação de planos operacionais

Em 2011 é proposta uma estratégia baseada em algoritmos de mineração de dados para validar plano de operação de voo a partir de predições de estados dos satélites do INPE (SOUZA, 2011). Trata-se de uma estratégia para validar plano de operação de voo de satélites, tendo como parte relevante uma ferramenta de *software* concebida com o propósito de gerar diagnóstico e realizar predições de estados operacionais do satélite, em função das ações contidas no plano, independente do planejador escolhido (SOUZA, 2011).

A estratégia de validação consiste em uma arquitetura composta por diversos componentes de *software* para verificação e validação de um plano de operação, antes da execução real ou durante uma operação real conforme pode ser observado na Figura 3.3 (SOUZA, 2011).

A Figura 3.3 ilustra dois módulos pontilhados dentro da estratégia, que indicam duas sequências distintas de execução (SOUZA, 2011):

- A primeira ocorre a partir de uma execução *off-line* do plano gerado pelo primeiro componente de *software*, o planejador de operações, cada ação do plano é executada e uma simulação do comportamento do satélite é realizada pelo segundo componente de *software*, um simulador de satélite. A saída fornecida pelo simulador consiste em um conjunto de parâmetros e telemetrias, que formam o estado operacional do satélite simulado, resultante da execução das ações do plano gerado pelo planejador de operações.

- Outra sequência de execução prevista para a estratégia acontece durante uma operação real do satélite. A partir de telemetrias recebidas pela estação terrena e enviadas como dados de entrada para a arquitetura de geração de diagnóstico, a ferramenta produz diagnóstico e predição dos estados do satélite por meio da classificação dessas telemetrias, o que desta forma indica a evolução do estado do satélite em função das ações do plano e sugere a aprovação ou rejeição do plano.

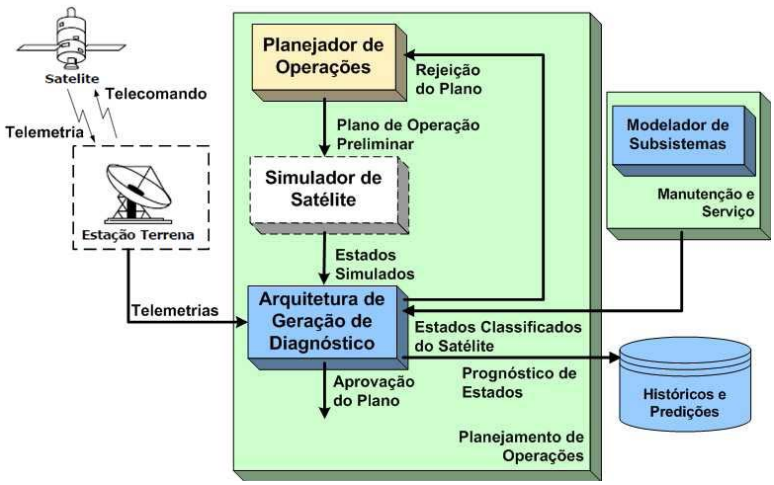

Figura 3.3 - Arquitetura de validação de plano de voo.
Fonte: Souza (2011)

Como parte relevante da estratégia de validação, o componente nomeado arquitetura de geração de diagnóstico, contém a ferramenta geradora de diagnóstico, concebida para fornecer predição de estados futuros do satélite a partir de parâmetros e telemetrias. Para isto são utilizadas técnicas de mineração de dados, implementadas por agentes que utilizam técnicas de IA, apresentadas na Figura 3.4 (SOUZA, 2011).

O trabalho propõe um componente de *software* capaz de analisar grande quantidade de dados fornecidos por um subsistema crítico para manutenção do satélite em órbita e de realizar diagnósticos e predições e sobre estados do satélite, utilizando para isto técnicas de mineração de dados (SOUZA, 2011).

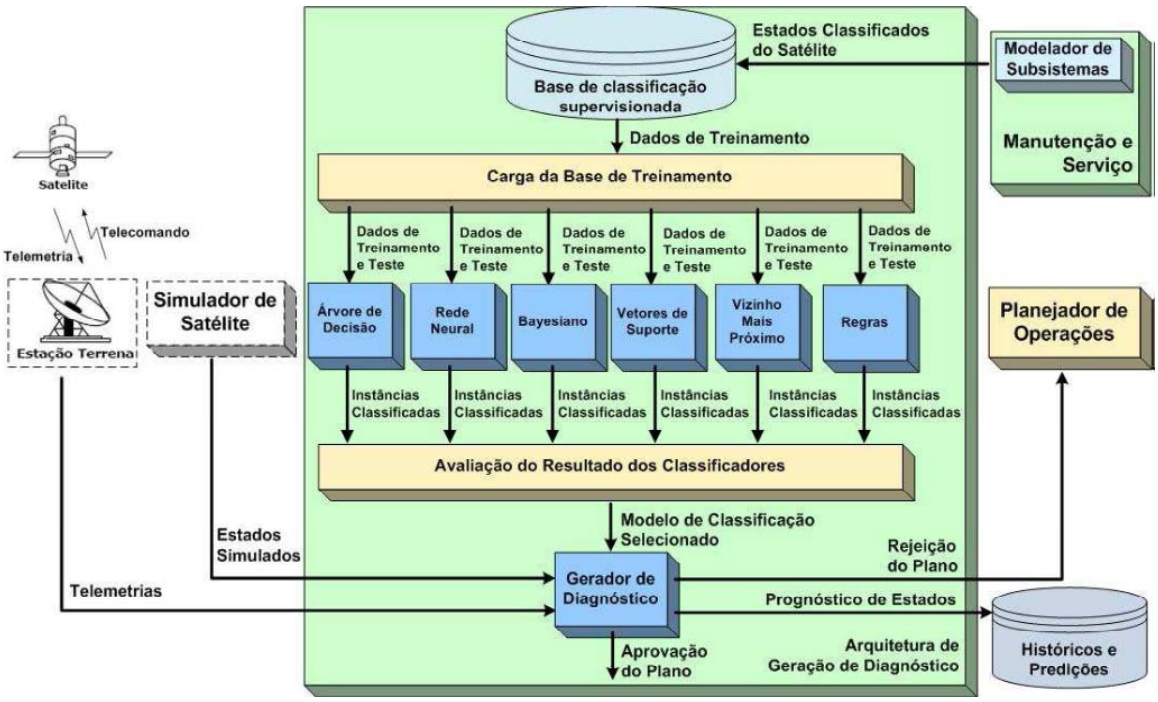

Figura 3.4 - Arquitetura para geração de diagnóstico.
Fonte: Souza (2011)

A arquitetura da Figura 3.4 apresenta os componentes de *software* e a sequência entre eles, que compõem as fases do processo de construção do modelo de classificação preditiva a ser selecionado. A definição das fases está

em conformidade com as fases do processo de descoberta de conhecimento em mineração de dados (SOUZA, 2011):

- fase de construção da base de classificação supervisionada;
- fase de construção dos modelos de classificação preditiva;
- fase de avaliação do resultado dos classificadores.

O objetivo da estratégia de validação é manter a integridade do sistema espacial, para isto, a concepção da arquitetura segue as recomendações das principais agências espaciais internacionais para segurança de sistemas autônomos baseados em planejamento usando Inteligência Artificial, o SPAAS (SOUZA, 2011).

3.4. Considerações

O trabalho SPAAS, apesar de possuir uma implementação, é uma metodologia teórica para a validação de sistemas autônomos em ambientes espaciais. A metodologia não define como deve ser feito e sim o que deve ser feito, independente se a avaliação é determinística ou não. A estratégia baseada em algoritmos de mineração de dados para validar plano de operação de voo a partir de predições de estados dos satélites do INPE (SOUZA, 2011) sugere a utilização de técnicas de IA, principalmente na fase de construção dos modelos de classificação preditiva, como redes neurais, para avaliação dos planos, tornando o processo não determinístico.

Por outro lado, o objetivo do presente trabalho é apresentar uma arquitetura para avaliação de planos de forma determinística, independente de planejador, através de Modelos de Estados, por este motivo o próximo capítulo apresenta uma revisão sobre este tema.

4 REPRESENTAÇÃO DE MODELOS DE ESTADO

Os conceitos de modelos de estados são apresentados em detalhes nesta seção por se tratar do modelo alvo dos estudos realizados durante o desenvolvimento deste trabalho.

Alguns sistemas podem ser descritos baseados em certas entradas e suas respectivas saídas resultantes. São os chamados sistemas reativos, os quais exigem uma análise dessas relações de entrada e saída ao longo do tempo. Sistemas reativos são sistemas que reagem continuamente a estímulos internos e externos. Alguns exemplos de sistemas reativos: telefones, automóveis, redes de comunicação, semáforos, sistemas embarcados de tempo real, sistemas aviônicos, etc. Devido às suas características especiais de responder aos estímulos, os sistemas reativos necessitam de técnicas para representá-los. Entre algumas técnicas, podem ser mencionadas: diagramas de estado ou máquina finita de Estados (MEF), redes de filas, redes de Petri e Statecharts (AMARAL; VIJAYKUMAR; MARTINS, 2003).

As diversas técnicas de modelagem encontradas na literatura diferem entre si pelo grau de formalismo proposto e mecanismos de modelagem disponibilizados. Cabe ao projetista, decidir qual o melhor modelo para representar o comportamento do sistema reativo. Cada uma delas possui vantagens e “deficiências”, quando analisadas sob a ótica do desempenho. Algumas dessas peculiaridades são discutidas a seguir. (PINHEIRO, 2012)

As ***redes de filas*** possuem uma base matemática bastante sólida, contudo sua representação gráfica oferece apenas os elementos fila e servidor, o que em muitos casos pode ser o suficiente. (FRANCÊS *et al.*, 2001)

Entretanto, em alguns sistemas reativos, desejam-se representar situações que não constituem necessariamente uma fila ou um servidor. Por exemplo, um processo em um computador pode se encontrar em três possíveis situações:

Processando, Bloqueado ou Pronto. Essas situações são estados abstratos que são facilmente representados em técnicas como redes de Petri e Statecharts, mas não são tratadas apropriadamente em redes de filas. (FRANCÊS *et al.*, 2001)

Redes de Petri possuem características bastante interessantes em sua representação. Uma das mais interessantes é a possibilidade da individualização de clientes, recurso que não é usual na maioria das técnicas. (FRANCÊS *et al.*, 2001)

Há situações em que essa característica é primordial, por exemplo, na migração de processos para balanceamento de carga entre máquinas, onde se pode desejar migrar exatamente um determinado processo, e por isso deve-se saber exatamente onde ele se encontra. Entretanto, há certa dificuldade na representação de processos paralelos (FRANCÊS *et al.*, 2001), mesmo utilizando-se da rede distribuição (rede elementar para representar criação de processos paralelos). Esse aspecto piora à medida que o modelo cresce, o que aliás é outro aspecto a ser ponderado.

A representação original de Redes de Petri, por não possuir nenhum mecanismo de hierarquia, tende a fazer com que os modelos cresçam substancialmente, quando a complexidade dos mesmos aumenta. Há algumas extensões que tentam minimizar esse efeito, como as RP Hierárquicas, que são baseadas em um elemento denominado superpágina. (MACIEL; LINS; CUNHA, 1996)

O "inconveniente" dessa extensão é que as superpáginas são caixas-pretas, ou seja, são retângulos que escondem uma complexidade que, em certos casos, é necessária para a compreensão do modelo. Além disso, algumas extensões desfiguram a notação original, o que, às vezes, parece ser outra técnica à parte, e não uma derivação das redes de Petri. (FRANCÊS *et al.*, 2001)

Como os modelos apresentados são extensões de MEFs, a seguir, é apresentada resumidamente a definição de MEF.

4.1. Máquinas de estados Finitos - MEF

As MEFs são máquinas abstratas que capturam as partes essenciais de máquinas concretas, que vão desde máquinas de vender jornais ou refrigerantes, passando por relógios digitais, elevadores, programas de computador e até mesmo o próprio computador digital (MENEZES, 2010).

Dentre as principais vantagens de MEFs estão: a sólida base teórica, o poder de expressividade e a existência de inúmeros algoritmos de interação com as máquinas. (PINHEIRO, 2012)

Uma MEF é uma representação de uma máquina composta por estados e eventos. Uma transição é caracterizada por dois eventos: um de entrada e um de saída; um estado origem e um estado destino. A máquina pode estar em apenas um estado por vez. Ao ocorrer um evento de entrada, a máquina pode responder com um evento de saída e uma transição para outro estado. (PINHEIRO, 2012)

A representação de uma MEF pode ser feita por um diagrama de estados, em que círculos representam os estados e arcos direcionados, as transições (PINHEIRO, 2012). Na Figura 4.1, pode-se observar um exemplo de uma MEF com quatro estados e oito transições, onde "q0" é o estado inicial.

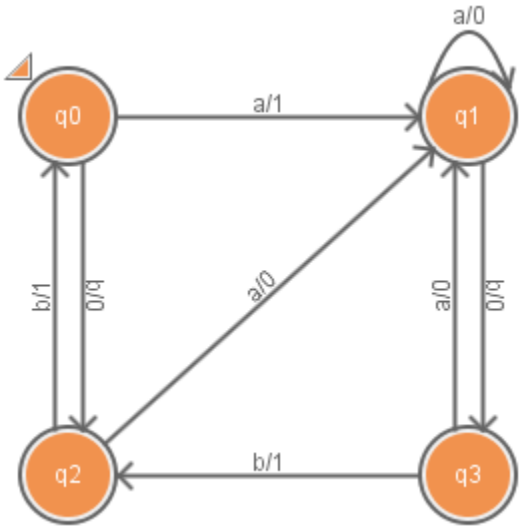

Figura 4.1 - Exemplo de MEF

Fonte: Adaptado de Pinheiro (2012)

Outra representação é a tabela de transição, em que os estados são representados por linhas e as entradas por colunas, como mostrado na Tabela 4.1.

Tabela 4.1 - Tabela de transição da MEF Figura 4.1

	Destino		Saída	
Entrada / Estados	a	b	a	b
q0	q1	q2	1	0
q1	q1	q3	0	0
q2	q1	q0	0	1
q3	q1	q2	0	1

4.1.1. Formalismo

Existem dois tipos de MEFs, Mealy e Moore: na máquina de Mealy, os eventos de saída estão associados às transições, ao ocorrer um evento de entrada, o

evento de saída ocorre durante a mesma transição. Na máquina de Moore, os eventos de saída estão associados aos estados; sendo assim, o evento de saída ocorre no seu estado destino (MENEZES, 2010).

Segundo Menezes (2010) a representação de uma MEF é definida como 7-upla: <I, Q, Z, Q0, F, O, Y>.

- I: alfabeto de entrada ou símbolos de entrada;
- Q: conjunto dos estados da máquina;
- Z: função de transição: Z: Q x I $\rightarrow$ Q; Mapeia um estado e uma entrada para o próximo estado;
- Q0: estado inicial da máquina, que deve ser um elemento do conjunto Q;
- F: conjunto de estados finais, que devem ser elementos do conjunto Q;
- O: alfabeto de saída;
- Y: função de saída:
    - para a representação de Mealy: Y: Q x I $\rightarrow$ O; Mapeia um estado e uma entrada para a saída;
    - para a representação de Moore: Y: Q $\rightarrow$ O; Mapeia um estado para a saída;

Considerando a MEF da Figura 4.1, tem-se para o estado "q0" as funções de saída, Y(q0,a) = 1 e Y(q0,b) = 0, e as funções de transição, Z(q0,a) = q1 e Z(q0,b) = q2. Pode-se estender essas notações para sequências de entrada aplicada a partir de um estado. Por exemplo, para o estado "q0", dada a sequência "abba", temos Y(q0,abba)=1010 e Z(q0,abba)=q1.

A representação formal para a MEF apresentada na Figura 4.1 pode ser definida como:

- alfabeto de entrada I = {a, b};
- conjunto de estados Q = {q0, q1, q2, q3};
- estado inicial Q0 = {q0}
- funções de transição:
    - Z(q0,a) = q1;
    - Z(q0,b) = q2;
    - Z(q1,a) = q1;
    - Z(q1,b) = q3;
    - Z(q2,a) = q1;
    - Z(q2,b) = q0;
    - Z(q3,a) = q1;
    - Z(q3,b) = q2;
- conjunto de estados finais F = {};
- alfabeto de saída O = {0, 1};
- funções de saída:
    - Y(q0,a) = 1;
    - Y(q0,b) = 0;
    - Y(q1,a) = 0;
    - Y(q1,b) = 0;
    - Y(q2,a) = 0;
    - Y(q2,b) = 1;
    - Y(q3,a) = 0;
    - Y(q3,b) = 1;

4.1.2. Propriedades de MEFs

As MEFs possuem propriedades importantes quanto sua estrutura, como (MENEZES, 2010):

- **Completude:** uma MEF é dita complemente especificada, ou completa, se ela trata todas as entradas pertencentes ao domínio de entrada (I) em todos os estados (Q). Caso contrário, a MEF é dita parcial;

- **Conectividade**: uma MEF é fortemente conexa se para cada par de estados (q_i, q_j) existe uma sequência de entrada que executa um caminho de transições com origem em q_i e destino a q_j. Se a partir do estado inicial for possível atingir todos os demais estados a MEF é inicialmente conexa;

- **Determinismo:** uma MEF é dita determinística quando há somente uma transição para cada par evento-estado;

- **Equivalência:** dois estados são equivalentes quando não houver sequência de entrada que gere saídas diferentes quando executadas a partir dos respectivos estados.

- **Reduzida ou mínima:** a MEF é reduzida ou mínima se não existem estados equivalentes.

A Figura 4.1 é um exemplo de MEF completa, reduzida, fortemente conexa e determinística.

Para a arquitetura proposta a propriedade mais importante é a determinística, que define que para cada par evento-estado está associada a somente uma transição, o que é assumido pelo processo de representação do domínio deste trabalho.

4.2. *Statecharts*

O principal problema na representação de sistemas reativos consiste na dificuldade de descrever o seu comportamento de forma clara, concisa e livre ambiguidade, uma vez que seu comportamento é dirigido por eventos complexos e inter-relacionados. Em alguns casos, o uso de MEFs se torna inviável, por conta de que pequenas variações no comportamento acarretam no crescimento exponencial dos estados (HAREL, 1987).

A técnica de especificação denominada *Statecharts* proposto por Harel (1987) permitiu uma representação clara e explícita de hierarquia, concorrência e interdependência entre componentes do sistema. (SILVA *et al.*, 2008)

Esta notação é uma extensão do diagrama tradicional de MEFs, ao qual foram adicionadas as características de: hierarquia de estados (profundidade), ortogonalidade (representação de atividades paralelas) e interdependência entre estados (mecanismos de comunicação) (HAREL, 1987).

Os elementos fundamentais para se especificar um sistema reativo em *Statecharts* são: Estados, Eventos, Condições, Ações, Expressões, Variáveis e Transições (SILVA *et al.*, 2008).

