MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-16719-TDI/1657

KPlanOO: UM META-MODELO ORIENTADO A OBJETOS PARA DESCRIÇÃO DE DOMÍNIOS E PROBLEMAS DE PLANEJAMENTO

Rodrigo Rocha Silva

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Mauricio Gonçalves Vieira Ferreira, e Nandamudi Lankalapalli Vijaykumar, aprovada em 14 de abril de 2010.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/379R4RP>

INPE
São José dos Campos
2010

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

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-16719-TDI/1657

KPlanOO: UM META-MODELO ORIENTADO A OBJETOS PARA DESCRIÇÃO DE DOMÍNIOS E PROBLEMAS DE PLANEJAMENTO

Rodrigo Rocha Silva

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Mauricio Gonçalves Vieira Ferreira, e Nandamudi Lankalapalli Vijaykumar, aprovada em 14 de abril de 2010.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/379R4RP>

INPE
São José dos Campos
2010

Dados Internacionais de Catalogação na Publicação (CIP)

Silva, Rodrigo Rocha.
Si38k KPlanOO: um meta-modelo orientado a objetos para descrição de domínios e problemas de planejamento / Rodrigo Rocha Silva. – São José dos Campos : INPE, 2010.
xxviii + 224 p. ; (INPE-16719-TDI/1657)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2010.
Orientadores : Drs. Mauricio Gonçalves Vieira Ferreira, e Nandamudi Lankalapalli Vijaykumar.

1. Inteligência artificial. 2. Planejamento. 3. Escalonamento. 4. Representação do conhecimento. 5. Reuso. 6. Ontologia. I.Título.

CDU 004.83

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

Computação Aplicada

Dr. **José Demisio Simões da Silva**

Presidente / INPE / SJCampos - SP

Dr. **Mauricio Gonçalves Vieira Ferreira**

Orientador(a) / INPE / SJCampos - SP

Dr. **Nandamudi Lankalapalli Vijaykumar**

Orientador(a) / INPE / SJCampos - SP

Dr. **Valcir Orlando**

Membro da Banca / INPE / SJCampos - SP

Dr. **Germano de Souza Kienbaum**

Membro da Banca / INPE / SJCampos - SP

Dra. **Selma Shin Shimizu Melnikoff**

Convidado(a) / USP / São Paulo - SP

Aluno (a): Rodrigo Rocha Silva

São José dos Campos, 14 de abril de 2010

"Construímos muros demais e pontes de menos".

Isaac Newton

A minha família, pais, irmão, vó Joana, tia Vivi e minha querida Day,
pelo apoio incondicional,
força, incentivo e amizade sem igual.

AGRADECIMENTOS

Primeiramente agradeço a Deus por me ajudar a superar todos os obstáculos e dificuldades possibilitando mais essa vitória em minha vida.

Não posso deixar de citar algumas pessoas que tiveram grande participação na realização deste trabalho. Primeiramente agradeço ao meu professor, orientador e amigo Mauricio, que desde a graduação vem contribuindo para minha formação com valiosos ensinamentos e oportunidades de crescimento pessoal e profissional. Ao professor Vijay por propiciar a tranqüilidade e confiança necessária para o desenvolvimento deste.

Ao colega Jun Tominaga, que sempre teve paciência para me explicar o que fosse necessário sobre operações de satélites, e aos colegas Francisco, Leonardo e Yasuo por toda ajuda na confecção de artefatos. Não posso deixar de agradecer ao colega Fabrício que durante várias viagens de ônibus me apresentou a área de planejamento, incentivando assim o desenvolvimento deste trabalho. Agradeço também a ajuda, as dicas e os ensinamentos do amigo Joubert que sempre me chamou a atenção para o poder de se criar componentes genéricos.

Agradeço especialmente minha mãe Jô e meu pai Mauricio que sempre incentivaram e apoiaram minhas decisões, e por me criarem em uma base familiar maravilhosa. Também agradeço à minha noiva Dayane, que me deu grande incentivo antes e durante a realização deste trabalho.

À minha avó Joana, meu irmão Diogo, meus tios Viviane e Saulo, que mesmo não entendendo o que eu fazia, sempre torceram para que tudo desse certo.

E a todos os meus colegas de trabalho, que estavam sempre dispostos a me escutar e trocar idéias.

RESUMO

Um problema evidente no desenvolvimento de Sistemas para Planejamento Automático é a modelagem e o reuso de domínios e problemas, devido à falta de ferramentas adequadas. Este trabalho propõe o desenvolvimento de um meta-modelo orientado a objetos (OO), genérico o suficiente para a descrição de vários domínios de aplicação da área de Planejamento Automático, como uma alternativa às atuais linguagens utilizadas em representação de problemas de planejamento. Adotou-se a disciplina de engenharia de domínio no desenvolvimento do modelo, com objetivo de promover a adoção sistemática da prática de reutilização de software em um nível de abstração mais alto que o de codificação. O modelo foi elaborado através de atividades de análise e projeto de domínio, possibilitando o desenvolvimento de uma infra-estrutura reutilizável, que captura a estrutura e o conhecimento de uma família de aplicações de Planejamento Automático. É apresentada a construção de um protótipo para servir de interface para captação dos dados do meta-modelo e também uma especificação XML *Schema* (XSD) que contempla os conceitos de modelagem, verificação e restrições do mesmo, para possibilitar a descrição de domínios de planejamento utilizando a proposta deste trabalho em XML. Apresenta-se o uso do modelo como ferramenta para descrição de três domínios e problemas de planejamento. São eles: um domínio de Logística apresentado em paralelo a apresentação da estrutura estática da proposta; o Mundo de Blocos; e o domínio de rastreio de satélite do Instituto Nacional de Pesquisas Espaciais. Também é apresentada uma extensão do modelo feita para servir como descrição do domínio de simulação de um satélite, promovendo a experimentação das metas propostas anteriormente.

KPLANOO: AN OBJECT-ORIENTED META-MODEL FOR DESCRIBING DOMAINS AND PROBLEMS FOR PLANNING

ABSTRACT

An evident problem from the development of Automatic Planning Systems is the modeling and reuse of domains and problems, due to lack of appropriate tools. In this context, the present work proposes the development of an object oriented (OO) meta-model, generic enough to allow the description of several domains from the Automatic Planning area, bringing an alternative to currently languages used for the representation of planning problems. The Domain Engineering discipline has been adopted in the model development, aiming to promote the systematic adoption of software reuse practice in a higher abstraction level than the coding. A model has been elaborated through domain analysis and project activities, enabling the development of a reusable infra-structure, that captures the structure and knowledge from a family of Automatic Planning applications. It's presented a prototype building to act as an interface for meta-model data collecting and also an XML Schema (XSD) that represents the modeling concepts as well as their validations and constraints, to enable a XML description of planning domains utilizing this work proposition. It's also presented the use of this model as a tool for the description of three domains and planning problems: a logistic domain problem, presented in parallel to the proposition static structure, the Block World and the National Institute for Space Research satellite tracking domain. Besides, it's presented an extension of the model as a domain description for a satellite simulation, promoting the experimentation of the earlier proposed aims.

LISTA DE FIGURAS

Pág.

Figura 2.1 - Modelo conceitual de planejamento 16
Figura 2.2 - Descrição de um problema em STRIPS 21
Fonte: Kucinskis (2007, p.21) 21
Figura 2.3 - Grafo de planejamento para o mundo de blocos 25
Figura 2.4 - Estrutura da geração de planos por satisfatibilidade 27
Figura 2.5 - Arquitetura simplificada de um KBS. 32
Figura 2.6 - Processo de planejamento clássico 34
Figura 2.7 - Tipos de ontologias, segundo seu nível de dependência em relação a uma tarefa ou ponto de vista particular 49
Figura 3.1 - Estrutura básica de um problema de planejamento em PDDL .. 56
Figura 3.2 - Exemplo de um arquivo de domínio de um satélite em PDDL .. 57
Figura 3.3 - Arquivo de problema em PDDL para o domínio do satélite 58
Figura 3.4 - Exemplo de descrição de domínio do mundo de blocos em OCL 61
Fonte: Kucinskis (2007, p.51). 61
Figura 4.1 - Abstrações identificadas para a descrição de domínios e problemas de planejamento representadas pelo meta-modelo. 73
Figura 4.2 - Estrutura estática do KPlanOO 81
Figura 4.3 - Destaque classe Type 83
Figura 4.4 - Diagrama de objetos da classe Type para o domínio de logística 84
Figura 4.5 - Destaque classe Element 84
Figura 4.6 - Destaque classe State 85
Figura 4.7 - Diagrama de objetos da classe State do domínio de logística .. 85
Figura 4.8 - Destaque classe ExpressionEvaluation 86
Figura 4.9 - Destaque classe SituationObject 86
Figura 4.10 - Destaque da interface IActivity e da classe Activity 88
Figura 4.11 - Diagrama de objetos da classe Function do domínio de logística 88
Figura 4.12 - Destaque da classe Action 89
Figura 4.13 - Diagrama de objetos da ação “dirigir” do domínio de logística com seus parâmetros e sua pré-condição. 89
Figura 4.14 - Diagrama de objetos da ação “dirigir” do domínio de logística com seus parâmetros e seu efeito. 90
Figura 4.15 - Destaque classe Domain 91
Figura 4.16 - Ilustração do estado inicial para um problema de planejamento no domínio logística. Fonte: Vaquero (2007, p.119). 92
Figura 4.17 - Ilustração do estado objetivo de um problema de planejamento no domínio logística. Fonte: Vaquero (2007, p.119). 92
Figura 4.18 - Destaque classe Init 93

Figura 4.19 - Diagrama de objetos dos estados dos objetos para o estado Inicial de um problema de planejamento do domínio de logística, descrito anteriormente 94
Figura 4.20 - Diagrama de objetos de funções a serem executadas para o estado Inicial de um problema de planejamento do domínio de logística, descrito anteriormente 94
Figura 4.21 - Destaque classe Goal 95
Figura 4.22 - Diagrama de objetos do estado objetivo de um problema de planejamento do domínio de logística, descrito anteriormente 95
Figura 4.23 - Destaque classe Goal 96
Figura 4.24 - Diagrama de objetos da modelagem um problema de planejamento do domínio logística, descrito anteriormente. 96
Figura 4.25 - Destaque classe Domain 97
Figura 4.26 - Estrutura de herança presente no KplanOO 98
Figura 5.1 – Possibilidades de entradas e saídas para o KPlanOO. 102
Figura 5.2 - Diagrama de pacotes de protótipo 108
Figura 5.3 - Diagrama de classes do pacote br.inpe.crc.sacs,kplanoo.dao 109
Figura 5.4 - Diagrama de classes do pacote br.inpe.crc.sacs,kplanoo.controller 110
Figura 5.5 - Diagrama de classes do pacote br.inpe.crc.sacs,kplanoo.view112
Figura 5.6 - Tela principal do protótipo 114
Figura 5.7 - Tela de especificação de tipos 115
Figura 5.8 - Tela de especificação de estados 116
Figura 5.9 - Tela de especificação de funções 117
Figura 5.10 - Tela de especificação de expressões de avaliação 118
Figura 5.11 - Tela de especificação de ações 119
Figura 5.12 - Tela de definição de domínios 120
Figura 5.13 - Tela de declaração de problemas 122
Figura 5.14 - Estrutura geral do XML Schema 125
Figura 5.15 - Estrutura para Action do XML Schema 126
Figura 5.16 - Estrutura para problem do XML Schema 126
Figura 5.17 - Representação parcial do elemento raiz (*KPlanOO*) do XKPS 127
Figura 5.18 - Representação do tipo Domain do XKPS 128
Figura 5.19 - Especificação de Type e sua coleção no XKPS 129
Figura 6.1 - Ilustração do mundo de blocos 132
Figura 6.2 - Diagrama de objetos da definição do tipo block e dos estados de domínio de mundo de blocos 133
Figura 6.3 - Diagrama de objetos da definição da ação pick-up 134
Figura 6.4 - Descrição XML para ação put-down 135
Figura 6.5 - Descrição XML para ação *stack* com sua pré-condição 136
Figura 6.6 - Descrição XML do efeito da ação stack e fechamento de tags abertas na Figura 6.5 137
Figura 6.7 - Descrição XML da pré-condição da ação unstack 138
Figura 6.8 - Ilustração do cenário inicial para o problema do Mundo de Blocos 139

Figura 6.9 - Ilustração do estado objetivo para problema do Mundo de Blocos 139
Figura 6.10 - Diagrama de objetos com a definição dos objetos para o problema do mundo de blocos. 139
Figura 6.11 - Diagrama de objetos com a definição do cenário inicial do problema do mundo de blocos 140
Figura 6.12 - Diagrama de objetos com a definição do cenário objetivo do problema do mundo de blocos 141
Figura 6.13 - Tela de Definição de ações com dados da ação SendTc preenchidos 144
Figura 6.14 - Trecho da descrição XML da ação SendTC 145
Figura 6.15 - Diagrama de objetos com as ações do domínio de rastreio de satélites do INPE. 145
Figura 6.16 - Trecho da descrição XML do domínio de rastreio de satélites do INPE com a definição de tipos e funções 146
Figura 6.17 - Trecho da descrição XML do problema para o domínio de rastreio de satélites com a definição dos objetos envolvidos no problema 147
Figura 6.18 - Tela do protótipo para definição de problemas com destaques 148
Figura 6.19 - Trecho da descrição XML com definição de funções a serem executadas no início do processo de planejamento. 149
Figura 6.20 - Trecho da descrição XML com definição de intervalo de tempo representado pela expressão “at”. 150
Figura 6.21 - Definição do objetivo para o problema de rastreio de satélites do INPE 151
Figura 6.22 - Parte do KPlanOO com a especialização para descrever um domínio de simulação de satélites 153
Figura 6.23 - Tela principal do protótipo especializada para o simulator 155
Figura 6.24 - Tela de especificação de parâmetros 156
Figura 6.25 - Tela de especificação de curvas de funções 156
Figura 6.26 - Tela de especificação de curvas de regras 157
Figura 6.27 - Arquitetura do novo sistema para planejamento autônomo de operações 158
Figura 7.1 - Linha de tempo com principais formas de representações do conhecimento para domínios de planejamento automático 164

LISTA DE TABELAS

Pág.

Tabela 2.1 - Eras do Planejamento Automático (Clássico e Neoclássico) com os Principais Planejadores e Linguagens 30

Tabela 3.1 - Comparação de características entre STRIPS e ADL..... 54

Tabela 3.2 - Comparação de características entre linguagens de descrição de domínios de planejamento..... 67

Tabela 4.1 - Comparação das características das linguagens de descrição de domínios de planejamento com KPlanOO 99

LISTA DE SIGLAS E ABREVIATURAS

ABSTRIPS	Abstraction-Based STRIPS
ADL	Action Description Language
AIAA	American Institute of Aeronautics and Astronautics
AIPS	Arificial Intelligence Planning Systems
AML	ASPEN Modeling Language
API	Application Programming Interface
ASPEN	Automated Scheduling and Planning Environment
BSD	Berkeley Software Distribution
CBA	Cuiabá
CBERS	China Brazil Earth Resource Satellites
COMAV	Computador Avançado
CCS	Centro de Controle de Satélites
CRC	Centro de Rastreio e Controle de Satélites
CSV	Comma Separated Values
DAO	Data Access Objects
DEA	Divisão de Eletrônica Aeroespacial
DOM	Document Object Model
DTD	*Document Type Definition*
EC	Engenharia do Conhecimento
ECP	European Conference on Planning
GPS	General Problem Solver

GIPO	Graphical Interface for Planning with Objects
GUI	Grafical User Interface
GRAPHPLAN	Planejador em Grafo
HSP	Planejador de Pesquisa Heurística
KBS	Knowledge Based Systems
KPlanOO	Knowledge of Planning Oriented Objects
LGPL	Lesser General Public License
IA	Inteligência Artificial
ICAPS	International Conference on Automated Planning and Scheduling
ICKEPS	Competição Internacional de Engenharia do Conhecimento para Planejamento e Escalonamento
INPE	Instituto Nacional de Pesquisas Espaciais
IPC	International Planning Competition
ITSIMPLE	Ambiente integrado de modelagem e análise de domínios de planejamento automático
JPL	Jet Propulsion Laboratory
NASA	National Aeronautics and Space Administration
NOAH	Nets of Action Hierarchies
OO	Orientação a Objetos
OCL	Object-Centered Language
PDDL	Planning Domain Definition Language
PlanIPOV	Planejamento Inteligente de Planos de Operação de Vôo
POV	Plano de Operações em Vôo
RASSO	Resources Allocation Service for Scientific Opportunities

RASSO_ml	RASSO modeling language
SCD	Satélite de Coleta de Dados
SPACEOPS	International Committee on Technical Interchange for Space Mission Operations and Ground Data Systems
SUBORD	Grupo de Supervisão de Bordo
STRIPS	Stanford Research Institute Problem Solver
SwingX	Swing eXtension
UML	Unified Modeling Language
TOVE	Toronto Virtual Enterprise
XKPS	XML KPlanOO Schema
XML	Extended Markup Language
XS	XML Schema
XSD	XML Schema Definition
XSDL	XML Schema Definition Language

SUMÁRIO

Pág.

1 INTRODUÇÃO 1

1.1. Motivação do trabalho de pesquisa 3

1.2. Objetivos do trabalho de pesquisa 8

1.3. Metodologia de trabalho e organização desta dissertação 11

2 ENGENHARIA DO CONHECIMENTO 13

2.1. Planejamento automático 13

2.1.1. Conceito 13

2.1.2. Histórico 20

2.2. Engenharia do conhecimento no contexto do planejamento automático 32

2.2.1. Conceito 32

2.2.2. Histórico 38

2.3. Análise de domínio 40

2.3.1. Conceitos 41

2.4. Representação do conhecimento e ontologias 43

2.5. Definição de Ontologia 46

2.6. Orientação objetos 50

2.7. Considerações 52

3 TRABALHOS CORRELATOS 53

3.1. Ferramentas para modelagem de domínios 53

3.1.1. STRIPS: O precursor 53

3.1.2. ADL: A primeira evolução 54

3.1.3. PDDL: o padrão que surgiu da união entre STRIPS e ADL 55

3.1.4. UML – Unified Modeling Language 59

3.1.5. OCL: Uma linguagem centrada em objetos 60

3.1.6. AML: Linguagem para modelagem de um planejador para missões espaciais. 62

3.1.7. RASSO_ml: Forte integração entre programação do sistema e o processo de planejamento. 63

3.1.8. ITSIMPLE: Ambiente integrado de modelagem e análise de domínios de planejamento automático 64

3.1.9. Mecanismos de apoio a engenharia do conhecimento 65

3.2. Considerações 67

4 UM META-MODELO ORIENTADO A OBJETOS PARA DESCRIÇÃO DE DOMÍNIOS E PROBLEMAS DE PLANEJAMENTO 69

4.1. Características dos domínios da área de planejamento automático 71

4.2. O meta-modelo proposto 71

4.2.1. Mecanismo de modelagem 75

4.2.2. Fundamentação do meta-modelo 76

4.2.3. Estrutura estática 80

4.3. Considerações 99

5 IMPLEMENTAÇÃO DE INTERFACES DE ENTRADA E VALIDAÇÃO DE DADOS. 101

5.1. Ferramentas utilizadas 102

5.1.1. Desenvolvimento da interface gráfica 102

5.1.2. Serializar objetos para XML 103

5.1.3. XSD (XML Schema Definition) 104

5.2. O Protótipo desenvolvido 107

5.2.1. Pacote br.inpe.crc.sacs.kplanoo.model 108

5.2.2. Pacote br.inpe.crc.sacs.kplanoo.dao 108

5.2.3. Pacote br.inpe.crc.sacs.kplanoo.controller 109

5.2.4. Pacote br.inpe.crc.sacs.kplanoo.gui 111

5.2.5. Pacote br.inpe.crc.sacs.kplanoo.view 111

5.2.6. Especialização do protótipo 112

5.2.7. Inclusão de restrições de integridade no protótipo 113

5.2.8. Uso do protótipo 113

5.2.9. Utilizando as funcionalidades do protótipo 114

5.2.9.1. Criação de Types 115

5.2.9.2. Definindo States 115

5.2.9.3. Criando Functions 116

5.2.9.4. Criando expressões de avaliação 117

5.2.9.5. Especificando ações 118

5.2.9.6. Definindo um domínio 120

5.2.9.7. Declaração de problemas 121

5.3. Um XML *Schema* para descrições de domínios e problemas de planejamento em KPlanOO 122

5.3.1. Regras usadas na concepção do XKPS 123

5.3.2. XKPS: Um XML Schema para KPLANOO 125

5.4. Considerações 129

6 ESTUDO DE CASO E ANÁLISE DOS RESULTADOS OBTIDOS 131

6.1. Domínio mundo de blocos 132

6.1.1. Modelando o domínio 132

6.1.2. Modelando o problema do mundo de blocos 138

6.2. Domínio de rastreio de satélites do INPE 141

6.2.1. Modelando o problema de rastreio de satélites 147

6.3. KPlanOO estendido para descrever um domínio de simulação de satélites 151

6.3.1. Modelando um exemplo simples 154

6.3.2. Especialização do protótipo para o domínio de simulação de satélites 154

6.3.3. Definindo parâmetros 155

6.3.4. Criando curvas 156

6.3.5. Definindo regras 157

6.4. Uma arquitetura integrada para projetos de planejamento de operações espaciais 157

7 CONCLUSÕES E TRABALHOS FUTUROS 161

7.1. Conclusões 161

7.1.1. Pontos desfavoráveis 164

7.2. Trabalhos futuros 165

REFERÊNCIAS BIBLIOGRÁFICAS..... 167

APÊNDICE A – DEFINIÇÃO EM XML DO DOMÍNIO DO MUNDO DE BLOCOS BASEADA NO XKPS 179

APÊNDICE B – DESCRIÇÃO XML DO DOMÍNIO DE RASTREIO DE SATÉLITES DO INPE 185

APÊNDICE C – DESCRIÇÃO XML DO PROBLEMA PARA O DOMÍNIO DE RASTREIO DE SATÉLITES DO INPE..... 209

APÊNDICE D – DESCRIÇÃO XML DO DOMÍNIO DE SIMULAÇÃO DE SATÉLITES..... 215

1 INTRODUÇÃO

Após o sucesso dos satélites SCD-1 e SCD-2 lançados em 1993 e 1998 respectivamente, que ultrapassaram em muito sua vida útil estimada, o Instituto Nacional de Pesquisas Espaciais (INPE) obteve o reconhecimento de sua capacitação na área espacial e iniciou um trabalho de amadurecimento no desenvolvimento de satélites mais avançados e de aplicações diversas.

Com os satélites da série China-Brazil Earth Resources Satellite (CBERS-1, 2 e 2B) voltados ao sensoriamento remoto, o Brasil aprimorou sua capacitação na engenharia de construção e das atividades de operação de satélites.

O Centro de Rastreio e Controle de Satélites (CRC) é o responsável, no INPE, pelas tarefas de monitoramento e controle em órbita de satélites. Uma tarefa estratégica na área de controle de solo de satélites artificiais é o planejamento das atividades operacionais envolvidas, de forma a usar o mínimo de recursos e gerar o máximo de retorno da missão espacial (dados científicos, dados tecnológicos e imagens), sem comprometer a segurança do satélite. Essa tarefa é complexa, pois envolve vários níveis de tomada de decisão (GONÇALVES, 2006).

Neste contexto as técnicas de Planejamento Automático e Escalonamento, provenientes da área de Inteligência Artificial, se apresentam como uma solução em potencial a ser explorada, visando a automatização do controle e operação de satélites (KUCINSKIS, 2007), pois como a maioria das instituições espaciais o INPE, por intermédio do CRC, tem como grande preocupação a diminuição do elevado custo de manutenção de suas operações espaciais. Assim através da automatização seria possível a redução do custo total das missões através da diminuição das equipes de operação, aumento da confiabilidade, bem como a obtenção de respostas mais rápidas e o reaproveitamento do conhecimento obtido.

Porém as pesquisas na área de Planejamento Automático permaneceram por muitos anos concentradas em resolver problemas clássicos e, durante este período, diversas técnicas de solução automática de problemas foram desenvolvidas. Além disto, a partir do final dos anos 90, ocorreu um aumento do uso das técnicas desta área para resolver problemas reais, tais como o controle de redes de satélites (KHATIB et al., 2003).

Os três maiores obstáculos para o Planejamento Automático de problemas reais que envolvem o controle de satélites são: (i) especificação dos requisitos, (ii) modelagem (geralmente na modelagem de ações) e (iii) análise de domínios, já que estes processos são considerados gargalos no desenvolvimento de aplicações reais (CARNIELLO, 2008).

Um quarto obstáculo, não evidenciado em Carniello (2008), é a não aplicabilidade de conceitos fundamentais da engenharia de software em projetos da área espacial. Enfatiza-se as seguintes premissas:

- o desenvolvimento de artefatos genéricos passíveis de reuso;

- a construção de aplicações a partir da integração de componentes corretos e possuidores de interfaces bem definidas;

Diante de tais obstáculos, a Engenharia de Conhecimento passou a ter uma grande importância na concepção dos problemas da área espacial. A especificação, modelagem e análise se tornaram essenciais para o melhor entendimento e classificação dos domínios e problemas de planejamento; a especificação, modelagem e análise dos domínios se tornam essenciais.

Devido aos fatos colocados e a necessidade de ferramentas que dêem auxilio durante a modelagem de domínios de planejamento, o presente trabalho investiga os problemas envolvidos na interação de informação pertinente à gestão do conhecimento do planejamento e escalonamento de atividades,

principalmente no âmbito da área espacial, oferecendo subsídios para a modelagem de domínios e problemas de planejamento.

É definindo um meta-modelo orientado a objetos para a descrição de domínios e problemas da área de Planejamento Automático, dispondo também de um vocabulário base para a descrição de problemas e domínios desta área, visando auxiliar o mapeamento do conhecimento necessário.

A proposta apresentada considera que a falta de uma padronização na representação do conhecimento sobre planejamento de atividades para o controle de satélites dificulta a compreensão por parte de todos os atores envolvidos no processo.

O meta-modelo proposto foi concebido inicialmente para domínios de planejamento da área espacial e domínios clássicos que serviram de base para o desenvolvimento inicial. Devido à iniciativa do INPE na automatização das operações de vôo, foi dado um foco maior na validação de domínios deste contexto, dentre eles um domínio de rastreio de satélite.

A descrição deste domínio atualmente é feita em parte por arquivos gerados a cada três semanas, contendo dados de previsão das próximas passagens do satélite correspondente sobre cada uma das estações terrenas envolvidas no rastreio (ORLANDO; KUGA, 2007).

1.1. Motivação do trabalho de pesquisa

Devido à importância dos satélites para incontáveis aplicações de interesse da sociedade brasileira, como a monitoração de vegetação, 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.

O controle de satélites é feito através de um planejamento gerado para cada satélite, sob a forma de um roteiro de operação em rotina denominado Plano de Operação em Vôo (POV)[1], que abrange tipicamente, um período que pode envolver vários dias, dependendo do satélite envolvido e da fase operacional (rotina ou crítica). Este planejamento é baseado em previsões de passagem obtidas através do software de dinâmica orbital que opera em tempo não real, a partir de arquivos de dados de telemetria e medidas de rastreio gravados pelo software aplicativo de tempo real durante passagens de satélites.

Atualmente, o INPE está operando quatro satélites: SCD1, SCD2 (Satélites de Coleta de Dados) e CBERS-2b (China-Brazil Earth Observation Satellites), além do satélite francês COROT que é rastreado por uma estação específica em Alcântara apenas para recepção de dados de telemetria e de carga útil. Com a previsão do aumento do número de satélites a serem controlados nos próximos anos, e face aos limitados recursos financeiros destinados a operação de satélites, a busca de soluções de automatização se faz importante.