Os estados descrevem componentes (e suas possíveis situações) de um determinado sistema. Os estados podem ser classificados em dois grupos: básicos e não básicos (SILVA *et al.*, 2008).

Os estados básicos são aqueles que não possuem subestados, que não possuem refinamento. Os estados não básicos são decompostos em subestados, classificados em dois tipos: XOR ou AND (HAREL, 1987).

A decomposição do tipo XOR, não permite que um estado não básico assuma mais de um subestado simultaneamente. Entretanto, a decomposição do tipo

AND, permite que o estado assuma mais de um subestado simultaneamente (HAREL, 1987).

Os eventos, função de transição em MEFs, são elementos que interferem no estado atual definindo a dinâmica do sistema. Opcionalmente, o evento pode receber uma condição (condição de guarda), que ao ser satisfeita executa a transição ao qual o evento está associado (HAREL, 1987).

Os eventos podem ser classificados em internos e externos. Os eventos internos são gerados pelo próprio sistema, sem a necessidade de um estímulo externo e são detectados automaticamente graças a graças à capacidade de *broadcasting* implícita ao formalismo *Statecharts*. Os eventos externos são aqueles gerados fora das fronteiras do sistema, e normalmente ocorrem por estímulo externo (SILVA *et al.*, 2008).

As ações, função de saída em MEFs, ocorrem dentro do sistema a partir da execução de um evento, e podem representar a mudança de uma expressão ou de uma variável (SILVA *et al.*, 2008).

Assim como em MEFs as representações gráficas em arco é que denotam a mudança de um estado para outro. Os rótulos nos arcos de transição definem a notação: evento[condição]/ação. (SILVA *et al.*, 2008). A Figura 4.2 apresenta um diagrama simples representado por *Statecharts*.

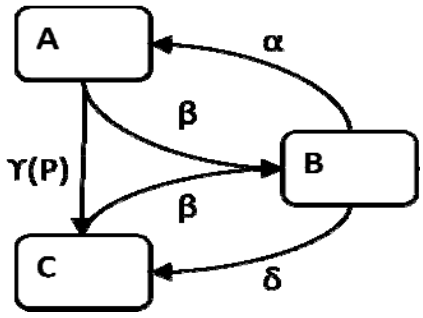

Figura 4.2 - Exemplo de *Statecharts*
Fonte: Harel (1987)

A técnica de especificação Statecharts pode ser considerada uma Máquina de Estados Finitos Estendidas (MEFE) que suporta estrutura hierárquica e concorrente de estados e um mecanismo de comunicação entre componentes através de eventos. (AMARAL; VIJAYKUMAR; MARTINS, 2003)

4.3. Considerações

A representação por modelos de estados têm sido amplamente empregada para a modelagem de sistemas reativos, variando de protocolos simples até complexos sistemas embarcados (LAI; LEUNG, 1995).

Estas técnicas possuem a característica de definir, precisamente, a ordem cronológica de suas interações. Isso explica o fato de Modelos de estados estarem sendo frequentemente utilizadas na especificação de sistemas reativos (AMARAL; VIJAYKUMAR; MARTINS, 2003).

Como já foi mencionado no capitulo 1, na seção Metodologia de pesquisa e organização da dissertação, a escolha por MEFs é por conta de que as técnicas como Redes de Petri e Teoria dos Conjuntos apresentam certa dificuldade na representação de paralelismo e sincronização entre componentes, (FRANCÊS *et al.*, 2001) o que pode ser claramente representado em Statecharts, que pode ser considerada uma extensão de Máquinas de Estados Finitas Estendidas (MEFE). (SILVA *et al.*, 2008)

A característica de paralelismo e sincronização é bastante importante para representação de sistemas reativos, o que justifica a escolha pela técnica de Modelos de estados e StateCharts, além da disposição de grande quantidade de literatura e de possuir um formalismo matemático que apresenta dentre outras propriedades a de determinismo.

O próximo capítulo apresenta a arquitetura proposta para a avaliação de planos operacionais, tomando como base o uso de MEFs e as características de *Statecharts*.

5 ARQUITETURA PROPOSTA

Este capítulo apresenta a arquitetura para uma ferramenta de avaliação determinística dos planos gerados por planejadores autônomos ou mesmo por operadores humanos.

A Figura 5.1 apresenta um diagrama da arquitetura proposta, que é divida em 3 partes: Entradas, Simulação, Saída.

Os artefatos de entrada são os planos operacionais que serão avaliados e as variáreis que descrevem a situação atual do sistema real. Esses parâmetros são enviados para um simulador capaz de interpretar o plano e simular os estados e transições de uma máquina de estados finitos (MEF).

O Domínio do Sistema é uma representação do sistema real ao qual o plano será executado. O domínio deve ser capaz de expressar todos os comportamentos do sistema e as interações com o ambiente ao qual ele se insere.

O plano consta de uma sequência de ações/eventos que deve ser executada para que um objetivo possa ser atingido dentro de um sistema. Os planos podem ser gerados por pessoas como gerentes de missão ou por sistemas planejadores autônomos, que fazem o uso de IA.

Ao executar a simulação, as ações são enviadas sequencialmente para o Domínio do Sistema, estimulando a transição entre os estados, possibilitando avaliar a situação final do sistema.

O Domínio do Sistema é descrito através de um Domínio de Estados e Transições denominado DET e um Domínio Lógico o DL que serão descritos com mais detalhes nas seções 5.4.1 e 5.4.2 respectivamente.

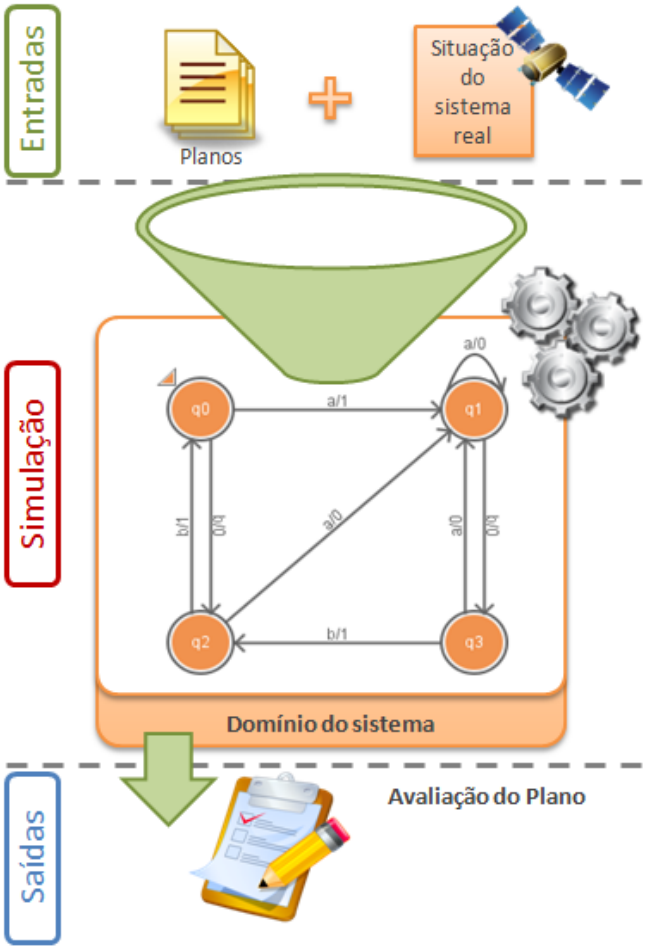

Figura 5.1 - Diagrama da arquitetura proposta

5.1. Requisitos funcionais para a ferramenta de avaliação

A seguir são apresentados os principais requisitos funcionais para a ferramenta de avaliação de planos.

RF1. A ferramenta deve permitir, de forma visual, a modelagem/desenvolvimento do Modelo de estados (Máquina de Mealy, Máquina de Moore e StateCharts), o Domínio de Estados e Transições;

RF2. A ferramenta deve permitir uma descrição comportamental nos estados e transições, o Domínio Lógico;

RF3. A ferramenta deve permitir a declaração de variáveis ambientais nos estados e transições;

RF4. A ferramenta deve permitir salvar/exportar o Modelo de Estados;

RF5. A ferramenta deve permitir abrir um Modelo de Estados salvo;

RF6. A ferramenta deve permitir a simulação de eventos que geram a mudança de estado no Modelo de Estados;

RF7. A ferramenta deve permitir que um plano em formato XML possa ser enviado para a simulação;

RF8. A ferramenta deve permitir que as variáveis possam ser alteradas e lidas através de comandos externos ao ambiente da ferramenta ou através do plano;

RF9. A ferramenta deve permitir a simulação interativa, através de um "*debugger*";

RF10. A ferramenta deve permitir o desenvolvimento ou configuração de um relatório com regras para avaliar a simulação do plano;

5.2. Requisitos não funcionais para a ferramenta de avaliação

RNF1. A ferramenta deve permitir que o simulador possa ser utilizado como uma biblioteca, para possibilitar a integração com outros sistemas, como por exemplo *softwares* planejadores e até mesmo sistemas espaciais como um satélite.

5.3. Casos de uso para a ferramenta de avaliação

A seguir são apresentados os principais casos de uso para a ferramenta de avaliação de planos.

5.3.1. Atores

Usuário-1: São os operadores e, gerentes de missão e engenheiros. Em geral pessoas que tem grande conhecimento do sistema que será avaliado pela ferramenta.

Sistema Externo: São sistemas que podem utilizar a ferramenta como uma biblioteca para avaliação das ações. A ferramenta pode servir de apoio a *softwares* planejadores ou até mesmo em temo real em um sistema espacial. Neste caso tanto o *software* planejador como o sistema espacial são Sistemas Externos.

5.3.2. Identificação dos casos de uso

A seguir são descritos os casos de uso, conforme Figura 5.2.

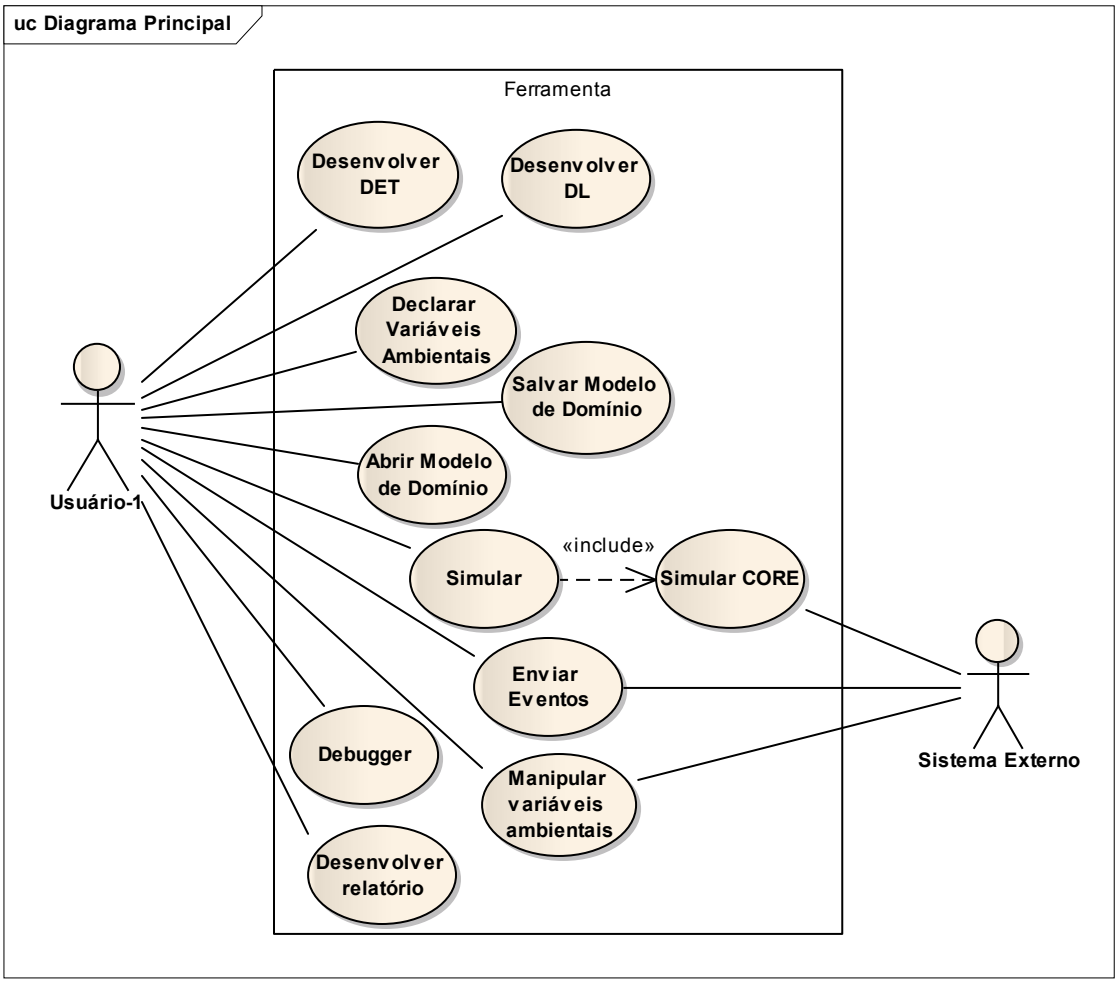

Figura 5.2 - Diagrama de casos de uso - Modelo principal da ferramenta

UC1. Desenvolver DET: Este caso de uso especifica a funcionalidade que permite o desenvolvimento do Domínio de Estados e Transições, o DET. Esta funcionalidade deve permitir que o usuário modele um determinado problema em modelos de estados, como Máquina de Mealy, Máquina de Moore e StateCharts. Maiores detalhes sobre o DET serão apresentados na seção 5.4.1;

Atores: Usuário-1;

Pré-condições: Não definida;

Pós-condições: Ter um Modelo de Domínio construído;

Requisitos funcionais: RF1;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Abrir a ferramenta;
2. Selecionar os itens que serão incluídos no modelo, sejam estados ou transições;
3. Caso de uso encerrado;

UC2. Desenvolver DL: Este caso de uso especifica a funcionalidade que permite o desenvolvimento do Domínio Lógico, o DL. Esta funcionalidade deve permitir que o usuário descreva comportamento no modelo de estados. Tanto no Estado quanto na transição deve haver uma AÇÃO de entrada e uma de saída. Nestas ações serão descritos os comportamentos. Estes comportamentos devem ser representados por uma linguagem de programação de alto nível, como, por exemplo, JAVA. Quando um evento externo for gerado, haverá uma mudança de Estado no DET. Nessa mudança de estado serão geradas as ações de entrada e saída das transições e dos estados envolvidos. Maiores detalhes sobre o DL serão apresentados na seção 5.4.2;

Atores: Usuário-1;

Pré-condições: Ter uma DET desenvolvida;

Pós-condições: Ter um Modelo de Domínio com o Domínio Lógico construído;

Requisitos funcionais: RF2;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Selecionar um evento ou transição;
2. Selecionar a ação de entrada ou de saída;
3. Descrever o comportamento;
4. Caso de uso encerrado;

UC3. Declarar Variáveis Ambientais: Este caso de uso especifica a funcionalidade que permite a declaração de variáveis ambientais no Domínio Lógico, o DL. Esta funcionalidade deve permitir que o usuário declare variáveis para serem utilizadas no Domínio Lógico. O Domínio Lógico irá operar sobre essas variáveis, utilizando para controle lógico e definição de valores. Essas variáveis devem apoiar na modelagem de sistemas que possuem variáveis continuas como, por exemplo, Temperatura ou a quantidade de combustível. Maiores detalhes sobre essas variáveis serão apresentados na seção 5.4.2;

Atores: Usuário-1;

Pré-condições: Ter uma DET e um DL desenvolvida;

Pós-condições: Ter um Modelo de Domínio com o Domínio Lógico construído;

Requisitos funcionais: RF3;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Selecionar um evento ou transição;
2. Selecionar a ação de entrada ou de saída;
3. Declarar a variável ambiental;
4. Caso de uso encerrado;

UC4. Salvar Modelo de Domínio: Este caso de uso especifica a funcionalidade que permite que o usuário possa salvar o Modelo de Domínio em arquivo para que posteriormente possa ser utilizado novamente;

Atores: Usuário-1;

Pré-condições: Ter um Modelo de Domínio Desenvolvido;

Pós-condições: Ter um arquivo no sistema de arquivos que represente o Modelo de Domínio;

Requisitos funcionais: RF4;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Acionar a opção de Salvar;
2. Sistema gera um arquivo no sistema de arquivos;
3. Caso de uso encerrado;

UC5. Exportar Modelo de Domínio: Este caso de uso especifica a funcionalidade que permite que o usuário possa exportar o Modelo de Domínio em arquivo para que posteriormente possa ser utilizado em simulação através de uma biblioteca específica para simulação. A exportação deve gerar um arquivo utilizado e binário para reduzir o tamanho físico do arquivo;

Atores: Usuário-1;

Pré-condições: Ter um Modelo de Domínio Desenvolvido;

Pós-condições: Ter um arquivo no sistema de arquivos que represente o Modelo de Domínio para simulação;

Requisitos funcionais: RF4;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Acionar a opção de Exportar;
2. Sistema gera um arquivo no sistema de arquivos;
3. Caso de uso encerrado;

UC6. Abrir Modelo de Domínio: Este caso de uso especifica a funcionalidade que permite que o usuário possa abrir um arquivo do Modelo de Domínio armazenado em disco.

Atores: Usuário-1;

Pré-condições: Ter um arquivo que represente o Modelo de Domínio no armazenado em disco;

Pós-condições: Ter o Modelo de Domínio aberto na ferramenta e disponível para uso;

Requisitos funcionais: RF5;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Acionar a opção de Abrir;
2. Sistema exibe o Modelo de domínio na ferramenta
3. Caso de uso encerrado;

UC7. Simular CORE: Este caso de uso especifica a funcionalidade que permite que o usuário possa simular a execução de eventos no Modelo de Domínio. Este caso de uso deve estar disponível em uma biblioteca que pode ser utilizada através de linha de comando, ou agregada a outro projeto, como, por exemplo, a ferramenta de modelagem que está sendo proposta. Ao executar esse caso de uso a ferramenta deve simular as Transições entre os Estados, que disparam as ações desenvolvidas no Domínio Lógico.

Atores: Usuário-1 e Sistema Externo;

Pré-condições: Ter um arquivo de simulação que represente o Modelo de Domínio no armazenado em disco; Ter um plano desenvolvido. Maiores detalhes do plano podem ser encontrados na seção 5.4.3.

Pós-condições: Ter o resultado da simulação;

Requisitos funcionais: RF6 e RF7;

Requisitos não funcionais: RNF1;

Fluxo de execução:

1. Executar a função da biblioteca, passando como parâmetro o plano e o Domínio Lógico preparado para simulação. Este arquivo pode ser obtido através do UC5;
2. Sistema executa a simulação;
3. Sistema apresenta resultado da simulação
4. Caso de uso encerrado;

UC8. Simular: Este caso de uso especifica a funcionalidade que permite que o usuário possa simular a execução de eventos no Modelo de Domínio. Este caso de uso inclui na execução o UC7. Este caso de uso deve estar disponível apenas na ferramenta de modelagem que está sendo proposta.

Atores: Usuário-1;

Pré-condições: Ter um Modelo de Domínio desenvolvido; Ter um plano desenvolvido. Maiores detalhes do plano podem ser encontrados na seção 5.4.3.

Pós-condições: Ter o resultado da simulação;

Requisitos funcionais: RF6 e RF7;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Selecionar a opção de simulação na tela principal;
2. Sistema executa a simulação;
3. Sistema apresenta resultado da simulação
4. Caso de uso encerrado;

UC9. Enviar Eventos: Este caso de uso especifica a funcionalidade que permite que o usuário possa enviar eventos durante a simulação de Modelo de Domínio. Esta funcionalidade simula a execução de um plano, ou seja, a sequencia de eventos enviada pode simular um plano de operações. Essa funcionalidade permite que a ferramenta possa ser utilizada na avaliação em temo real.

Atores: Usuário-1 e Sistema Externo;

Pré-condições: Ter um Modelo de Domínio desenvolvido e estar executando uma simulação;

Pós-condições: Ter o resultado da simulação;

Requisitos funcionais: RF6;

Requisitos não funcionais: RNF1;

Fluxo de execução:

1. Enviar o evento;

2. Sistema executa a simulação;

3. Sistema deve aguardar novamente o passo 1.

4. Caso de uso encerrado;

UC10. Manipular variáveis ambientais: Este caso de uso especifica a funcionalidade que permite que o usuário possa alterar os valores das varáveis ambientais. A ferramenta deve possuir uma opção para que o usuário possa ler e escrever nas variáveis declaradas no Domínio Lógico.

Atores: Usuário-1 e Sistema Externo;

Pré-condições: Ter um Modelo de Domínio desenvolvido;

Pós-condições: Ter o valor da variável ambiental manipulado;

Requisitos funcionais: RF8;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Selecionar a opção para manipulação de variáveis;

2. Selecionar a variável que deseja manipular;

3. Executar a manipulação;

4. Caso de uso encerrado;

UC11. Debugger: Este caso de uso especifica a funcionalidade que permite que o usuário possa executar a simulação de forma interativa. Este funcionalidade deve permitir que o usuário possa acompanhar passo a passo cada um dos eventos enviados.

Atores: Usuário-1;

Pré-condições: Ter um Modelo de Domínio desenvolvido;

Pós-condições: Ter a simulação interativa sendo executada;

Requisitos funcionais: RF9

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Selecionar a de execução interativa;
2. Selecionar a opção de enviar eventos;
3. Executar o passo 2 até que chegue ao final da simulação;
4. Caso de uso encerrado;

UC12. Desenvolver relatório: Este caso de uso especifica a funcionalidade que permite o desenvolvimento do relatório de avaliação da simulação. Este relatório deve ser desenvolvido usando o Domínio Lógico, o DL. Esta funcionalidade deve permitir que o usuário descreva comportamento e as regras deliberativas para a avaliação do Modelo de Domínio. O projeto possui uma ação de saída que é executada sempre que uma simulação termina. Assim como no UC2, estes relatório deve ser desenvolvido por uma linguagem de programação de alto nível,

como por exemplo JAVA. Quando a simulação for concluída, a ferramenta irá executar a ação de saída do projeto e por consequência irá executar o código do relatório que estiver descrito nesta ação.

Atores: Usuário-1;

Pré-condições: Ter um Modelo de Domínio desenvolvido;

Pós-condições: Ter o relatório da simulação para avaliação;

Requisitos funcionais: RF10;

Requisitos não funcionais: Não definido;

Fluxo de execução:

1. Selecionar a ação de saída do projeto;
2. Descrever o relatório;
3. Caso de uso encerrado;

5.4. Especificação da arquitetura

Nesta seção serão apresentados os detalhes mais relevantes sobre a arquitetura da ferramenta proposta.

A implementação do simulador e os padrões dos parâmetros de entradas são apresentados em maiores detalhes no capítulo 6.

5.4.1. DET - Domínio de Estados e Transições

Os planos operacionais devem ser executados dentro do Modelo DET, que é representado através de MEF. Cada comando gera um evento para MEF, que executa a transição e muda de estado. Essa dinâmica deve representar o comportamento do sistema real, ao qual o plano será executado. Um mesmo DET pode ser usado para avaliação de vários planos.

A Figura 5.3 apresenta o exemplo didático de um DET para uma máquina de refrigerantes. No exemplo são representados os estados que a máquina pode assumir. Para o funcionamento básico da máquina são necessárias duas moedas de R$ 0,50 centavos, somando um total de R$ 1,00, para que um refrigerante possa ser liberado. O usuário pode escolher o sabor, "*Flavor pressed*", ou pedir o dinheiro de volta, "*Coin return pressed*".

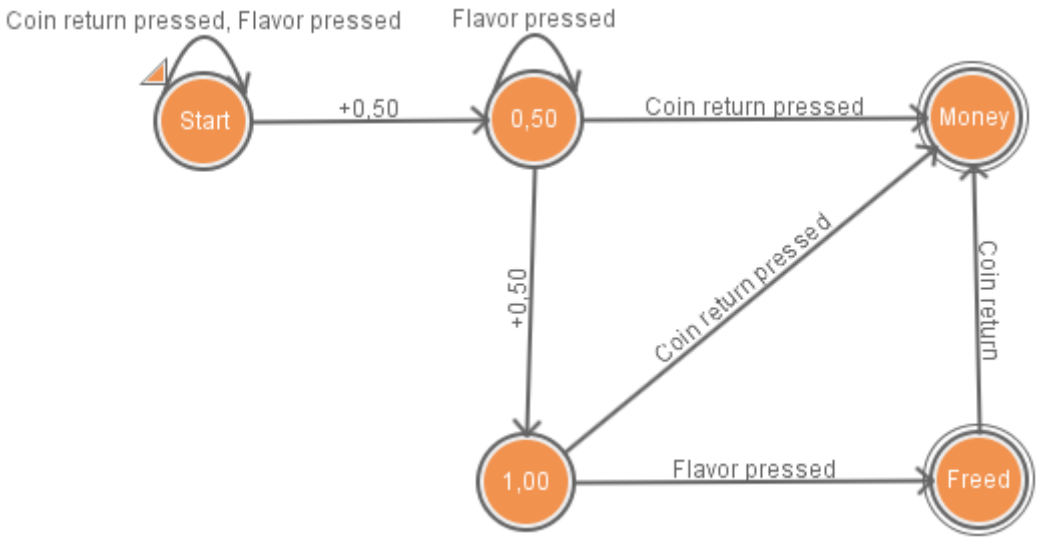

Figura 5.3 - Exemplo de um DET para máquina de refrigerante

Dada uma sequência ordenada de ações, ou seja, o plano, conforme pode ser observado na Figura 5.4, a execução básica consta de uma simulação das transições entre os estados do domínio.

```
1. +0,50
2. +0,50
3. Flavor pressed
```

Figura 5.4 - Exemplo de um plano operacional válido

A execução deste plano na máquina de refrigerante tem como resultado alcançar o estado *"Freed"*, como pode ser observado na Figura 5.5, um estado final que indica a liberação do refrigerante, um plano válido, considerando a alcançabilidade do estado final.

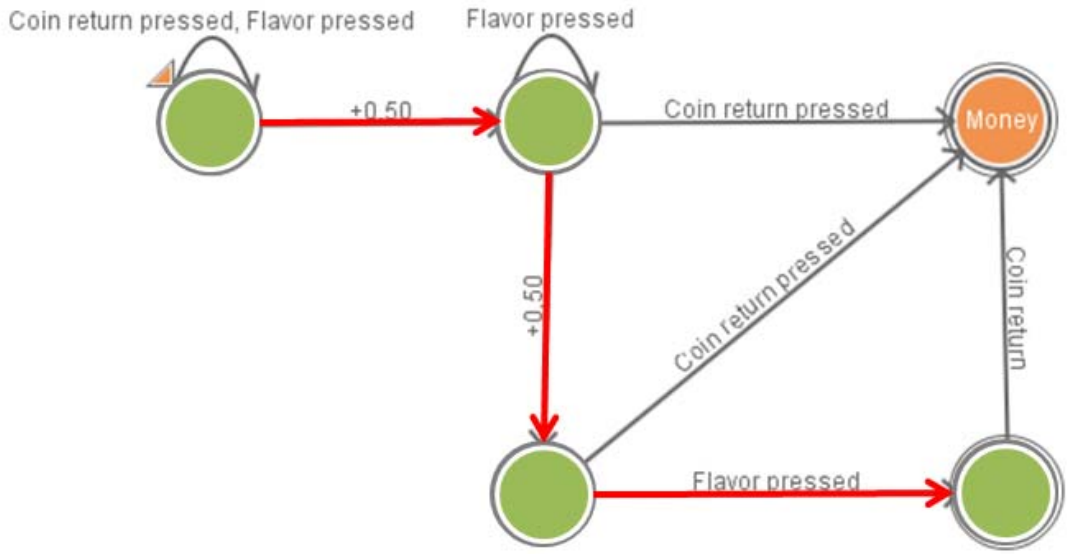

Figura 5.5 - Simulação da execução do plano válido

Considerando um novo plano, apresentado na Figura 5.6, é possível observar que o estado "*Freed*" não é alcançado, se for considerado a alcançabilidade do estado final, esse plano passa a ser inválido, conforme pode ser observado na Figura 5.7.

```
1. +0,50
2. Flavor pressed
```

Figura 5.6 - Exemplo de um plano operacional inválido

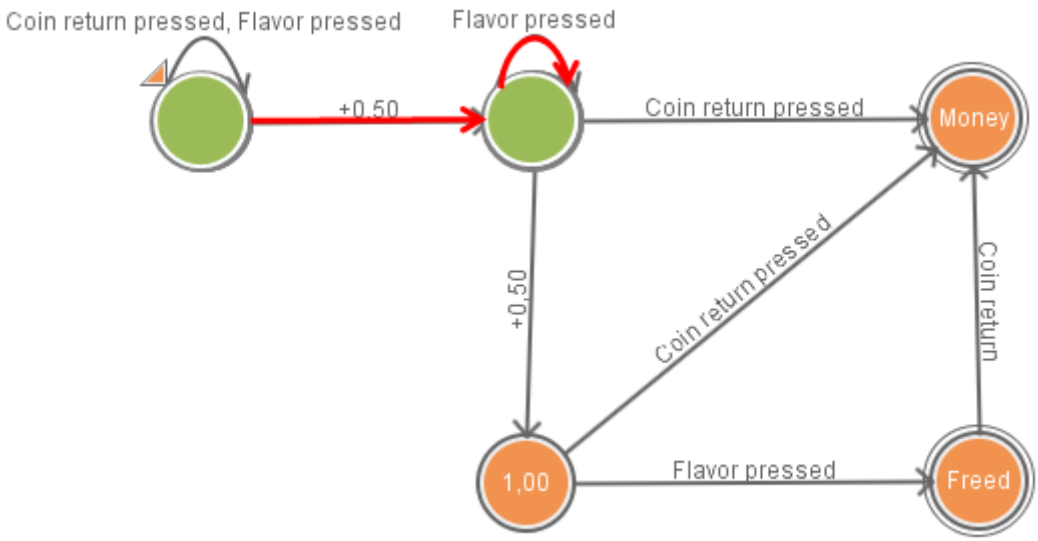

Figura 5.7 - Simulação da execução do plano inválido

Sob o ponto de vista da alcançabilidade de estados esta abordagem já seria o suficiente para a avaliação do plano, no entanto, é comum encontrar situações em que determinada variável do sistema pode assumir uma grande variação de valores, como, por exemplo, temperatura, carga de uma bateria, velocidade e etc. A modelagem desta situação em uma MEFE levaria o domínio a uma explosão do número de estados. Como forma de solucionar esse problema, a arquitetura propõe a combinação do DET a um DL, onde é possível modelar estes tipos de variáveis.

Desta forma as MEFEs passam a se chamar DET que é executada em paralelo ao DL, constituindo o Domínio do Sistema, conforme pode ser observado na Figura 5.8.

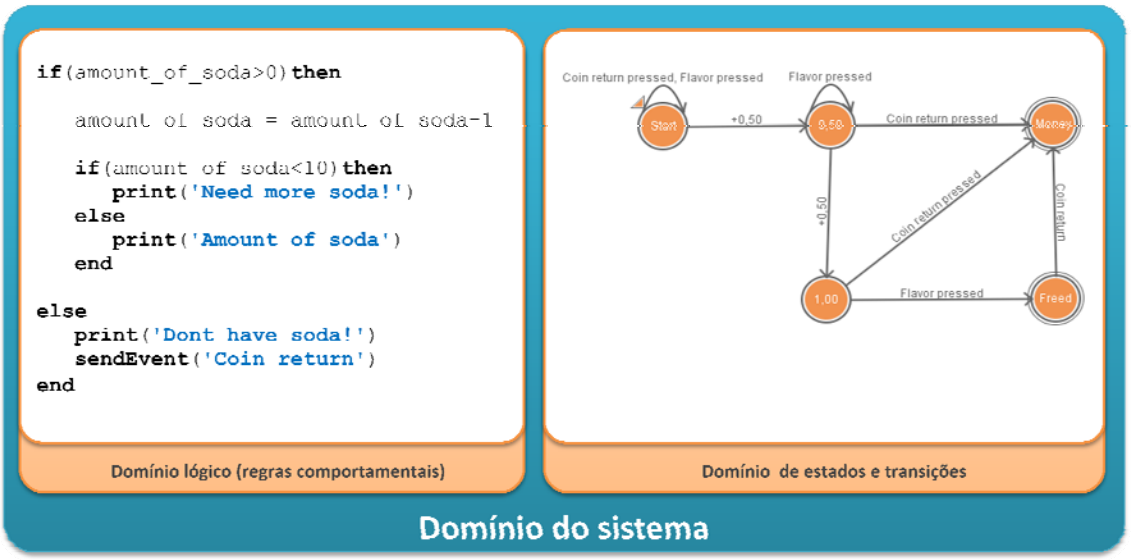

Figura 5.8 - Domínio Lógico e Domínio de Estados e Transições

5.4.2. DL - Domínio lógico

O Domínio Lógico (DL) é utilizado para descrever algoritmos e lógicas para representação de comportamentos de variáveis contínuas, similar aos ***"recursos"*** do planejamento temporal. No entanto, o DL permite a descrição lógica para representação de comportamento, conforme pode ser observado na Figura 5.8 a variável "*amount_of_soda*" representa a quantidade de refrigerante da máquina.

O ponto de união do DL e do DET é realizado através de ações dos objetos "*estados*", "*transições*" e "*eventos"*, que possuem as ações "*de entrada*" e "*de saída*". Nestas ações é possível descrever as lógicas que representam os comportamentos das variáveis continuas.

Durante a simulação o domínio do sistema assume um estado atual. A mudança de um estado para outro gera um evento "*de saída*" no objeto origem e um evento "*de entrada*" no objeto destino, conforme pode ser observado na Figura 5.9.

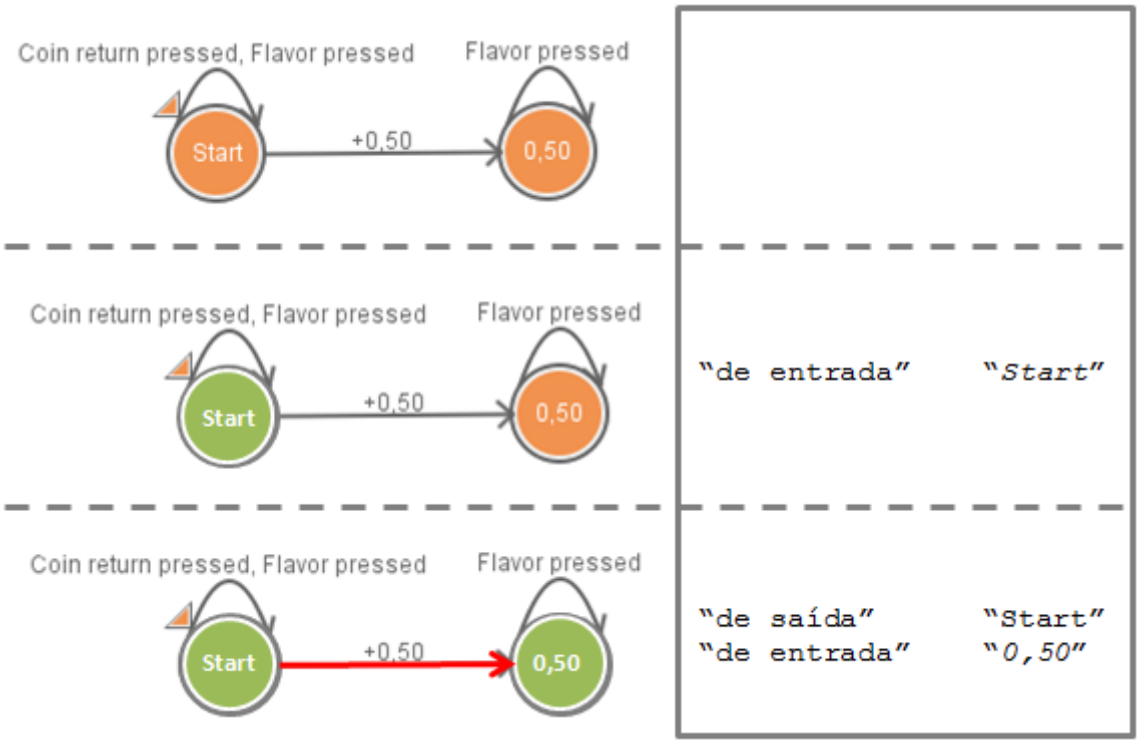

Figura 5.9 - Esquema de geração de eventos

5.4.3. Definição do Plano em formato XML

O plano deve ser escrito em XML conforme os detalhes que podem ser observados na Figura 5.10.