O CRC já vem a certo tempo trabalhando em aspectos relacionados a automatização do controle de satélites, tendo desenvolvido programas que geram automaticamente o POV para cada um de seus satélites, a partir de um arquivo de predição de passagens gerado pelo software de dinâmica orbital. Cada um deles, foi desenvolvido para atender à operação de um satélite específico (ORLANDO; KUGA, 2007). Porém, no cenário atual, não é mais suficiente que uma aplicação seja correta e atenda aos requisitos funcionais

[1] O plano de operação em vôo lista cronologicamente, para um determinado período, todas as ações de controle a serem executadas com o satélite em cada passagem futura contida no arquivo de previsão de passagem usado como entrada em seu processo de geração, como por exemplo: telecomandos a serem enviados, telemetrias a serem monitoradas e medidas de rastreio a serem executadas. (Orlando and Kuga, 2007)

apenas para os quais foi desenvolvida. Idealmente, essa mesma aplicação deve atender a novos requisitos ou se adaptar para atender a aplicações semelhantes dentro do mesmo domínio do sistema com o passar do tempo.

Nesse contexto, além das perspectivas *de fazer a coisa certa* e *fazer de forma certa*, os sistemas para automatização do controle de satélites devem estar preparados para futuras necessidades (*fazer a próxima coisa*). Motivado por essa questão o CRC tem desenvolvido de vários trabalhos de pesquisa, envolvendo dissertações de mestrado e teses doutorados, com a temática de automatização de operações e controle de operações espaciais.

A proposta de Gonçalves (2006), aborda uma arquitetura de um sistema de Planejamento Inteligente de Planos de Operação de Vôo (PlanIPOV), focada na geração automática de POVs para a fase operacional de rotina de satélites.

Uma arquitetura Multi-Agente de Planejamento de controle de Satélites, nomeada como MAPSat, foi desenvolvida por Carniello (2008). Essa arquitetura é constituída por agentes que gerenciam a alocação de recursos de solo para o rastreio de múltiplos satélites e planejam as operações de controle destes satélites.

Também foi desenvolvido um trabalho por Tominaga (2010), que consiste em um simulador de satélites, cuja finalidade é a validação de software de planejamento de operações em vôo experimental. O simulador é um sistema especialista baseado em regras, que atualiza seu estado interno a partir de um estado inicial combinado a uma fila de eventos. O histórico de estados é exportado para diagnosticar o plano, rejeitando ou aprovando o mesmo.

Assim visando à automatização do processo de geração dos POVs, os trabalhos citados utilizam a técnica de planejamento e escalonamento da área de inteligência artificial, já que a comunidade de pesquisa desta técnica após

um longo período focada apenas na solução de domínios clássicos de planejamento (domínios conhecidos e simplificados) passou a demonstrar, na década de 90, um grande interesse em aplica-lá em problemas reais de engenharia onde a complexidade se torna um grande complicador.

Porém um ponto importante no desenvolvimento destas técnicas foi, por muito tempo, negligenciado – embora fosse um dos aspectos fundamentais para o progresso do Planejamento Automático: “Não importa o quão eficientes ou poderosas são as técnicas e mecanismos de planejamento, o sucesso de suas aplicações depende do conhecimento do domínio que é fornecido a elas. Se o modelo do domínio fornecido é falho, o resultado da aplicação das técnicas será também falho” (MCCLUSKEY et al., 2003).

Devido à complexidade de sistemas reais de planejamento, torna-se de alta relevância os estudos sobre métodos, técnicas, ambientes e ferramentas de suporte ao processo de especificação e projeto dos domínios de planejamento.

Como observado por Cardoso (2005) há, ainda, uma grande dificuldade em descrever os domínios reais, através da PDDL (*Planning Domain Definition Language*), uma linguagem de descrição de domínios em planejamento automático criada por Drew McDermott em 1998 que, atualmente, é tida como padrão para a descrição de domínios e problemas de planejamento. No trabalho de Kucinskis (2007) foi gerado um pseudo-script chamado de RASSO_ml para gerar descrições de domínios de planejamento, dada as restrições do que existe atualmente, no âmbito da modelagem de conhecimento para a área de planejamento.

Assim a motivação deste trabalho é oferecer subsídios para a modelagem de domínios e problemas de planejamento para as aplicações que utilizam esta técnica. É esperado que estes subsídios se constituam em uma ferramenta de auxilio para as aplicações que visam à automatização da geração de POV’s e

consequentemente a automatização, na área de operação de satélites artificiais.

Dessa forma, além das perspectivas de modelar domínios e problemas atuais (fazer a coisa certa) e fazer da maneira mais eficiente possível (fazer de forma certa), é proposto um modelo que esteja preparado para descrever novos domínios e problemas atendendo futuras necessidades (a próxima coisa), no contexto da automatização do controle de satélites:

I. **Faça a coisa certa:** esse tópico está diretamente ligado à validação ou, em outras palavras, à garantia de que o que está sendo descrito (modelado) é o que realmente é desejado. Nesse novo cenário, não somente os domínios da aplicação precisam ser compreendidos, como esses domínios podem mudar rapidamente durante o processo de concepção. Além disso, essa abordagem faz com que a comunicação, a interface e a documentação desses domínios possam ocorrer sem ambigüidades, inconsistências e contradições, tornando os domínios mais compreensíveis e fazendo com que as suas aplicações tenham uma maior mantenabilidade. Por fim, esse tópico adiciona uma nova perspectiva ao conceito de reutilização: ao invés de reutilizarmos somente componentes de código, é importante que possamos desenvolver e reutilizar elementos mais abstratos, como, por exemplo, a experiência embutida em padrões de solução, arquiteturas de estruturação de elementos do domínio e, idealmente, estruturas capazes de capturar, de forma explícita, o conhecimento disponível em um conjunto de problemas.

II. **Faça de forma certa:** esse tópico está relacionado à verificação, ou melhor, à garantia de que o que é desejado está sendo construído de maneira correta. Até então, sistemas de planejamento têm sido usados, principalmente, como ferramentas de apoio e automatização de tarefas

no contexto do controle de satélites. Nesse novo cenário, vários sistemas computacionais poderão vir realizar algumas atividades do CRC, de modo que o seu processo de desenvolvimento deve ser conduzido com rigor de outras disciplinas da engenharia. É importante salientar, porém, que as linguagens de representação de domínios são ferramentas que devem servir aos propósitos de um desenvolvedor humano, e não o contrário. Desse modo, é fundamental que o emprego abordagens formais seja conduzido com a visão mais abrangente, de que requisitos de naturezas diferentes requerem abordagens diferentes.

III. **Faça a próxima coisa:** esse tópico está relacionado com a propriedade de adaptabilidade em suas várias formas: (i) extensibilidade - diz respeito à habilidade do modelo ser estendido sem que isso ocasione mudanças profundas em sua estrutura, como, por exemplo, para abrigar a introdução de uma nova abstração do sistema a ser contemplada pelos novos satélites; (ii) flexibilidade – o modelo deve ser capaz de promover mudanças em seus mecanismos de interface ou em suas tecnologias de armazenamento, sem que isso tenha impacto na arquitetura dos sistemas; (iii) escalabilidade - a demanda dessa característica está relacionada à necessidade de adaptação ao novo cenário de implementação, de aplicações e de crescente heterogeneidade das plataformas de hardware e sistemas operacionais. O modelo foi construído a fim de prever mudanças de tecnologias e de plataformas, além de considerar previamente a necessidade da redundância de informações.

1.2. Objetivos do trabalho de pesquisa

Estudando os princípios e métodos para o desenvolvimento e construção de Sistemas de Planejamento Automático para a área espacial, o trabalho proposto, apresenta o desenvolvimento de um meta-modelo orientado a

objetos que tenta ser genérico o suficiente para descrever o maior número possível de domínios relacionados a missões espaciais. Procurando ser uma alternativa às atuais linguagens utilizadas em representação de problemas de planejamento, permitindo maior flexibilidade nas definições do domínio através da componentização do modelo, possibilitando assim a inclusão e especialização do modelo através de suas abstrações e relacionamentos, já que linguagens como a PDDL é incapaz de se definir, por exemplo, a prioridade dos eventos exógenos[2] quando o tempo de um evento superpõe o tempo de outro evento (CARDOSO, 2005).

Sendos os objetivos principais do trabalho:

- Criar uma alternativa para descrição de diversos domínios de planejamento;

- Permitir maior flexibilidade nas definições do domínio através da componentização do modelo;

- Ser amplo representando uma grande variedade de problemas, mas restritivo o bastante para permitir que algoritmos eficientes operem sobre ele;

- Facilitar o correto mapeamento dos dados de entrada necessários para a geração da definição de domínios;

- Facilitar a entrada de dados através de interfaces amigáveis;

- Armazenamento e reaproveitamento de conhecimento;

[2] Eventos exógenos são eventos fora do controle do planejador, mas que afetam o estado do modelo. Em um plano para controlar um carro com o objetivo de levar seus passageiros de um ponto a outro em uma

No modelo proposto são contemplados os requisitos de modelagem necessários para aplicações de planejamento. Para que esta abordagem seja válida como representação do conhecimento, o modelo proposto procura ser suficientemente amplo, reapresentando uma grande variedade de problemas, mas restritivo o bastante para permitir que algoritmos eficientes operem sobre ele, estabelecendo restrições e imposições em sua estrutura. Viabiliza a modelagem de problemas que envolvam tempo e restrições de recursos na geração de planos.

A abordagem que o trabalho apresenta auxilia também na representação do conhecimento do domínio de planejamento, facilitando assim o correto mapeamento dos dados de entrada necessários para a geração da definição do domínio. Possibilitando a representação do conhecimento ser obtida através de interfaces humano computador e interfaces de comunicação entre sistemas, sendo que estas poderão ser criadas conforme necessidades e motivações de uso do modelo proposto, chegando-se a interfaces mais amigáveis do que as linguagens atuais.

São também objetivos deste trabalho:

1) Mostrar que alguns exemplos significativos descritos em trabalhos relacionados a automatização de controle de satélites são cobertos pelo modelo proposto e que, a partir deles, seja possível reutilizar elementos pré-existentes na construção de novos domínios e problemas de planejamento.

2) Apresentar a viabilidade da construção de interfaces humano computador e de comunicação para o modelo proposto através de um protótipo, isto é, uma implementação limitada que forneça

cidade, por exemplo, a abertura e o fechamento dos semáforos encontrados pelo caminho são eventos exógenos, pois estão fora do controle do planejador. (Kucinskis, 2007)

funcionalidades básicas para que seja possível a descrição de domínios e problemas de planejamento baseado no modelo proposto.

1.3. Metodologia de trabalho e organização desta dissertação

Esta dissertação traz o levantamento geral sobre Planejamento Automático e principalmente sobre as Engenharias de Conhecimento e de Domínios, Meta-Modelos e Ontologias aplicados a esta área. Também foi feito um estudo sobre os planos de operação de vôo dos satélites controlados pelo INPE. Para tal, o seu conteúdo foi organizado da seguinte maneira:

CAPÍTULO 2 – ENGENHARIA DO CONHECIMENTO: far-se-á uma descrição dos principais conceitos envolvidos no processo de especificação, modelagem e análise de domínios de planejamento e escalonamento de atividades. Neste capítulo são apresentadas áreas como Planejamento Automático, Engenharia do Conhecimento aplicada a Planejamento e Análise de Domínio. Por último são descritos conceitos sobre Ontologias e Orientação a Objetos tais conceitos estão inseridos no contexto da modelagem e descrição de domínios.

CAPÍTULO 3 – TRABALHOS CORRELATOS: este capítulo discute algumas linguagens e modelos para descrição de domínios que de alguma forma se assemelham com o *KPlanOO,* e também para a descrição de sistemas em geral. Essas linguagens e modelos são descritos brevemente com o objetivo de levantar as necessidades de representação e de ferramentas que auxiliem no processo de entendimento, caracterização e modelagem de domínios envolvidos no planejamento automático.

CAPÍTULO 4 – UM META-MODELO ORIENTADO A OBJETOS PARA DESCRIÇÃO DE DOMÍNIOS DE PLANEJAMENTO: neste capítulo apresentamos o modelo proposto para modelagem e descrição de domínios e

problemas de planejamento. Os principais conceitos do modelo são apresentados através de um domínio de planejamento clássico da literatura.

CAPÍTULO 5 - PROTÓTIPOS DE INTERFACES PARA O KPLANOO: este capítulo inclui a descrição de um protótipo que foi implementado como interface de captação de dados do *KPlanOO* e um XML *Schema* que possibilita a definição de domínios baseados no *KPlanOO* em XML.

CAPÍTULO 6 – APRESENTAÇÃO E ANÁLISE DOS RESULTADOS OBTIDOS: neste capítulo são apresentados os resultados obtidos durante os testes realizados para a validação do modelo. O capítulo apresenta dois diferentes domínios e problemas descritos através do *KPlanOO*: o Mundo de Blocos e o domínio de rastreio de satélite do INPE. Com os estudos de casos são apresentados alguns benefícios da utilização do modelo, bem como aluguns de seus conceitos. Também é apresentada uma extensão do modelo feita para servir como descrição para o domínio de simulação de um satélite, promovendo a experimentação das metas propostas.

CAPÍTULO 7 – CONCLUSÕES E TRABALHOS FUTUROS: neste capítulo são apresentadas as conclusões e os trabalhos futuros vislumbrados.

2 ENGENHARIA DO CONHECIMENTO

Visando analisar o atual cenário da área de Planejamento Automático, para poder contribuir com métodos e ferramentas de suporte ao processo de modelagem de domínios e problemas de planejamento reais, neste capítulo são descritas brevemente as principais áreas e técnicas associadas ao processo de modelagem de sistemas, principalmente os de planejamento automático.

Primeiramente, como o trabalho está inserido na área de Planejamento Automático, serão apresentados os principais conceitos desta área. Complementando essa descrição será apresentada a área de Engenharia do Conhecimento para Planejamento, área esta que abrange aspectos inerentes à especificação e modelagem de domínios de planejamento, sendo que alguns desses aspectos estão relacionados à Engenharia de Requisitos, esta área também será abordada neste capítulo.

Uma vez que o uso de ontologias para representar os conceitos do domínio permite que estes possam ser reutilizados em várias aplicações, apresentam-se os conceitos sobre a área de ontologia e algumas disciplinas relacionadas, como: Análise de Domínio, Modelos de Domínios, Representação do conhecimento e Ontologias. Para salientar a necessidade abordada neste trabalho, durante a apresentação dos tópicos serão expostos breves históricos.

2.1. Planejamento automático

2.1.1. Conceito

Segundo Vaquero (2007), 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.

Escalonamento é definido por Zweben (1993), como “o processo de atribuir tempo e recursos a tarefas em um plano, satisfazendo ainda uma série de restrições de domínio”. Planejamento e Escalonamento são áreas da Inteligência Artificial (IA) que estudam esses processos deliberativos computacionalmente.

A área de Planejamento está presente em cenários 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 containeres, entre outros (GHALLAB et al, 2004).

Uma abordagem provável para os cenários referidos seria estudá-los, representá-los e desenvolver sistemas de planejamento específico, ou seja, direcionados para um caso específico com fortes dependências do domínio. Porém esta abordagem não interessa à área de Planejamento Automático, pois, esta área da IA almeja estudar e desenvolver sistemas de planejamento para domínios variados (sistemas de planejamento independente do domínio) utilizando especificações e representações do problema, bem como conhecimentos sobre o domínio. Evidentemente, o desenvolvimento de sistemas independente do domínio se torna mais complexo.

O maior esforço em pesquisa na área de Planejamento Automático é no desenvolvimento de sistemas de planejamento, chamados de planejadores (*planners*). A idéia básica do processo de planejamento é simular o comportamento de um ambiente ao qual é aplicada uma seqüência de ações (também chamadas na literatura existente de tarefas, atividades ou operadores). As características do ambiente são representadas de forma lógica por um conjunto de estados, e cada ação aplicada ao ambiente pode modificar

um ou mais de seus estados. A meta é encontrar a seqüência correta de ações que leve o ambiente de seu estado inicial a um estado desejado, ou estado-objetivo.

Duas principais características são desejáveis nos planejadores: eficiência (no sentido de conduzir para uma solução satisfatória) e desempenho (no sentido de não perder tempo em buscas e tentativas frustradas). Planejar no mundo real (o mundo das máquinas, dos elevadores, dos sistemas reais que interagem com o elemento humano, o mundo das entidades automatizadas), é um grande desafio, pois, enquanto um sistema planeja suas ações, o ambiente (a parte do mundo que o envolve e influencia os parâmetros do planejamento) é modificado paralelamente por outras entidades nas quais o planejador não tem controle. Isso ocorrendo, seja na forma de contingências ou de concorrência entre máquinas, agrega uma dificuldade ainda maior gerando, por exemplo, uma reformulação na seqüência de ações, ou plano.

Para um planejador planejar este deve possuir 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) e o 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 de ações especificadas no domínio levando em consideração suas precondições, procurando atingir o objetivo estabelecido. Após a execução da ação, seus efeitos são aplicados sobre o modelo. Caso o estado obtido não seja condizente com a meta esperada, o processo segue, até que a mesma seja alcançada. O conjunto das ações executadas com sucesso é chamado de “plano”.

Observa-se que a especificação do problema e o conhecimento sobre os domínios fornecidos ao planejador são fundamentais no processo de planejamento.

A Figura 2.1 apresenta um diagrama representando um modelo conceitual de Planejamento, visando facilitar o entendimento dos conceitos de Planejamento e do papel do planejador no sistema como um todo.

Figura 2.1 - Modelo conceitual de planejamento
Fonte: Ghallab et al. (2004, p.9).

Podemos observar na Figura 2.1 os três principais elementos que, tradicionalmente, fazem parte de um sistema de planejamento: 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 recebidas (um sistema estado-transição pode ser representado por um grafo direcionado onde os nós são estados e os arcos são ações ou eventos). Como visto na mencionada figura acima, tanto as ações do controlador quanto os eventos exógenos (Eventos) podem contribuir para a evolução do Sistema Σ.

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 et al, 2004).

No modelo conceitual o planejador tem a finalidade de descobrir quais ações devem ser aplicadas a cada estado de Σ para que os objetivos sejam alcançados. Os objetivos podem ser especificados de várias maneiras, por exemplo, uma simples descrição de um estado, ou uma descrição de uma condição a ser satisfeita durante a evolução dos estados do sistema Σ. Os objetivos podem ser especificados através de funções de penalidade associadas a um determinado estado onde o objetivo passa a ser aperfeiçoar esta função.

As observações (*Observations*), parciais ou não, e o status de execução do plano (*Execution status*) são características que tornam o modelo conceitual da Figura 2.1 mais próximo do real. As observações, ou seja, conhecimento sobre o estado do sistema Σ, podem contribuir para um controle mais eficaz sobre a evolução de Σ, possibilitando ao controlador tratar as diferenças entre o sistema estado-transição Σ e o mundo real. O status de execução pode trazer o modelo para um cenário ainda mais realístico onde o planejador leva em consideração um retorno de informações (*feedback*) do plano fornecido. Esse retorno permite ao planejador realizar supervisões e revisões dos planos sintetizados para um eventual replanejamento, o que cria um sistema fechado de planejamento (GHALLAB et al, 2004).

O Planejamento Automático aborda os diversos aspectos que constam no modelo conceitual apresentado, porém nem todos os conceitos são tratados de forma trivial. 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.

Essas simplificações e suposições são descritas a seguir (GHALLAB et al, 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 Seqüenciais:** Um plano-solução de um problema de planejamento é uma seqüência 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.

Com as limitações e suposições apresentadas é natural imaginar que o planejamento torna-se apenas um processo trivial de busca por um caminho no grafo de estados, o que é um problema bem conhecido e bem resolvido na Ciência da Computação. De fato, se possuíssemos o grafo que representa Σ explicitamente não haveria muito planejamento a se fazer. Todavia, mesmo no domínio de planejamento mais simples, o grafo de Σ pode ser tão grande que representá-lo explicitamente seria inviável. Conseqüentemente, é necessário representar o domínio de forma implícita com uma representação compacta do sistema Σ tornando menos onerosa a tarefa de busca por uma solução.

Devido ao crescimento da pesquisa na área de Planejamento Automático, muitas das limitações e simplificações apresentadas anteriormente foram, ao longo do tempo, sendo relaxadas ou até mesmo eliminadas. Essa dissolução nas suposições, tratadas por planejadores capazes de lidar com problemas mais complexos, gerou o chamado Planejamento Neoclássico. Alguns exemplos de dissoluções nas suposições descritas acima são: a suposição A0 tornou-se dissoluta após a inserção de variáveis de estado numéricas produzindo uma explosão no número de estados; outro exemplo se encontra na suposição A4, pois no planejamento neoclássico já era possível expressar objetivos mais complexos como situações a serem evitadas, estados intermediários a serem atingidos, e outros; a suposição A6 também podia ser relaxada, pois passou-se a considerar ações com duração durante o processo de planejamento, entre outras suposições (GHALLAB et al, 2004).

A representação e modelagem dos domínios, ou seja, a representação de Σ é um dos aspectos mais relevantes em Planejamento Automático. Todos os

sistemas/domínios de planejamento Σ, 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 (GHALLAB et al, 2004). Neste trabalho considera-se somente a Representação Clássica e suas evoluções, pois esta tem as características mais adequadas para modelagem de domínios de planejamento para a área espacial.

Características da representação Variável de Estados são encontradas na Representação Clássica. A linguagem mais conhecida e utilizada desta representação é a chamada PDDL (MCDERMOTT, 1998). A PDDL é uma das principais linguagens utilizadas na área de pesquisa de Planejamento Automático. Esta linguagem será detalhada nos próximo capítulo.

2.1.2. Histórico

A pesquisa na área de planejamento teve inicio na década de 60 através de trabalhos científicos que tinham seu foco no desenvolvimento de algoritmos gerais de problemas (principalmente com o uso de lógica de primeira ordem) como, por exemplo, o GPS (*General Problem Solver*) de Ernest e Newell (1969), Newell e Simon (1972) e QA3 de Green (1969). A meta dos algoritmos era encontrar, automaticamente, soluções para uma variada gama de problemas da forma mais genérica possível dos mais variados domínios.

Inicialmente os algoritmos de planejamento eram conhecidos como métodos de busca empregados a provadores de teorema que utilizavam linguagens formais como Cálculo de Situações (*Situation Calculus*) (KOWALSKI; SERGOT, 1969) ou Cálculo de Eventos (*Events Calculus*) (KOWALSKI;, SERGOT,, 1969) para representação dos domínios e problemas. Apesar do uso dessas linguagens

apresentarem um avanço considerável em relação à representação de conhecimento para o desenvolvimento de sistemas de planejamento. Ainda não existia um planejador capaz de usufruir, de forma eficaz, das representações dos domínios durante a obtenção das soluções dos problemas. Foi neste cenário que, no inicio dos anos 70, um grupo de pesquisadores do Instituto de Pesquisa de Stanford criou um sistema de planejamento, chamado STRIPS (*Stanford Research Institute Problem Solver*) (FIKES; NILSSON, 1971), que se tornaria, além de uma referência, um pioneiro na área.

De formulação simples, o STRIPS marcou o início da Era Clássica do Planejamento Automático que se estendeu até o começo da década de 90 (GHALLAB et al, 2004). Este planejador se tornou tão notório pela sua formulação e representação de ações (ou operadores) que permaneceu quase duas décadas como paradigma vigente. Como seu algoritmo era muito complexo, várias propostas de melhoria surgiram nesta era. A Figura 2.2 traz um exemplo da representação em STRIPS para o problema do mundo dos blocos, que é definido a seguir.

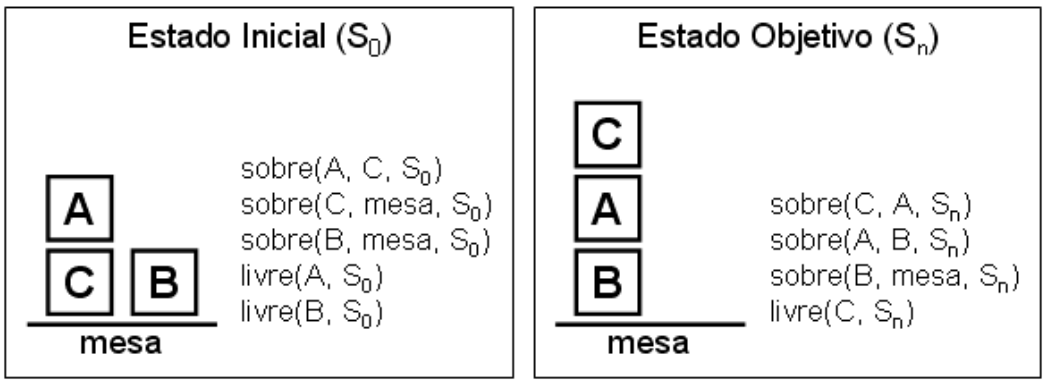

Figura 2.2 - Descrição de um problema em STRIPS
Fonte: Kucinskis (2007, p.21)

Perante 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.

Na década de 80 houve uma desaceleração na pesquisa desta área da IA. Não foram obtidos muitos resultados devido a um desânimo da comunidade de planejamento frente às limitações da geração automática de planos com as técnicas existentes. Os planejadores mais famosos nesta década foram o SIPE (o primeiro a lidar com replanejamento) (WILKINS, 1983) (WILKINS, 1984), TWEAK (CHAPMAN, 1987), PRODIGY (MINTON, 1988) e o ABTWEAK (YANG; TENENBERG, 1990), porém mesmo com o empenho dos pesquisadores, nenhum planejador robusto o suficiente para resolver problemas genéricos foi desenvolvido neste período.

A representação de domínios, principalmente a representação de ações, ganhou um grande avanço em 1989 com a introdução da linguagem ADL por 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.

Em meados dos anos 90 ocorreu uma grande busca por melhorias no 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, nesta época, os planejadores O-PLAN (CURRIE; TATE; 1991) SNLP (MCALLESTER; ROSENBLITT, 1991), POP (BARRET; WELD, 1992) e UCPOP (PENBERTHY; WELD, 1992). Estes tinham seu embasamento, nos planejadores STRIPS e o HTN (EROL et al., 1994), proveniente do planejador NOAH, que fortalecia o planejamento hierárquico. As

técnicas de planejamento hierárquico têm sido muito utilizadas em aplicações reais de planejamento como, por exemplo, logística, robótica, planejamento de processo de manufatura, planejamento e escalonamento de espaçonaves, entre outras (GHALLAB et al, 2004).

Os planejadores desenvolvidos na *era do planejamento clássico* foram chamados de *"sistemas clássicos de planejamento"* e são de um modo geral, sistemas restritos de transição de estados que utilizam métodos como busca para frente ou para tras, tanto no espaço de estados quanto no espaço de planos (GHALLAB et al, 2004), para solucionar os problemas de planejamento. Durante a era clássica, grande atenção foi dada ao desenvolvimento e melhoria das técnicas de planejamento, mas a criação e o desenvolvimento de novas linguagens de modelagem direcionadas aos domínios de planejamento foram deixados em segundo plano, assim como um melhor entendimento e especificação dos problemas e domínios deste tipo.

Faz-se necessário frisar que, desde o inicio da era clássica, o Planejamento Automático se tornou uma área de pesquisa muita ativa envolvendo diversos institutos de pesquisa, universidades e grandes empresas.

Porém mesmo com os significativos progressos, diversos problemas, mesmo que simples, não foram solucionados com um desempenho computacional satisfatório pelos planejadores. 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. Sua simplicidade aliada ao seu desempenho superior aos planejadores da época estimulou o desenvolvimento e a pesquisa de novas técnicas de planejamento. Esse planejador marcou o início da Era Neoclássica da área que fez reviver a pesquisa sobre os problemas clássicos de planejamento, pois naquele momento o planejamento clássico parecia estar perdendo forças

devido ao descontentamento com expressividades dos modelos e com a complexidade dos algoritmos. O desenvolvimento de técnicas neoclássicas de planejamento gerou novos espaços de busca e novos algoritmos que permitiram um aumento significativo no tamanho e na complexidade dos problemas de planejamento que podiam ser resolvidos (GHALLAB et al, 2004).