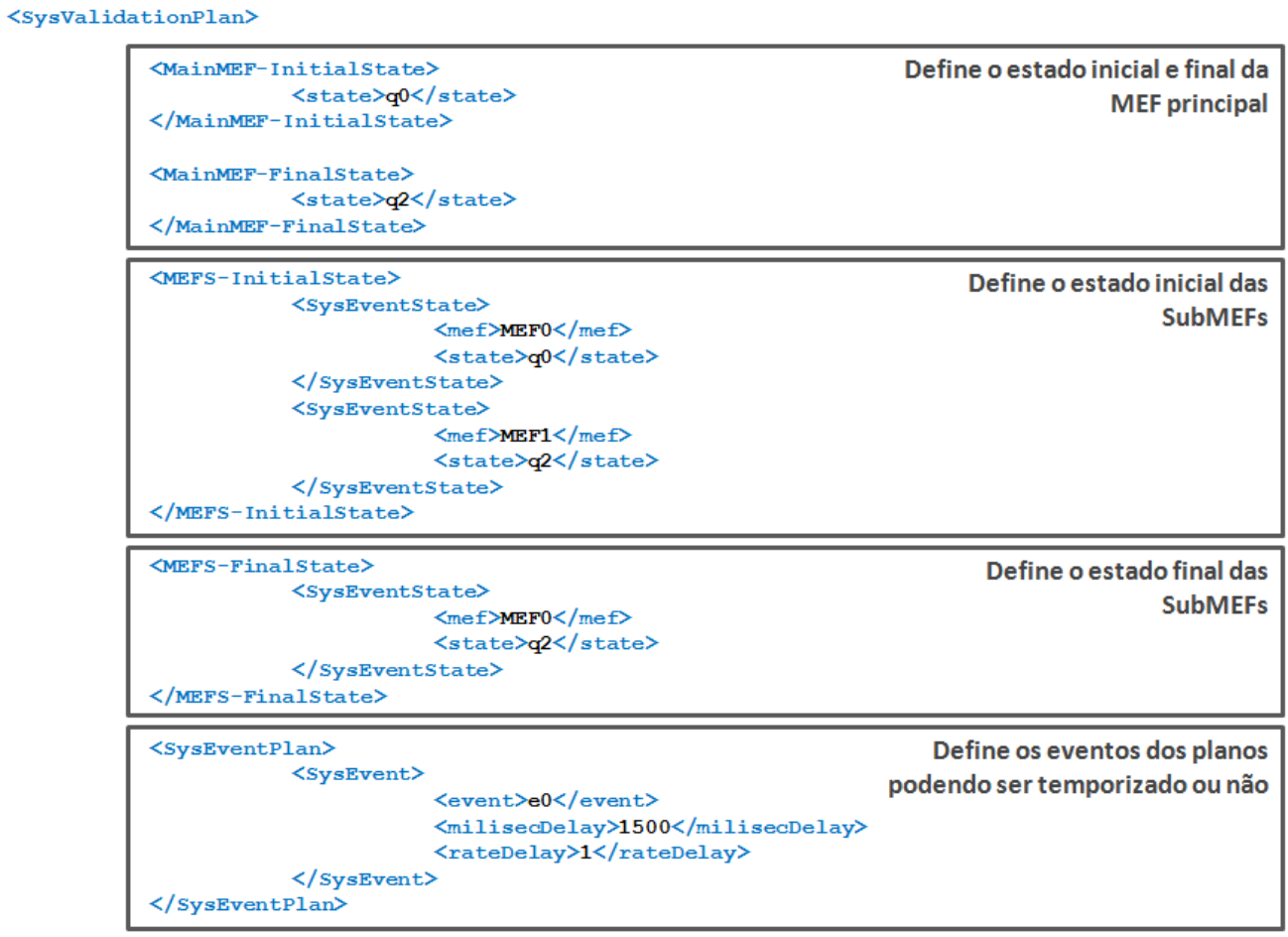

Figura 5.10 - Formato do plano de execução

O plano é dividido em 4 partes:

1. Definição do estado inicial e final da MEF principal;
2. Definição do estado inicial das SubMEFs;
3. Definição do estado final das SubMEFs.
4. Definição dos eventos que serão enviados sequencialmente durante a simulação.

Cada evento possui 3 informações importantes:

1. **event:** Identifica o evento que será gerado;
2. **milisecDelay:** Identifica o tempo de atraso do comando;
3. **rateDelay**: Identifica a taxa do atraso. Tempo de atraso é dado por: *milisecDelay / rateDelay*

O objetivo do plano é definido pelos Estados finais de cada MEF.

5.5. Diagrama de atividades

Nesta seção é apresentado o diagrama de atividade principal do sistema, que é o de simulação e pode ser observado na Figura 5.11.

Este diagrama mostra o funcionamento principal da ferramenta, que ao iniciar uma simulação necessita ser informado um Modelo de Domínio. Em seguida o usuário define a objetivo final ao qual o plano deve conduzir a simulação.

O objetivo final é definido apenas pela alcançabilidade, ou seja, se o estado final definido no plano foi alcançado, ou se as variáveis ambientais estão dentro de determinados padrões e critérios.

O objetivo de alcançabilidade é definido no próprio plano, nos estados finais. Os objetivos dados pela analise das variáveis ambientais devem ser descritos e modelados no relatório de avaliação, no DL.

A atualização das variáveis ambientais com os valores reais é importante para simular exatamente as condições ao qual o sistema a ser avaliado se encontra. Essas variáveis podem ser atualizadas através da funcionalidade definida UC10, ou através de comando **$var(<nome_da_variavel>,<valor>)** no plano operacional.

Exemplo:

```
<SysEvent>
    <event>$var(<nome_da_variavel>,<valor>)</event>
</SysEvent>
```

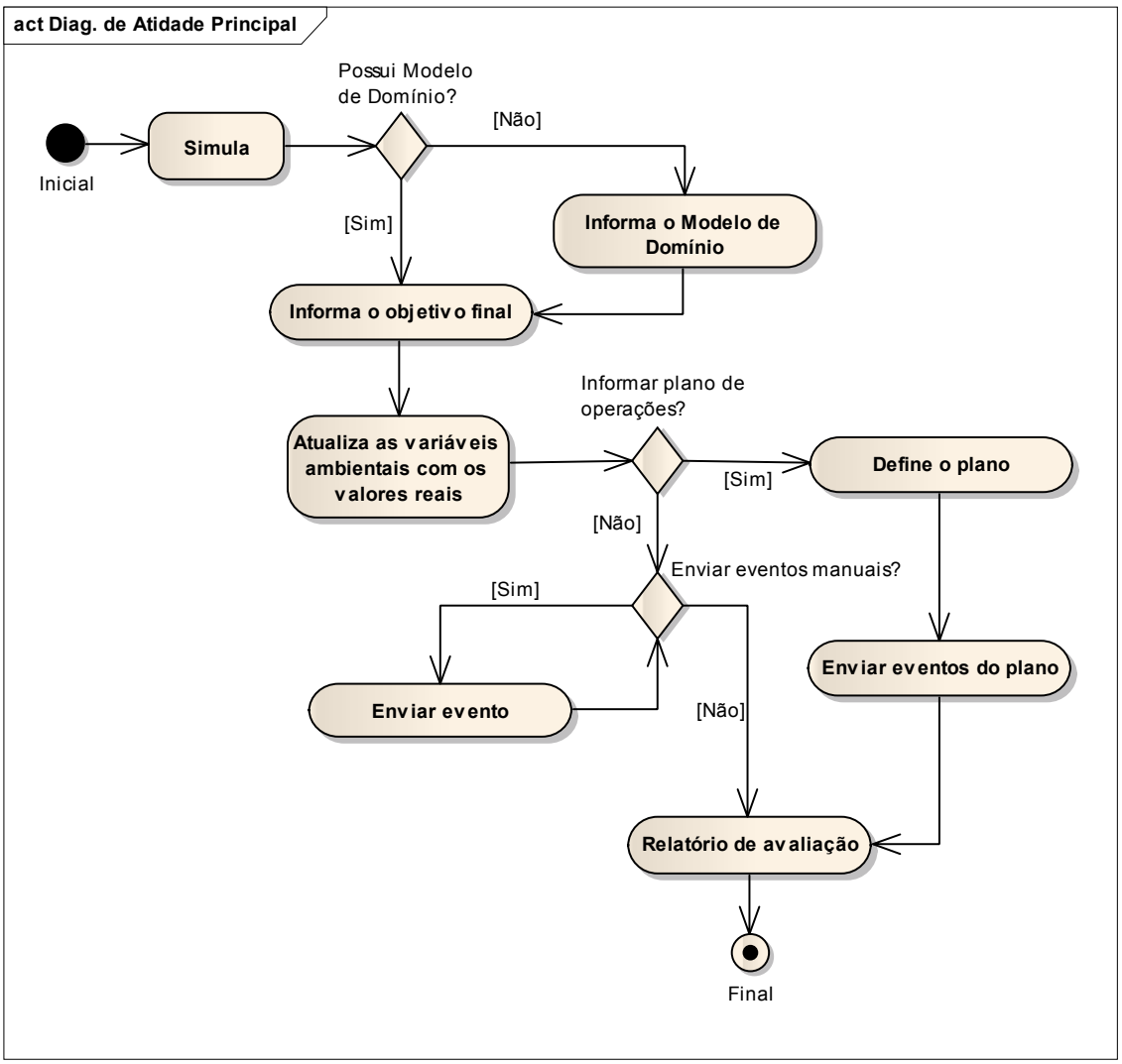

Figura 5.11 - Diagrama de atividade de funcionamento da arquitetura

Caso o usuário opte por não informar um plano, pode enviar eventos manuais para simular a execução de um plano. Esta funcionalidade pode ser interessante para simulações em tempo real. Outra alternativa é enviar um plano completo.

Após a execução de todos os eventos, sejam manuais ou através do plano, o sistema apresenta o relatório de avaliação definido pelo usuário no DL.

5.6. Considerações

A arquitetura representa três passos essenciais no processo de avaliação, são eles:

a) O desenvolvimento do modelo do domínio que representa a união do DET com o DL que representa algoritmos e lógicas para descrever comportamentos de variáveis continuas.

b) Execução do plano sob o modelo de domínio. Trata-se de fazer com que a sequência de eventos seja simulada no modelo do domínio para que possa ser avaliada no passo seguinte.

c) A avaliação da execução do plano, que significa observar os estados e as variáveis ambientais descritas no DL para deliberar sobre o plano através de critérios definidos pelos engenheiros e técnicos de missão.

A arquitetura proposta preocupa-se principalmente com os atributos do domínio lógico (DL) e as mudanças que ocorrem ao longo do tempo. Estes atributos são chamados de variáveis de estado ou variáveis ambientais. Os comportamentos e saídas do modelo de estados são definidos pelas alterações das variáveis de

estado e condições definidas nas ações dos estados ou transições, dependendo do tipo de máquina utilizada.

A arquitetura tem como objetivo avaliar o plano, e assume como sendo o processo de determinar a valia ou o valor de; apreciar ou estimar, o preço, o merecimento, calcular, estimar; fazer a apreciação; ajuizar. Neste sentido a arquitetura permite determinar os valores das variáveis de estado após a simulação e permite que o operador ou gerente de missão possa definir regras deliberativas (ajuizar) sobre a segurança, confiabilidade e alcançabilidade do objetivo do plano.

A utilização desta arquitetura para avaliação de planos significa um aumento na segurança, uma diminuição dos riscos e dos custos de operação, em se tratando de planejadores autônomos, agrega principalmente determinismo ao processo. Além disso, a arquitetura é independente do planejador, podendo ser utilizada como parte do processo de planejamento, servindo como retroalimentação na busca por um plano com qualidade.

A arquitetura define como fazer e pode ser utilizada como parte das recomendações do SPAAS (BLANQUART *et al.*, 2004), além de que a utilização de Modelos de Estados torna a arquitetura determinística, do ponto de vista que todos os estados são conhecidos.

O próximo capítulo apresenta uma ferramenta para validar a arquitetura proposta.

6 A FERRAMENTA - ATOM SYSVAP

Para validar a arquitetura proposta neste trabalho uma ferramenta foi implementada e recebeu o nome de Atom SysVap (*Atom system for validation of finite automatons and execution plans*).

O Atom foi implementado para satisfazer todos os requisitos definidos na arquitetura proposta, e permite:

a) A representação dos modelos de DET em máquinas de Mealy, Moore e *Statecharts*;

b) A execução e avaliação de planos na interface de desenvolvimento do modelo de domínio;

c) A execução e avaliação de planos através de linha de comando, para permitir que a ferramenta possa ser acoplado a qualquer outro sistema;

d) A execução em modo iterativo (*Debug/Real time*), através do envio de eventos durante a execução/simulação do plano, para simular a interferência externa ou do ambiente.

e) A leitura de variáveis ambientais para avaliação dos resultados e determinar se o plano pode ou não ser aceito.

O Atom foi desenvolvido em Java e utiliza a linguagem Lua para desenvolvimento do DL, atendendo os requisitos RF2, RF3 e RF10. Lua é uma linguagem de programação de extensão projetada para dar suporte à programação procedimental em geral e que oferece facilidades para a descrição de dados. (IERUSALIMSCHY; FIGUEIREDO; CELES, 2006)

Lua foi criada em 1993 por Roberto Lerusalimschy, Luiz Henrique de Figueiredo e Waldemar Celes, membros da *Computer Graphics Technology*

Group na PUC-Rio, a Pontifícia Universidade Católica do Rio de Janeiro, no Brasil.

O Atom foi implementado através de dois componentes, como mostrado na Figura 6.1. O componente Atom IDE (*integrated development environment*) implementa a interface gráfica com o usuário e depende do Atom CORE para realizar a execução/simulação dos planos no domínio atendendo principalmente o requisito RNF1.

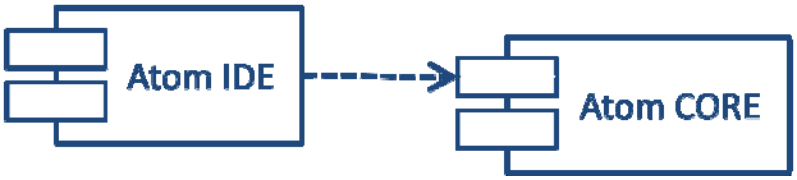

Figura 6.1 - Componentes do Atom SysVap

O Atom IDE foi projetado para o desenvolvimento do modelo de domínio do sistema, representado através de Modelo de Estados. Os Modelos de Estados possuem elementos que serão tratados e manipulados como objetos pelo Atom IDE. Esses objetos são: o Projeto; as MEFE; os Estados; as Transições; e os Eventos.

6.1. Atom IDE

No componente Atom IDE estão concentradas todas as implementações referentes a interface com usuário, de forma a proporcionar uma boa usabilidade, conforme mostrado na Figura 6.2.

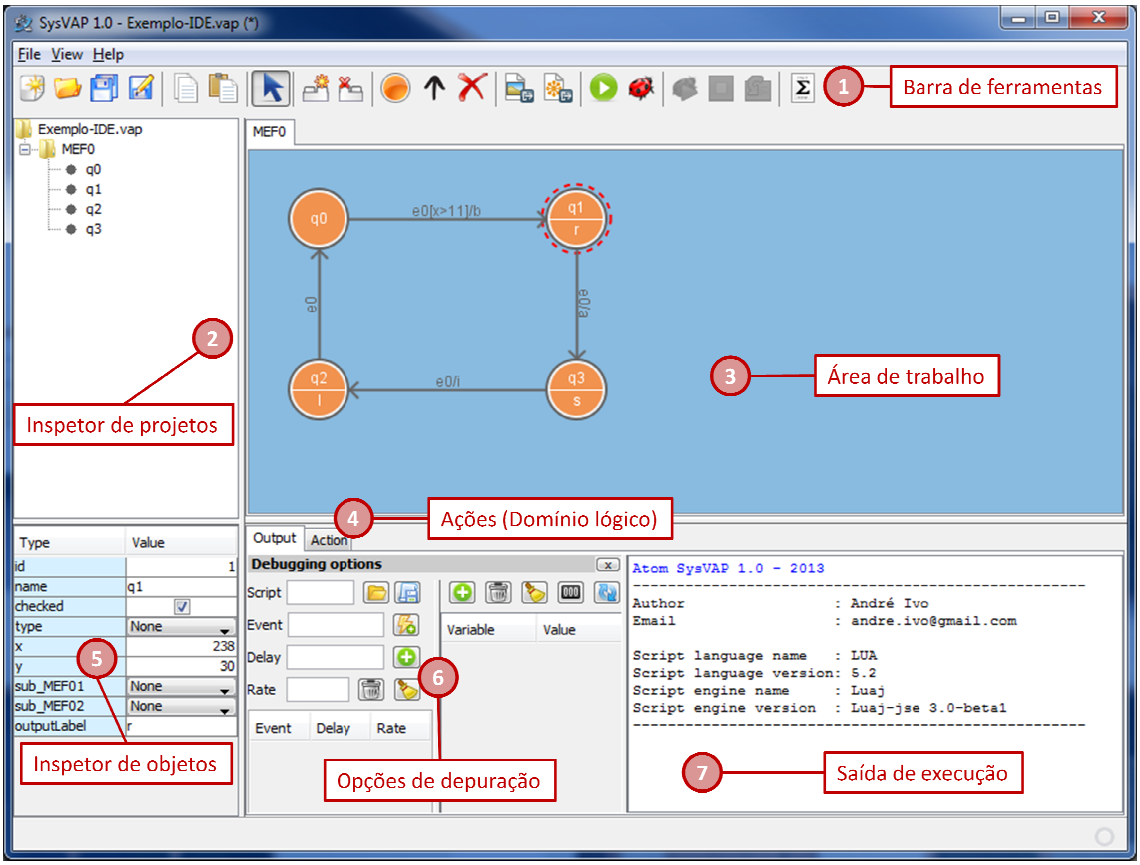

Figura 6.2 - IDE do Atom SysVap

A interface do Atom IDE é dividida em 7 áreas: Barra de ferramentas; Inspetor de projeto; Área de trabalho (DET); Ações (DL); Inspetor de objetos; Opções de depuração e Saída de execução.

1. **Barra de ferramenta**: Disponibiliza as opções para interação com o projeto, conforme pode ser observado na Figura 6.3, atende os requisitos RF4, RF5, RF6 e RF9.

Figura 6.3 - Atom SysVap - Ferramentas para interação com projeto

2. **Inspetor de objetos**: O inspetor de objetos descreve a hierarquia do projeto, com suas MEFs e os estados de cada máquina, conforme pode ser observado na Figura 6.4, as máquinas são identificadas como *"MEF<num>"* e os estados com *"q<num>"*.

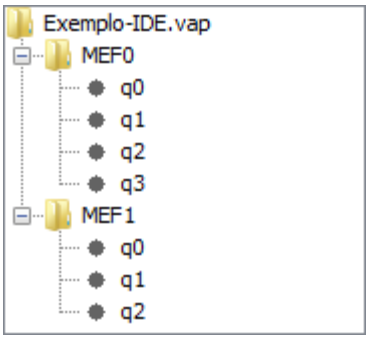

Figura 6.4 - Atom SysVap - Inspetor de objetos

3. **Área de trabalho (DET)**: A área de trabalho disponibiliza um espaço para o desenvolvimento e modelagem do DET, que atende o requisito RF1,

conforme pode ser observado na Figura 6.5. Alguns requisitos das representações principalmente de StateCharts são atendidos no DL.

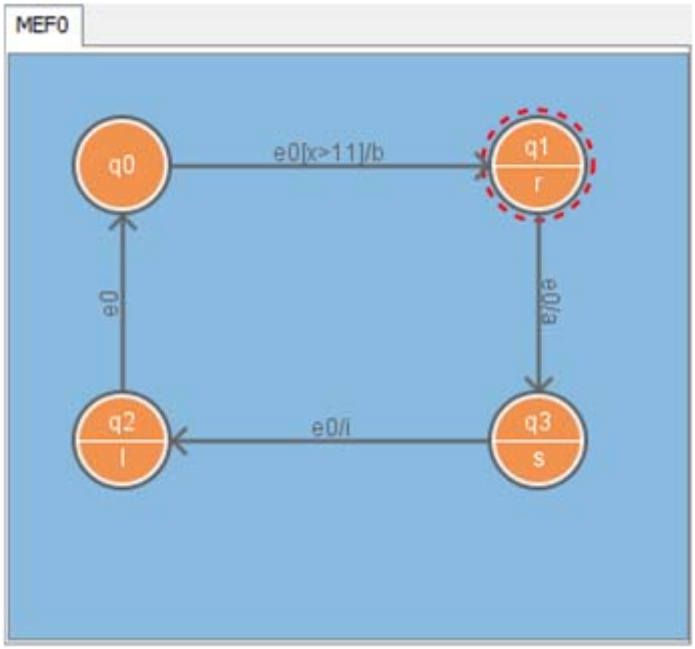

Figura 6.5 - Atom SysVap - Área de trabalho

4. **Ações (DL):** A opção de ações permite a modelagem lógica para representar o comportamento de variáveis continuas como por exemplo consumo de uma bateria, variação de temperatura e etc. e pode ser observado na Figura 6.6 atendendo aos requisitos RF2, RF3 e RF10. No DL é utilizada a linguagem Lua (IERUSALIMSCHY; FIGUEIREDO; CELES, 2006). Maiores informações sobre a linguagem pode ser encontrada no site http://www.lua.org/.

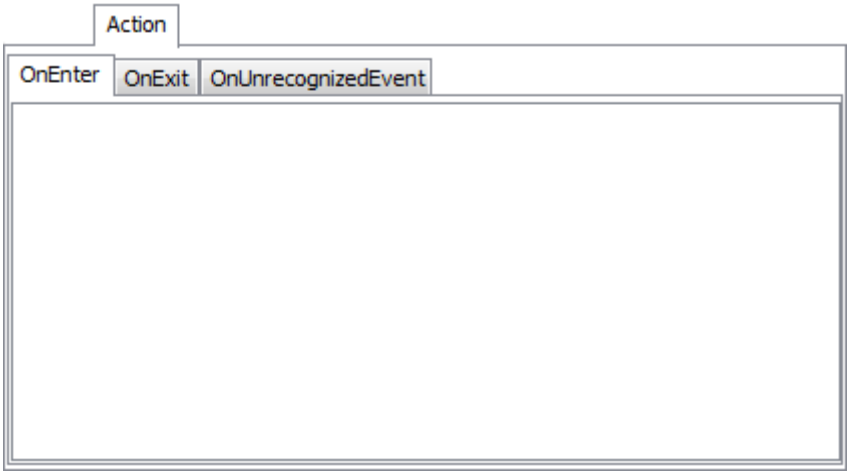

Figura 6.6 - Atom SysVap - Ações (DL)

O DL é implementado através eventos *"OnEnter", "OnExit"* e *"OnUnrecognizedEvent"* dos objetos do DET. O DL disponibiliza 2 funções importantes para a simulação: *sendEvent()* e *stopAll()*.

A função *sendEvent()* permite gerar eventos através do DL, função que representa a capacidade de "*broadcasting"* intrínseca ao formalismo de *StateCharts* A função recebe como parâmetro uma string que representa o evento, como por exemplo: *sendEvent('trocar')*.

A função *stopAll()* termina a simulação e não recebe nenhum parâmetro, para chamá-la por exemplo: *stopAll().*

Além das funções é possível acessar os objetos e suas propriedades. São 9 os objetos disponíveis para acessar no DL, conforme pode ser observado na Tabela 6.1.

Por exemplo, *MEF0_state:getName()* retorna o nome do estado atual da *MEF0.* Para acessar o nome do primeiro elemento da lista de objetos de estados finais da *MEF0,* basta executar *MEF0_finalState:get(0):getName().*

Os detalhes do tipo de cada objeto podem ser encontrados no **Apêndice A**.

Tabela 6.1 - Atom SysVap - Objetos disponíveis no DL

Objeto	Tipo	Descrição
sysProject	SysProject	Representa o projeto de Domínio do sistema
initialMEF	SysMEF	Representa a MEF inicial
event	SysEvent	Representa o último evento enviado durante a simulação
MEF	SysMEF	Representa a MEF
<MEF_Nome>_initialState	SysState	Representa o estado inicial da MEF
<MEF_Nome>_finalState	List<SysState>	Representa a lista de estados finais da MEF
<MEF_Nome>_state	SysState	Representa o estado atual da MEF
<MEF_Nome>_transition	SysTransition	Representa a última transição executada na MEF durante simulação
<MEF_Nome>_event	SysEventTransition	Representa o último evento executado na MEF durante simulação

5. **Inspetor de objetos**: O inspetor de objeto permite a alteração de propriedades dos objetos das MEFs, conforme pode ser observado na Figura 6.7. O conjunto de propriedades muda para cada tipo de objeto. Existem 4 tipos de objetos: Estados, Transições, Eventos e o próprio projeto.

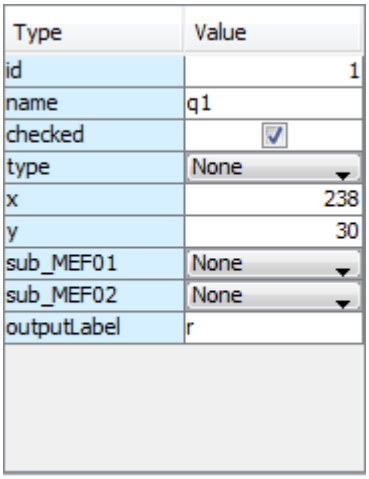

Figura 6.7 - Atom SysVap - Inspetor de Objetos

As propriedades mais importantes, no caso do estado são *"sub_MEF01"* e *"sub_MEF02"* que indicam quais são as submáquinas dos estados, representando a propriedade de ortogonalidade dos *StateCharts*.

Tanto nas transições quanto nos estados existe a propriedade *"outputLabel"* que indica a saída. A propriedade de saída nos estados indica uma representação de Moore e a propriedade de saída nas transições indica uma representação de Mealy.

As transições possuem a propriedade "*guardCondition*" que representa a condição de guarda da transição. A condição de guarda é uma função booleana que deve retornar verdadeiro ou falso (*true or false*) e indica se a transição deve ocorrer ou não. Nesta propriedade é definida pelo modelo lógico, desta forma as condições devem utilizar as variáveis descritas no modelo.

As transições também possuem a propriedade "*timeout*", que habilita a função de temporização, que executa a transição assim que o tempo for esgotado, representação também intrínseca do formalismo de StateCharts. A propriedade "*milisec_timeout*" indica quanto tempo a transição deve aguardar antes de executar mudança de estado, se o *"timeout"* estiver habilitado.

O projeto possui uma propriedade muito importante para simulação do "*rateDelay*" que indica qual a taxa de execução das funções de temporização. Toda função de temporização é dividida por "*rateDelay*", por exemplo, no caso das transições com timeout: *Tempo = milisec_timeout / rateDelay*. Essa propriedade também é utilizada no envio de comandos temporizados dos planos operacionais.

6. **Opções de depuração**: Esta opção é dividida em duas partes como pode ser observada na Figura 6.8.

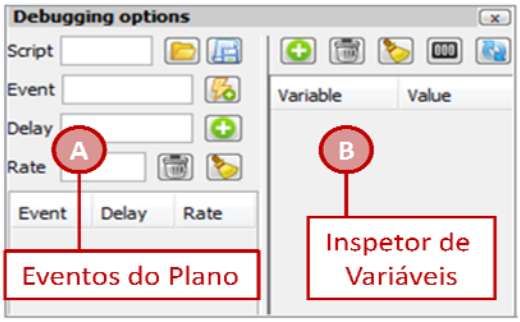

Figura 6.8 - Atom SysVap - Opções de depuração

A. **Eventos do Plano**: Nesta opção é possível definir os eventos para simulação do plano, atendendo ao requisito RF6, além da possibilidade de carregar um arquivo XML que represente o plano, atendendo o requisito RF7. O campo "*Script"* define o nome do arquivo XML que foi carregado com o plano, conforme Figura 5.10.

   O campo "*Event"* indica qual evento deve ser enviado; o campo "*Delay"* indica quantos milissegundos o evento atrasa antes de ser executado; e o campo "*Rate"* indica a taxa de execução do evento específico, similar "*rateDelay",* no entanto quando definido tem prioridade sob o "*rateDelay*". Por exemplo: *Tempo = milisec_timeout / Rate.*

B. **Inspetor de Variáveis:** Esta opção é utilizada para visualizar o valor de determinada variável ou objeto durante o processo de depuração. As variáveis devem estar definidas no DL.

7. **Saída de execução**: Esta área é utilizada para exibir todas informações sobre a simulação do domínio. Ao utilizar o comando "*print"* no DL o sistema imprimirá o resultado na saída de execução.

6.2. Atom CORE

No Atom CORE foram implementadas todos os mecanismos de execução e simulação de um plano sob um modelo de domínio. O componente Atom CORE pode ser utilizado como biblioteca java, podendo ser incorporado a outro programa, ou através de linha de comando, que recebe como argumento os arquivos que representam o modelo de domínio e o plano, o que atende o requisito RNF1.

1. **Através de linha de comando**: para executar o Atom CORE através de linha de comando basta ter o Java instalado na máquina e executar a biblioteca passando como parâmetros o domínio (exportado para simulação) e o plano no formato XML.

   Como por exemplo: *java -jar AtomCore.jar aval.evap plan.xml.*

2. **Como uma biblioteca**: Para utilizar como biblioteca basta adicionar o *AtomCORE.jar* no projeto, utilizar as classes *SysProject, SysMEF, SysState, SysTransition e SysEventTransition* para construção do DET e utilizar as propriedades *actionOnEnter, actionOnExit e actionOnUnrecognizedEvent* para construção do DL.

   Para iniciar uma simulação basta utilizar a classe *SysSimulation* e enviar os eventos através da função *sendEvent()*, como pode ser observado na Figura 6.9.

```
SysProject sysProject = new SysProject ();
...
...
SysSimulation simulation = new SysSimulation(exportProject);
simulation.sendEvent("trocar");
```

Figura 6.9 - Atom CORE- Exemplo de código para simulação

O uso do Atom CORE como uma biblioteca permite que programas possam avaliar planos em tempo real, enviar eventos, realizar a leitura das variáveis de ambiente, característica que permite replanejamento quando acoplado diretamente com o sistema.

6.3. Considerações

A implementação da ferramenta permite explorar e validar a arquitetura proposta. O próximo capítulo apresenta a utilização da arquitetura proposta e da ferramenta para avaliar planos em experimentos controlados.

7 ESTUDO DE CASO

Para avaliar a funcionalidade do Atom SysVap foram desenvolvidos dois experimentos, didáticos e hipotéticos, que exploram as funcionalidades do Atom SysVap para expor os recursos da arquitetura proposta e da ferramenta, sendo:

- Câmera CCD (*charge-coupled device*) embarcada em um satélite: Experimento mostra um sistema simples e estático de acionamento de uma câmera CCD. O objetivo deste experimento é apresentar a modelagem e a avaliação de um plano no sistema de forma didática para apresentar os conceitos básicos;

- Sistema de suprimento de energia (TOMINAGA, 2010): Este experimento mostra a modelagem de um sistema de suprimento de energia, o ambiente e a interação entre sistema/ambiente, além da avaliação do sistema. O objetivo deste experimento é apresentar de forma detalhada o passo à passo da modelagem de uma sistema e como explorar os recursos da arquitetura e da ferramenta.

7.1. Experimento 1 - Acionamento de câmera CCD

Para avaliar a funcionalidade do Atom SysVap foi desenvolvido um experimento, hipotético, simples e didático, de acionamento de uma câmera CCD (charge-coupled device) embarcada em um satélite, conforme a Figura 7.1. O exemplo é hipotético e utilizado apenas para validação do conceito. Durante a apresentação do exemplo serão utilizados valores hipotéticos, como por exemplo se a carga da bateria apresentar 55% da capacidade.

O exemplo pretende representar um sistema espacial e como regra de negócio, deve aguardar por um telecomando para estabelecer uma conexão, e assim

que estabelecida o sistema deve permitir ligar o equipamento CCD e capturar uma foto. Para que uma nova captura seja realizada o equipamento CCD deve ser desligado e religado. A cada captura o sistema consome o equivalente a 3% do suprimento total de energia que ao atingir um limite de: 60% entra em estado de alerta B1; 40% passa para alerta B2; 20% passa para alerta B3. A capacidade de armazenamento também é controlada no sistema. A capacidade total de armazenamento é de 30 fotos, que ao atingir 50% emite um alerta F1 e F2 ao atingir mais que 80% da capacidade.

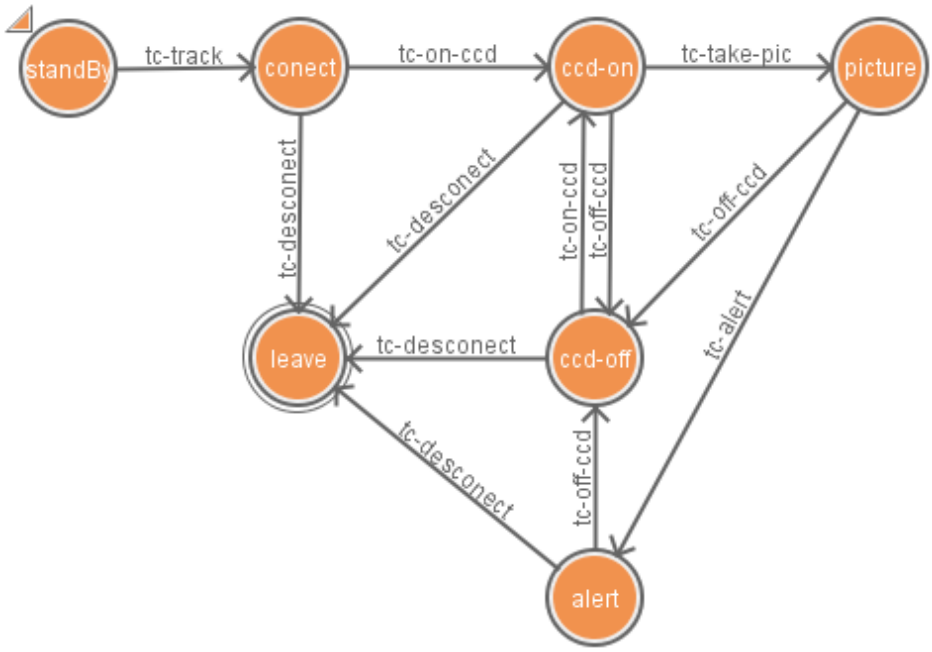

Figura 7.1 - Modelo de domínio para Câmera CCD

O modelo matemático da MEF que representa o domínio proposto na Figura 7.1 é definido por:

I: {tc-alert, tc-desconect, tc-off-ccd, tc-on-ccd, tc-take-pic, tc-track}

Q: {alert, ccd-off, ccd-on, conect, leave, picture, standBy}

Q0: {standBy}

Z: (standBy,tc-track) = conect

Z: (conect,tc-on-ccd) = ccd-on

Z: (conect,tc-desconect) = leave

Z: (ccd-on,tc-take-pic) = picture

Z: (ccd-on,tc-off-ccd) = ccd-off

Z: (ccd-on,tc-desconect) = leave

Z: (ccd-off,tc-on-ccd) = ccd-on

Z: (ccd-off,tc-desconect) = leave

Z: (alert,tc-off-ccd) = ccd-off

Z: (alert,tc-desconect) = leave

Z: (picture,tc-off-ccd) = ccd-off

Z: (picture,tc-alert) = alert

F: {leave}

No DL foram modelados os comportamentos através das variáveis ambientais. No estado "*picture*" foram adicionados os comportamentos e alertas sobre o consumo de energia e capacidade de armazenamento, conforme Figura 7.2.

```
power_suply = power_suply-3
if(power_suply<60)then
            power_suply_alert = 'B1'
            sendEvent('tc-alert')
elseif(power_suply<40)then
            power_suply_alert = 'B2'
            sendEvent('tc-alert')
elseif(power_suply<20)then
            power_suply_alert = 'B3'
            sendEvent('tc-alert')
end

if(storage_suply<30)then
            storage_suply = storage_suply+1
            if(storage_suply>=15)then
                        storage_suply_alert = 'F1'
                        sendEvent('tc-alert')
            elseif(storage_suply>=24)then
                        storage_suply_alert = 'F2'
                        sendEvent('tc-alert')
            end
else
            print('Not enough space to store the photo.')
            sendEvent('tc-alert')
end
```

Figura 7.2 - Ação executada no estado "*picture*"

No estado "*alert*" foi modelo o comportamento para quando o suprimento de energia estiver abaixo de 3% o sistema é forçado a desligar, conforme pode ser observado na Figura 7.3.

```
if(power_supply<=3)then
            power_supply_alert = 'B4'
            print('Power supply not enough to continue.'..
                  'System was forced to shut down.')
            sendEvent('tc-desconect')
end
```

Figura 7.3 - Ação executada no estado "*alert*"

O Atom permite que sejam modelados os critérios de avaliação do plano no modelo de domínio. No caso do exemplo da Figura 7.1, pretende-se verificar, se o plano levou ao estado final previsto, se houve algum alerta para o suprimento de energia e de armazenamento além de verificar se ouve algum comando no plano que não foi reconhecido pelo sistema. Para o este modelo de domínio foi modelada a avaliação conforme Figura 7.4.

```
print('-------------------------------------------------')
print('------          Simulation Report          -------')
print('-------------------------------------------------')
print('')
print('Power supply..: '..power_supply..'%')
print('Storage supply: '..storage_supply..' photo')
print('')
if(MEFO_finalState:get(0):getName()==MEFO_state:getName())then
          print('The expected final state was reached by the simulation!')
else
          print('The expected final state was NOT reached by the simulation!')
end
print('')
print('-------------------------------------------------')
print('-------               Alerts               -------')
print('-------------------------------------------------')
if(power_supply_alert~='')then
          print('Warning system: Powar reached the limit '..power_supply_alert..'!')
end
if(storage_supply_alert~='')then
          print('Warning system: Storage reached the limit '..storage_supply_alert..'!')
end
print('')
print('-------------------------------------------------')
print('-------        Unrecognized event          -------')
print('-------------------------------------------------')
for key,value in pairs(unrecEvent) do
          print(value)
end
print('-------------------------------------------------')
```

Figura 7.4 - Avaliação dos planos executados no domínio do sistema

Definido o modelo de avaliação o domínio está pronto para ser utilizado na avaliação dos planos. O desenvolvimento do modelo de domínio é um dos componentes de entrada para o Atom, como os planos e a definição das variáveis ambientais.

Para realizar a avaliação, define-se o estado inicial das variáveis ambientais, que neste ensaio será de 61% de suprimento de energia, supondo que o sistema estivesse em funcionamento.