As pesquisas na era neoclássica do Planejamento Automático direcionaram-se, basicamente, para o aperfeiçoamento do GRAPHPLAN, para o uso de novas técnicas de planejamento e também para o uso de novas formas de representar e modelar domínios. No início do planejamento neoclássico destacaram-se técnicas como:

- **Planejamento com Grafos (Planning-graph):** Técnica que utiliza estruturas poderosas de alcançabilidade para organizar e restringir o espaço de busca. Destacam-se nesta técnica planejadores como o GRAPHPLAN e seus sucessores (KUCINSKIS, 2007);

O GRAPHPLAN possui duas etapas para a geração de planos: a primeira consiste da geração de um 'grafo de planejamento' com representações das ações e estados. A segunda, de um algoritmo simples de busca exaustiva no grafo gerado. O que o torna especial é justamente a criação do grafo, que reduz consideravelmente o espaço de busca para o algoritmo (KUCINSKIS, 2007).

O grafo de planejamento é constituído por níveis, cada um representando um instante no tempo, e possui dois tipos de nós: os que representam estados e os que representam ações. Os nós que representam os estados encontram-se nos níveis pares do grafo. O conjunto de nós de um mesmo nível par representa o conhecimento do mundo em determinado instante. O primeiro nível do grafo representa o estado inicial, e o último, o estado-objetivo. Nos níveis ímpares estão representadas as ações cujas pré-condições – em termos de estados –

estão presentes no nível imediatamente anterior (KUCINSKIS, 2007). A Figura 2.3 ilustra um grafo de planejamento simplificado para o problema do mundo de blocos, na mesma configuração apresentada na Figura 2.2.

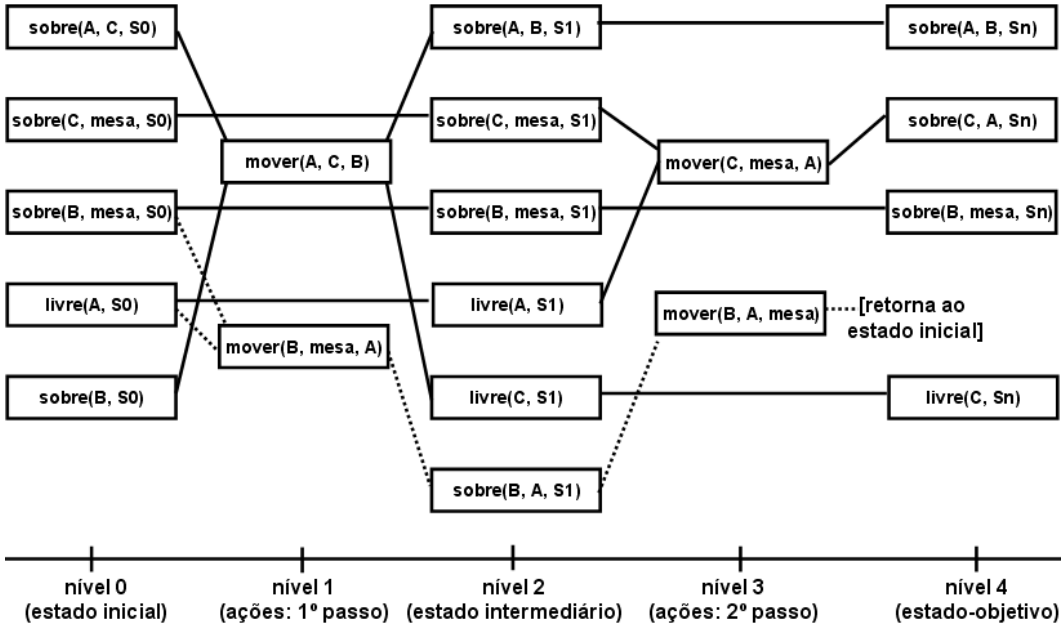

Figura 2.3 - Grafo de planejamento para o mundo de blocos
Fonte: Kucinskis (2007, p.42).

Para gerar este grafo (a primeira etapa da execução), o GRAPHPLAN insere os nós com as literais que descrevem o estado inicial, e em um nível 'n' par, as literais do estado-objetivo. São então verificadas todas as ações cujos operadores de pré-condição (PRE) para o estado inicial são verdadeiros, e estas são inseridas no nível um (KUCINSKIS, 2007).

Os operadores ADD e DEL de cada ação são aplicados, e as literais resultantes são colocadas no nível seguinte do grafo (nível dois), junto com as literais não alteradas por ação alguma, que são transferidas para o próximo nível. Novamente são selecionadas ações cujos operadores PRE sejam verdadeiros no nível dois, e inseridas no nível três. Este processo continua até que o grafo atinja as literais do estado-objetivo.

Muitas das ações inseridas no grafo são mutuamente exclusivas (mutex), e o controle destas ações é a base da geração de planos por análise de grafos. Duas ações são consideradas mutex caso satisfaçam os seguintes critérios:

- Efeitos inconsistentes: o efeito de uma ação é a negação do efeito de outra ação;
- Interferência: uma ação elimina a pré-condição de outra ação;
- Pré-condições inconsistentes: duas ações do nível 'i' são geradas a partir de pré-condições que são mutuamente exclusivas no nível 'i-1'.

O grafo gerado compõe um espaço de busca bastante reduzido com relação ao original. O GRAPHPLAN então dispara um processo de busca exaustiva no grafo para encontrar o melhor plano – que no exemplo da Figura 2.3 é composto por duas ações:

```
mover(A, C, B) -> mover(C, mesa, A)
```

- **Planejamento baseado em satisfatibilidade (SAT):** dada uma expressão booleana composta por 'n' elementos (variáveis) cujos valores possíveis são 'verdadeiro' e 'falso', existe algum conjunto de atribuições de verdadeiro e falso para os elementos que torne a expressão inteira verdadeira? (KUCINSKIS, 2007) Pode-se citar o planejador SATPlan (KAUTZ; SELMAN, 1996) como um dos destaques desta técnica. Alguns planejadores combinavam técnicas de Planejamento com Grafos e SAT como, por exemplo, o BLACKBOX (KAUTZ; SELMAN 1999); A Figura 2.4 apresenta as etapas do processo de planejamento baseado em SAT.

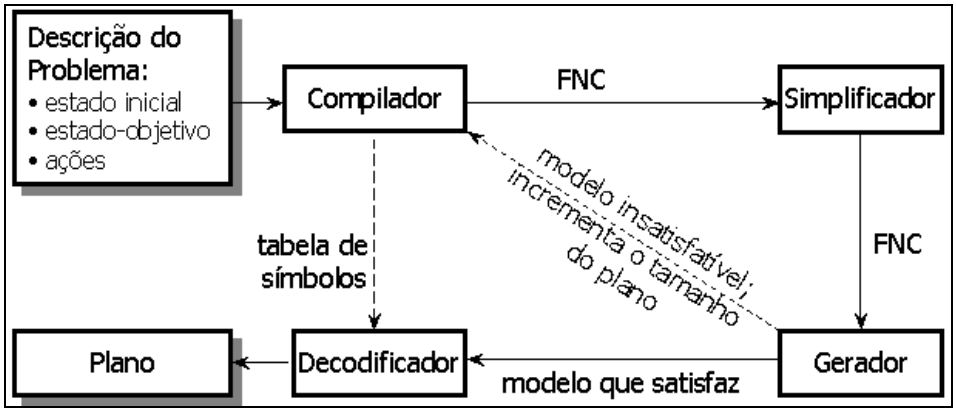

Figura 2.4 - Estrutura da geração de planos por satisfatibilidade
Fonte: Kucinskis (2007, p.41).

Um Compilador recebe a descrição do problema e gera uma fórmula normal conjuntiva (FNC) proposicional equivalente a um plano de tamanho ‘n’. A FNC é uma conjunção de cláusulas booleanas no formato:

```
((estado1 OR estado2) AND (estado3)) ...
```

A FNC gerada é então enviada ao Simplificador. Cabe a ele reduzir o tamanho da fórmula através de técnicas de otimização em tempo polinomial: eliminação de literais e propagação de cláusula única. O Gerador tenta então encontrar um conjunto de atribuições para os elementos da FNC que resulte numa expressão verdadeira. Caso nenhum conjunto de atribuições torne a expressão verdadeira, o processamento retorna ao Compilador, que aumenta o tamanho estimado do plano e gera uma nova fórmula.

O processo segue até que o Gerador seja capaz de encontrar os valores para tornar a FNC verdadeira. Quando isso ocorre, o Decodificador traduz a fórmula em um plano aplicável ao estado inicial do problema e o encaminha para execução (KUCINSKIS, 2007) .

- **Satisfação de Restrição (*Contraint Satisfaction*):** De forma similar ao anterior esta técnica traduz o problema de planejamento em um problema de satisfação de restrições trazendo para a área do Planejamento Automático métodos eficientes de planejamento. Um exemplo de planejador que utiliza técnicas avançadas de Satisfação de restrição é o DESCARTS (JOSLIN; POLLACK, 1996). Satisfação de Restrição em planejamento já havia aparecido algum tempo na comunidade de planejamento. Algoritmos como MOLGEN (STEFICK, 1981) (que prove uma boa ilustração do gerenciamento de restrições) e UMCP (EROL et al., 1994) (algoritmo hierárquico que usufrui bastante desta técnica) já utilizavam este tipo de técnica que mesmo tendo surgindo antes de 1995, é considerada uma técnica neoclássica (GHALLAB et al, 2004).

Hector Geffner ao apresentar a técnica de Planejamento por Busca Heurística em 1998 estabeleceu um novo paradigma para Planejamento Automático, provendo assim um novo estímulo para esta área de pesquisa. O HSP - *Heuristic Search Planner* (BONET; GEFFNER, 1999) - se tornou uma das técnicas mais rápida de planejamento, sendo que a definição de uma função de custo que guia o processo de busca pela solução foi sua maior contribuição.

Já que os maiores esforços da comunidade de planejamento ainda eram o desenvolvimento e pesquisa de planejadores, ainda em 1998 começaram a surgir esforços para definição de uma linguagem comum para modelagem dos domínios de planejamento. Almejando comparar os planejadores existentes e incentivar ainda mais a pesquisa nesta área da IA, criou-se a linguagem de definição de domínios de planejamento chamada PDDL – *Planning Domain Definition Language* (MCDERMOTT, 1998). Esta linguagem foi utilizada na primeira competição de planejadores chamada de IPC – *International Planning Competition* – que ocorreu durante um dos principais congressos na área de Planejamento Automático, o AIPS (*Artificial Intelligence Planning Systems* -

1998). Nesta competição os planejadores resolviam problemas clássicos de planejamento, bem como problemas reais simplificados.

O objetivo da IPC era comparar o desempenho dos planejadores, unir os pesquisadores da área, estabelecer uma evolução nas terminologias e motivar o desenvolvimento de algoritmos, para que estes se tornassem cada vez mais eficientes. A PDDL foi uma ótima ação na tentativa de padronizar a forma de modelar domínios de planejamento, já que até então cada planejador tratava sua própria linguagem de modelagem de domínios como “input”. De fato, a PDDL tornou-se um padrão na modelagem de domínios de planejamento, mas mesmo com as sucessivas evoluções, esta vem recebendo críticas até os dias de hoje, principalmente por não ser tão intuitiva (BACCHUS, 2003) (BODDY, 2003) (GEFFNER, 2003) (MCCLUSKEY, 2003) (MCDERMOTT, 2003) (SMITH, 2003).

Como conseqüências dos avanços apresentados, a cada competição planejadores mais eficientes eram apresentados, por exemplo, o FF (HOFFMANN; NEBEL, 2001) destacou-se na edição de 2000 com um planejamento por busca heurística, o LPG, destaque da competição de 2002, que combinava técnicas do GRAPHPLAN e técnicas de busca heurísticas, e o METRIC-FF (HOFFMANN, 2002) que se caracterizava como uma evolução do FF capaz de tratar restrições numéricas e funções de otimização. Em 2004 os planejadores Fast (*Diagonally*) Downward (HELMERT, 2004), SGPLAN (WAH; CHEN, 2004) e SATPLAN'04 (KAUTZ; SELMAN, 1999) tiveram desempenhos excelentes, principalmente no que se refere ao tempo de resposta. Na Tabela 2.1 mostramos uma tabela, que contempla o planejamento clássico e o neoclássico, destacando os principais planejadores, bem como algumas linguagens.

Tabela 2.1 - Eras do Planejamento Automático (Clássico e Neoclássico) com os Principais Planejadores e Linguagens

Era	Ano	Planejador	Linguagem
Planejamento Clássico	1971	STRIPS	
	1974	ABSTRIPS	
	1977	NOAH e NONLIN	
	1983	SIPE	
	1987	TWEAK	
	1988	PRODIGY	
	1989		ADL
	1990	ABTWEAK	
	1991	O-PLAN e SNLP	
	1992	POP e UCPOP	
Planejamento Neoclássico	1995	GRAPHPLAN	
	1998		PDDL
	1999	SATPLAN e DESCARTS	
	2000	FF	
	2002	CPG e METRIC-FF	
	2004	DIAGONALLY, SGPLAN E SATPLAN´04	

Os avanços obtidos começaram a chamar, cada vez mais, a atenção de novas áreas de pesquisa, com grandes empresas e institutos de pesquisa fazendo com que os algoritmos evoluíssem mostrando o potencial em alcançar os problemas reais. Com isso, os congressos e competições na área de planejamento ganharam grande visibilidade e a cada ano novas idéias são agregadas aos planejadores e a área de Planejamento Automático como um todo.

As principais conferências, AIPS e ECP, (*European Conference on Planning*) se fundiram em 2003 criando a conferência anual chamada ICAPS (*International Conference on Automated Planning and Scheduling*). Esta fusão, além de fortalecer as tendências do Planejamento Automático, fortaleceu a presença do conceito de escalonamento (*Scheduling*). O escalonamento já estava presente nos congressos anteriores, mas este conceito ganhou espaço, principalmente na competição IPC-02, quando a PDDL evoluiu e se tornou capaz de expressar ações com duração e recursos. Essa evolução foi

chamada de PDDL 2.1 (FOX; LONG, 2003) e esta linguagem vem evoluindo juntamente com as competições.

Mesmo com o grande avanço do planejamento automático e com o sucesso das competições e conferências, a comunidade de planejamento percebeu que, de fato, muita atenção estava sendo dada apenas à pesquisa e o desenvolvimento dos algoritmos e pouca atenção era dada para o processo de modelagem de problemas reais como um todo. Essa percepção estava diretamente relacionada com o objetivo de atingir problemas reais. Esse processo de modelagem de domínios de planejamento envolve, por exemplo, análise de requisitos, os métodos de modelagem, as linguagens de representação e modelagem dos domínios, ontologias, análise e verificação de domínios, entre outros. Estes conceitos, inerentes a Engenharia do Conhecimento (*Knowledge Enginnering*) e Engenharia de Requisitos (*Requirements Engineering*), eram negligenciados até então, mas percebeu-se que não importa o quão eficiente é o mecanismo de planejamento ou escalonamento, eles são tão eficientes quanto o conhecimento que estão usando. Em outras palavras, se o modelo do domínio de planejamento e/ou escalonamento for falho então a solução fornecida por um planejador será também falha. Assim, a participação dessas áreas da engenharia vem se tornando cada vez mais evidente e importante para a evolução da pesquisa em Planejamento Automático (GONÇALVES, 2006).

A seguir é apresentada uma breve descrição dos conceitos da Engenharia do Conhecimento aplicada ao Planejamento Automático, também é descrito como e quando a Engenharia do Conhecimento iniciou sua participação na área de Planejamento Automático em IA.

2.2. Engenharia do conhecimento no contexto do planejamento automático

2.2.1. Conceito

A Engenharia do Conhecimento estuda os princípios e métodos para o desenvolvimento e construção de Sistemas Baseados em Conhecimento (*Knowledge Based Systems* - KBS) (STUDER et al., 1998). Os KBS's são softwares que baseando-se nos dados armazenados em uma base de conhecimento resolvem problemas normalmente utilizando-se de máquinas de inferência provenientes da área da inteligência artificial, tendo como objetivo auxiliar e/ou substituir especialistas humanos em suas tarefas. A Figura 2.5 apresenta uma arquitetura simplificada de um KBS.

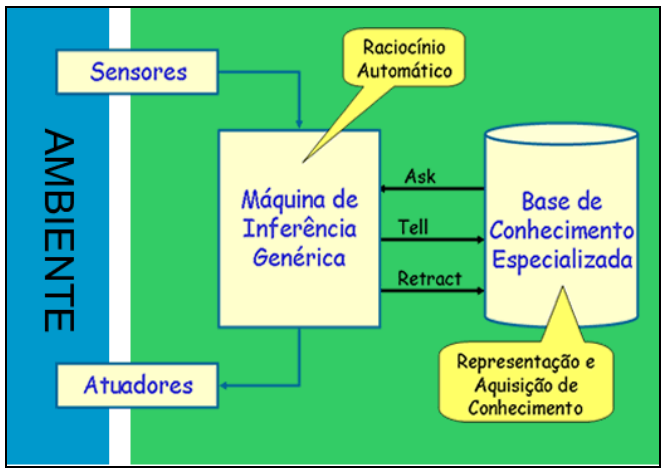

Figura 2.5 - Arquitetura simplificada de um KBS.

Nesta arquitetura tem-se a Base de Conhecimento que contém, por exemplo, as representações de ações e eventos do domínio, bem como regras; e a Máquina de inferência genérica, responsável pelo raciocínio automático sobre a Base de Conhecimento e os fatos para a resolução de problemas.

Os KBS são utilizados por uma vasta gama de aplicações, dentre elas estão as aplicações baseadas em Planejamento.

Por muito tempo, pesquisas na área de IA focaram no desenvolvimento de formalismos, mecanismos de inferência e ferramentas para operacionalizar os Sistemas Baseados em Conhecimento. Tipicamente, os esforços iniciais de desenvolvimento eram restritos a realização de KBS de pequeno porte com o objetivo de estudar a viabilidade das diferentes abordagens (STUDER et al., 1998).

Embora esses estudos apresentassem resultados promissores, a aplicação da tecnologia de KBS em uso comercial, onde seriam necessários KBS de grande porte, falhou em muitos casos, devido principalmente à abordagens de aplicações acadêmicas. Esta situação pôde ser diretamente comparada e relacionada com uma situação similar no desenvolvimento de sistemas de software tradicionais, chamada de “crise do software”, no final da década de 60, onde os meios nos quais pequenos protótipos acadêmicos estavam sendo desenvolvidos não obtiveram sucesso para o design e a manutenção de sistemas comerciais grandes e de longa duração. Da mesma forma que a “crise do software” resultou no estabelecimento da disciplina Engenharia de Software, o cenário insatisfatório do desenvolvimento dos KBS gerou também a necessidade de abordagens mais metodológicas, bem como métodos disciplinados (STUDER et al., 1998).

Nesse contexto, a área da Engenharia do Conhecimento surgiu com um objetivo similar ao da Engenharia de Software: tornar o processo de construção de KBS uma disciplina da engenharia. Este objetivo requer uma análise do próprio processo de construção e manutenção desses sistemas e também o estudo e o desenvolvimento de métodos e processos disciplinados de eliciação, de especificação e de modelagem, bem como linguagens, métodos de solução de problemas e ferramentas especializadas para o design de KBS.

A Engenharia do Conhecimento aplicada à área de Planejamento Automático pode ser vista como um caso especial da grande área da EC. O processo de

planejamento envolve a manipulação de conhecimento de fenômenos complexos, tais como ações (GONÇALVES, 2006). A Figura 2.6 mostra o processo de planejamento, onde o conhecimento (Objetivos, Estado Inicial e Ações) é o ponto de partida de todo o processo.

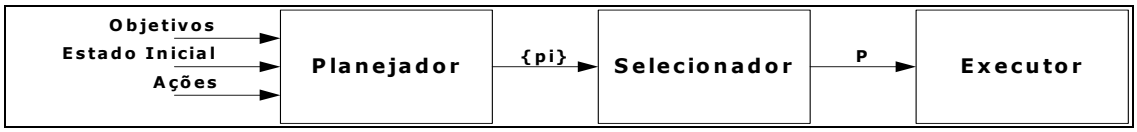

Figura 2.6 - Processo de planejamento clássico
Fonte: Feber (1999, p.152).

Em alguns trabalhos, Lee McCluskey (2002) esclarece algumas terminologias definidas em Planejamento Automático. O conhecimento associado a uma determinada aplicação real de planejamento é denominado domínio. Qualquer forma de representação simbólica de parte do domínio pode ser chamada de descrição do domínio, por exemplo, um conjunto de documentos descrevendo o domínio, possivelmente em linguagem natural. O termo especificação do domínio é uma descrição abstrata do domínio definitiva e fechada. É esperado que uma especificação do domínio seja completa e precisa. A representação simbólica do conhecimento que pode ser utilizada para realizar e executar operações, de modo similar ao domínio, é chamado de modelo do domínio. O modelo do domínio é uma especificação do domínio na forma operacional, contendo detalhes explícitos das propriedades e da dinâmica do domínio apropriados para ser processado por um mecanismo de planejamento. Os fenômenos presentes no modelo do domínio devem corresponder àqueles presentes no domínio real (MCCLUSKEY, 2002).

Como visto nos conceitos de planejamento, os planejadores precisam deste conhecimento, o modelo do domínio e suas respectivas representações dos problemas, para produzir seqüências de ações de modo a atingir metas e diretrizes estabelecidas, especialmente para os planejadores baseados em busca heurística. Como as aplicações de planejamento possuem uma forte dependência das bases de conhecimento, é possível afirmar que a disciplina

Engenharia do Conhecimento e seus respectivos processos são de extrema importância para esta área especifica da IA.

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., 2003).

Capturar e representar corretamente o conhecimento em aplicações de planejamento não é uma tarefa simples. Os aspectos do conhecimento nestas aplicações são reconhecidos como sendo a maior dificuldade no projeto. Experiências com planejadores adaptados a aplicações como, por exemplo, aeroespaciais e militares (WILKINS, 1999) (TATE et al., 1996) (MUSCETTOLA et al., 1998), apontaram que aspectos da EC são os que requerem mais atenção. O processo de representação do conhecimento envolvido em aplicações de planejamento está fortemente relacionado com a identificação e representação das ações que afetam os objetos do domínio. Esse conhecimento deve ser adequado para permitir que o planejador raciocine e construa planos de maneira eficiente.

Muitos conceitos inerentes às Engenharias de Software, Requisitos e outras engenharias são de grande importância para os processos da EC em domínios de planejamento. É possível citar alguns conceitos, processos e aspectos fundamentais que devem ser levados em consideração na EC para Planejamento Automático:

- **Análise de Requisitos:** analisar e entender profundamente os requisitos do domínio de planejamento para uma melhor descrição e especificação dos domínios. Ao realizar uma análise rigorosa pode-se economizar tempo e recursos nas demais fases do desenvolvimento de sistemas de planejamento automático como, por exemplo, a modelagem do domínio.

- **Os pontos de vista (*viewpoints*) no processo de EC:** o processo de aquisição de conhecimento da EC envolve papéis ou pontos de vistas diferentes para que se possa ter uma ampla visão do domínio de planejamento. Pontos de vista provenientes dos especialistas (*experts*) em planejamento, dos engenheiros do domínio, dos especialistas no domínio, especialistas em softwares, *stakeholders* e usuários são extremamente importantes para uma boa evolução do processo e um bom entendimento do domínio. Naturalmente, uma pessoa pode assumir mais de um ponto de vista. Como esse processo pode lidar com diversas visões, assim como diversos níveis de conhecimento, é necessário o uso de ferramentas de suporte para que todos tenham uma visão unificada do domínio e este se torne consistente.

- **Ferramentas de Suporte ao processo de EC:** as ferramentas de suporte são fundamentais nos processos de análise, aquisição, modelagem, verificação e validação de modelos e/ou dos sistemas de planejamento. Estas ferramentas auxiliam os projetistas tanto durante os processos de EC quanto na manutenção dos domínios modelados.

- **Modelagem de domínio:** o processo de modelagem de domínios de planejamento envolve a representação do domínio utilizando alguma linguagem de modelagem. Esse processo cria o modelo do domínio. No planejamento automático, parte-se do pressuposto que o modelo

contenha uma descrição declarativa da dinâmica do domínio real e que o aspecto mais importante no modelo é a descrição das ações. Este modelo declarativo deve ser desenvolvido e elaborado independentemente do planejador ou outro software que contribua no processo. Este fato tende a facilitar a verificação e validação do modelo do domínio. Um modelo criado independentemente do planejador pode ser utilizado por uma variedade de planejadores, ou mecanismos de planejamento, e até mesmo em outras aplicações que não necessariamente são de planejamento.

- **Métodos de modelagem:** boas práticas de modelagem assim como métodos para especificação de domínios podem ser utilizados para guiar todos os processos da EC. Esses métodos contribuem para que o processo de modelagem seja organizado e estruturado e muitas vezes sistemático para que não haja possíveis perdas de informação sobre o domínio.

- **Verificação:** a verificação é o processo de correção e refinamento dos modelos dos domínios, realizado de maneira informal ou semi-formal, baseado, geralmente, na aprovação dos participantes no processo de desenvolvimento.

- **Validação:** a validação de modelo do domínio de planejamento é o processo que promove a qualidade do modelo através de métodos formais de identificação e correções de erros e inconsistências (MCCLUSKEY et al., 2003).

- **Representação do Conhecimento:** em geral, o conhecimento de domínios de planejamento a ser capturado e modelado são as ações, objetivos, atividades, recursos, tempo, restrições. Esse conhecimento

deve ser representado utilizando alguma linguagem de representação. Atualmente, a linguagem padrão de representação dos modelos utilizado pela comunidade de pesquisa em Planejamento Automático é a PDDL. Todavia, esta linguagem não foi desenvolvida com uma perspectiva baseada na Engenharia do Conhecimento, o que reforça a necessidade da busca de outras representações do conhecimento, ainda que transferíveis para a PDDL. Esta linguagem será mais bem detalhada nos próximos tópicos.

2.2.2. Histórico

O conceito Engenharia do Conhecimento para Planejamento começou a aparecer a partir de 1999 como um dos tópicos de interesse da comunidade de planejamento. Um dos grandes precursores na introdução do conceito da Engenharia do Conhecimento na área de Planejamento Automático foi o pesquisador Lee McCluskey, além dos pesquisadores envolvidos na formação de linguagem de modelagem (por exemplo, a PDDL). O pesquisador, juntamente com seus colegas de pesquisa, já havia realizado trabalhos na área da EC para Planejamento desde o inicio da década de 90 (MCCLUSKEY; PORTEOUS, 1993) (MCCLUSKEY et al., 1995).

A entrada dos conceitos da EC na comunidade de planejamento foi devida a diversos aspectos como, por exemplo, o descontentamento com a linguagem padrão PDDL, a busca por domínios mais complexos onde as abordagens clássicas não eram suficientes e eficazes, entre outros. Novamente, esse cenário de insatisfação pode ser relacionado e comparado tanto com a “crise do software” quanto com o próprio surgimento da EC conforme descrito anteriormente.

A Engenharia do Conhecimento aplicada a Planejamento teve alguns marcos de grande importância. Um importante marco foi o trabalho realizado pelo pesquisador McCluskey e seus colegas sob o título “*Knowledge Engineering for*

Planning ROADMAP" (MCCLUSKEY et al., 2003). Este trabalho teve início em 2000 e continha os principais conceitos da EC para a comunidade de planejamento, bem como os passos que a própria comunidade deveria tomar para que as aplicações reais fossem alcançadas. Esse trabalho teve um papel fundamental no despertar da comunidade que na época, conforme visto anteriormente focalizava esforços em domínios clássicos e, principalmente, nas técnicas de planejamento (planejadores).

Entre 1997 e 1998 ocorreu outro importante marco, o aparecimento de algumas ferramentas de planejamento que contemplavam os aspectos da EC. Antes de 1997 as ferramentas de aquisição de domínios de planejamento eram apenas consideradas verificadores de sintaxe e de erros. As duas ferramentas pioneiras na área eram o "*Common Process Editor*" (TATE et al., 1998), presente no sistema O-PLAN, e o "*Act Editor*" (MYERS; WILKINS, 1997), presente no SIPE. Essas ferramentas e editores surgiram devido à necessidade de desenvolvimento de aplicações reais de planejamento onde o auxílio na modelagem durante os processos de engenharia era fundamental. Ambas as ferramentas eram específicas para determinados domínios, mas estas foram construídas para auxiliar a superar os problemas encontrados durante o desenvolvimento de modelos de domínios complexos, fato que contribui para a relevância da EC para planejamento.

Mesmo com poucas ferramentas, com o passar do tempo, cresceu o interesse nesta área, assim alguns focos de pesquisa para sistemas e ambientes capazes de analisar domínios, foram surgindo. O objetivo destes sistemas e ferramentas era processar um modelo de domínio e utilizar ao máximo as informações disponíveis para que estas fossem enviadas a um planejador. A análise de domínios auxiliaria o usuário a modelar e refinar o domínio de interesse. Esses sistemas seriam capazes de: verificar se uma ação (ou operador) modelada está consistente, ou seja, não produzirá um estado inválido se o estado atual for válido; raciocinar sobre as ações e as restrições

para que sejam visualmente verificadas pelo usuário; verificar objetivos inconsistentes, entre outros.

2.3. Análise de domínio

A disciplina de análise de domínio começou a ser considerada importante pela comunidade de Planejamento Automático a partir da necessidade de se conhecer melhor os domínios reais mais complexos, e também do consenso relacionado à importância do desenvolvimento para reuso e com reuso neste cenário.

O termo Análise de Domínio foi introduzido por Neighbors (NEIGHBORS, 1981) com a seguinte definição:

> *A Análise de Domínio é uma tentativa de identificar os objetos, operações e relações entre o que peritos em um determinado domínio percebem como importante.*

Por exemplo, em um domínio de controle de satélite em solos, objetos típicos são estação terrena, telecomando, telemetria e zona de silêncio. Operações e ações incluem envio de telecomando temporizado em solo, envio de telecomando não temporizado e calibração. 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.

É natural acreditar que 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).

Esta definição informal ilustra bem a idéia inicial, porém necessitamos de uma definição mais rigorosa para servir de base para as discussões que aparecem no decorrer do capítulo. Esta definição é construída a partir de alguns termos apresentados a seguir.

2.3.1. Conceitos

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 certa comunidade. Esta definição cobre duas perspectivas (ARANGO, 1994):

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

2. 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 ambigüidades 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: União de atividades com o propósito de diminuir a complexidade da nossa percepção de um determinado domínio, impondo uma coerente organização aos dados adquiridos através de experimentos, elicitação de especialistas e engenharia reversa de sistemas existentes (GUIZZARDI, 2000).

Conforme ressaltado por Neighbors (1981), 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, conhecimento. No entanto, essa atividade se mostrou complexa e geralmente cara, dependendo da criação de toda uma cultura de apoio no ambiente em que o desenvolvimento para reuso e com reuso vá ocorrer. Além disso, o sucesso depende diretamente do modelo gerado no que diz respeito à relevância e à qualidade da representação dos itens de informação, bem como de suas relações.

No trabalho de Arango (1994), é evidenciada a dificuldade de se identificar, capturar e organizar estes itens apropriados de informação, de forma que se torna clara a necessidade de um processo bem definido e estruturado. Deste modo, a análise de domínio toma a forma de um conjunto interdisciplinar que envolve adaptações de técnicas comumente usadas nas áreas de engenharia de software, engenharia de requisitos, modelagem conceitual, aquisição de conhecimento e representação de conhecimento, podendo ser vista como versão em meta-nível da engenharia de requisitos e como um processo de engenharia do conhecimento direcionado a auxiliar uma tarefa particular de resolução de problemas.

Baseado no processo para análise de domínio descrito por Guizzardi (2000) este trabalho se enquadra na fase de Análise dos dados e modelagem do domínio; nesta fase, o conhecimento capturado é inicialmente avaliado quanto à consistência, correção e completude e, então, modelado identificando entidades, relações, funções e axiomas comuns às diversas fontes analisadas. O modelo para descrição de domínios e problemas para a área de planejamento automático que é proposto aqui é um modelo conceitual (que mostra a presença de entidades do domínio e como elas se relacionam) e por um léxico (ou dicionário) do domínio. Este léxico apresenta de maneira menos ambígua possível as definições dos elementos que compõem o modelo conceitual, desempenhando um papel fundamental quanto à economia de tempo, minimização dos problemas de comunicação e, conseqüentemente, promoção de um diálogo mais eficiente e consistente acerca do domínio modelado.

Outras técnicas usuais abordadas neste trabalho também são: modelos de entidades e relacionamentos, modelos de objetos, pré-condições e invariantes. Uma cautela observada nessa proposta é que o modelo produzido não reflete decisões de implementação. Esta etapa finaliza com a definição da hierarquia, abstração e classificação das entidades. Esta atividade é baseada em fatos como interdependência ou exclusão mútua das entidades selecionadas. Tais características também foram levadas em consideração neste trabalho.

2.4. Representação do conhecimento e ontologias

Acredita-se, hoje em dia, que a representação formal do conhecimento tenha começado na Índia do primeiro milênio A.C. com o estudo da gramática de Shastric Sanscrit. No entanto, da forma como a vemos atualmente, esta disciplina tem um relação muito próxima com os trabalhos realizados na Grécia

antiga, principalmente por Aristóteles (384-322 A.C.) nos campos da lógica, ciências naturais e filosofia metafísica (RUSSEL; NORVING, 1995).

As primeiras discussões no campo da Inteligência Artificial focavam a questão da representação do ponto de vista do problema e não do conhecimento. Com a proliferação dos sistemas especialistas, a representação do conhecimento era feita com o objetivo claro de extrair o conhecimento do perito e formalizá-lo em uma base de conhecimento, ou seja, a mente do perito era vista como um "mina" e o papel do engenheiro de conhecimento era explorá-la. Por outro lado, a maior parte dos esforços para incorporar conhecimento aos sistemas concentrava-se na construção de mecanismos uniformes e gerais de representação. A forma como este processo era conduzido teria uma influência direta em alguns aspectos (FALBO, 1998):

- Uma vez que a máquina de inferência era de propósito geral, a estratégia para resolver um problema era embutida como parte da base de conhecimento. Desta forma, era praticamente impossível separar os conhecimentos do domínio, da aplicação e da tarefa a ser realizada, tornando a reutilização do conhecimento praticamente inviável. O conhecimento do domínio não podia ser usado em outras aplicações dado que era adquirido para uma tarefa específica.

- O problema da reutilização era ainda agravado pelo modo no qual o conhecimento é associado e conseqüentemente disponibilizado por parte dos peritos. O conhecimento elicitado de especialistas em entrevistas é disponibilizado de forma bastante compilada através de heurísticas, o que dificulta a separação dos seus diversos tipos e praticamente inviabiliza o seu reuso.

- O uso do conhecimento do especialista como única fonte de conhecimento é, por si só, uma falha. Outras fontes de conhecimento,

> como literatura técnica e sistemas existentes, desempenham papéis igualmente importantes, devendo ser utilizadas de forma complementar. Ao relegar estas fontes, a estratégia de transferência do conhecimento do especialista para o sistema não apenas tornava a tarefa de aquisição mais difícil, como também reforçava o problema da superficialidade.

Toda vez que um sistema especialista tivesse sendo construído em um mesmo domínio, mas com o objetivo de realizar uma diferente tarefa, todo o processo de elicitação e codificação do conhecimento deveriam ser refeito, expondo o processo a erros e inconsistências que já poderiam ter sido resolvidas, além de provocar perda de tempo, esforço e conseqüentemente recursos. Diante desta situação, surge, então, a necessidade de uma nova abordagem para construção desta classe de sistemas, buscando um processo que pudesse modelar e isolar os diferentes tipos de conhecimento, possibilitando o reuso em seu mais alto nível de abstração: o reuso de conhecimento.

Clancey (1993) sugere a alteração desta perspectiva, argumentando que o foco da Engenharia de Conhecimento deve ser a modelagem de sistemas e não a tentativa de reproduzir a maneira como os especialistas raciocinam, defendendo a visão de que uma base de conhecimento deve ser vista como um produto de uma atividade de modelagem e não um repositório de conhecimento especializado. Desta forma, a modelagem passa a ser o aspecto central da Engenharia de Conhecimento e a aquisição de conhecimento passa a ser essencialmente um processo construtivo, no qual o engenheiro de conhecimento usa todos os tipos de informação disponíveis e estabelece as decisões finais de modelagem. Dentro da comunidade de representação do conhecimento surgiu, então, um grupo de defensores da idéia de que o conhecimento embutido em uma determinada porção da realidade poderia (e deveria) ser representado em um nível de abstração tal que fosse independente e reutilizável ao longo de várias tarefas (GUARINO, 1997).

Com esta abordagem, esta comunidade entrou em um território anteriormente explorado unicamente por filósofos da ciência e da linguagem, fazendo com que, devido à imposição de sua disciplina, esta área fosse investigada de forma mais rápida e profunda do que quando era um domínio exclusivo da filosofia. Ao produto desta área inicialmente criada por Aristóteles com seu abrangente sistema de classificação, taxonomização e de representação do conhecimento de forma geral, chamamos hoje de Ontologias (GUIZZARDI, 2000).

2.5. Definição de Ontologia

Segundo Gruber(1995), uma ontologia é uma explícita especificação de uma ”conceitualização”, Guarino (1998) estende essa definição dizendo que uma ontologia é uma especificação parcial e explícita que tenta, da melhor forma possível, aproximar a estrutura de mundo definida por uma conceituação. Uma ontologia, portanto, passa a ter compromisso apenas com a consistência em um determinado domínio e não com a completude. Ao conjunto de elementos de um domínio que podem ser representados em uma ontologia é dado o nome de universo de discurso. Zlot et al. (2002) vão além e definem que uma ontologia consiste de uma especificação de objetos, conceitos e outras entidades que são assumidas como existentes, além de relações entre conceitos e restrições expressas através de axiomas.

Apesar de sua difusão e do longo tempo em que vem sendo usado, ainda não há um consenso (principalmente na comunidade de Ciência da Computação) sobre a semântica do termo "ontologia". Em alguns casos, ele é usado apenas como um nome mais rebuscado, denotando o resultado de atividades familiares como modelagem de domínio e análise conceitual. No entanto, em muitos outros casos, as ditas ontologias apresentam algumas peculiaridades como a forte ênfase na necessidade de uma abordagem altamente formal e interdisciplinar, na qual a filosofia e a lingüística desempenham um papel fundamental.

Nesse trabalho, o termo ontologia de acordo com a definição de Guarino (1998), ou seja, ontologias são tratadas como um artefato computacional composto de um vocabulário de conceitos, suas definições e suas possíveis propriedades, um modelo gráfico mostrando todas as possíveis relações entre os conceitos e um conjunto de axiomas formais que restringem a interpretação dos conceitos e relações, representando de maneira clara e não ambígua o conhecimento do domínio. É importante realçar que, de posse dessa base de conhecimento formalizada como uma teoria lógica, a ontologia não descreve apenas conhecimento imediato, isto é, conhecimento factual que pode ser obtido diretamente a partir da observação do domínio, mas também conhecimento derivado, ou seja, conhecimento obtido através de inferência sobre o conhecimento imediato disponível. Um modelo de domínio utilizando-se ontologias, portanto, não é somente uma hierarquia de termos, mas uma infra-estrutura teórica que versa sobre o domínio em questão.

Segundo Guarino (1998), com base em seu conteúdo as ontologias podem ser classificadas nas seguintes categorias:

- **meta-ontologias:** também chamadas de Ontologias Genéricas ou Ontologias Fundamentais, que são reutilizáveis (ou aplicáveis) em diferentes domínios.

- **ontologias de domínio:** são reutilizáveis em um dado domínio provendo vocabulários sobre os conceitos dentro de um domínio e seus relacionamentos, sobre as atividades que envolvem este domínio e sobre as teorias e princípios elementares que governam aquele domínio.

- **ontologias de tarefas:** expressam conceituações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto

é, descrevem o vocabulário relacionado a uma atividade ou tarefa genérica, tal como, diagnose ou vendas;

- **ontologias de aplicações:** que contêm o conhecimento necessário para modelar situações específicas de uma tarefa em um domínio particular.

- **ontologias de representação:** explicam as conceituações que fundamentam os formalismos de representação de conhecimento.

O modelo desenvolvido neste trabalho apesar de ter sua motivação maior no domínio da área espacial, mais precisamente no que se refere ao controle de atividades de satélites, baseou-se nos conceitos de meta-ontologia, ontologias de domínio e ontologias de aplicações para possibilitar a representação de todo o conhecimento para o processo de planejamento, de diversos domínios e problemas.

O enfoque das pesquisas em ontologias genéricas visa construir teorias básicas do mundo, de caráter bastante abstrato, aplicáveis a qualquer domínio. Entre os trabalhos nesta categoria, destacam-se os projetos *CYC* (LENAT, 1995), *WORDNET* (MILLER, 1990), *Generalized Upper Model* (BATEMAN et al., 1994) e as ontologias de Sowa (1995) e de Dahlgren (1995). Estes trabalhos estão bastante alinhados com o uso de ontologias nas áreas filosóficas de categorização e lingüística e procuram descrever a natureza das coisas. Tipicamente, ontologias genéricas definem conceitos tais como coisa, estado, evento, processo, ação, etc., com o intuito de serem especializados na definição de conceitos em uma ontologia de domínio, estes conceitos são adequados para o a modelagem do domínio de Planejamento Automático.

Ontologias de domínio é o tipo mais comumente desenvolvido, sendo que diversos trabalhos são encontrados na literatura, enfocando áreas como

química (GÓMEZ-PÉREZ et al., 1996), modelagem de empreendimento - TOVE (*Toronto Virtual Enterprise*) (GRUNINGER, 2009) (USCHOLD; GRUNINGER, 1996), Design – DORPA (VAREJÃO, 1999) e YMIR(ALBERTS, 1994), modelagem de processos de software(FALBO, 1998), entre outros.

Guarino (1998) propõe que ontologias sejam construídas segundo seu nível de generalidade, como é mostrado na Figura 2.7. Os conceitos de uma ontologia de domínio ou de tarefa devem ser especializações dos termos introduzidos por uma ontologia genérica. Os conceitos de uma ontologia de aplicação, por sua vez, devem ser especializações dos termos das ontologias de domínio e de tarefas correspondentes.

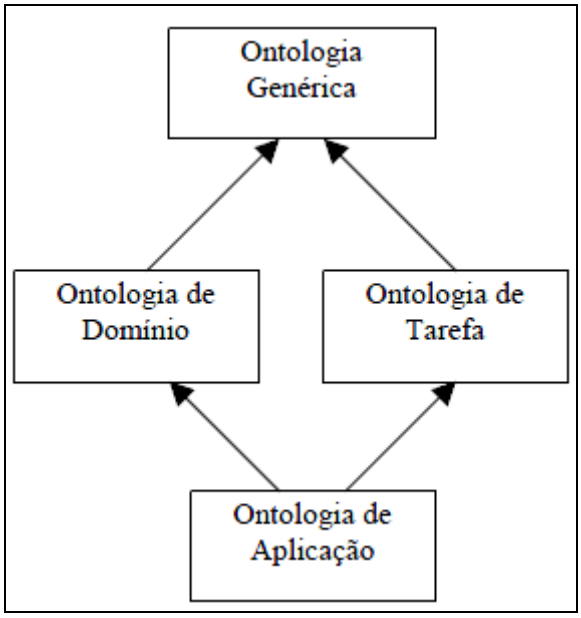

Figura 2.7 - Tipos de ontologias, segundo seu nível de dependência em relação a uma tarefa ou ponto de vista particular
Fonte Guarino (1998).

Um projeto de ontologia tem abordagens baseadas em projetos de orientação a objetos (RUMBAUGH et al., 1991) (BOOCH et al., 1998). Porém, o desenvolvimento de ontologias se difere da modelagem de classes e relacionamentos numa modelagem orientada a objetos (representada em UML). Noy & McGuinness (2001) esclarecem que a modelagem orientada a

objetos se foca principalmente sobre métodos em classes - um analista toma decisões baseado em propriedades operacionais de uma classe, enquanto um engenheiro de ontologias toma decisões baseado em propriedades estruturais de uma classe, explicitando suas relações e formalizando os conceitos. Como resultado, uma estrutura de classe e relações em uma ontologia pode ser diferente de uma estrutura em uma modelagem orientada a objetos para um domínio similar (BOOCH et al, 1998).

2.6. Orientação objetos

O sucesso da metodologia de Orientação Objetos (OO), nas últimas duas décadas, trouxe inovações para auxiliar o desenvolvimento de planejamento automático, como uma tecnologia que dispõe de mecanismos que facilitam a modelagem de domínios variados. Por exemplo: Liu e McCluskey (2000) proporão a *Object-Centered Language* (OCL) com a finalidade de representar modelos de domínios como um conjunto de objetos sujeitos a várias restrições. A idéia de representação baseada em objetos está claramente alinhada com uma abordagem baseada em modelos.

O uso da OO, na modelagem de domínios de planejamento automático, propicia algumas de vantagens. Estas vantagens podem ser destacadas na forma dos seguintes tópicos:

a) **Maior consistência nas visões dos modelos:** Na OO, o mesmo conjunto de visões dos modelos é usado em todas as fases do desenvolvimento do projeto. Estas visões dos modelos vão agregando detalhamentos até chegarem a implementação propriamente dita. Por exemplo, os objetos, classes e dados identificados têm representação direta no código.

b) **Maior abstração do domínio do problema:** Na OO, os conceitos de classe e de objeto mantêm forte acoplamento com os dados e as

operações que manipulam estes dados, do mesmo modo que ocorre com os elementos do domínio real. Isto torna a tecnologia intuitiva, permitindo que os desenvolvedores e pessoas envolvidas no domínio do problema tenham facilidades para compreender e validar as descrições feitas. (SILVA, 1996)

c) **Manutenção facilitada e maior estabilidade frente às mudanças:** Conceitos como abstração, encapsulamento, herança e polimorfismo fazem com que as alterações ou as extensões da aplicação sejam estáveis, sem agregar maiores complicações. Por exemplo, o conceito de encapsulamento permite que a classe sofra mudanças internas, sem alterar a sua interação com as demais classes do sistema. Outro tópico que vem ao encontro de facilitar a manutenção é a boa coesão dos sistemas orientados a objetos. Esta boa coesão dos aspectos dos sistemas ocorre, novamente, através do conceito de encapsulamento, inerente às classes e aos objetos, uma vez que seus dados mantêm-se fortemente acoplados com o comportamento (TKARCH; PUTTICK, 1994).

d) **Reuso facilitado:** O reuso poderá ocorrer nas situações em que há semelhança nas descrições de domínios de problema (SILVA, 1996). Esta possibilidade de reuso pode ser facilitada através do conceito de herança. As classes genéricas, da estrutura de herança, facilitam o reuso através da adição de novas classes ou extensão das já existentes. Isto pode ocorrer sem grandes alterações no código fonte. São as diferenças que deverão ser codificadas. Através do conceito de herança, é permitido a especificação incompleta dos objetos, sendo que esta pode ser refinada, satisfazendo as necessidades do domínio do problema. No entanto, cabe uma ressalva, estas duas formas de utilizar o conceito de herança não garantem que o reuso irá ocorrer automaticamente.

e) **Melhor suporte ao conceito de confiabilidade:** Através dos conceitos de abstração e encapsulamento, com o auxílio de pré e pós-condições, é possível projetar interfaces, bem definidas, de interação entre os objetos. Isso resulta em um controle maior da forma como os objetos interagem, melhorando a confiabilidade e diminuindo a ocorrência de erros no sistema. (DOUGLASS, 1998)

2.7. Considerações

Apresentou-se neste capítulo, o estudo teórico realizado para o desenvolvimento do meta-modelo. 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. Por outro lado, com o estudo das abordagens, foi possível a definição de um meta-modelo inicial. Ainda nesse capítulo, foram introduzidos os conceitos de engenharia do conhecimento, análise de domínio, orientação a objetos e ontologias, o que possibilitou formar uma boa base quanto a conceitos inerentes a modelagem e representação de domínios.

O meta-modelo desenvolvido neste trabalho utiliza a OO como metodologia, sua base foi constituída da generalização das propriedades estruturais do domínio da área de Planejamento Automático, chegando assim em um modelo estruturado de classes que se relacionam de forma que estabeleçam conceitos e métricas fortes para a modelagem do domínio em questão. No próximo capítulo apresentam-se algumas linguagens e modelos para descrição de domínios que de alguma forma se assemelham com este trabalho e também de sistemas em geral.

3 TRABALHOS CORRELATOS

Neste capítulo serão apresentados alguns trabalhos relacionados à modelagem e análise de domínios de planejamento que, de alguma forma, se assemelham com este trabalho e também de sistemas em geral. Linguagens de especificação e modelagem de domínios, são descritas brevemente, com o objetivo de levantar as necessidades de representação e de ferramentas que auxiliem no processo de entendimento, caracterização e modelagem de domínios que envolvam planejamento automático.

São apresentadas algumas linguagens para representação de domínios em planejamento, um conjunto de instruções para um planejador e um ambiente de modelagem e análise de domínios de planejamento.

3.1. Ferramentas para modelagem de domínios

3.1.1. STRIPS: O precursor

Em 1971 o STRIPS primeiro sistema de planejamento foi anunciado. A representação de ações e estados baseados em literais e seu modelo de ações tiveram muito mais influência na área de planejamento do que sua abordagem algorítmica. A esta forma de representação foi dado o mesmo nome do sistema planejador.

A descrição em termos de literais que representam estados, ações, pré-condições e efeitos provou ser capaz de modelar os mais variados domínios e problemas. Diversas extensões surgiram com o passar dos anos e, mesmo novas linguagens, como a ADL e a PDDL, continuam sendo baseadas nestes mesmos conceitos fundamentais (KUCINSKIS, 2007).

Com a vigente necessidade de se trabalhar com recursos e tempo em planejamento em domínios reais, um grande desafio foi imposto para este tipo

de representação, o que vem motivando o estudo de novas formas de modelagem, técnicas inspiradas na Orientação a Objetos como a linguagem OCL e também como a proposta por este trabalho.

3.1.2. ADL: A primeira evolução

A linguagem ADL (*Action Descripition Language*) surgiu na tentativa de suprir as limitações da linguagem STRIPS.

Comparando-a com a linguagem STRIPS, pode ser destacada a inclusão de efeitos condicionais, a permissão de disjunção nos objetivos, suporte de variáveis com tipos definidos e o fato de ADL assumir a hipótese de domínios abertos, em que literais não mencionados são desconhecidos.

É fácil ver que, através da linguagem ADL, é possível representar uma gama maior de problemas. Na Tabela 3.1 é apresentada uma comparação das características da representação utilizada na STRIPS e na ADL.

Tabela 3.1 - Comparação de características entre STRIPS e ADL

STRIPS	ADL
Apenas literais positivos nos estados	Literais Positivos e Negativos nos estados
Hipótese de domínio fechado	Hipótese de domínio aberto
Efeito P $\wedge \neg$Q: adicionar P e apagar Q	Efeito P $\wedge \neg$Q: adicionar P e $\neg$ Q e apagar $\neg$P e Q
Apenas proposições nos objetivos	Variáveis quantificadas
Objetivos são conjunções	Objetivos podem ser conjunções e/ou disjunções
Efeitos são conjunções	Efeitos condicionais permitidos: When P:E
Não suporta igualdade	Suporta igualdade
Não suporta tipos	Suporta tipos

3.1.3. PDDL: o padrão que surgiu da união entre STRIPS e ADL

A linguagem PDDL (*Planning Domain Definition Language*) (MCDERMOTT, 1998) é uma combinação de linguagens STRIPS e ADL com tarefas hierárquicas, recursos, tempo e metas com otimizações.

Foi desenvolvida especialmente para a primeira competição de planejadores realizada durante o congresso internacional AIPS (*Artificial Intelligence Planning Systems*), em 1998, com o objetivo de ser uma especificação padrão para representar os problemas de planejamento.

Assim passou a ser possível comparar a eficiência dos planejadores de IA para resolver problemas reais.

Abaixo as principais características da PDDL:

- A PDDL é uma representação direcionada às ações do domínio;
- As ações representadas em PDDL são baseadas em ações do modelo STRIPS (FIKES; NILSON, 1971) onde as pré-condições e efeitos de uma ação representam a dinâmica da execução desta no domínio;
- Possui características da linguagem ADL (PEDNAULT, 1989) que incluem a representação de efeitos condicionais nas ações, assim como qualificadores e quantificadores universais;
- Definição de restrições;
- Especificação de ações hierárquicas compostas por sub ações (MCDERMOTT, 1998);

- Devido ao fato da linguagem ser padronizada é possível que um mesmo problema representado em PDDL seja tratado por vários planejadores diferentes (portabilidade de problemas entre agentes planejadores).

A PDDL está em contínuo desenvolvimento desde sua criação. Uma das principais evoluções da PDDL foi a chamada PDDL 2.1 (FOX; LONG, 2003) que é utilizada como referência neste trabalho. A PDDL 2.1 foi desenvolvida almejando representar domínios de planejamento determinísticos que envolvam tempo e que necessitem de recursos de manipulação algébrica, incorporando também características da linguagem ADL (PEDNAULT, 1989).

A representação de um problema de planejamento em PDDL é composto por dois arquivos distintos: um que descreve o domínio e outro que especifica o problema. A descrição do domínio é única para os diversos problemas, que são compostos por descrições do estado inicial e metas a serem alcançadas, conforme pode ser observado na Figura 3.1. A separação da definição do domínio e dos problemas é um fator positivo já que para uma mesma definição de domínio é possível raciocinar sobre diversos problemas.

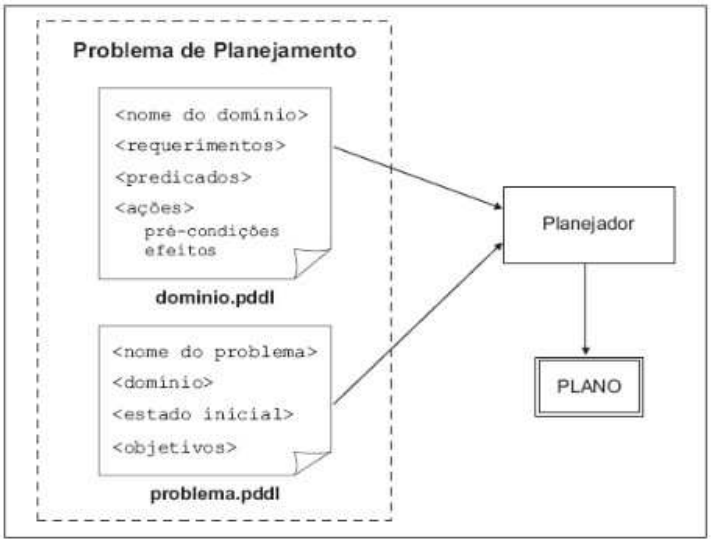

Figura 3.1 - Estrutura básica de um problema de planejamento em PDDL

No arquivo de domínio são especificados o nome do domínio, os requisitos da linguagem, os predicados existentes e as ações disponíveis. Já o arquivo para descrição do problema contém o nome do problema, o domínio, os objetos, o estado inicial e o objetivo.

A Figura 3.2 apresenta um exemplo de domínio em PDDL, representando estados e ações relativas à operação de um satélite.

```
(define (domain satellite)
 (:requirements :strips :equality :typing :fluents :durative-actions)
 (:types satellite direction instrument mode)
 (:predicates
             (on_board ?i - instrument ?s - satellite)
             (supports ?i - instrument ?m - mode)
             (pointing ?s - satellite ?d - direction)
             (power_avail ?s - satellite)
             (power_on ?i - instrument)
             (have_image ?d - direction ?m - mode) )
 (:functions (slew_time ?a ?b - direction))

 (:durative-action turn_to
   :parameters (?s - satellite ?d_new - direction ?d_prev - direction)
   :duration (= ?duration (slew_time ?d_prev ?d_new))
   :condition (and (at start (pointing ?s ?d_prev)) )
   :effect (and  (at end (pointing ?s ?d_new))
                 (at start (not (pointing ?s ?d_prev)))))

 (:durative-action switch_on
   :parameters (?i - instrument ?s - satellite)
   :duration (= ?duration 2)
   :condition (and (over all (on_board ?i ?s))
                      (at start (power_avail ?s)))
   :effect (and (at end (power_on ?i))
                (at start (not (power_avail ?s)))))

  (:durative-action switch_off
   :parameters (?i - instrument ?s - satellite)
   :duration (= ?duration 1)
   :condition (and (over all (on_board ?i ?s))
                      (at start (power_on ?i)))
   :effect (and (at start (not (power_on ?i)))
                (at end (power_avail ?s))) )

  (:durative-action take_image
   :parameters(?s- satellite ?d - direction ?i - instrument ?m - mode)
   :duration (= ?duration 7)
   :condition (and    (over all (on_board ?i ?s))
                      (over all (supports ?i ?m) )
                      (over all (power_on ?i))
                      (over all (pointing ?s ?d))
                      (at end (power_on ?i)))
   :effect (and (at end (have_image ?d ?m)))))
```

Figura 3.2 - Exemplo de um arquivo de domínio de um satélite em PDDL
Fonte: adaptada de Gonçalves (2006, p.65).