O objetivo do plano é definido como capturar 2 fotos e desconectar do sistema, conforme Figura 7.5

```
1. tc-track           //Comando para iniciar a conexão
2. tc-on-ccd          //Comando para ligar equipamento ccd
3. tc-ccd-test        //Comando para testar o ccd
4. tc-take-pic        //Comando para capturar a 1° foto e armazenar
5. tc-off-ccd         //Comando para desligar equipamento ccd
6. tc-system-test     //Comando para testar o sistema
7. tc-on-ccd          //Comando para ligar o ccd
8. tc-take-pic        //Comando para capturar a 2° foto e armazenar
9. tc-off-ccd         //Comando para desligar o sistema
10.tc-desconect       //Comando para desconectar
```

Figura 7.5 - Plano para obter 2 fotos

Conforme modelo matemático, I: {tc-alert, tc-desconect, tc-off-ccd, tc-on-ccd, tc-take-pic, tc-track}, o domínio não prevê comandos como o "tc-ccd-test" e "tc-system-test", que não foram reconhecidos, corretamente, pela avaliação do Atom, conforme mostrado na Figura 7.6. Além disso a avaliação mostra que após o plano ser avaliado o suprimento de energia é de 55%, o que gerou um alerta B1 e 2 fotos foram capturadas, lembrando que os valores apresentados são hipotéticos.

```
Started simulation - CameraCCD.vap
-------------------------------------------------
-------------------------------------------------
------          Simulation Report          -------
-------------------------------------------------

Power supply..: 55%
Storage supply: 2 photo

The expected final state was reached by the simulation!

-------------------------------------------------
-------                Alerts                -------
-------------------------------------------------
Warning system: Power reached the limit B1!

-------------------------------------------------
-------        Unrecognized event          -------
-------------------------------------------------
Event "tc-ccd-test" unrecognized on state ccd-on!
Event "tc-system-test" unrecognized on state ccd-off!
-------------------------------------------------

Finished simulation - CameraCCD.vap
-------------------------------------------------
```

Figura 7.6 - Avaliação realizada pelo Atom SysVap

7.2. Experimento 2 - sistema de suprimento de energia

O experimento proposto consiste em um satélite virtual simples, cuja operação se resume a ligar e desligar duas cargas úteis. A modelagem deste exemplo inclui apenas o subsistema de suprimento de energia do satélite, sendo isto necessário e suficiente para caracterizar as atividades rotineiras desta missão. (TOMINAGA, 2010)

A Figura 7.7 representa o diagrama de blocos do satélite virtual considerado. A energia em bordo é produzida por meio de um gerador composto por conjuntos de células solares (SAG). Este conjunto alimenta o controlador de carga da bateria (BCC) que, por sua vez, alimenta uma bateria recarregável (BAT) e um regulador de barramento principal (MBR). O barramento principal fornece energia para as cargas-úteis (PL1 e PL2) e para equipamentos de serviço (SL).

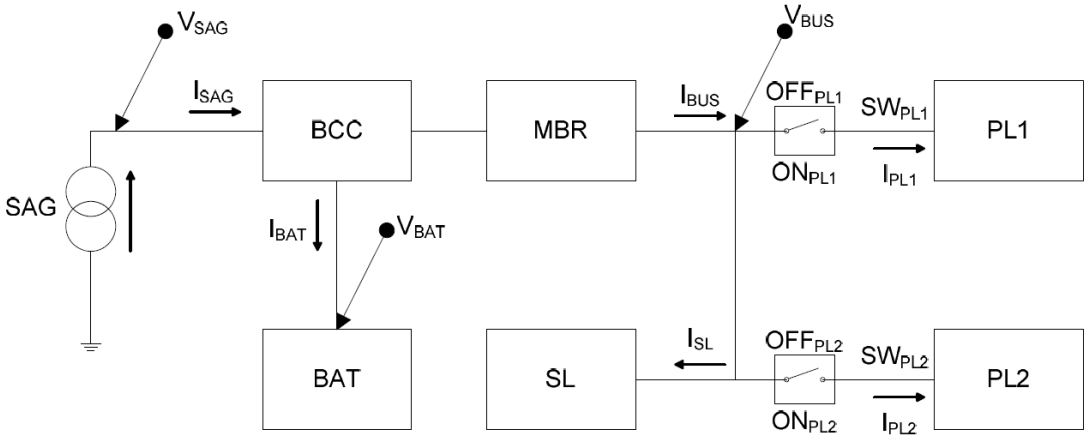

Figura 7.7 - Diagrama do subsistema de suprimento de energia do satélite
Fonte: Tominaga (2010)

As cargas-úteis do satélite, PL1 e PL2, são operadas de forma independente. Os telecomandos ONPL1 e ONPL2 ligam, respectivamente, as cargas-úteis PL1 e PL2. Os telecomandos OFFPL1 OFFPL2 as desligam, de forma análoga. As telemetrias do satélite incluem corrente do gerador (ISAG), tensão de bateria (VBAT), tensão e corrente do barramento (VBUS e IBUS), corrente de entrada das cargas-úteis (IPL1 e IPL2) e estado da chave de alimentação das cargas-úteis (SWPL1 e SWPL2).

De forma simplificada, as atividades rotineiras desta missão consistem em temporizar a execução de comandos de liga e desliga das cargas-úteis (ONPL1, ONPL2, OFFPL1 e OFFPL2). O objetivo da simulação deste tipo de operação consiste em monitorar a profundidade de descarga (DOD = *Depth of Discharge*) da bateria. Operacionalmente, é requisitado que o DOD da bateria não ultrapasse o limite de segurança correspondente a 20%.

Para uma simulação o mais realista possível é importante considerar os eventos externos. Eventos externos são previamente avaliados segundo modelos de ambiente espacial, e fornecidos pela equipe de dinâmica de voo na forma de previsões de eventos orbitais. Isto permite que fenômenos externos e procedimentos a serem executados possam ser agrupados em uma única fila de eventos temporizada.

Para o experimento em questão os eventos orbitais importantes são os momentos que o satélite está iluminado pelo Sol ou eclipsado pela Terra, pois afetam o carregamento da bateria.

Desta forma é possível identificar 2 modelos de domínios:

- **Sistema de satélite:** modelo virtual do produto satelital que responde a eventos ambientais e/ou eventos sistêmicos, com objetivo de simular o comportamento do produto real que receberá o plano. No caso do experimento deve representar o sistema de energia e as cargas úteis.

- **Sistema ambiental:** modelo do ambiente ao qual o sistema de satélite está inserido. Simula os fenômenos ambientais que interferem no produto. No caso do experimento deve representar os eventos orbitais.

7.2.1. Modelo de domínio

Por simplicidade e por não fazer parte deste trabalho, as questões de circuitos elétricos, como cálculos de corrente serão suprimidas e trocadas por constantes.

As regras hipotéticas são:

1 A profundidade de descarga (DOD) da bateria será dada em percentual (%);

2 Quando a célula solar está sendo iluminada pelo Sol a taxa de carregamento é 5% a cada 60 segundos. Sendo necessários 1200 segundos (20 minutos) para ter uma recarga de 0% à 100% da bateria;

3 A carga útil PL1 consome 1(%) da bateria a cada 60 segundos quando está iluminada além de afetar o carregamento, passando para uma taxa de 2% a cada 60 segundos;

4 A carga útil PL2 consome 1(%) da bateria a cada 60 segundos quando está iluminada além de afetar o carregamento, passando para uma taxa de 2% a cada 60 segundos;

5 A carga útil PL1 consome 2(%) da bateria a cada 60 segundos quando não está iluminada;

6 A carga útil PL2 consome 2(%) da bateria a cada 60 segundos quando não está iluminada;

7 O consumo total das duas cargas uteis ligadas simultaneamente é de 5(%) da bateria a cada 60 segundos quando está iluminada além de afetar o carregamento, passando para uma taxa de 2% a cada 60 segundos;

8 O consumo total das duas cargas uteis ligadas simultaneamente é de 10(%) da bateria a cada 60 segundos quando não está iluminada;

7.2.1.1. Modelo de domínio geral

A identificação de dois sistemas paralelos, o satelital e o ambiental, sendo que o ambiental é independente e influencia no satelital, sugere a construção de um modelo que comporte o paralelismo das modelos, a ortogonalidade prevista nos *StateCharts*. Para isto o modelo geral é construído como sendo um estado de "*erro*" e um "*main*", onde serão descritas as máquinas paralelas, conforme pode ser observado na Figura 7.8.

Figura 7.8 - DET geral

O estado "*main*" possui duas subMEFs, uma que representa o modelo de domínio do sistema de satélite e outra que representa o modelo de domínio do sistema ambiental.

O DL geral, na ação "*OnEnter*" do estado "*erro*", apresenta apenas uma mensagem de erro para histórico e gerar relatório para futura análise, conforme pode ser observado na Figura 7.9.

```
print('')
print('-------------------------')
print('- Erro !')
print('- DOD é igual a '..bateria..'%')
print('-------------------------')
```

Figura 7.9 - DL geral - Definição das variáveis

7.2.1.2. Modelo de domínio para o sistema de satélite

Baseado na Figura 7.7, onde basicamente os telecomandos ONPL1, ONPL2 OFFPL1 E OFFPL2, atuam para provocar a mudança de estado do sistema, ligando ou desligando as cargas uteis PL1 e PL2, foi desenvolvido o modelo de domínio para o sistema de satélite, conforme Figura 7.10.

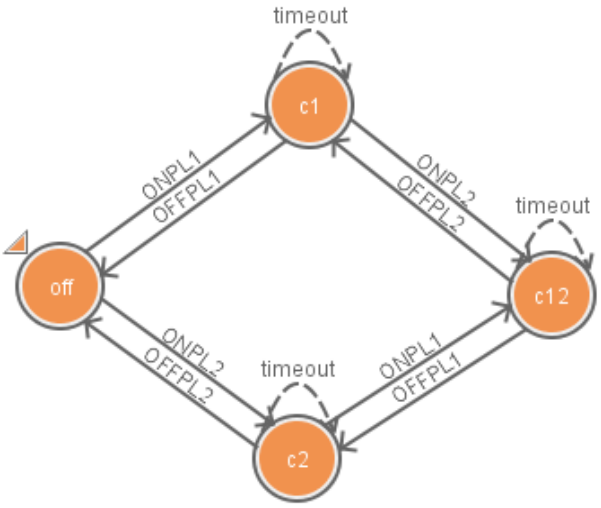

Figura 7.10 - DET para o sistema satélite

A transição "*timeout*" foi utilizada para modelar a taxa de consumo de energia da bateria.

O estado inicial "*off*" representa que nenhum equipamento está ligado. O estado "*c1*" representa que a carga útil PL1 está ligada, o estado "*c2*" que a carga útil PL2 está ligada e o estado "*c12*" que ambas as cargas úteis estão ligadas.

O modelo de DL para o sistema de satélite foi utilizado para modelar as regras sobre consumo de energia da bateria. No evento "*OnEnter*" do projeto, foram definidas as variáveis inicias, como a bateria, se o sistema está em eclipse ou não e qual das cargas úteis estão ligadas, conforme pode ser observado na Figura 7.11

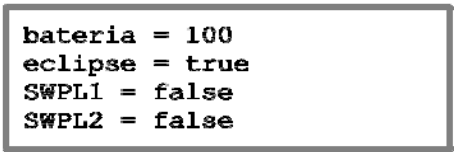

Figura 7.11 - DL satelital - Definição das variáveis

Nas transições foram modeladas as mudanças nas variáveis de controle que indicam quais cargas estão ligadas e algumas mensagens para apresentação de um histórico de execução. Conforme pode ser observado na Figura 7.12.

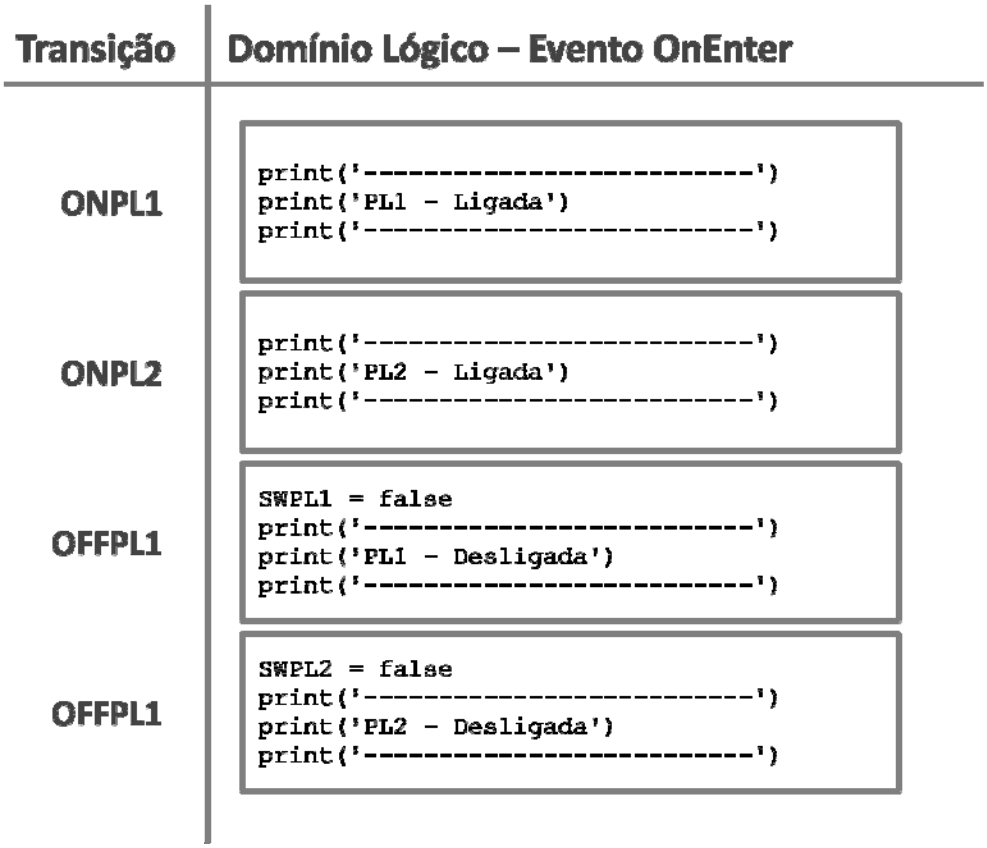

Figura 7.12 - DL satelital - Definição dos eventos de transição

Nos estados foram modelados o consumo da bateria e a algumas mensagens para apresentação de um histórico de execução. Na Figura 7.13 é possível observar que tanto para o estado *"c1"* e *"c2"* o consumo de bateria é de 2% se a variável eclipse for igual a "*true"* (verdadeiro) e de 1% se for "*false"* (falso). No estado "*c12*" o consumo de bateria é de 10% se a variável eclipse for igual a "*true"* (verdadeiro) e de 5% se for "*false"* (falso). Caso a variável bateria seja menor que 20% é gerado um evento "*ALERT*" que conduzirá o sistema a um erro, previsto no domínio geral, conforme pode ser observado na Figura 7.8.

Estado	Domínio Lógico – Evento OnEnter

C1	SWPL1 = true
	if(eclipse==true)then
	bateria = bateria-2
	else
	bateria = bateria-1
	End
	print('CONSUMO..... < :'..bateria..'%')
	if(bateria<20)then
	sendEvent('ALERT')
	end

C2	SWPL2 = true
	if(eclipse==true)then
	bateria = bateria-2
	else
	bateria = bateria-1
	End
	print('CONSUMO..... < :'..bateria..'%')
	if(bateria<20)then
	sendEvent('ALERT')
	end

C12	SWPL1 = true
	SWPL2 = true
	if(eclipse==true)then
	bateria = bateria-10
	else
	bateria = bateria-5
	End
	print('CONSUMO..... < :'..bateria..'%')
	if(bateria<20)then
	sendEvent('ALERT')
	end

Figura 7.13 - DL satelital - Definição dos eventos dos estados

7.2.1.3. Modelo de domínio para o sistema ambiental

O modelo ambiental baseado no sistema apresentado na Figura 7.7 refere-se a dinâmica da orbita satélite-terra, e terra-sol. Essa dinâmica define quando o sol ilumina o satélite que interage com os painéis solares gerando energia necessária para carregar as baterias do satélite, conforme pode ser observado na Figura 7.14.

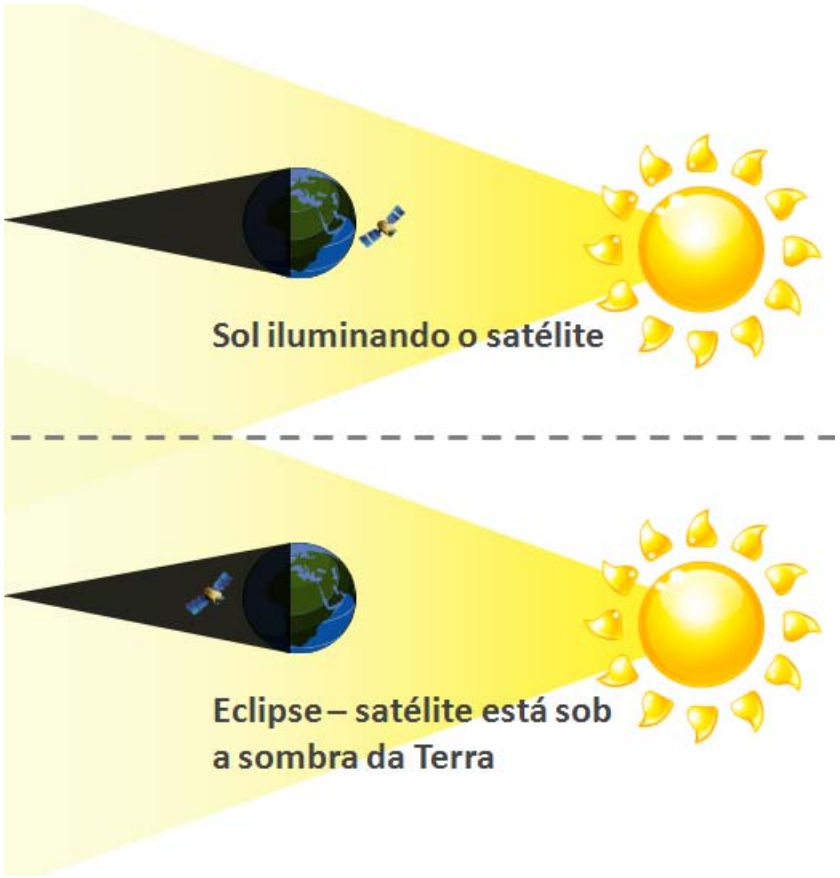

Figura 7.14 - Esquema da dinâmica de voo de um satélite

Baseado na dinâmica exibida na Figura 7.14, foi desenvolvido o modelo de domínio para representar o ambiente, conforme pode ser observado na Figura 7.15.