A Figura 3.3 apresenta um arquivo de problema para o domínio apresentado na Figura 3.2.

```
(define (problem sat-example)
(:domain satellite)
(:objects
      satellite0 - satellite
      instrument0 - instrument
      image1 - mode
      spectrograph2 - mode
      thermograph0 - mode
      GroundStation2 - direction
      Phenomenon4 - direction)
(:init
      ;; Definição do valor da função
      (= (slew_time Phenomenon4 GroundStation2) 39.73)
      (= (slew_time GroundStation2 Phenomenon4) 40.00)
      ;;Estado inicial do ambiente
      (supports instrument0 thermograph0)
      (on_board instrument0 satellite0)
      (power_avail satellite0)
      (pointing satellite0 GroundStation2))
(:goal (and
      (have_image Phenomenon4 thermograph0))))
```

Figura 3.3 - Arquivo de problema em PDDL para o domínio do satélite
Fonte: adaptada de Gonçalves (2006, p.67).

A descrição do problema contém objetos (*objects*), que são instanciações dos tipos definidos no domínio, e as descrições dos estados inicial (*init*) e objetivo (*goal*) do domínio. Em Gonçalves (2006) é possível ver um detalhamento maior deste domínio.

A PDDL possui um formalismo para as definições do domínio e do problema. Toda a definição formal da linguagem pode ser encontrada na especificação da PDDL 2.1 (FOX; LONG, 2003) ou em versões posteriores como a PDDL 2.2 (EDELKAMP; HOFFMANN, 2004) e PDDL 3.0 (GEREVINI; LONG, 2005).

No trabalho de (GONÇALVES, 2006), foram observadas algumas limitações da linguagem PDDL.

A primeira limitação encontrada foi em relação à dificuldade na geração dos arquivos de problemas a cada mudança do ambiente durante a fase de modelagem da base de conhecimento do domínio de rastreio de satélite. Após a geração correta do arquivo de domínio constatou-se que uma grande parte do trabalho se concentrava na geração dos arquivos de problemas.

Observou-se que a estrutura oferecida pela linguagem PDDL não permitia a classificação dos predicados no arquivo de domínio.

Uma limitação também encontrada na linguagem é de não permitir definir prioridades nos eventos exógenos. Isto significa que se dois eventos exógenos tem o mesmo horário de ocorrência, o planejador não consegue encontrar um plano válido.

Outra limitação da linguagem, também relacionada à prioridade, é de não permitir o estabelecimento de prioridades nos sub-objetivos a serem alcançados no plano.

3.1.4. UML – Unified Modeling Language

A UML – é uma linguagem semi-formal de modelagem, de propósito geral, que se tornou unanimidade e um padrão na modelagem e especificação principalmente de sistemas orientados a objetos. Foi padronizada pela OMG (*Object Mangement Group*) entre 1996 e 1997 (D´SOUZA; WILLS, 1999). Como a própria OMG descreve em OMG (2009):

> *A UML é uma linguagem gráfica para visualização, especificar, modelagem, construção e documentação de artefatos de um sistema. Ela representa a união das melhores práticas na modelagem orientada a objeto.*

A OMG (2009) define os seguintes objetivos para a UML:

- Fornecer ao usuário uma linguagem visual expressiva, pronta para o uso no desenvolvimento de modelos de negócios;

- Fornecer mecanismos para extensões e mecanismos de especialização para apoiar os conceitos essenciais;

- Ser independente de linguagem de implementação;

- Encorajar o crescimento do número de ferramentas com abordagem orientadas a objetos;

- Suportar conceitos de desenvolvimento de níveis mais elevados tais como colaborações, padrões e componentes;

- Integrar as melhores práticas de desenvolvimento de software.

Neste trabalho utilizamos o Diagrama de Classes, que descreve as classes (entidade) que formam o domínio, bem como as associações entre elas definindo a estrutura estática do domínio; e o Diagrama de Objetos que representa os objetos existentes no sistema e como eles interagem em um determinado instante.

3.1.5. OCL: Uma linguagem centrada em objetos

Implementada em PROLOG e com a idéia de se representar modelos de domínios como um conjunto de objetos sujeitos a várias restrições, a Object-Centered Language (OCL) proposta de Liu e McCluskey (2000), apesar de ser uma representação baseada em objetos, se mostra ainda restritas por descreverem o modelo como um conjunto de predicados, pré-definidos.

Um modelo em OCL é composto por objetos dinâmicos ou estáticos, agrupados em classificações. Isso é o que se chama em Orientação a Objetos (OO) de ‘classe’, ou na PDDL de type. Cada objeto dinâmico existe em um de seus conjuntos de estados (chamados substates), caracterizados por predicados. A aplicação de um operador irá mover os objetos de um conjunto de estados para outro. A OCLh adiciona a estes conceitos o relacionamento hierárquico entre sorts, permitindo criar objetos a partir da composição de outros, como ocorre em OO. A Figura 3.4 traz um trecho de uma descrição do domínio do mundo de blocos em OCL (KUCINSKIS, 2007).

```
sorts(primitive_sorts, [bloco]) .

objects(bloco,
        [bloco1, bloco2, bloco3, bloco4, bloco5, bloco6, bloco7]) .

predicates([
        sobre_bloco(bloco, bloco),
        sobre_mesa(bloco),
        bloco_livre(bloco),
        ]) .

substate_classes(bloco, B, [
      [sobre_bloco(B, B1), bloco_livre(B), ne(B, B1)],
      [sobre_bloco(B, B1), ne(B, B1)],
      [sobre_mesa(B), bloco_livre(B)],
      [sobre_mesa(B)] ]) .
```

Figura 3.4 - Exemplo de descrição de domínio do mundo de blocos em OCL
Fonte: Kucinskis (2007, p.51).

A instrução sorts define o tipo bloco. Este tipo é instanciado pela instrução *objects*, onde são criados os objetos *bloco1* a *bloco7*. A instrução *predicates* traz os atributos de um objeto do tipo bloco: ‘sobre_bloco’, ‘sobre_mesa’ e ‘bloco_livre’. Finalmente, a instrução *substate_classes* define explicitamente todos os possíveis sub-estados em que um objeto de determinado tipo pode se encontrar, indicando quais combinações de predicados são legais.

A OCL tem como grande contribuição permitir uma melhor estruturação e maior representatividade do conhecimento sobre o mundo sendo modelado. Mas, por ainda se basear em descrição por predicados, a linguagem é dependente da representação explícita de qualquer mudança nos estados do modelo,

exatamente como em STRIPS ou PDDL. De qualquer forma, a OCL representa um grande passo em direção à descrição de domínios de forma totalmente orientada a objetos (KUCINSKIS, 2007).

3.1.6. AML: Linguagem para modelagem de um planejador para missões espaciais.

Em 1997 foi apresentado o planejador para missões *espaciais Automated Scheduling and Planning Environment* (ASPEN – Fukunaga et al., 1997) desenvolvido pelo *Jet Propulsion Laboratory* (JPL) da NASA com o objetivo de ser um ambiente para o desenvolvimento aplicações de planejamento e escalonamento de propósito geral.

Para o ASPEN foi criado uma linguagem expressiva de modelagem de restrições, que permite ao usuário definir naturalmente o domínio da aplicação; nomeada como ASPEN *Modeling Language* (AML). Para se descrever um modelo em AML é necessário a confecção de sete componentes básicos: parâmetros, dependências entre parâmetros, restrições temporais, recursos, variáveis de estado, reservas e atividades.

Parâmetro é uma variável com um domínio restrito e bem-definido. Este domínio pode ser um subconjunto dos inteiros, por exemplo. Outros tipos de parâmetros aceitos incluem números de ponto flutuante, booleanos e *strings*.

A restrição temporal é o relacionamento entre o instante de início de uma atividade e o instante de término de outra qualquer.

Variáveis de estado representam todos componentes que possam fazer parte do domínio.

No ASPEN recursos representam perfil de consumo de um recurso físico no tempo. Os tipos de recursos disponíveis são os consumíveis e não-consumíveis.

Reservas são requisitos para a execução de atividades, com relação a recursos ou variáveis de estado. Por exemplo, uma atividade pode ter uma reserva de dez watts de energia. Isso significa que, durante a atividade, são consumidos estes dez watts. Pode-se especificar se este consumo será calculado no início ou no fim da atividade (KUCINSKIS, 2007).

Finalmente, a atividade é a estrutura central no ASPEN. Uma atividade representa uma ação ou um passo em um plano. Ela possui um momento de início, duração e um momento de término. Atividades podem consumir um ou mais recursos e podem conter sub-atividades, o que permite o uso de planejamento hierárquico (KUCINSKIS, 2007).

3.1.7. RASSO_ml: Forte integração entre programação do sistema e o processo de planejamento.

Kucinskis (2007) propôs uma arquitetura para o serviço de replanejamento embarcado nomeada como *Resources Allocation Service for Scientific Opportunities* (RASSO), composto por um modelo de satélite, um compositor de problemas e um planejador para o Computador Avançado (COMAV) que vem sendo desenvolvido pelo Grupo de Supervisão de Bordo (SUBORD) da Divisão de Eletrônica Aeroespacial (DEA) (KUCINSKIS, 2007), que seguindo uma tendência mundial, pretende unificar as tarefas de supervisão de bordo e o controle de atitude e órbita do satélite, que são usualmente realizadas por computadores distintos. Além de efetuar o controle e comunicação com experimentos (KUCINSKIS, 2007).

Devido às limitações de representações baseada em predicados herdadas do STRIPS e linguagens modernas de planejamento como PDDL e a OCL, como:

- A carência no que se refere à representação de recursos e de seu consumo;

- A forma limitada de descrição de pré-condições e efeitos – não há estruturas de loops ou seleções de casos, e nem todas as linguagens baseadas em predicados possuem condicionais;

- O fato de que modelos descritos nestas linguagens devem ser interpretados por um analisador (*parser*) e convertidos em estruturas de dados, antes de serem utilizados pelo planejador.

Foram criadas instruções para o planejador, que fazem parte do modelo, de forma a permitir uma forte integração entre a programação do sistema e o processo de planejamento, tornando a ligação entre o modelo, o planejador e o restante do software mais natural. Estas instruções foram implementadas através de macros da linguagem C, usada no desenvolvimento do COMAV.

As macros ocultam a criação e a alimentação das estruturas, vetores, funções e outros elementos utilizados pelo planejador, e fazem com que a descrição do domínio seja mais legível e próxima à linguagens de planejamento, como a AML. A este conjunto de instruções foi dado o nome de RASSO *modeling language*, ou apenas RASSO_ml.

3.1.8. ITSIMPLE: Ambiente integrado de modelagem e análise de domínios de planejamento automático

Neste trabalho, Vaquero (2007) propôs um ambiente integrado de modelagem e análise de domínio de planejamento automático, inserido no contexto de ciclo de vida de projeto que, além de favorecer as boas práticas já presentes no

desenvolvimento de software, sistemas de informação e sistemas de engenharia em geral, focalizam principalmente as fases iniciais do ciclo de vida de projeto, tais como a definição do problema e o design (modelagem, análise, design preliminar e testes de protótipos). O Ambiente proposto contempla não somente a modelagem e análise de domínios, mas também processos que antecedem essas atividades, como a análise de requisitos e geração de especificações que serão tratadas de forma similar ao desenvolvimento de sistemas de engenharia em geral.

O Ambiente visa integrar as ferramentas e linguagens mais adequadas e mais utilizadas nas fases iniciais de projeto para que os progressos já atingidos sejam utilizados. O processo de Modelagem de Domínios de Planejamento segue uma abordagem orientada a objetos utilizando a UML, através de seus diagramas. Devido ao fato dos domínios de planejamento possuírem características específicas, algumas regras são sugeridas durante a modelagem utilizando a UML de modo a gerar um processo disciplinado de modelagem para este tipo de problema (VAQUERO, 2007).

3.1.9. Mecanismos de apoio a engenharia do conhecimento

Mecanismos de apoio a desenvolvedores durante a modelagem de domínios de planejamento quase escassas antes da primeira Competição Internacional de Engenharia do Conhecimento para Planejamento e Escalonamento (ICKEPS) em 2005. Antes do ICKEPS 2005 poucos trabalhos tratavam da elaboração e desenvolvimento desses mecanismos. O trabalho do pesquisador McCluskey et at. (2003) onde conceitos e sugestões de processos de aquisição e modelagem de domínios de planejamento eram propostos através de um ambiente de planejamento idealizado, foi um dos primeiros a abordar o tema. Assim, este trabalho passou ser a principal referência da área da EC para Planejamento. Como fruto dessa linha de pesquisa, McCluskey juntamente com Simpson idealizaram plataformas e ferramentas que contribuíssem nos

processos de modelagem, verificação e validação de modelos. A ferramenta pioneira, desenvolvida dentro deste ideal, se chamou GIPO (*Graphical Interface for Planning with Objects*) (SIMPSON et al., 2001).

A ferramenta GIPO é uma das únicas com foco direcionado aos processos de aquisição de conhecimento e modelagem do domínio. Esta também possui uma abordagem orientada a objetos durante a modelagem, o que vem mostrando ser uma abordagem interessante no contexto de planejamento. GIPO se tornou uma ferramenta de referência nesta área, mas a mesma possuía algumas restrições tais como: a utilização de uma linguagem não tão conhecida fora do âmbito acadêmico, a OCL (*Object Centered Language* (SIMPSON et al., 2001)); projetistas sem conhecimento em planejamento automático ou conceitos de IA podem ter dificuldades no uso da mesma.

Ferramentas como "*Common Process Editor*" (TATE et al., 1998) e "Act Editor" (MYERS; WILKINS, 1997) também se destacaram antes da competição, mas estas eram ferramentas específicas a alguns domínios. De fato, para lidar com problemas reais de planejamento são realmente necessários ferramentas e métodos que auxiliem o projetista a modelar seus domínios com grande flexibilidade e portabilidade. As ferramentas podem contribuir para a síntese de modelos, análises automáticas de domínios, correções de sintaxe, remover redundâncias, visualização do modelo, verificação, validação entre outras contribuições. Essas contribuições são similares àquelas encontradas em ferramentas de apoio à modelagem e desenvolvimento de softwares ou de sistemas de engenharia em geral.

Diante deste cenário o desenvolvimento de uma ferramenta que considera a modelagem de domínios de planejamento, se fez necessário.

A Tabela 3.2 faz uma comparação de características das principais linguagens de representação de conhecimento para domínios e problemas do planejamento automático.

Tabela 3.2 - Comparação de características entre linguagens de descrição de domínios de planejamento

Características	STRIPS	ADL	PDDL	AML	OCL
Tipos Definidos		X	X	X	
Definição de Estados	X	X	X	X	X
Ações	X	X	X	X	X
Pré-condições	X	X	X	X	X
Efeitos	X	X	X		X
Efeitos Condicionais		X	X		
Recursos				X	
TEMPO				X	
Literais Abertos		X	X		X
Objetivos Predominantes					
Prioridade de Eventos (Exógenos)					
Subações			X		
Portabilidade de Plataforma					
Consistência entre Domínio e Problema					X
Baseada em Objetos					X
Objetos Dinâmicos					X
Subestados					X
Definição de Operadores(Expressões de Avaliação)					X
Estruturas de seleção e loops(Expressões de Avaliação)					
Subdomínios					
Interface para planos					
Abstração de Entidades					
Sintaxe textual	X	X	X	X	
Intervalo de valores					
Definição de Atividades Condições					

3.2. Considerações

Neste capítulo é foram apresentadas as linguagens STRIPS, ADL, PDDL, AML, OCL e RASSO_ML para descrição de domínios. Este estudo possibilitou o levantamento de benefícios e limitações destas, contribuindo de forma significativa para a elaboração do meta-modelo. Também foi feito um estudo

sobre a UML para que o meta-modelo fosse apresentado em uma linguagem de modelagem rica e poderosa, independente de plataforma de implementação.

No próximo capítulo é apresentado o modelo proposto para modelagem e descrição de domínios e problemas de planejamento. Trata-se de um meta-modelo orientado a objetos genéricos para domínios de planejamento automático, possibilitando a descrição de uma variedade de diversos domínios, o modelo é composto por um conjunto de classes que se relacionam de maneira a estabelecer regras para a modelagem dos domínios de planejamento.

4 UM META-MODELO ORIENTADO A OBJETOS PARA DESCRIÇÃO DE DOMÍNIOS E PROBLEMAS DE PLANEJAMENTO

Os conceitos apresentados no Capitulo 2 foram estudados de forma a levantar aspectos e características de um ambiente de modelagem de um domínio de planejamento automático. O propósito foi examinar a classe de problemas da área espacial e também de outras áreas como: Jogos, Controle de Tráfego e etc, que são evidenciados pela necessidade do uso do planejamento automático, e as técnicas existentes para modelá-los. Para tal foram estudados métodos, linguagens e ferramentas que contribuíssem para os processos de modelagem de domínios de um sistema deste tipo.

No decurso de muitos anos, domínios e problemas de planejamento modelados por pesquisadores da área eram considerados clássicos, ou seja, os domínios eram simplificados de forma a dispensar processos rigorosos de especificação e modelagem. O enunciado destes domínios e problemas eram bem definidos viabilizando assim representá-los diretamente em linguagens, como a PDDL, que muitas vezes não possibilitavam uma clara visão do domínio, e/ou não disponibilizava mecanismos eficientes de verificação e validação de modelos, e/ou são mais próximas de linguagem de máquina a ser interpretada por sistema de planejamento. As pesquisas de técnicas de planejamento eram voltadas basicamente aos problemas clássicos, não existia a necessidade de métodos disciplinados de especificações e modelagem.

Porém quando pesquisadores da área passaram a abordar problemas reais de engenharia, o uso de fundamentos e conceitos práticos aplicados a modelagem de domínios de sistemas reais começaram a ser, além de necessário, fundamental para a evolução das ferramentas, bem como um melhor entendimento destes problemas reais. Existem trabalhos referentes à modelagem e descrição de domínios de planejamento na literatura como os descritos no capítulo 3, mas pode-se dizer que poucos contemplavam um

modelo genérico para descrição de diversos domínios relacionado a missões espaciais e também outras áreas.

A proposta de um Meta-Modelo Genérico para Descrição e Modelagem de Domínios de Planejamento (referenciado aqui apenas como Meta-Modelo), apresentado neste capítulo, inserido no contexto de projetos da área espacial, propicia a aplicação das boas práticas de modelagem provenientes da metodologia OO, focando principalmente os benefícios desta, tais como o uso de padrões de projetos já conceituados na literatura para a implementação de vários tipos de serviços e algoritmos de planejamento, permitem uma grande flexibilidade na implementação de planejadores e possíveis serviços.

Visando permitir maior flexibilidade nas definições do domínio, o modelo foi construído levando em consideração as reais necessidades de um problema real de planejamento da área espacial e também de alguns problemas como de logística, seqüenciamento de carros e alguns problemas clássicos da área de planejamento automático.

Discute-se no início deste capítulo alguns pontos relevantes dos domínios de planejamento que devem ser levados em consideração para a modelagem destes. O modelo proposto é então apresentado baseado no cenário das áreas do Planejamento Automático e da Engenharia do Conhecimento aplicada ao Planejamento Automático apresentado no Capítulo 2.

Neste tópico é apresentado o conceito do Modelo, suas interfaces, classes, atributos e relacionamentos. Por fim são discutidos aspectos de alguns trabalhos já desenvolvidos no contexto de modelagem de domínios de planejamento, apresentando a necessidade de um Meta-Modelo como o proposto.

4.1. Características dos domínios da área de planejamento automático

De acordo com a fundamentação teórica apresentada no Capítulo 2, a área de Planejamento Automático em IA estuda o raciocínio automático inteligente de planejamento envolvendo basicamente objetos, ações, eventos, objetivos, atividades, tempo e recursos. A base principal do Planejamento Automático é conhecimento. Conforme observado, o planejamento trabalha com conhecimento de fenômenos complexos tais como as ações. Assim, a representação do conhecimento através de um modelo de domínio acaba sendo um dos principais aspectos no desenvolvimento de sistemas de planejamento (MCCLUSKEY et. al., 2003).

Os desafios encontrados na formação do modelo de domínios de uma aplicação de planejamento são proporcionais ao tamanho e complexidade desta.

Algumas propriedades dos domínios de planejamento como: tempo, recursos, meios de comunicação, por vezes dificultam a modelagem e o entendimento do mesmo, principalmente quando se trata de domínios reais. Um domínio pode ser composto por: número infinito de ações e estados, efeitos de ações desconhecidos, estados desconhecidos, incertezas, restrições e relacionamentos desconhecidos, ambiente não observável, eventos exógenos, entre outros.

4.2. O meta-modelo proposto

Mediante as evidencias da necessidade de pesquisas na área de planejamento automático que eram apresentados por trabalhos e panoramas desta área, os desafios encontrados nos domínios e a falta de ferramentas de modelagem destes, motivaram o desenvolvimento de um Meta-Modelo para descrição de domínios de planejamento, apresentado neste tópico. Este trabalho teve seu desenvolvimento a partir dos seguintes aspectos:

- A necessidade de unir a teoria à prática de planejamento (MCDERMOTT; HENDLER, 1995) (GIL et al., 1995) (GHALLAB et al, 2004);

- O fato de pesquisadores almejarem trabalhar com problemas reais de planejamento, incluindo o CCS no INPE;

- Pode ser observado em alguns trabalhos que por menor que seja a variação na representação dos domínios de planejamento o desempenho de um planejador pode ser impactado. Isso ressalta a importância de se ter uma ferramenta bem elaborada para a descrição de domínios de planejamento;

- Falta de métodos e ferramentas de modelagem de domínios de planejamento automático (MCCLUSKEY et al., 2003);

- A PDDL é viável apenas para problemas clássicos e não para problemas reais já que problemas reais possuem grandes desafios de modelagem;

- Dificuldade na visualização e entendimento das linguagens utilizadas em Planejamento Automático;

Também foi levado em consideração o trabalho de McCluskey et al. (2003), intitulado “Knowledge Engineering for Planning ROADMAP”, onde são levantados vários aspectos sobre modelagem de domínios, linguagem de representação, ferramentas e ambientes de suporte à engenharia do conhecimento, ontologias para planejamento, entre outros aspectos.

Baseado nestes aspectos, trabalhos que contribuam com conhecimento e experiência na área de planejamento são bem reconhecidamente importantes para toda comunidade de pesquisa e para possíveis usuários em aplicações práticas, como o CRC, por exemplo.

Inserido neste contexto este trabalho descreve, focado nas fases de modelagem e análise do modelo do domínio, um Meta-Modelo que abstrai conceitos e fundamentos da área de planejamento para servir como ferramenta para modelagem e descrição de domínios desta área.

O principal objetivo do Meta-Modelo é ser uma ferramenta de alta flexibilidade, com baixo acoplamento, alta coesão e grande reutilização de componentes para modelagem de domínios de sistemas de planejamento automático. O Meta-Modelo proposto consiste em classes que abstraem a base do conhecimento para descrição de domínios e problemas de planejamento. Na Figura 4.1 observam-se as abstrações identificadas e representadas pelo Meta-Modelo.

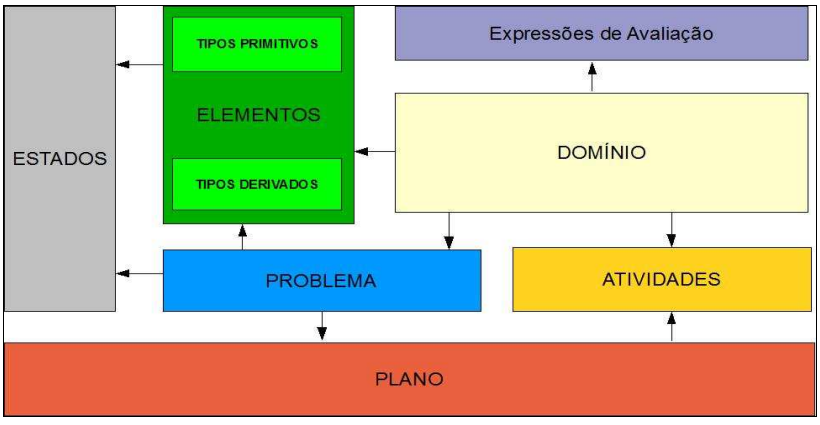

Figura 4.1 - Abstrações identificadas para a descrição de domínios e problemas de planejamento representadas pelo meta-modelo.

Estas abstrações, foram estipuladas a partir do estudo feito sobre o conhecimento necessário para a execução de planejadores automáticos e

necessidades específicas para modelagem de domínios e problemas reais como o da área espacial.

- *Estados*: Todos os elementos do domínio serão constituídos por estados que estarão em formas de cenários, situações e condições dos objetos modelados;

- *Elementos*: Os elementos irão definir os objetos, os respectivos tipos de um domínio e restringir como instâncias destes tipos poderão ser relacionadas e utilizadas;

- *Domínio*: O domínio será composto por um conjunto de outras instâncias, direta ou indiretamente formando assim sua estrutura de propriedades e características;

- *Expressões de Avaliação*: Expressões que irão avaliar condições e situações estabelecidas, podendo receber e retornar valores;

- *Atividades*: Ações e funções que farão parte do domínio, sendo que estas terão propriedades, valores e variáveis inerentes ao contexto do planejamento;

- *Problema*: Problemas para resolução através de planejamento para um dado domínio, no problema serão representados os estados iniciais, objetivos e o conjunto de objetos associados a este;

- *Plano*: Resultado da execução de um planejador para um problema baseado no domínio associado a este. Um plano é composto por uma coleção de atividades (ações) que deverão ser executadas na ordem fornecida.

Este trabalho propõe um mecanismo de modelagem e análise do modelo do domínio direcionado ao design de domínios de planejamento automático, utilizando a metodologia OO e a linguagem UML como forma de representação.

O Meta-Modelo proposto, chamado de *KPlanOO* (*Knowledge of Planning Oriented Objects*) através de sua estrutura (classes, relacionamentos, herança e interfaces), impõe regras que garantem a consistência da modelagem dos domínios.

O *KPlanOO* está inserido principalmente no processo de Especificação, Modelagem, Análise do Modelo do Domínio no ciclo de vida de um projeto de software de Planejamento Automático. Este trabalho propõe um mecanismo de modelagem e análise do modelo do domínio direcionado ao design de domínios de planejamento automático, utilizando a metodologia OO e a linguagem UML como forma de representação.

Os tópicos a seguir apresentam o *KPlanOO* e aspectos relacionados ao processo de modelagem utilizado pelo mesmo. Utilizar-se-á, no decorrer da exposição do KPlanOO e do processo de modelagem como exemplo, o domínio clássico de planejamento chamado Logística. Trata-se de um domínio simples (e fechado), composto por: Caminhões e Aviões que transportam Pacotes entre Lugares. Caminhões se locomovem entre Locais da mesma Cidade e Aviões somente entre Aeroportos.

4.2.1. Mecanismo de modelagem

A carência de mecanismos de modelagem de domínios de planejamento é um dos principais fatores para a dificuldade na modelagem destes domínios. Os Mecanismos e ferramentas de modelagem freqüentemente surgem através de experiências de profissionais que acumularam vivência no desenvolvimento de

sistemas. Porém em Planejamento Automático, mecanismos e ferramentas de modelagem estão ainda surgindo, sendo este trabalho um exemplo.
Essa proposta é embasada principalmente nas boas práticas de modelagem provenientes do desenvolvimento de software orientado a objetos, onde a UML é amplamente utilizada. Modelos criados em UML normalmente são desenvolvidos a fim de torná-los o mais próximo possível da implementação, utilizando-se de termos e estruturas que facilitem a implementação.

Neste trabalho foi criado um Meta-Modelo utilizando o diagrama de classes da UML, visando possibilitar que domínios sejam descritos e modelados o mais próximo do real. Alguns aspectos foram cuidadosamente tratados para que planejadores pudessem operar sobre o *KPlanOO*, respeitando princípios da área de Planejamento Automático abordados no capítulo 2. Estes aspectos serão apresentados durante a explicação do *KPlanOO* e da modelagem de domínios através do mesmo.

Nos tópicos a seguir são apresentadas, primeiramente, a fundamentação para a construção do *KPlanOO*, suas entidades, relacionamentos e atributos através de um diagrama de classes da UML e em seguida a proposta de modelagem do domínio e problemas de planejamento que faz uso do mesmo.

4.2.2. Fundamentação do meta-modelo

Sowa (1999) define o termo representação de conhecimento como a aplicação da lógica e da ontologia na tarefa de construir modelos computacionais para algum domínio e para algum propósito. A Modelagem de Domínios se aplica na Gestão de Conhecimento pela introdução de técnicas estruturadas ou formais de descrição de domínios. Em particular, na atual adoção da UML em diversos campos do conhecimento.

Nesta proposta foram levados em consideração os conceitos de ontologias e modelos de domínios no desenvolvimento do *KPlanOO* como um mecanismo de descrição de domínios de planejamento automático.

Baseado nas definições de Guarino (1998) sobre ontologia para ciência da computação, onde é estabelecido que: "*uma ontologia é uma teoria lógica para relacionar o significado pretendido de um vocabulário formal, isto é, seu comprometimento com uma conceitualização particular do mundo*"; e na definição de Fonseca (2000) para componentes em que "*ontologia é uma teoria de especificação de vocabulário relativo a um determinado domínio definindo entidades, classes, funções e relacionamentos entre componentes*".

O conceito de Meta-Modelo, busca a idéia, da descoberta do todo, como pode ser observado em "Os investigadores", onde Boorstin cita que "Einsten via o pequeno e o grande, o atômico e o cósmico, como um enigma, a fim de descobrir o todo, explicado por leis e pela razão, inspirado pelo que ele chamava de seu sentimento religioso cósmico." (BOORSTIN, 2003); e que segundo Gómez-Sanz (2001) um Meta-Modelo também pode ser visto como uma representação dos tipos de entidades que podem existir em um modelo, suas relações e restrições de aplicações.

O *KPlanOO* foi desenvolvido orientado a objetos como um modelo do domínio da área de planejamento automático, pois assim alcançamos o frisado por Gómez-Pérez (1999) que, para a construção de uma ontologia, cinco tipos de componentes têm que ser levados em conta: conceitos (termos ou classes, e seus domínios de valores), relacionamentos, funções (relações especiais onde o n-ésimo elemento da relação é único para os n-1 elementos precedentes), axiomas (modelam sentenças que são sempre verdadeiras) e instâncias.

O uso combinado de ontologias e modelos de domínios já foi abordado em outros trabalhos como (MANGAN et al., 2001), (FARIA, 2000) e (GUIZZARD, 2000-A), porém até o momento não foi proposto como modelagem do domínio

Planejamento Automático um meta-modelo que seja genérico o suficiente para a descrição de domínios relacionado a problemas reais (dentre eles missões espaciais) e que seja uma alternativa às atuais linguagens utilizadas em representação de problemas de planejamento.

As características: flexibilidade, baixo acoplamento, alta coesão e reutilização de componentes, presentes no *KPlanOO* supre o primeiro nível de divergência em ontologia descrito por Klein (2001), que é o da linguagem de descrição ou Meta-Modelo onde as divergências inclusas neste nível são as diferenças sintáticas, diferenças no significado das primitivas, nas diferentes linguagens e diferenças na expressividade das linguagens, pois o modelo proposto é baseado em objetos inerentes ao escopo do Planejamento Automático não sendo acoplado a nenhuma primitiva pré-estabelecida.

A construção do *KPlanOO* apesar de ter sua motivação maior no domínio da área espacial, mais precisamente no que se refere ao controle de atividades de satélites, baseou-se nos tipos de ontologia apresentados por SOUZA et al., (1998) chegando a um Meta-Modelo genérico capaz de modelar todo o conhecimento para o processo de planejamento, cobrindo diversos domínios e problemas e, em especial, aqueles da área espacial.

Abaixo os tipos de ontologias utilizadas como base para o desenvolvimento do Meta-Modelo proposto:

- *Meta-Ontologias*, também chamadas de Ontologias Genéricas ou Ontologias Fundamentais, que são reutilizáveis (ou aplicáveis) em diferentes domínios.

- *Ontologias de Domínio* são reutilizáveis em um dado domínio provendo vocabulários sobre os conceitos dentro de um domínio e

seus relacionamentos, sobre as atividades que envolvem este domínio e sobre as teorias e princípios elementares que governam aquele domínio.

- *Ontologias de Aplicações* que contêm o conhecimento necessário para modelar situações específicas de uma tarefa em um domínio particular.

Um projeto de ontologia tem abordagens baseadas em projetos de orientação a objetos (RUMBAUGH et. al., 1991) (BOOCH et. al., 1999). Porém, o desenvolvimento de ontologias se difere da modelagem de classes e relacionamentos numa modelagem orientada a objetos (representada em UML). Noy & McGuinness (2001) esclarecem que a modelagem orientada a objetos foca principalmente os métodos em classes — um analista toma decisões baseado em propriedades operacionais de uma classe, enquanto um engenheiro de ontologias toma decisões baseado em propriedades estruturais de uma classe, explicitando suas relações e formalizando os conceitos. Como resultado, uma estrutura de classe e relações em uma ontologia pode ser diferente de uma estrutura em uma modelagem orientada a objetos para um domínio similar (BOOCH et al., 1999).

Assim o *KPlanOO* apesar de ter utilizado a OO como metodologia, foi desenvolvido baseando-se na generalização das propriedades estruturais do domínio Planejamento Automático, chegando assim a um modelo estruturado de classes componentizadas que se relacionam de forma a estabelecer conceitos e métricas fortes para a modelagem do domínio em questão.

Utilizou-se o Diagrama de Classes para a modelagem da estrutura estática do Meta-Modelo e é utilizado o Diagrama de Objetos para a representação do modelo do domínio de planejamento a ser modelado com o *KPlanOO*.

4.2.3. Estrutura estática

O Diagrama de Classes é o mais utilizado e o mais conhecido, dentre os diagramas que a UML disponibiliza, para representar e modelar a estrutura estática de domínios com uma abordagem orientada a objetos. Este diagrama é utilizado para representar o modelo estático conceitual (abstrato) do domínio de planejamento mostrando as entidades (chamadas aqui de “classes” ou “tipos abstratos”) existentes, os seus relacionamentos, atributos, características, métodos (similar a funções) e restrições.

A versão atual do *KPlanOO* que será apresentado é uma evolução das versões apresentadas no *IV Fórum de Inteligência Artificial da Região Sul* (Silva et. al., 2008) e no *II Seminário de Pesquisa em Ontologia no Brasil* (Silva et. al., 2009). A presente versão apresentada na *SpaceOps 2010 Conference Delivering on the Dream*, organizada pela AIAA (American Institute of Aeronautics and Astronautics) é composta por 18 classes com seus respectivos relacionamentos e atributos e 2 Interfaces. Sua estrutura estática encontra-se no diagrama de classes apresentado na Figura 4.2.

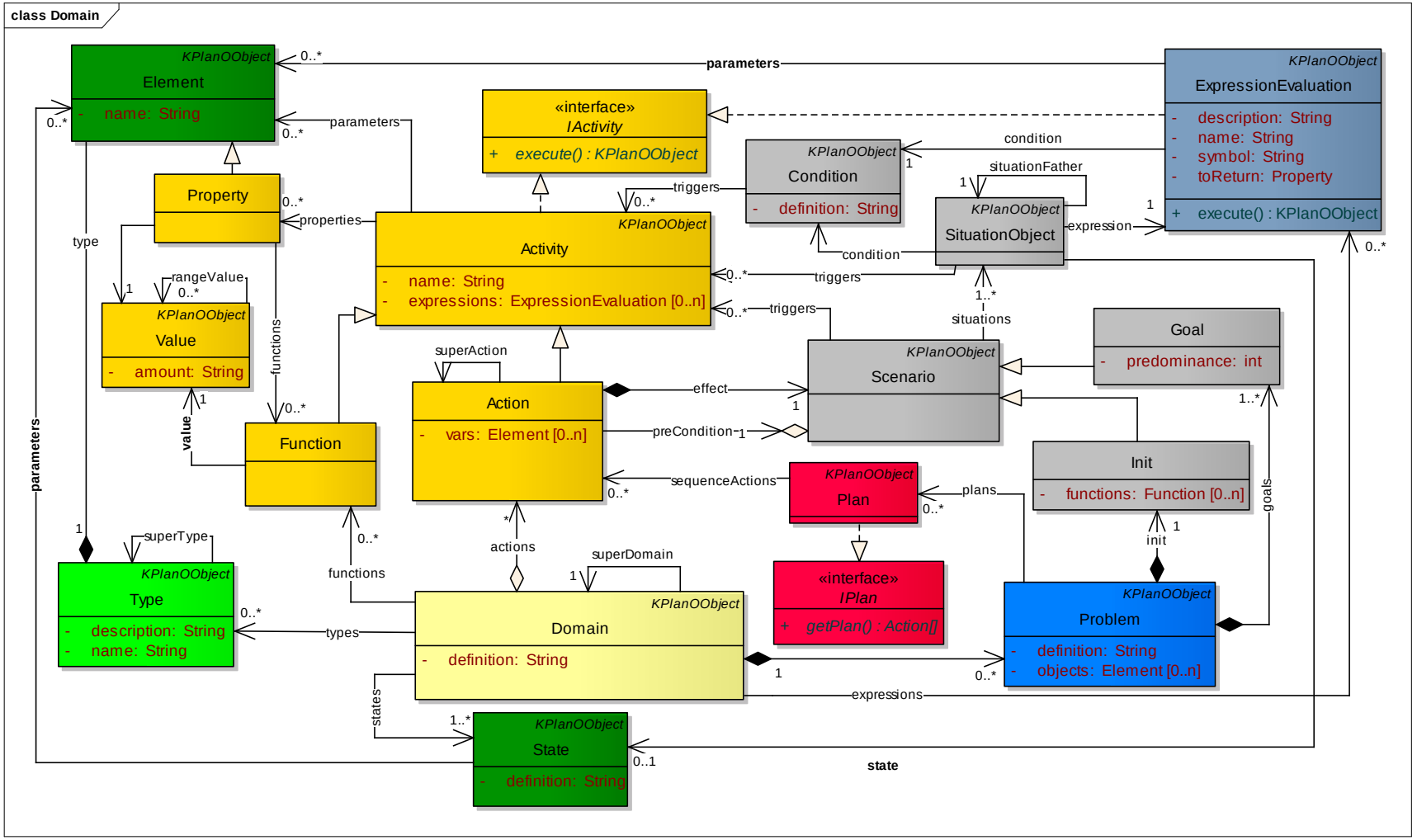

Figura 4.2 - Estrutura estática do KPlanOO

Os atributos e métodos das classes mais suas associações, fornecem uma boa visão e noção da semântica do modelo. Qualquer relacionamento entre duas ou mais classes é considerado uma associação.

As associações simples, agregação e composição geralmente fazem com que o modelo do domínio se torne semanticamente mais próximo do domínio real, pois demosntram o grau de realacionamento entre as entidades. As associações simples, interligam classeS identificando algum significado apropriado.

Agregações e Composições já expressam significados como, por exemplo, “possui” (“pertence”) e “constituído de”, respectivamente. Normalmente associações desses tipos possuem nomes, com uma direção semântica, *rolenames* e multiplicidades em ambos os lados. Os *rolenames* representam as navegações entre as classes de uma associação, e as multiplicidades representam restrições (regras), indicando a maneira como os objetos se relacionam que permitem o raciocínio sobre situações e estados válidos do modelo.

Pode-se observar, no *KPlanOO,* as regras de relação através da multiplicidade como: um problema estará associado a um e somente um domínio(*1*); já em um domínio pode ter nenhum, um ou muitos problemas associados (*0..**).

O diagrama apresentado na Figura 4.2 apresenta as associações que fazem sentido no domínio de Planejamento Automático como, por exemplo, a associação “*effect*” entre as classes *Action* e *Scenario* apresentando um significado visual para usuários não especialistas em planejamento ou em modelagem.

Algumas informações são muito importantes no Diagrama de Classe e que foram muito utilizadas no desenvolvimento do *KPlanOO*. Por exemplo, os

conceitos de relações flexíveis (associação) e relações rígidas (definidos por Ghallab et a. (2004)). As relações flexíveis representam as associações que mudam durante a evolução do ambiente (durante o ciclo de vida dos objetos). As relações rígidas representam associações que não variam de estado para estado. Um bom exemplo de relação flexível é a associação "*sequenceActions*" entre as classes *Plan* e *Action,* em que um plano pode ter sua seqüência de ações alteradas durante o processo de planejamento. Uma relação rígida, pode ser observada através da associação (restrição) *Type*, entre as classes *Element* e *Type*, representando os tipos disponíveis no domínio podendo-se dizer que é imutável. Normalmente as associações de agregação e composição também são relações rígidas nos domínios de planejamento.

Utilizou-se o conceito de herança proveniente da orientação a objetos para melhorar o modelo, para entidades que possuíam estruturas e características em comum. As Heranças podem ser adicionadas através da generalização de classes, por exemplo, a classe *Activity* é vista como uma superclasse de *Action* e *Function* onde estas duas classes herdam todas as características da classe *Activity,* incluído as associações.

A baixo é apresentado a descrição de cada classe do *KPlanOO* e também diagramas de objetos com os respectivos valores da descrição do domínio Logística.

A classe *Type* na Figura 4.3 é a abstração dos tipos que podem ser criados em um domínio, ex: Inteiro, Satélite, Região, Veículo, etc...

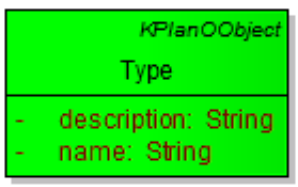

Figura 4.3 - Destaque classe Type

No domínio de logística existiria os tipos apresentados na Figura 4.4:

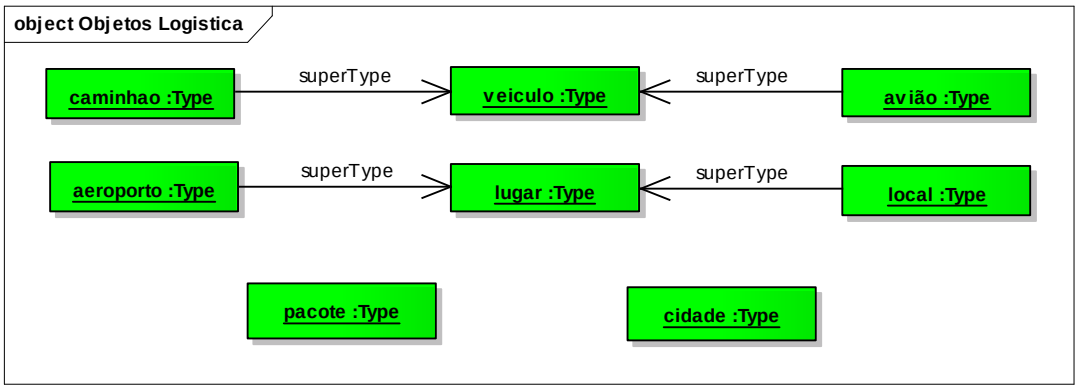

Figura 4.4 - Diagrama de objetos da classe Type para o domínio de logística

Element na Figura 4.5 abstrai os objetos que irão fazer parte do problema, também abstrai os parâmetros dos estados e funções e variáveis necessárias na descrição de ações. Todo elemento deverá ter um tipo associado, sendo que este tipo deve estar associado ao domínio em que o elemento está sendo declarado.

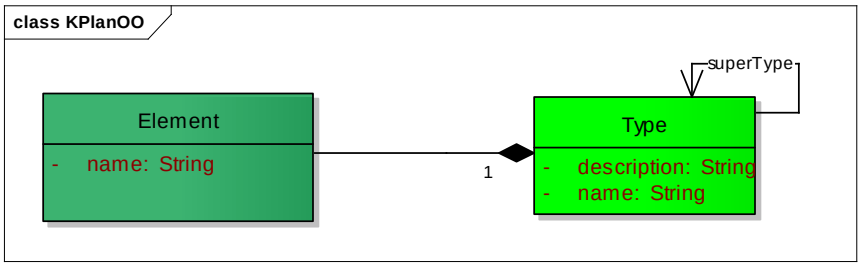

Figura 4.5 - Destaque classe Element

State na Figura 4.6 representa estados definidos no domínio que serão aplicados aos objetos de problemas. Pode-se fazer uma analogia deste conceito com o conceito de predicados de outras Linguagens de Modelagem de Domínios como PDDL.

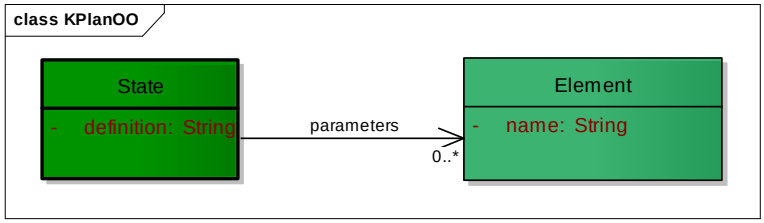

Figura 4.6 - Destaque classe State

A Figura 4.7 mostra o diagrama de objetos com possíveis estados para o domínio de logística com seus respectivos parâmetros:

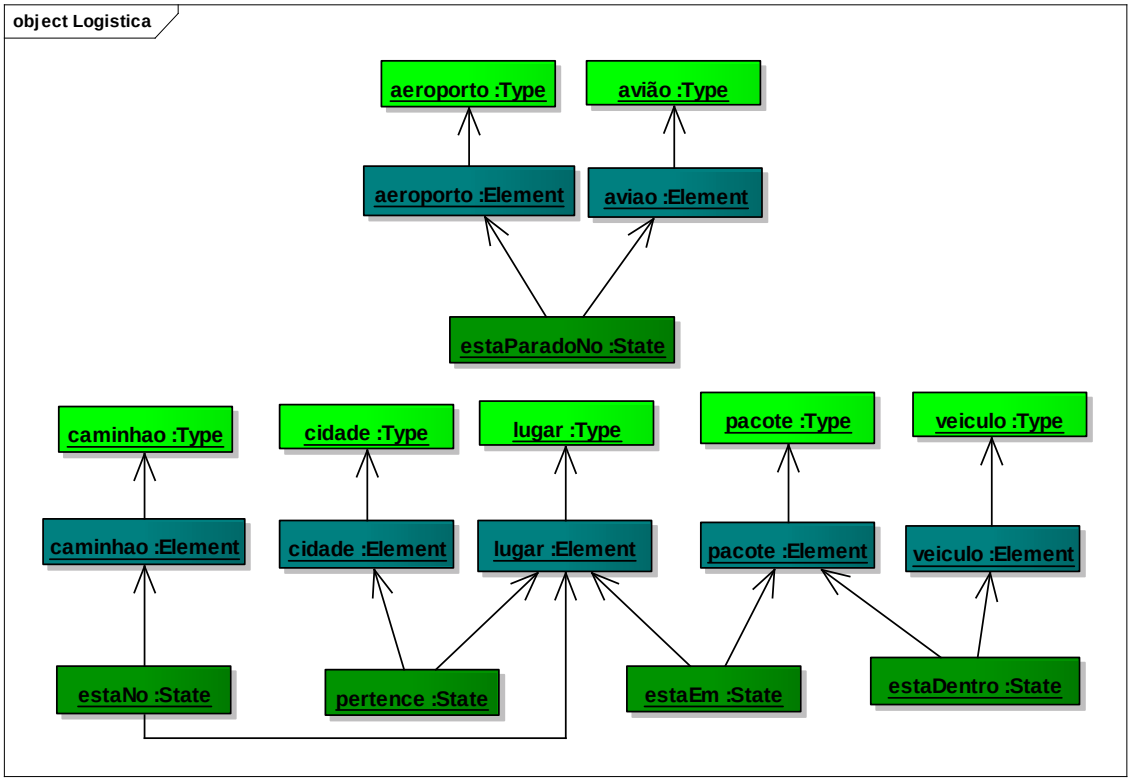

Figura 4.7 - Diagrama de objetos da classe State do domínio de logística

ExpressionEvaluation na Figura 4.8 é uma abstração de expressões de avaliação, estas serão aplicadas em elementos passados como parâmetros. Esta classe dispõe de quatro atributos, um para representar o nome da expressão, um para a descrição, um para um símbolo para esta descrição e um atributo que representa o resultado que será gerado pela expressão. Após a aplicação da expressão de avaliação será gerada uma condição sob o objeto

avaliado. Exemplos de expressões que poderão ser criadas: > (maior), < (menor), + (soma), &(and) |(or), etc....

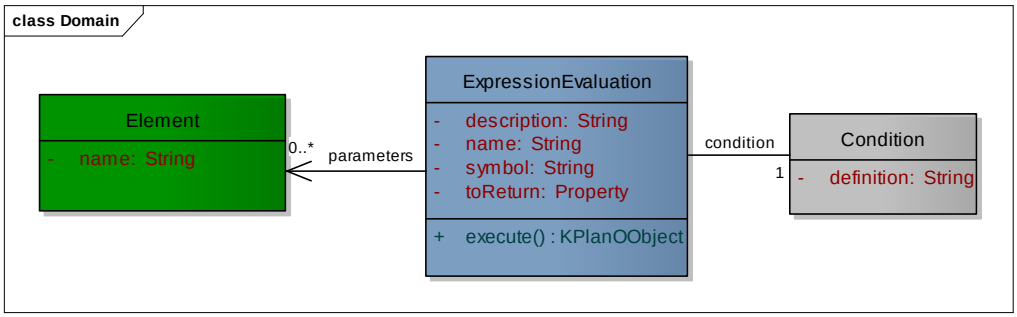

Figura 4.8 - Destaque classe ExpressionEvaluation

SituationObject Figura 4.9 é a entidade que representa as situações dos objetos que serão manipulados durante o planejamento. As situações é o conjunto formado pelo estado dos objetos pelas condições atreladas a este estado ou/e por uma expressão de avaliação. É possível ter uma super situação e eventos associados a cada situação, que são executados dada a situação.

Condition na Figura 4.9 é a abstração de condições que um objeto em um determinado estado se encontra, cada condição pode ter eventos associados, que são definidos por uma lista chamada "*triggers*".

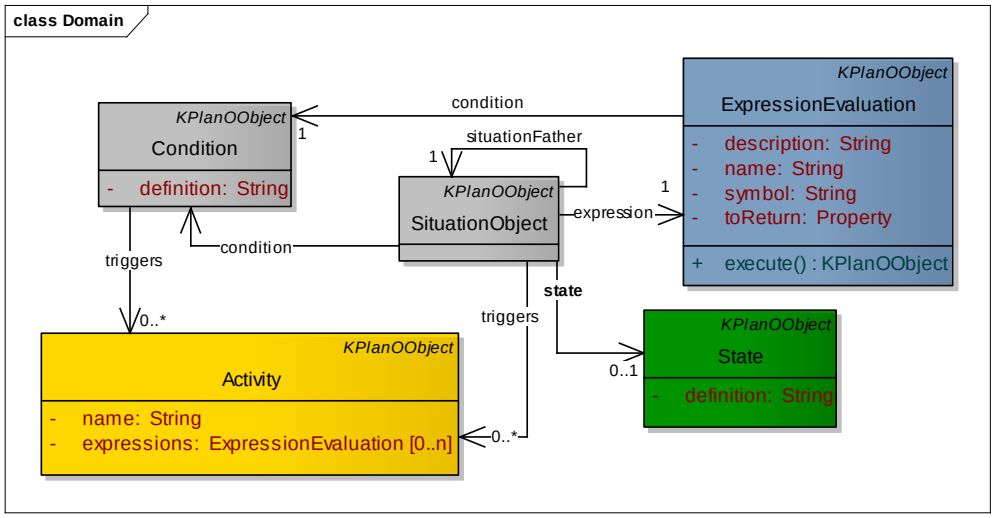

Figura 4.9 - Destaque classe SituationObject

IActivity na Figura 4.10 é a interface para atividades de um domínio.

Activity na Figura 4.10 é a abstração de atividades de um domínio que podem ser utilizadas no processo de planejamento, estas possuem uma lista de elementos que representam seus parâmetros e também podem ter uma lista de propriedades.

Value na Figura 4.10) representa um valor que será configurado no domínio tanto para uma função como para uma propriedade. A classe *Value* tem um auto-relacionamento que possibilita a configuração de um intervalo de valores.

Property na Figura 4.10 é a entidade que representa propriedades de uma atividade, essas propriedades podem ter seus valores definidos através da execução de uma função.

Function na Figura 4.10 abstrai possíveis funções que possam fazer parte de um domínio, possui uma lista de propriedades herdadas que possibilitam a configuração de valores iniciais ou uma gama de valores. Também possui uma lista de elementos que herdada que representam os parâmetros destas funções. Com esta abstração é possível inserir regras utilizando-se de um conjunto de *StateObject*.

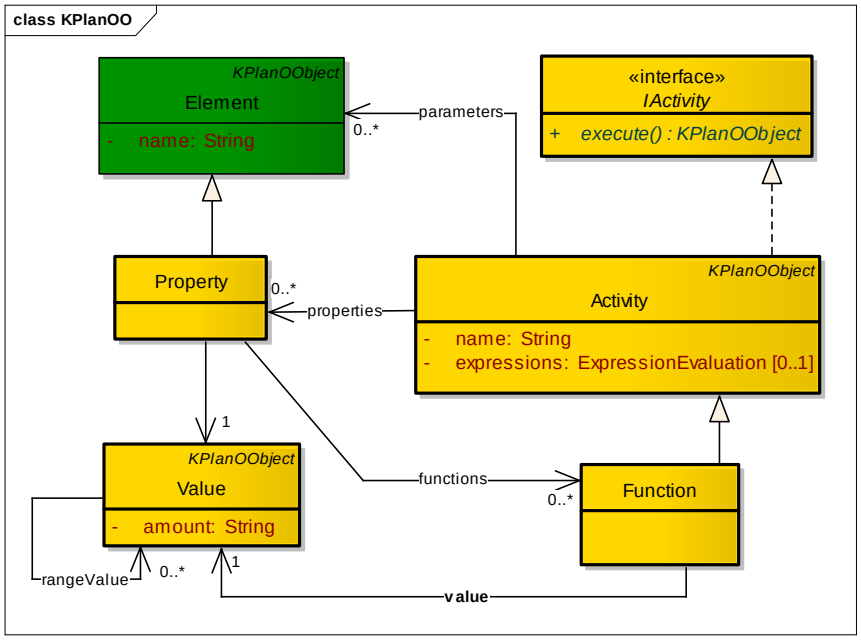

Figura 4.10 - Destaque da interface IActivity e da classe Activity

A Figura 4.11 apresenta o diagrama de objetos com funções estabelecidas para o domínio Logística:

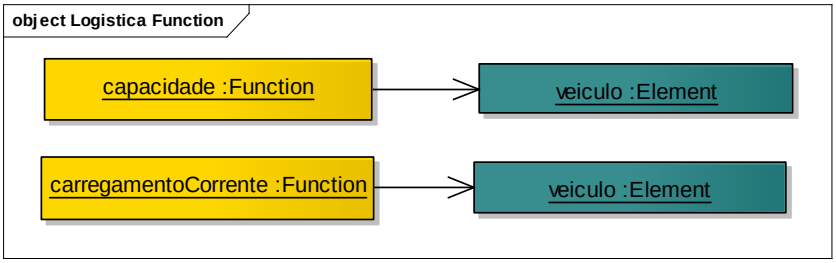

Figura 4.11 - Diagrama de objetos da classe Function do domínio de logística

Action na Figura 4.12 é a abstração das ações do domínio, que contempla a especificação de “SuperAções” através de uma auto relacionamento. Além da lista de elementos herdados que representa seus parâmetros, define-se uma outra lista para variáveis de cada ação específica. Cada ação possui um cenário do domínio como efeito e um cenário como pré-condição que são definidos através de relações com a entidade *State*.