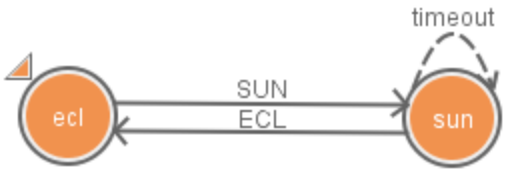

Figura 7.15 - DET para o ambiente

O estado "*ecl*" indica que o sistema está em eclipse e não gera energia para as baterias do sistema de satélite. O estado "*sun*" indica que os sistema está sendo iluminado o que gera energia para o satélite.

A geração de energia é dada em %/segundo e para modelar essa taxa foi utilizado uma transição "*timeout*" que ocorre a cada ciclo conforme previsto nas regras hipotéticas do experimento.

As transições "SUN" e "ECL" representam a mudança ambiental do sistema, são considerados eventos externos previstos pela equipe de dinâmica de voo.

Nas transições foram modeladas algumas mensagens para apresentação de um histórico de execução. Conforme pode ser observado na Figura 7.16.

Transição	Domínio Lógico – Evento OnEnter

SUN	`print('--------------------------')`
	`print('Evento externo - SOL')`
	`print('--------------------------')`

ECL	`print('--------------------------')`
	`print('Evento externo - ECLIPSE')`
	`print('--------------------------')`

Figura 7.16 - DL ambiental - Definição dos eventos de transição

O modelo de DL para o sistema ambiental foi utilizado para modelar as regras sobre o carregamento de energia da bateria. No evento "*OnEnter*" estado "*sun*" foi modelada as regras de carregamento, conforme pode ser observado na Figura 7.17.

```
eclipse = false
if(bateria<100) then
            if((SWPL1 == true) or (SWPL2 == true))then
                        bateria = bateria+2
            else
                        bateria = bateria+5
            end

            bateria = math.min(100,bateria)
end
print('CARREGANDO...  >:'..bateria..'%')
```

Figura 7.17 - DL ambiental - Definição do estado "*sun*"

7.2.1.4. Simulação e avaliação

Foram considerados 2 planos hipotéticos para simulação deste experimento. Neste experimento é importante considerar que existem dois subsistemas em paralelo, o ambiental, que é independente, e influencia no satelital que representa o satélite.

A. Plano 1

Para uma simulação realista é importante inicializar as variáveis do modelo de domínio com os parâmetros reais, no caso do experimento a variável a ser atualizada é a “*bateria*", conforme pode ser observado na Tabela 7.1.

Tabela 7.1 - Comandos de controle

Tempo	Comando de controle	Descrição
00:00:00	$var(bateria,100)	Atualiza variável bateria

O plano a ser enviado ao satélite é descrito na Tabela 4.1, que representa uma lista de comando que devem ser enviados de forma ordenada e temporizada.

Tabela 7.2 - Plano 1 do experimento 2

Tempo	Telecomando	Descrição
00:00:01	ONPL1	Ligar carga útil PL1
00:00:07	ONPL2	Ligar carga útil PL2
00:12:08	OFFPL1	Desligar carga útil PL1
00:12:09	OFFPL2	Desligar carga útil PL2

Além do plano é importante considerar a dinâmica orbital, que descreve os eventos orbitais conforme pode ser observado na Tabela 7.3.

Tabela 7.3 - Eventos orbitais 1 do experimento 2

Tempo	Evento	Descrição
00:00:00	ECL	Evento externo eclipse
00:06:07	SUN	Evento externo iluminar
00:12:07	ECL	Evento externo eclipse

Para a simulação é importante descrever uma única lista de eventos que representem a união dos comandos de controle, telecomando e os eventos orbitais ordenados, conforme Figura 7.18.

```
1. $var(bateria,100)   - Delay 0 ms          - Tempo 00:00:00
2. ECL                 - Delay 0 ms          - Tempo 00:00:00
3. ONPL1               - Delay 1000 ms       - Tempo 00:00:01
4. ONPL2               - Delay 60000 ms      - Tempo 00:00:07
5. SUN                 - Delay 360000 ms     - Tempo 00:06:07
6. ECL                 - Delay 360000 ms     - Tempo 00:12:07
7. OFFPL1              - Delay 1000 ms       - Tempo 00:12:08
8. OFFPL2              - Delay 1000 ms       - Tempo 00:12:09
```

Figura 7.18 - Plano 1 unificado do experimento 2

A simulação do plano apresentado na Figura 7.18 leva o sistema ao estado de erro, pois o DOD (profundidade de descarga) chegou a 18%, conforme pode ser observado no relatório da Figura 7.19.

```
Started debug simulation - Sistema-Energia.vap
------------------------------------------------
-------------------------
PL1 - Ligada
-------------------------
CONSUMO..... < :98%
CONSUMO..... < :96%
-------------------------
PL2 - Ligada
-------------------------
CONSUMO..... < :86%
CONSUMO..... < :76%
CONSUMO..... < :66%
CONSUMO..... < :56%
CONSUMO..... < :46%
CONSUMO..... < :36%
-------------------------
Evento externo - SOL
-------------------------
CARREGANDO...  >:38%
CONSUMO..... < :33%
CARREGANDO...  >:35%
CONSUMO..... < :30%
CARREGANDO...  >:32%
CONSUMO..... < :27%
CARREGANDO...  >:29%
CONSUMO..... < :24%
CARREGANDO...  >:26%
CONSUMO..... < :21%
CARREGANDO...  >:23%
CONSUMO..... < :18%

-------------------------
- Erro !
- DOD igual a 18%
-------------------------

Finished simulation - Sistema-Energia.vap
------------------------------------------------
```

Figura 7.19 - Relatório da simulação do plano 1 unificado

B. Plano 2

Os dois planos são bastante parecidos, considerando apenas 1 alteração, o evento orbital "*SUN*" acontece antes do envio do telecomando "*ONPL2*", conforme pode ser observado nas Tabela 7.4 e Tabela 7.5.

Tabela 7.4 - Plano 2 do experimento 2

Tempo	Telecomando	Descrição
00:00:01	ONPL1	Ligar carga útil PL1
00:06:07	ONPL2	Ligar carga útil PL2
00:12:08	OFFPL1	Desligar carga útil PL1
00:12:09	OFFPL2	Desligar carga útil PL2

Tabela 7.5 - Eventos orbitais 2 do experimento 2

Tempo	Evento	Descrição
00:00:00	ECL	Evento externo eclipse
00:06:01	SUN	Evento externo iluminar
00:12:07	ECL	Evento externo eclipse

O plano completo é apresentado na Figura 7.20 onde pode ser observado a alteração do evento orbital "SUN" e do telecomando "*ONPL2*".

```
1. $var(bateria,100)   - Delay 0 ms          - Tempo 00:00:00
2. ECL                 - Delay 0 ms          - Tempo 00:00:00
3. ONPL1               - Delay 1000 ms       - Tempo 00:00:01
4. SUN                 - Delay 360000 ms     - Tempo 00:06:01
5. ONPL2               - Delay 60000 ms      - Tempo 00:06:07
6. ECL                 - Delay 360000 ms     - Tempo 00:12:07
7. OFFPL1              - Delay 1000 ms       - Tempo 00:12:08
8. OFFPL2              - Delay 1000 ms       - Tempo 00:12:09
```

Figura 7.20 - Plano 1 unificado do experimento 2

A simulação do plano apresentado na Figura 7.20 mostra que o sistema não apresentou nenhum problema e que o DOD (profundidade de descarga) chegou a 69%, conforme pode ser observado no relatório da Figura 7.21.

```
Started debug simulation - Sistema-Energia.vap
------------------------------------------------
-------------------------
PL1 - Ligada
-------------------------
CONSUMO..... < :98%
CONSUMO..... < :96%
CONSUMO..... < :94%
CONSUMO..... < :92%
CONSUMO..... < :90%
CONSUMO..... < :88%
-------------------------
Evento externo - SOL
-------------------------
CARREGANDO...  >:90%
CONSUMO..... < :89%
CARREGANDO...  >:91%
-------------------------
PL2 - Ligada
-------------------------
CONSUMO..... < :86%
CONSUMO..... < :81%
CARREGANDO...  >:83%
CONSUMO..... < :78%
CARREGANDO...  >:80%
CARREGANDO...  >:82%
CONSUMO..... < :77%
CONSUMO..... < :72%
CARREGANDO...  >:74%
CONSUMO..... < :69%
CARREGANDO...  >:71%
-------------------------
Evento externo - ECLIPSE
-------------------------
-------------------------
PL1 - Desligada
-------------------------
CONSUMO..... < :69%
-------------------------
PL2 - Desligada
-------------------------

Finished simulation - Sistema-Energia.vap
------------------------------------------------
```

Figura 7.21 - Relatório da simulação do plano 2 unificado

8 CONCLUSÕES

Este trabalho apresentou uma arquitetura para avaliação de planos utilizados na operação de satélites através de Modelos de estados. A arquitetura foi validada através da construção da ferramenta Atom SysVap, que permite a modelagem de domínios através de representações gráficas como MEFs, MEFEs e *Statecharts*, simulação/avaliação de planos além da integração do sistema avaliador Atom CORE a outros sistemas, que pode viabilizar novas pesquisas na área de replanejamento.

Tanto a arquitetura proposta quanto a ferramenta atendem os requisitos propostos pela ESA no trabalho SPASS de (BLANQUART *et al.*, 2004), sendo apropriada à aplicação do filtro de segurança, pois a técnica utilizada não requer grande capacidade de processamento e memória características importante para sistemas executados em computadores de bordo dos satélites.

8.1. Contribuições do trabalho

A arquitetura proposta confere a característica determinística à avaliação de planos de voo para o controle de satélites, o que traz maior segurança e confiabilidade à utilização, pesquisa e desenvolvimento de novos sistemas para planejamento autônomo. Esta característica permite atender o requisito imposto pela crença dos gerentes de missão e engenheiros de que todas as decisões tomadas em sistemas espaciais devem ser absolutamente previsíveis.

Independente se o planejador é um gerente de missão, ou um sistema autônomo, a arquitetura proposta permite a avaliação de planos operacionais antes da sua execução nos sistemas reais o que pode ajudar a mitigar riscos em uma missão espacial.

A utilização de MEFEs na modelagem de Domínios de Sistemas contribui para uma representação concisa, clara e livre de ambiguidade, problema tradicional e recorrente na representação de sistemas reativos.

O Atom SysVap pode ser utilizado como alternativa a PDDL, que é uma linguagem de definição de domínios de planejamento, utilizada para representação de sistemas reativos. Ao desenvolver um novo sistema planejador é possível utilizar o Domínio do Sistema descrito no Atom SysVap como base para busca por um plano operacional. O Atom SysVap possui uma biblioteca desenvolvida em Java que permite essa integração.

A biblioteca do Atom SysVap também pode ser utilizada para avaliação em tempo real dos planos, podendo ser acoplado diretamente ao sistema real o que viabilizaria o processo de replanejamento.

O Atom SysVap além de servir ao propósito de avaliação de planos de voo, deve servir de apoio acadêmico ao aprendizado sobre linguagens formais e autômatos finitos, como as MEFs. MEFEs e *StateCharts*. O Atom SysVap possui interface amigável e de fácil utilização.

Outros trabalhos relacionados diretamente à elaboração desta dissertação foram apresentados em congresso internacional, o SPACEOPS (*International Committee on Technical Interchange for Space Mission Operations*), e foram submetidos 2 artigos completos, um para a revista IEEE América Latina na área Engenharia III que possui qualificação B1 e outro para a revista IEEE Aerospace & Eletronic System na área Engenharia III que possui qualificação A1.

8.2. Trabalhos futuros

Como trabalho futuro novas pesquisas podem ser desenvolvidas para que a ferramenta Atom SysVap possa importar e exportar domínios escritos em PDDL, ou mesmo ser acoplado diretamente a sistemas embarcados. O Atom SysVap também pode ter sua atuação expandida para outras áreas, como a representação de funções e rotinas de *softwares* além da geração de casos testes para estas rotinas.

REFERÊNCIAS BIBLIOGRÁFICAS

AMARAL, A. S. M. S. D.; VIJAYKUMAR, N. L.; MARTINS, E. Geração automática de casos de teste de conformidade para software de aplicações em protocolos de comunicação. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 2003, São José dos Campos, SP. **Anais...** São José dos Campos: INPE,2003. Disponível em: http://urlib.net/lac.inpe.br/worcap/2003/10.29.11.18. Acesso em 10/12/2013.

ARANGO, G. Domain analysis methods. In: WORKSHOP ON SOFTWARE ARCHITECTURE, 1994, Los Angeles - EUA. **Anais...** USC Center for Software Engineering, 1994.

ARANGO, G.; PRIETO-DÍAZ, R. Domain analysis concepts and research directions. In: WORKSHOP ON SOFTWARE ARCHITECTURE, 1994, Los Angeles - EUA. **Anais...** USC Center for Software Engineering, 1994.

BARRETT, A.; WELD, D. . S.; ETZIONI, O.; HANKS, S.; HENDLER, J.; KNOBLOCK, C.; KAMBHAMPATI, R. Partial-order planning: evaluating possible efficiency gains. **Artificial Intelligence**, v. 67, p. 71--112, 1993. ISSN 92-05-01.

BENSON, S.; NILSSON, N. Reacting, planning and learning in an autonomous agent. In: FURUKAWA, K.; MICHIE, D.; MUGGLETO, S. **Machine intelligence 14:** applied machine intelligence. New York, NY, USA: Oxford University Press, 1996. p. 29-62.

BIANCHO, A. C.; AQUINO, A. C.; FERREIRA, M. G. V.; SILVA, J. D. S.; CARDOSO, L. S. Software agents for multiple satellite automated planning and control. In: AIAA INTERNACIONAL COMMUNICATIONS SATELLITE SYSTEMS CONFERENCE, 24TH (ICSSC 2006), 2006, San Diego, Califórnia. **Anais...** AIAA, 2006. p. 06

BLANQUART, J.-P.; FLEURY, S.; HERNEK, M.; HONVAULT, C.; INGRAND, F.; PONCET, J.-C.; POWELL, D.; STRADY-LÉCUBIN, N.; THÉVENOD, P. Software safety supervision on-board autonomous spacecraft. In: EUROPEAN CONGRESS ERTS, EMBEDDED REAL TIME SOFTWARE, 2., 2004, Toulouse, France. **Anais...** ERTS-2, 2004. p. 11. Disponível em: http://homepages.laas.fr/~felix/publis-pdf/erts04-2.pdf. Acesso em 10/12/2013.

BLUM, A.; FURST, M. Fast planning through planning graph analysis. **Artificial Inteligence**, Montreal, v. 90, p. 1636--1642, 1995.

CANTONI, L. F. A. **Avaliação do uso da linguagem PDDL no planejamento de missões para robôs aéreos**. Belo Horizonte, MG: UFMG, 2010.

CARDOSO, L. S. **Aplicação da tecnologia de agentes de planejamento em operação de satélites**. 2006. p. 167. (INPE-14092-TDI/1075). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais São José dos Campos, 2006b. Disponível em: http://urlib.net/sid.inpe.br/MTC-m13@80/2006/05.04.17.36 Acesso em: 10/12/2013.

CARDOSO, L. S.; FERREIRA, M. G. V.; ORLANDO, V. An intelligent system for generation of automatic flight operation plans for the satellite control activities at INPE. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS WILL BE HOSTED BY THE, 9TH, 2006a, Rome, Italy. **Anais...** The American Institute of Aeronautics and Astronautics (AIAA), 2006. p. 9.

CHISTOPHER, L.; ZUCKERMAN, I. An inductive approach to learning search control rules for planning. **Artificial Intelligence**, UK, v. 101, p. 63--98, 1998.

CURRIE, K.; TATE, A.; BRIDGE, S. O-Plan: the open planning architecture. **Artificial Intelligence**, v. 52, p. 49--89, 1991.

ESTLIN, T. A.; MOONEY, R. J. Multi-strategy learning of search control for partial-order planning. In: THIRTEENTH NATIONAL CONFERENCE ON AI, 1996, Portland - OR. **Anais...** AAAI Press / The MIT Press, 1996. p. 843-848. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=BD595B83960034881DEBDD222B4167FF?doi=10.1.1.26.6519&rep=rep1&type=pdf. Acesso em 10/12/2013.

FIKES, R. E.; NILSSON, N. J. STRIPS: a new approach to the application of theorem proving to problem solving. **Artificial Intelligence**, v. 2, p. 189, 1971.

FOX, M.; LONG, D. PDDL 2.1: an extension to PDDL for expressing temporal planning domains. **Journal of Artificial Intelligence Research**, v. 20, p. 61--124, 2003.

FRANCÊS, C. R. L.; SANTANA, M. J.; VIJAYKUMAR, N. L.; CARVALHO, S. V. D.; SANTANA, R. H. C. Statecharts Estocásticos e Queuing Statecharts**:** novas

abordagens para avaliação de desempenho baseadas em especificação statecharts. In: 15O SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES2001), 2001, Rio de Janeiro, RJ. **Anais...** SBES, 2001. p. 316-331. Disponível em: http://lasdpc.icmc.usp.br/artigos/avaliacao-de-desempenho/2000-2002/frances-sbes2001.pdf. Acesso em 10/12/2013.

FUENTETAJA, R.; BORRAJO, D. Improving control-knowledge acquisition for planning by active learning. In: EUROPEAN CONFERENCE ON MACHINE LEARNING (ECML'06), 2006, Berlin (Germany). **Anais...** Springer Verlag, 2006. p. 138--149. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.7090&rep=rep1&type=pdf. Acesso em 10/12/2013.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated planning:** theory and practice. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

GIUNCHIGLIA, E.; MARATEA, M. Improving plan quality is SAT-based planning. In: INTERNATIONAL CONFERENCE OF THE ITALIAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE, 11., 2009, Italy. **Anais...** Springer, 2009. p. 253-263. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.167.242&rep=rep1&type=pdf. Acesso em 10/12/2013.

HAREL, D. Statecharts: a visual formalism for complex systems. **Sci. Comput. Program.**, Amsterdam, The Netherlands, The Netherlands, v. 8, p. 231--274, 1987.

IERUSALIMSCHY, R.; FIGUEIREDO, L. H. D.; CELES, W. **Lua reference manual**. Rio de Janeiro: Lua.Org, 2006.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Plano diretor 2011-2015**. São José dos Campos, SP, 2011.

JÓNSSON, A. K.; MORRIS, P. H.; MUSCETTOLA, N.; RAJAN, K.; SMITH, B. Planning in Interplanetary Space**:** theory and pratice. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE PLANNING SYSTEMS, 5., 2000, USA. **Anais...** AAAI, 2000. p. 177-186. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.724&rep=rep1&type=pdf. Acesso em 10/12/2013.

KAMBHAMPATI, S.; KATUKAM, S.; QU, Y. Failure driven search control for partial order planners: an explanation based approach. **Artificial Intelligence**, v. 88, p. 253--315, 1996.

KUCINSKIS, F. D. N. **Alocação dinâmica de recursos computacionais para experimentos científicos com replanejamento automatizado a bordo de satélites**. 2007. p. 165. (INPE-14798-TDI/1241). Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2007. Disponível em: http://urlib.net/sid.inpe.br/mtc-m17@80/2007/04.23.11.42. Acesso em 10/12/2013.

LAI, R.; LEUNG, W. Industrial and academic protocol testing: the gap and the means of convergence. **Computer Networks and ISDN Systems**, p. 537-547, January 1995.

LIFSCHITZ, V.; MCCAIN, N.; REMOLIA, E.; TACCHELLA, A. Getting to the airport: the oldest planning problem in AI. In: MINKER, J. **Logic-based artificial intelligence**. Norwell: Kluwer Academic Publishers, 2000. p. 147--165.

LUGER, G. F. **Inteligência artificial:** estruturas e estratégias para a solução de problemas complexos. 4. ed. Brasil: Bookman, 2004.

MACIEL, P. R. M.; LINS, R. D.; CUNHA, P. R. F. Introdução às Redes de Petri e Aplicações. In: ESCOLA DE COMPUTAÇÃO, 10., 1996, Campinas-SP. **Anais...** Campinas: Unicamp. Instituto de Computação, 1996.

MARTINS, E.; SABIÃO, S. B.; AMBROSIO, A. M. ConData: a tool for automating specification-based test case generation for communication systems. **Software Quality Journal**, v. 8, n. 4, p. 303-319, Dec. 1999. ISSN 0963-9314.

MCALLESTER, D.; ROSENBLITT, D. Systematic Nonlinear Planning. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 9., 1991, Anaheim, California. **Anais...** AAAI Press, 1991. p. 634--639. Disponível em: http://dspace.mit.edu/bitstream/handle/1721.1/6588/AIM-1339.pdf?sequence=2. Acesso em 10/12/2013.

MCCARTHY, J. Programs with common sense. In: TEDDINGTON CONFERENCE ON THE MECHANISATION OF THOUGHT PROCESSES, 1968, Cambridge MA. **Anais...** MIT Press, 1968. p. 403--418. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.9028&rep=rep1&type=pdf. Acesso em 10/12/2013.

MCCLUSKEY, T. L.; ALER, R.; BORRAJO, D.; HASLUM, P.; JARVIS, P. **Knowledge engineering for planning roadmap**. Huddersfield: University of Huddersfield, 2000. Disponível em: http://planet.hud.ac.uk/road.pdf. Acesso em 10/12/2013.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. **PDDL - The Planning Domain Definition Language**. Yale Center for Computational Vision and Control, 1998. (TR-98-003).

MENEZES, P. B. **Linguagens formais e autômatos**. 6. ed. Porto Alegre, RS: ARTMED® EDITORA S. A., 2010.

MINTON, S.; CARBONELL, J. G.; KNOBLOCK, C. A.; KUOKKA, D. R.; ETZIONI, O.; AND GIL, Y. Explanation-based learning: a problem solving perspective. **Artificial Intelligence**, v. 40, p. 63-118, 1989.

NAU, D. S.; MUÑOZ-AVILA, H.; CAO, Y.; LOTEM, A.; MITCHELL, S. Total-Order Planning with Partially Ordered Subtasks. In: IJCAI, 2001, San Francisco, CA, USA. **Anais...** Morgan Kaufmann Publishers Inc., 2001. p. 425-430. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.7579&rep=rep1&type=pdf. Acesso em 10/12/2013.

NEIGHBORS, J. M. **Software construction using components**. University of California, Irvine - EUA, 1980. p. 75. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=AEF9A9F02D2D753C73F0F19DA7BD5593?doi=10.1.1.21.4397&rep=rep1&type=pdf. Acesso em 10/12/2013.

NEWELL, A.; SHAW, J. C.; SIMON, H. A. Report on a general problem-solving. In: IFIP CONGRESS, 1959, Paris - France. **Anais...** UNESCO, 1959. p. 256--264. Disponível em: http://www.textfiles.com/bitsavers/pdf/rand/ipl/P-1742_A_Variety_Of_Intelligent_Learning_In_A_General_Problem_Solver_Jul59.pdf. Acesso em 10/12/2013.

PEDNAULT, E. ADL**:** exploring the middle ground between STRIPS and situation calculus. In: INTL. CONF. ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING, 1., 1989, Toronto. **Anais...** Morgan Kaufmann, 1989. p. 324--332

PENBERTHY, J. S.; WELD, D. UCPOP**:** a sound, complete, partial order planner for ADL. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE REPRESENTATION AND REASONING, 3., 1992, Boston - MA. **Anais...** Morgan Kaufmann, 1992. p. 103--114. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.5549&rep=rep1&type=pdf. Acesso em 10/12/2013.

PÉREZ, A. M. The goal is to generate better plans. In: SPRING SYMPOSIUM SERIES, WORKSHOP ON GOAL-DRIVEN LEARNING, 1994, California. **Anais...** AAAI,1994. Disponível em: http://aaaipress.org/Papers/Symposia/Spring/1994/SS-94-02/SS94-02-011.pdf. Acesso em 10/12/2013.

PÉREZ, M. A.; CARBONELL, J. G. Control Knowledge to Improve Plan Quality. In: INTERNATIONAL CONFERENCE ON AI PLANNING SYSTEMS, 2., 1994, Chicago. **Anais...** AAAI,1994. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.5647&rep=rep1&type=pdf. Acesso em 10/12/2013.

PETRENKO, A.; YEVTUSHENKO, N.; LEBEDEV, A.; DAS, A. Nondeterministic state machines in protocol conformance testing. In: PROTOCOL TEST SYSTEMS, 6., 1993, Pau, France. **Anais...** Amsterdam: North-Holland Publishing Co. p. 363--378

PINHEIRO, A. C. **Subsídios para a aplicação de métodos de geração de casos de testes baseados em máquinas de estados**. 2012. p. 95. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) - USP, São Carlos, 2012. Disponível em: http://www.teses.usp.br/teses/disponiveis/55/55134/tde-10092012-100529/publico/DissertacaoREVISADA_ArineizaPinheiro.pdf. Acesso em 10/12/2013.

RABENAU, E.; DENIS, M.; JAYARAMAN, P. Implementation of a mission planning system for an interplanetary mission. In: CONFERENCE ON SPACE OPERATIONS (SPACEOPS-2002), 7., 2002, Houston, Texas, USA. **Anais...** National Aeronautics and Space Administration, 2002. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.200.1162&rep=rep1&type=pdf. Acesso em 10/12/2013.

RABENAU, E.; PESCHKE, S. Experience gained with a mission planning system for mission to Mars. In: CONFERENCE ON SPACE OPERATIONS

(SPACEOPS-2004), 8., 2004, Montreal, Quebec. **Anais...** Space Operations Branch, Canadian Space Agency, 2004. p. 7. Disponível em: http://arc.aiaa.org/doi/pdf/10.2514/6.2004-301-152. Acesso em 10/12/2013.

RUSSELL, S.; NORVIG, P. Classical planning, planning and acting in the real world. In: _____ **Artificial intelligence:** a modern approach. 3. ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2003. Cap. 10-11, p. 366--416.

SACERDOTI, E. D. Planning in a Hierarchy of Abstraction Spaces, Artificial Intelligence. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 3., 1974, Stanford, USA. **Anais...** Morgan Kaufmann Publishers Inc., 1974. p. 412--422. Disponível em: http://ijcai.org/Past%20Proceedings/IJCAI-73/PDF/045.pdf. Acesso em 10/12/2013.

SACERDOTI, E. D. **A structure for plans and behaviors**. USA: Elsevier, 1977. v. 3.

SILVA, M. S. D.; VIJAYKUMAR, N. L.; FRANCÊS, C. R. L.; CARVALHO, S. V. D.; CARDOSO, D. L. O uso combinado dos statecharts e processo markoviano de decisão no processo de avaliação de desempenho de sistemas. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL-SBPO, 40., 2008, João Pessoa. **Anais...** Rio de Janeiro:Sobrapo, 2008 p. v. 1, p.1-15

SIMÃO, A. S.; AMBRÓSIO, A. M.; FABBRI, S. C. P.; AMARAL, A. S. M. S.; MARTINS, E.; MALDONADO, J. C. PLAVIS/FSM**:** an environment to integrate fsm-based testing tool. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 2005, Uberlândia, MG. **Anais...** SBES, 2005. p. 06. Disponível em: http://urlib.net/sid.inpe.br/iris@1916/2005/11.21.18.33. Acesso em 10/12/2013.

SIMÃO, A.; PETRENKO, A.; YEVTUSHENKO, N. Generation reduced test for FSMs with extra states. In: INTERNATIONAL CONFERENCE ON TESTING OF SOFTWARE AND COMMUNICATION SYSTEMS AND 9TH INTERNATIONAL FATES WORKSHOP, 2009, Eindhoven, The Netherlands. **Anais...** Springer-Verlag, 2009. p. 129--145.

SIMON, H. A. **The sciences of the artificial**. 3. ed. Cambridge, MA: The MIT Press, 1996.

SOUZA, P. B. D. **Uma estratégia baseada em algoritmos de mineração de dados para validar plano de operação de voo a partir de predições de**

estados dos satélites do INPE. 2011. p. 171. (sid.inpe.br/mtc-m19/2011/04.15.19.12-TDI). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2011. Disponível em: http://urlib.net/rep/8JMKD3MGP7W/39GL532?languagebutton=pt-BR. Acesso em 10/12/2013.

SOUZA, P. B.; FERREIRA, M. G. V.; SILVA, J. D. S. O Estado da Arte em Teste de Planejamento na Área Espacial. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 2007, São José dos Campos, SP. **Anais...** São José dos Campos: INPE, 2007. Disponível em: http://urlib.net/rep/8JMKD3MGP8W/37D4N7S?languagebutton=pt-BR. Acesso em 10/12/2013.

STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: principles and methods. **IEEE Transactions on Data and Knowledge Engineering**, USA, v. 25, p. 161--197, 1998.

TATE, A. Generalizing project networks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE - IJCAI-77, 1977, MA. **Anais...** IJCAI, 1977. p. 888-893. Disponível em: http://ijcai.org/Past%20Proceedings/IJCAI-77-VOL2/PDF/071.pdf. Acesso em 10/12/2013.

TOMINAGA, J. **Simulador de satélites para verificação de planos de operação em voo**. 2010. p. 174. (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/rep/8JMKD3MGP7W/37HL3J8?languagebutton=pt-BR. Acesso em 10/12/2013.

VAQUERO, T. S. **itSIMPLE:** ambiente integrado de modelagem e análise de domínios de planejamento automático. 2007. p. 316. Dissertação (Mestrado em Computação Aplicada) - USP, São Paulo - SP, 2007. Disponível em: http://www.teses.usp.br/teses/disponiveis/3/3152/tde-19072007-174135/publico/DissertacaoRevisadaVaqueroTS.pdf. Acesso em 10/12/2013.

APÊNDICE A - DIAGRAMA DE CLASSES

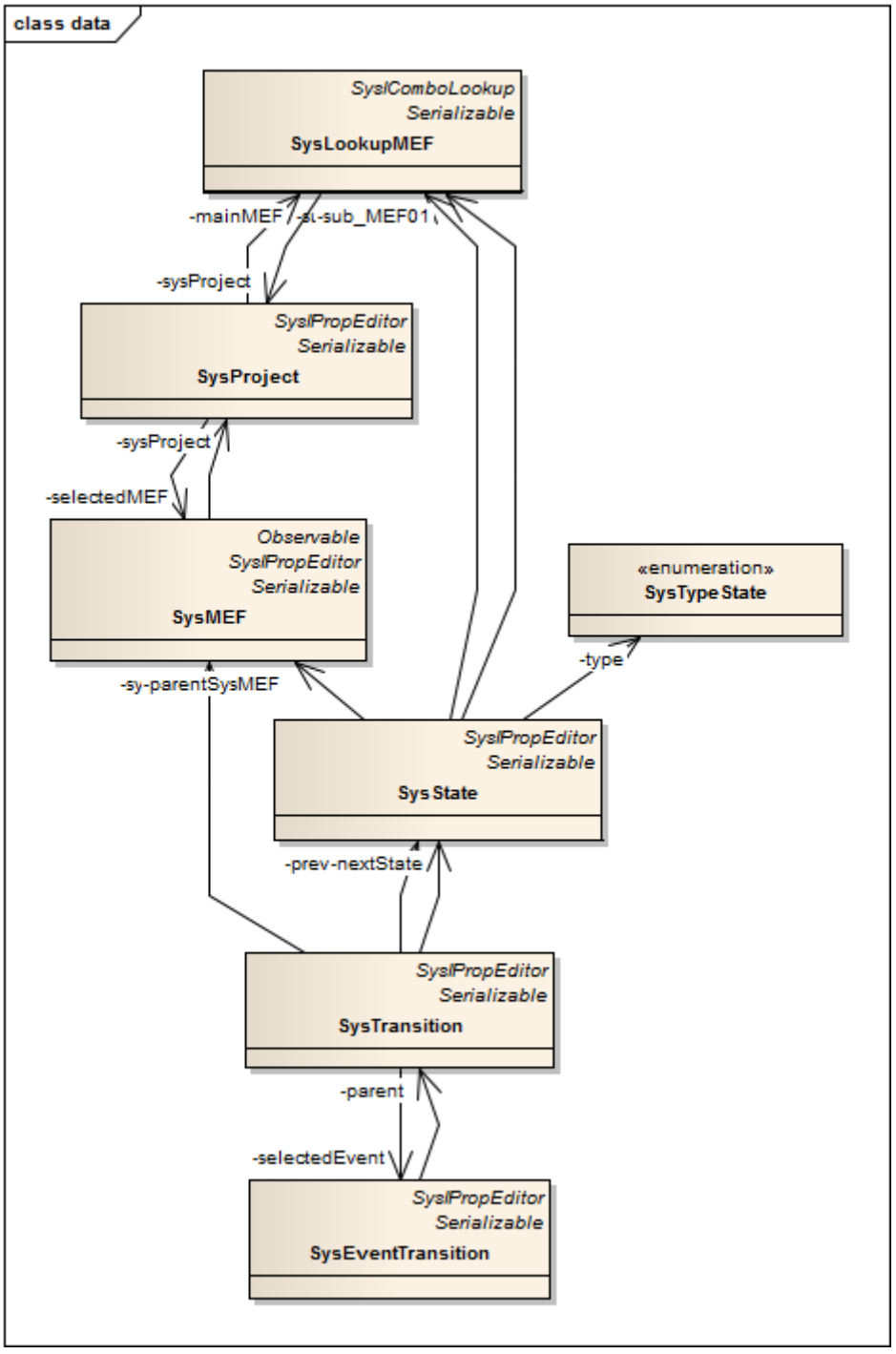

Figura A.8.1 - Diagrama de Relacionamento de Classes

Figura A.8.2 - Descrição da Classe que representa o projeto

Figura A.8.3 - Descrição da Classe que representa a MEF

Figura A.8.4 - Descrição da Classe que representa o estado

Figura A.8.5 - Descrição da Classe que representa a transição

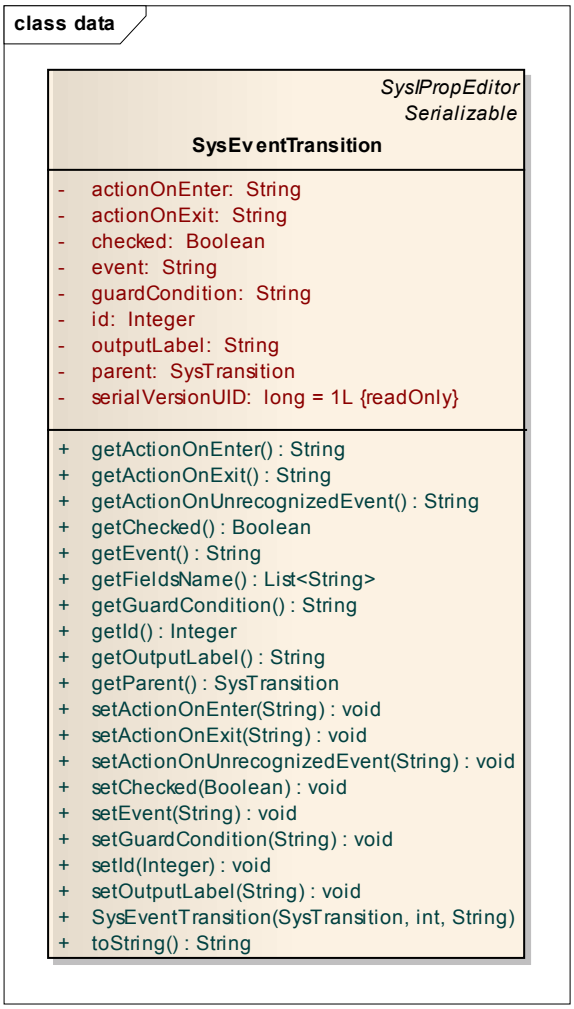

Figura A.8.6 - Descrição da Classe para os eventos das transições

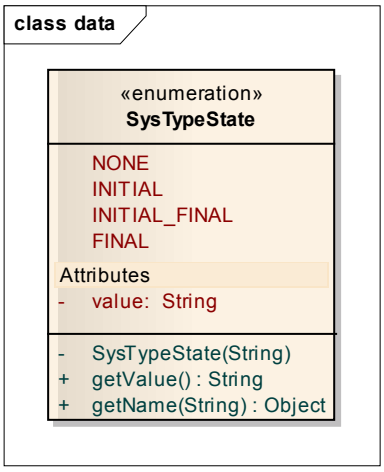

Figura A.8.7 - Descrição da Classe que representa os tipos dos estados

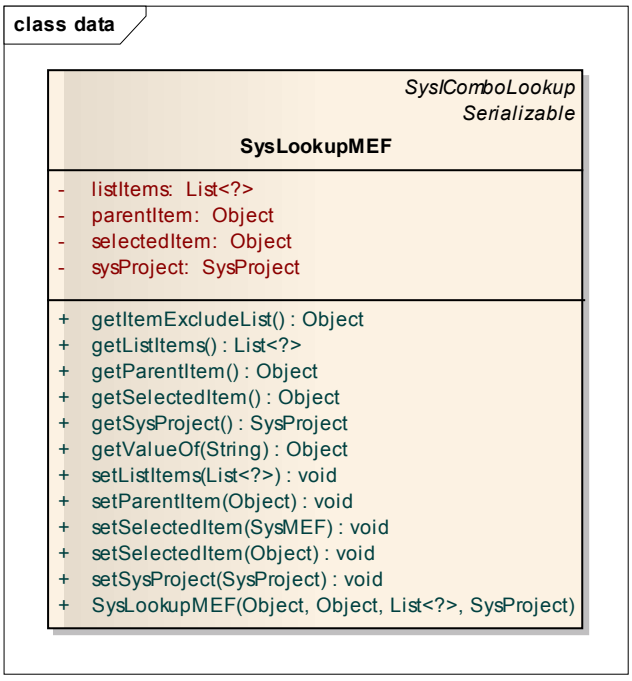

Figura A.8.8 - Descrição da Classe que representa a lista de MEF