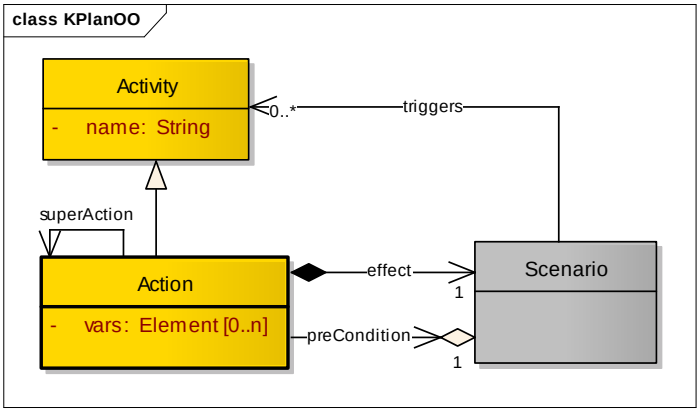

Figura 4.12 - Destaque da classe Action

A Figura 4.13 apresenta o diagrama de objetos com a ação “dirigir” do domínio de logística com seus respectivos parâmetros:

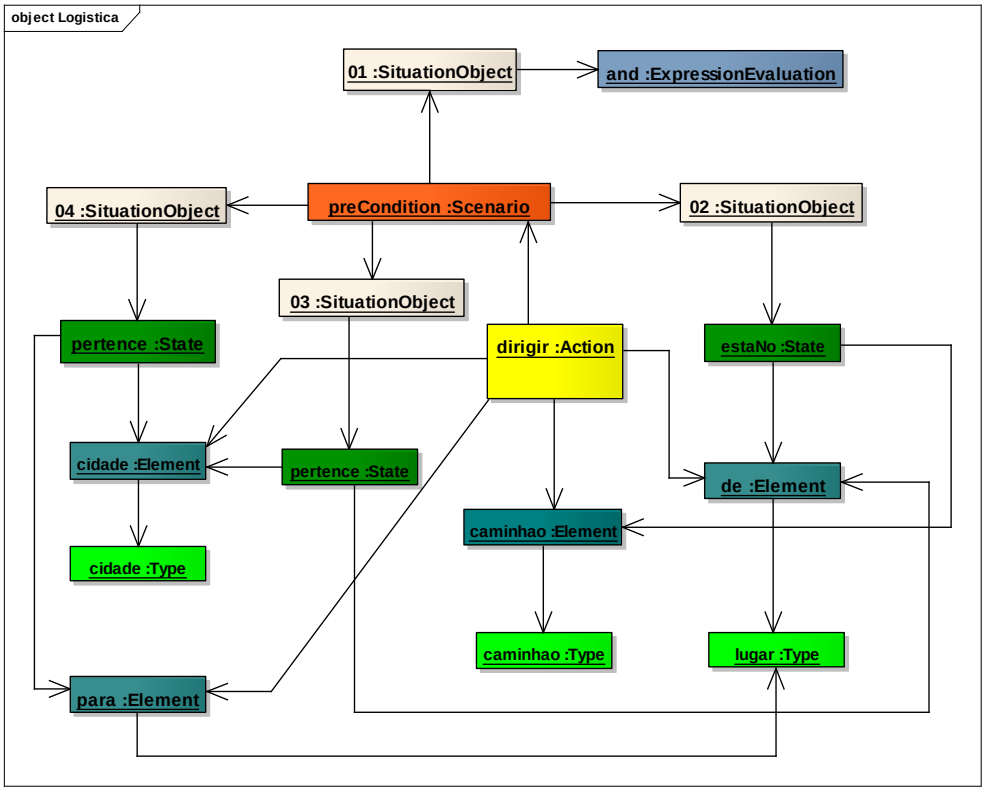

Figura 4.13 - Diagrama de objetos da ação “dirigir” do domínio de logística com seus parâmetros e sua pré-condição.

A Figura 4.14 apresenta o diagrama de objetos com a ação “dirigir” do domínio de Logística, com seus parâmetros e seu efeito:

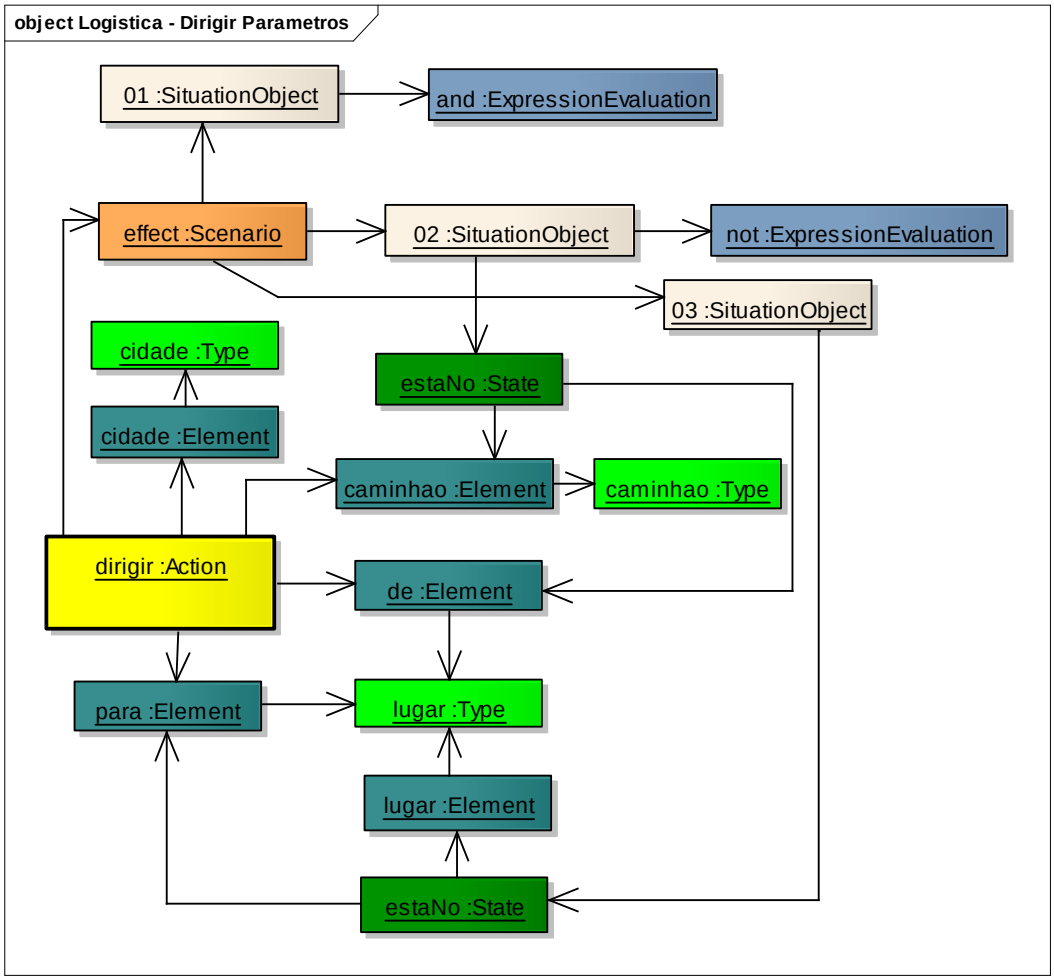

Figura 4.14 - Diagrama de objetos da ação “dirigir” do domínio de logística com seus parâmetros e seu efeito.

Domain na Figura 4.15 representa a descrição do domínio de planejamento, composto por uma lista de ações, possui uma lista de estados prováveis para objetos de problemas com tipos definidos. É também possível definir um conjunto de funções, e o domínio pode ter várias instâncias de problema.

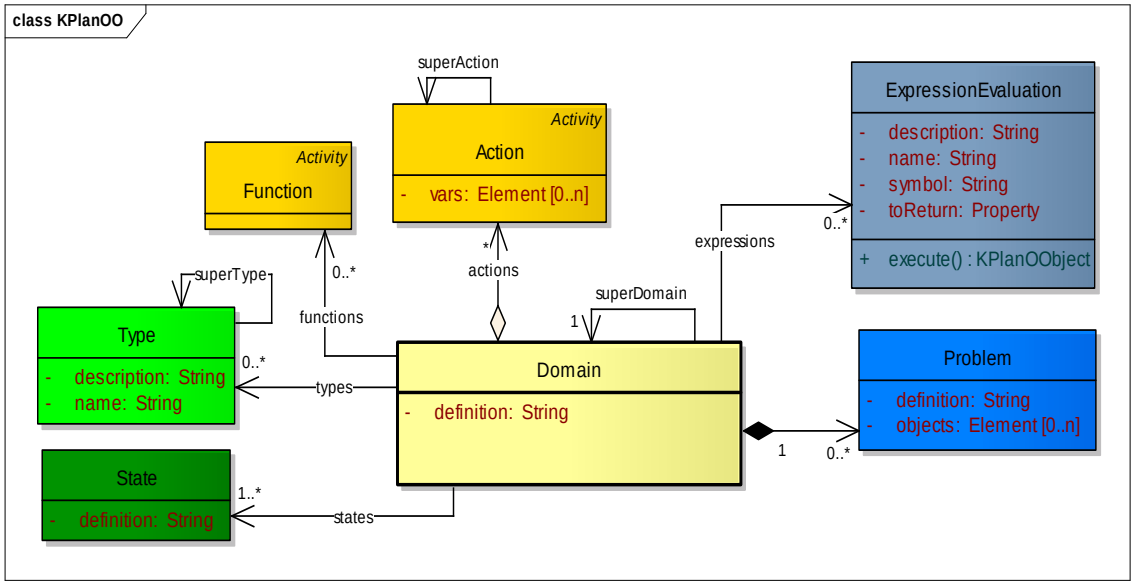

Figura 4.15 - Destaque classe Domain

As associações, multiplicidades, atributos, restrições e classes impostas pelo *KPlanOO* devem ser seguidas e obedecidas durante a definição dos domínios e problemas de planejamento de modo a construir estados válidos em relação à estrutura estática definida.

Após a modelagem do domínio de planejamento é possível realizar a modelagem de problemas. Conforme visto anteriormente, um problema de planejamento é referente a um modelo de domínio de planejamento, geralmente caracterizado por uma situação onde dois estados são conhecidos: o estado inicial (*Init*) e o estado objetivo (*Goal*).

Um possível problema de planejamento do domínio de logística poderia ser explicado da seguinte forma:

- **Estado Inicial:** O ***caminhão1***, com capacidade para ***4 pacotes*** e com nenhum carregamento, encontra no ***localSP1*** da cidade ***saopaulo***. Os pacotes ***pkg1*** e ***pkg2*** se encontram no ***localSP2*** e o avião1 (com capacidade para 10 pacotes e com nenhum carregamento) se encontra no ***aeroportoSP1*** também da cidade ***saopaulo***. Já o

caminhão2, com capacidade para ***2 pacotes*** e com nenhum carregamento, se encontra no ***localRJ1*** da cidade ***riodejaneiro***. O ***pkg3*** se encontra no ***localRJ2*** e o pacote ***pkg4*** e ***avião2*** (com capacidade para ***10 pacotes*** e com nenhum carregamento) se encontram no ***aeroportoRJ*** também da cidade ***riodejaneiro***.

- **Estado Objetivo:** no estado desejado os pacotes ***pkg1*** e ***pkg2*** devem ser entregues no ***localRJ1*** da cidade ***riodejaneiro*** e os pacotes ***pkg3*** e ***pkg4*** devem ser entregues no ***localSP1*** da cidade ***saopaulo***.

A Figura 4.16 e a Figura 4.17 ilustram melhor os estados apresentados a cima.

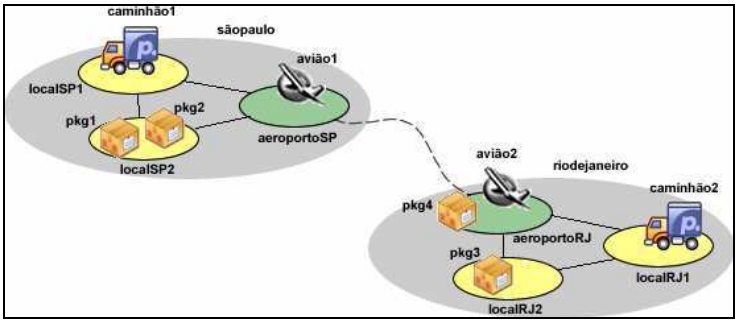

Figura 4.16 - Ilustração do estado inicial para um problema de planejamento no domínio logística. Fonte: Vaquero (2007, p.119).

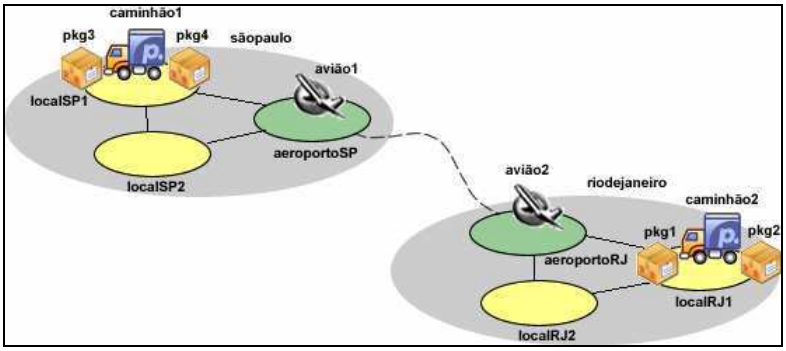

Figura 4.17 - Ilustração do estado objetivo de um problema de planejamento no domínio logística. Fonte: Vaquero (2007, p.119).

A baixo são apresentadas as entidades do KPlanOO que contemplam a modelagem de problemas de planejamento.

Init na Figura 4.18 representa o estado inicial de um problema possui uma lista de *Function's* que contempla funções a serem executadas no inicio do processo de planejamento. *Init* é uma especialização da classe *Scenario*.

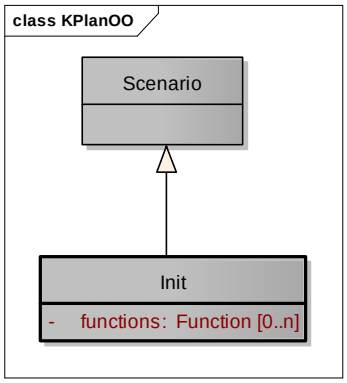

Figura 4.18 - Destaque classe Init

Na Figura 4.19 e Figura 4.20 é apresentado através do diagrama de objetos o estado inicial descrito acima para o domínio Logística, modelado com o *KPlanOO*, sendo que na Figura 4.19 é apresentado o conjunto de estados dos objetos envolvidos no problema e na Figura 4.20 as funções a serem executadas no início do processo de planejamento.

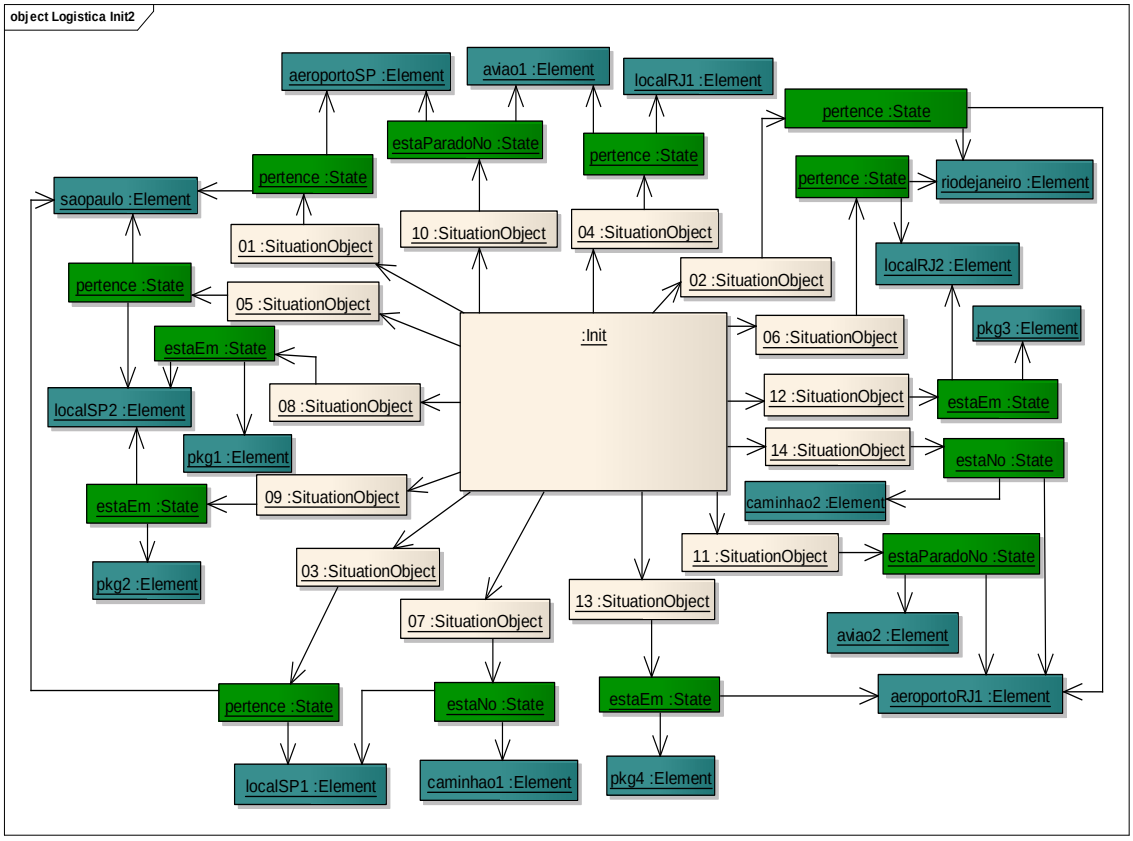

Figura 4.19 - Diagrama de objetos dos estados dos objetos para o estado Inicial de um problema de planejamento do domínio de logística, descrito anteriormente

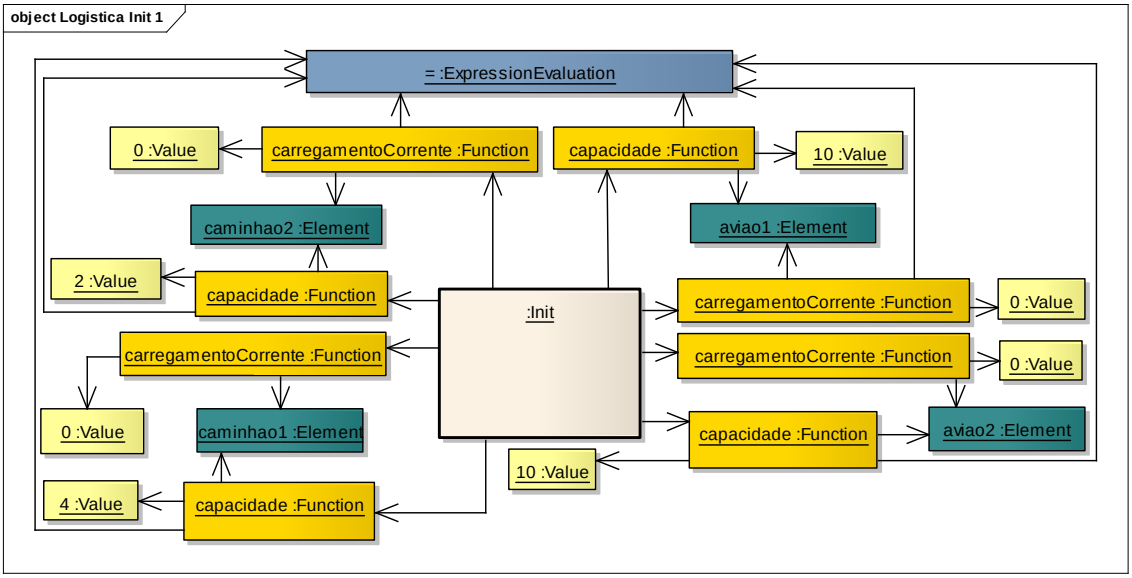

Figura 4.20 - Diagrama de objetos de funções a serem executadas para o estado Inicial de um problema de planejamento do domínio de logística, descrito anteriormente

Goal na Figura 4.21 é uma abstração responsável por representar a meta a ser atingida por um planejador, através de estados para os objetos declarados no problema. Possui um atributo *predominance* que estabelece prioridade da meta em relação às outras definidas para o mesmo problema.

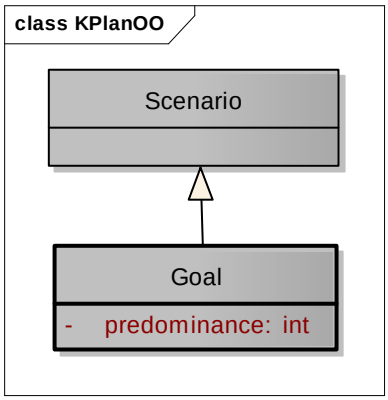

Figura 4.21 - Destaque classe Goal

Na Figura 4.22 é apresentado através do diagrama de objetos o estado objetivo descrito anteriormente para o domínio de logística, modelado com o *KPlanOO*.

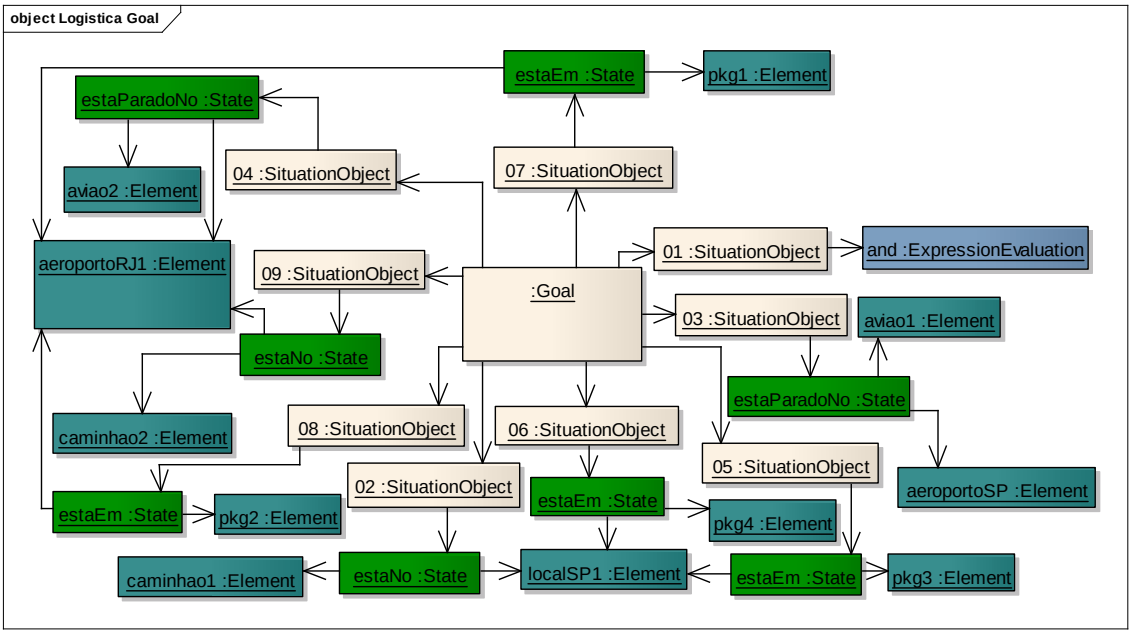

Figura 4.22 - Diagrama de objetos do estado objetivo de um problema de planejamento do domínio de logística, descrito anteriormente

Problem na Figura 4.23 é a entidade com a finalidade de representar a descrição dos problemas para um dado domínio.

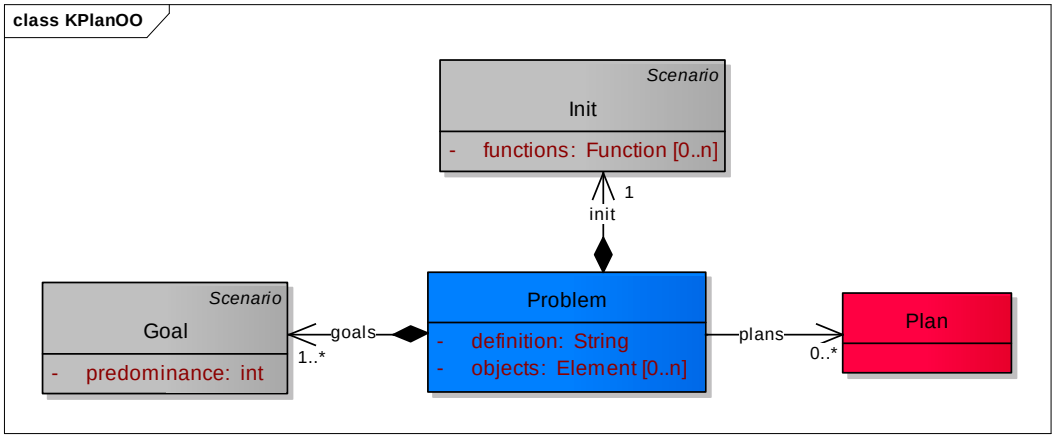

Figura 4.23 - Destaque classe Goal

Na Figura 4.24 é apresentado, através do diagrama de objetos o a estrutura do problema descrito anteriormente para o domínio Logística, com seus objetos e as respectivas referências para as instâncias de *Init* e *Goal*, definidas anteriormente.

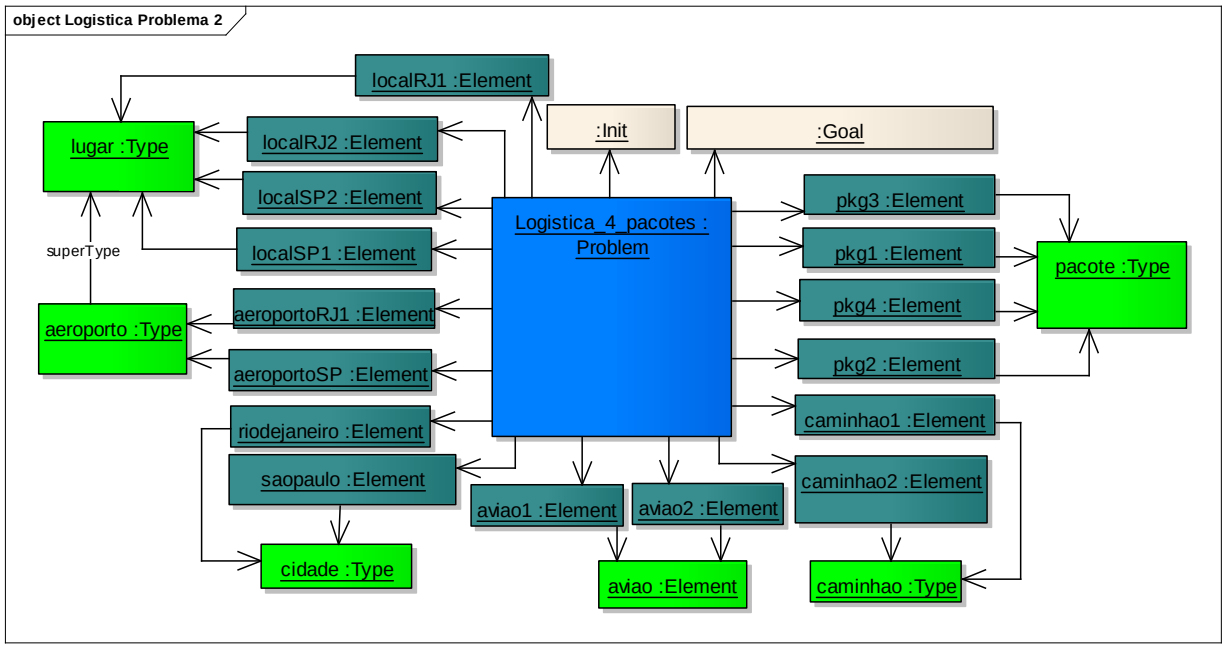

Figura 4.24 - Diagrama de objetos da modelagem um problema de planejamento do domínio logística, descrito anteriormente.

IPlan na Figura 4.25 é a interface para os planos gerados.

Plan (Figura 4.25) é a entidade que representa os planos gerados.

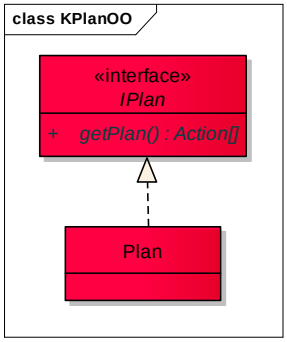

Figura 4.25 - Destaque classe Domain

Visando possibilitar o uso de padrões de projeto - baseados no mecanismo de herança como: *Bridge*, *Chain of Responsibility*, *Composite*, *Decorator*, *Observer* ou *Strategy* (GAMMA et. al.,1995) - na construção de planejadores OO que utilizem o *KPlanOO* como ferramenta para modelagem de domínios de planejamento, todas as classes conceituais do modelo são sub-classes direta ou indiretamente de uma classe abstrata do modelo nomeada como *KPlanOObject*, esta classe possui apenas um atributo "*id*". Na Figura 4.26 é possível ver a estrutura de herança existente no modelo.

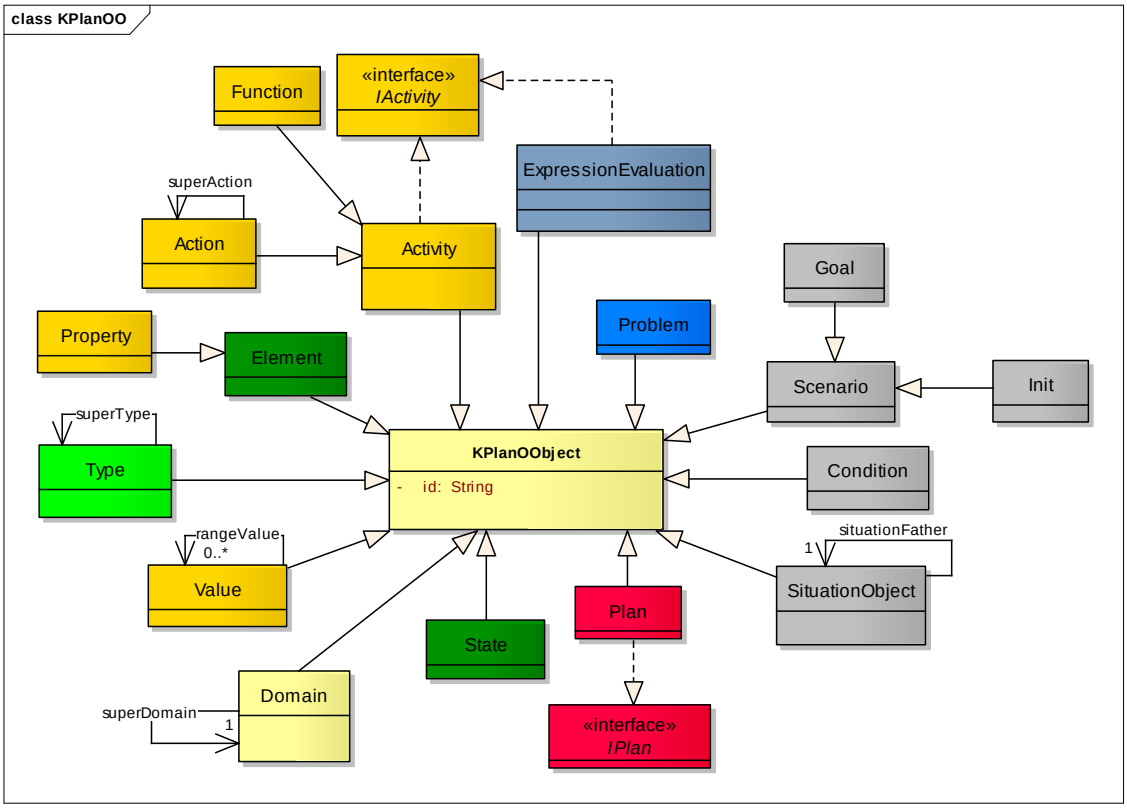

Figura 4.26 - Estrutura de herança presente no KplanOO

Características como a apresentada pela interface *IActivity*, que segue a especificação do padrão de projeto "*command*", tornam o *KPlanOO* altamente extensível e possibilita seu reuso. O padrão *command* encapsula uma requisição como objeto, para que os clientes parametrizem diferentes requisições e produzam operações reversíveis, assim cada implementação da interface *IActivity* pode ser manipulada da maneira mais conveniente para cada processo de planejamento.

Na Tabela 4.1 apresenta-se a Tabela 3.1 do capítulo 3, alterada incluíndo o *KPlanOO* onde são observadas as características das linguagens e da proposta desse trabalho.

Tabela 4.1 - Comparação das características das linguagens de descrição de domínios de planejamento com KPlanOO

Características	STRIPS	ADL	PDDL	AML	OCL	KPlanOO
Tipos Definidos		X	X	X		X
Definição de Estados	X	X	X	X	X	X
Ações	X	X	X	X	X	X
Pré-condições	X	X	X	X	X	X
Efeitos	X	X	X		X	X
Efeitos Condicionais		X	X			X
Recursos				X		X
TEMPO				X		X
Literais Abertos		X	X		X	X
Objetivos Predominantes						X
Prioridade de Eventos (Exógenos)						X
SubAções			X			X
Portabilidade de Plataforma						X
Consistência entre Domínio e Problema					X	X
Baseada em Objetos					X	X
Objetos Dinâmicos					X	X
SubEstados					X	X
Definição de Operadores(Expressões de Avaliação)					X	X
Estruturas de seleção e loops(Expressões de Avaliação)						X
Subdomínios						X
Interface para planos						X
Abstração de Entidades						X
Sintaxe textual	X	X	X	X		
Intervalo de valores						X
Definição de Atividades Condições						X

4.3. Considerações

Inicialmente neste capítulo foram apresentadas algumas características dos Domínios da área de Planejamento automático. Após isso, foi apresentado o meta-modelo proposto, nomeado *KPlanOO*, que tem por objetivo ser uma ferramenta de alta flexibilidade, com baixo acoplamento, alta coesão e grande reutilização de componentes; para ser usado na modelagem de domínios de sistemas de planejamento automático.

Foram apresentadas as abstrações identificadas para a Descrição de Domínios e Problemas de Planejamento presentes no Meta-Modelo. Apresentou-se a fundamentação do meta-modelo e sua estrutura estática. Simultaneamente a apresentação da estrutura estática do *KPlanOO,* foi apresentado a modelagem de um domínio de logística utilizando o meta-modelo proposto.

Outra importante consideração desse capítulo é a idéia de que o meta-modelo proposto pode ser utilizado tanto para a modelagem de domínios da área espacial como de outras áreas, visto que na construção do mesmo foram consideradas abordagens da técnica de planejamento automático.

No próximo capítulo será apresentado todo o processo de construção do protótipo para o uso do meta-modelo aqui definido e uma especificação XML *Schema Definition* (XSD) que contempla os conceitos de modelagem, verificação e restrições do mesmo, que servem de meios descrição de domínios de planejamento utilizando a proposta deste trabalho. Além disso, serão apresentadas algumas ferramentas estudadas e uma visão das funcionalidades do protótipo.

5 IMPLEMENTAÇÃO DE INTERFACES DE ENTRADA E VALIDAÇÃO DE DADOS

No capítulo anterior foram discutidos os conceitos referentes ao Meta-Modelo proposto, que tem como foco o processo de modelagem de domínios de planejamento.

Após apresentado esse processo e estudadas as ferramentas disponíveis com objetivos similares, apresentam-se as ferramentas utilizadas para a construção de um protótipo que permite a criação e a validação de modelos com base no *KPlanOO* e fornecer suporte, principalmente, à modelagem de domínios de planejamento, tanto acadêmicos quanto reais, para os desenvolvedores.

Neste capítulo também é detalhado o desenvolvimento do protótipo, sua especialização para a área espacial e quais restrições de integridade já estão cobertas no mesmo. Também é apresentado as funcionalidades do protótipo por meio da visualização de suas interfaces.

Por fim, é apresentada a definição de todo o meta-modelo em XSD, o que possibilita que domínios e problemas de planejamento sejam modelados com ferramentas específicas para se trabalhar com XML (*eXtended Markup Language*), classes em linguagens de programação ou diversos outros aplicativos que incluem hoje um amplo suporte ao padrão XSD.

Tanto o protótipo quanto a definição XSD, tem por finalidade possibilitar a modelagem de domínios e problemas de planejamento com o *KPlanOO*, facilitando a entrada de dados de modelos de domínios e problemas de planejamento, verificar a consistência desses modelos com o meta-modelo e possibilitar que sejam acoplados geradores de códigos. A Figura 5.1 mostra uma visão em alto nível as possibilidades de entrada e saída para o KPlanOO.

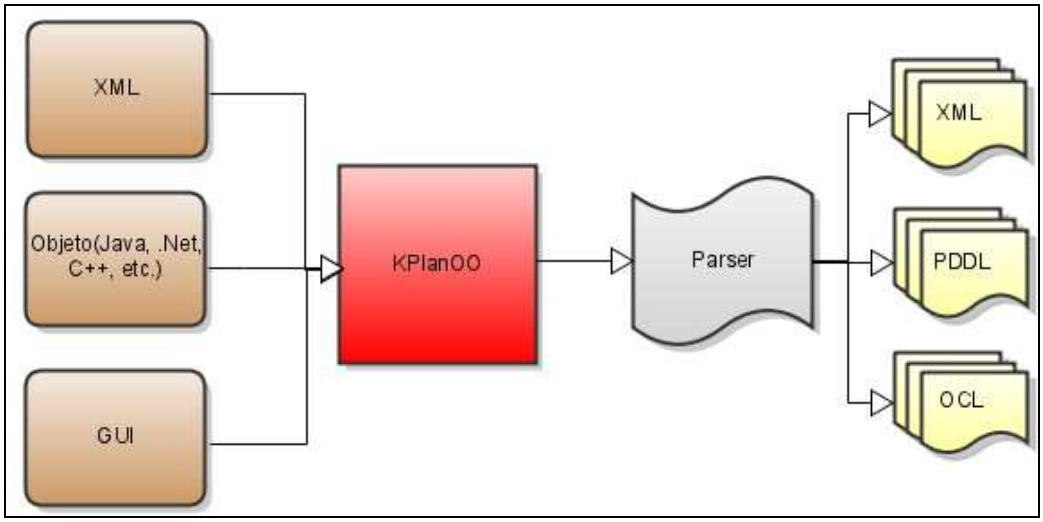

Figura 5.1 – Possibilidades de entradas e saídas para o KPlanOO.

Observa-se na Figura 5.1, que idéia é que independete de como o os dados sejam capturados, pode-se haver geradores de código (parser) para diferentes linguagens, ou formas de representação, de domínios e problemas de planejamento.

5.1. Ferramentas utilizadas

Para o desenvolvimento do protótipo foram utilizadas basicamente três ferramentas: Plataforma Java 6, onde utilizou-se a linguagem de programação e também a API Swing para a construção da interface gráfica; a biblioteca de componentes Swing SwingX (*Swing eXtension*); e o XStream, como ferramenta para serializar objetos para xml.

Nesta seção, será explicada a escolha da API Swing e da biblioteca SwingX e é apresentada uma visão geral da ferramenta XStream.

5.1.1. Desenvolvimento da interface gráfica

O protótipo foi desenvolvido utilizando a API Swing da plataforma Java. Esta API dá suporte à criação de componentes GUI (*Grafical User Interface*) como botões, menus, tabelas, e vários outros, que possibilita a criação de uma

interface mais amigável e de fácil controle. Optou-se pelo desenvolvimento em Swing por sua flexibilidade de desenvolvimento de novos componentes, o que possibilita o reuso e compartilhamento destes componentes em outros sistemas que venham ser desenvolvidos pelo CRC.

Outro fator que levou a escolha do Swing como ferramenta para a construção da interface gráfica foi por ela ser escrita e baseada em java, possibilitando assim uma integração fácil com outras ferramentas e frameworks.

Utilizou-se os componentes *jxTaskPaneContainer* e *jxTaskPane* da biblioteca SwingX afim de conseguir uma boa organização nos menus.

O SwingX é um subprojeto da SwingLabs (SWINGX, 2009) que possui uma série de outros projetos e funciona como uma comunidade de desenvolvedores de aplicativos desktop e um repositório de futuras novidades no núcleo do Swing. A SwingLabs conta com o apoio da *Sun Microsystems* e os projetos contam com a participação de engenheiros da Sun, que trabalham no projeto Swing. Além disso os projetos estão sob licença LGPL(*Lesser General Public License*) e alguns em BSD(*Berkeley Software Distribution*), e a comunidade ainda conta com um local para *brainstorming* que ajudam na formulação e incremento de novas funcionalidades e novos componentes para o projeto.

5.1.2. Serializar objetos para XML

Para auxiliar a serialização de objetos java, instanciados por meio do protótipo desenvolvido, para xml foi utilizada a biblioteca XStream (XSTREAM, 2009). Esta é uma biblioteca de código aberto para serialização de objetos Java para xml, facilitando desta forma a persistência dos dados em xml, além disso também facilita o processamento do xml. Essa tarefa pode ser executada manipulando o objeto carregado a partir do xml, isso acontece porque o próprio XStream se encarrega de fazer o processo inverso da serialização do xml e

carregar o objeto, retirando do programador a parte chata do processamento do xml, dando mais produtividade, diminuindo substancialmente a geração de erros de programação e ainda por cima apresentando um ótimo desempenho.

A baixo é apresentado as principais características do XStream:

- Facilidade de uso. Geralmente só se precisa conhecer uma classe.
- Não é necessário configurar o mapeamento manualmente. Pois a biblioteca se baseia no mecanismo de Reflexão pelo qual é possível descobrir dinamicamente quais os atributos de uma classe Java e os respectivos valores de seus objetos.
- XML legível e mais compacto que a serialização nativa de Java.
- Não exige que os objetos sigam alguma regra em particular.
- Permite referências duplicadas e circulares dos objetos.
- Pode ser integrada com outras APIs.
- A estratégia de conversão pode ser configurada.
- Realiza uma avaliação prévia se o documento XML não está mal-formado, exibindo eventuais mensagens de erros.

5.1.3. XSD (XML Schema Definition)

O padrão XSD (XML *Schema Definition*) ou XS (XML *Schema*) é a recomendação oficial do W3C desde 2001 para validação de documentos XML. Esse padrão consegue suprir as limitações da DTD (*Document Type*

Definition), além de fornecer diversas funcionalidades. Através do XSD é possível construir tipos próprios derivados de tipos mais básicos, realizar relacionamentos entre elementos de dados dentro do XML (de forma similar aos relacionamentos entre tabelas), etc.

Um documento XSD é em sua essência um documento XML. Isso quer dizer que ele deve obedecer às mesmas regras que um documento XML exige (em especial as regras relacionadas à sua boa formação). Um documento XSD também possui outras necessidades que um documento XML não necessita. Para que essas necessidades sejam atendidas é preciso definir as partes de um documento XSD:

- **Namespace** - um container para os nomes utilizados, funciona como um agrupador e geralmente tem seu nome associado a um nome único (comumente uma URL).

- **Declaração de elementos** - Os elementos são declarados utilizando-se a *tag* “element”. Essa *tag* contém alguns atributos que podem ser utilizados. Os principais são:

    - *name*: especifica o nome do elemento;

    - *type*: especifica o tipo de dados do elemento;

    - *minOccurs*: especifica a quantidade mínima de vezes que o elemento pode aparecer;

    - *maxOccurs*: especifica a quantidade máxima de vezes que o elemento pode aparecer (A palavra *unbounded* pode ser utilizada para especificar uma quantidade ilimitada).

- **Declaração de atributos** - De uma forma geral as declarações de atributos se parecem muito com as declarações de elementos. Essas declarações possuem alguns atributos. Os principais são:

  - *name*: especifica o nome do atributo;

  - *type*: especifica o tipo de dados do atributo;

  - *use*: especifica a utilização do atributo (requerido, opcional ou proibido);

- **Tipos de dados** - O XML Schema possui diversos tipos de dados (além da possibilidade de criar tipos próprios). Os mais comuns são:

  - *xsd:string* – string de caractéres de comprimento ilimitado;

  - *xsd:boolean* – valor booleano (true, false, 1 ou 0);

  - *xsd:decimal* – número decimal;

  - *xsd:float* – ponto flutuante;

  - *xsd:date* – Uma data no calendário gregoriano;

  - *xsd:dateTime* – Um instante específico no calendário gregoriano;

  - *xsd:integer* – Um número inteiro.

- **Grupos de modelos** - Os grupos de modelos são construções que permitem que elementos sejam especificados dentro de outros elementos e, se desejado, obedeçam a uma ordem ou escolha específica através de conectores. Os três conectores permitidos são *sequence, all* e *choice*.

O mecanismo de funcionamento do XML *Schema* (XSD), possibilita uma validação da estrutura do documento XML em dois níveis. O primeiro para aplicações *on-line* que utilizam o *parser* (conjunto de regras de validação) para otimizarem os processos das APIs (*Application Programming Interface*) que integram o programa que está sendo utilizado. O segundo nível realiza uma adequação das rotinas a partir do modelo de informações do XML para o *parser* e o XML *Schema*.

Neste trabalho foi desenvolvido um XML *Schema*, o XKPS, para descrição de domínios e problemas de planejamento automático. O XKPS é baseado no *KPlanOO*, e permite usar ferramentas genéricas para editar, validar e processar documentos XML. Além disso, um domínio ou um problema de planejamento expresso de acordo com o XKPS é obrigatoriamente anotado corretamente, contribuindo para uma confiabilidade maior da descrição.

5.2. O Protótipo desenvolvido

Nesta seção será apresentada a estrutura geral do protótipo, não foi escolhida nenhuma linguagem diagramática para ser utilizada ou avaliada, pois o foco do mesmo foi no processo de consistência de modelos e geração de código. O protótipo é dividido nos seguintes pacotes principais: *modelo*, *dao, controller, view*, e *gui*. Na apresentação de cada pacote, serão suprimidos atributos e métodos secundários para facilitar a leitura. A Figura 5.2 apresenta o Diagrama de Pacotes do protótipo desenvolvido e as próximas seções detalham cada pacote assim como as classes e arquivos que compõem cada um desses.

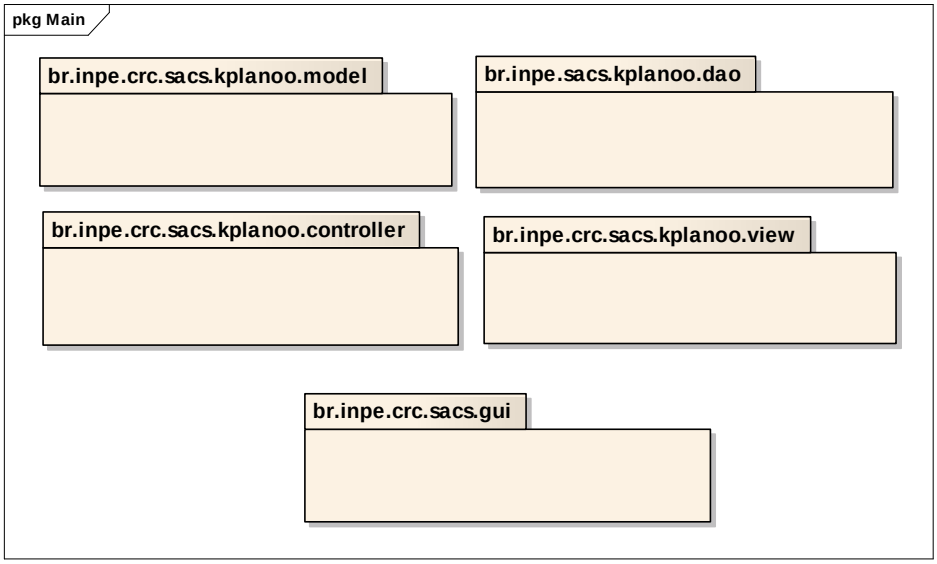

Figura 5.2 - Diagrama de pacotes de protótipo

5.2.1. Pacote br.inpe.crc.sacs.kplanoo.model

Neste pacote são armazenadas as classes que representam as entidades do *KPlanOO*, apresentadas na Figura 4.2. Nestas classes são feitos mapeamentos, utilizando o recurso de anotações do XStream, para estrutura XML.

5.2.2. Pacote br.inpe.crc.sacs.kplanoo.dao

Neste pacote são armazenadas as classes responsáveis pela persistência dos dados capturados através do protótipo. Estas classes são implementações do padrão *Data Access Objects* (DAO) que tem por finalidade isolar a camada de persistência, facilitando assim a manutenção e o reuso. Visando possibilitar implementações para que diversos mecanismos de persistência possam ser desenvolvidos, foi criada a interface *IDAOKPlanOO* que estabelece a assinatura dos métodos de persistência com um parâmetro do tipo *KPlanOObject*.

Três implementações foram realizadas, uma para XML, outra para PDDL e outra para armazenamento em um banco de dados relacional utilizando a API de manipulação de dados JDBC. O Diagrama de classes UML desse pacote é apresentado pela Figura 5.3.

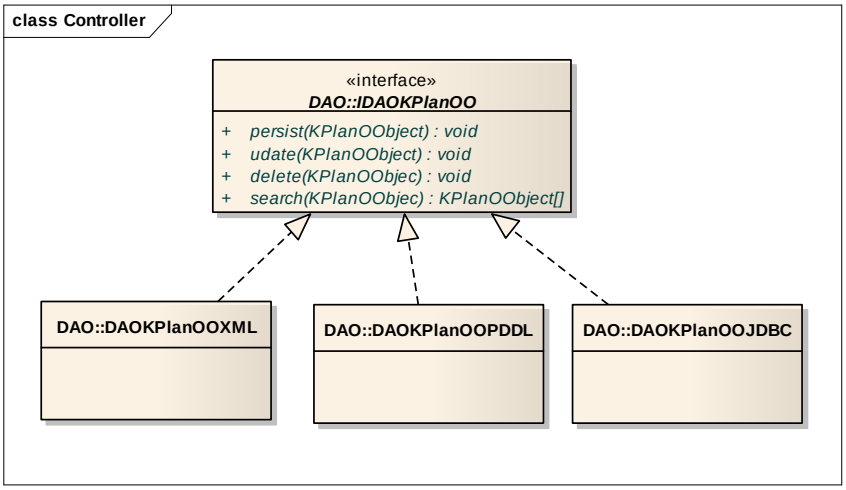

Figura 5.3 - Diagrama de classes do pacote br.inpe.crc.sacs,kplanoo.dao

5.2.3. Pacote br.inpe.crc.sacs.kplanoo.controller

Este pacote contém as classes que determinam o fluxo da apresentação e também que disponibilizam serviços e funcionalidades da camada lógica. Tais classes seguem o padrão *Facade* que tem por finalidade fornecer uma interface unificada para um conjunto de interfaces do sistema, e o padrão *Command* que encapsula uma requisição como objeto possibilitando assim a parametrização de diferentes requisições. O padrão *Facade* é implementado pela interface de programação *IFacadeKPlanOO* e o padrão *Command* pela interface de programação *IValidator*.

Temos a classe *FacadeKPlanOO* que por ser uma implementação de *IFacadeKPlanOO* implementa os quatro métodos de manipulação de objetos do tipo *KPlanOObject* estabelecidos por esta. Os métodos desta interface delegarão as respectivas operações de persistência do objeto do tipo *KPlanOObject*, enviado como parâmetro, a um DAO específico. O parâmetro

do tipo inteiro enviador, irá representar a forma de persistência (0 – todos, 1 – XML, 2 – PDDL, 3 – JDBC, etc...), este DAO será uma instância contida em uma coleção.

Porém antes da execução do método do DAO, são executados possíveis validadores que são implementações da interface *IValidator*. Inicialmente foram implementados dois validadores, um que valida a estrutura estática das instâncias de um objeto que representa um domínio conforme as regras estabelecidas no *KPlanOO*, o outro valida a instância de um problema baseados também nas regras do *KPlanOO*. A Figura 5.4 apresenta o Diagrama de Classes UML desse pacote.

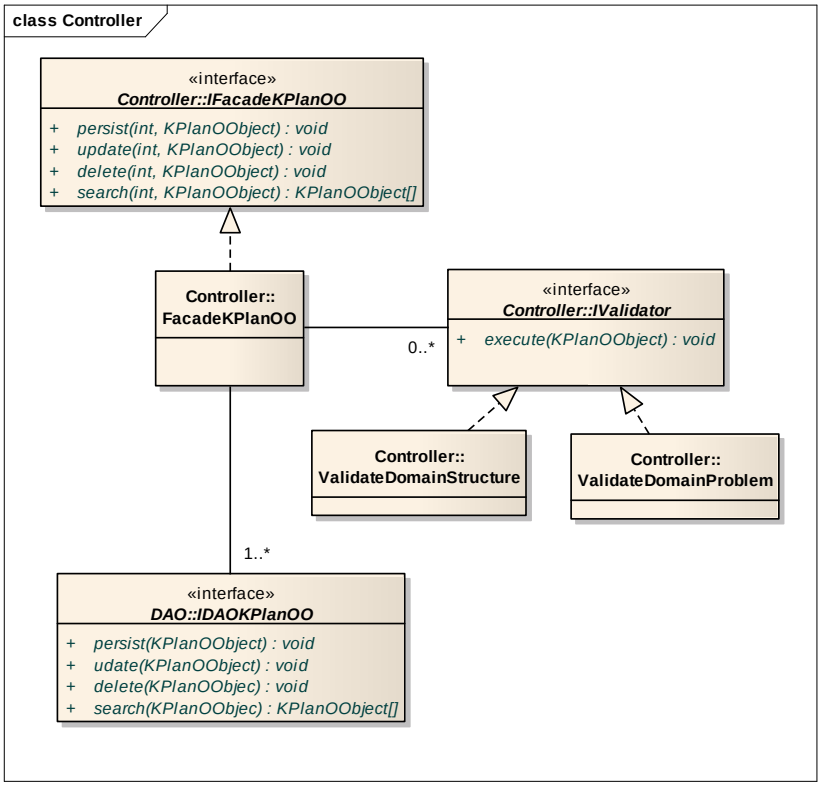

Figura 5.4 - Diagrama de classes do pacote br.inpe.crc.sacs,kplanoo.controller

A abordagem de implementação utilizando os padrões descritos aqui, só é possível graças a abstração *KPlanOObject* presente no *KPlanOO*. Com esta

solução é possível que novas aplicações que utilizem o *KPlanOO* sejam desenvolvidas utilizando esta abordagem de controle de fluxo e validação de informações. Isso é bastante positivo pois; além de estabelecer um padrão de desenvolvimento; possibilita uma maior facilidade na manutenção dos sistemas, uma vez que a solução está baseada em padrões amplamente utilizado no desenvolvimento de software.

5.2.4. Pacote br.inpe.crc.sacs.kplanoo.gui

Para a construção do protótipo foi criado uma coleção de componentes gráficos especializados, com o objetivo de facilitar a construção de telas e o reuso dos mesmos. Estes componentes são especializações de componentes da API Swing do Java; tendo sido criadas especializações para os seguintes tipos de componentes: campos de texto curtos e longos, lista de seleção, tabelas e lista de opções simples e múltiplas. Estes componentes foram especializados basicamente para que já tivessem um rótulo, métodos de acesso a conteúdos e inserção de eventos.

5.2.5. Pacote br.inpe.crc.sacs.kplanoo.view

Neste pacote são armazenadas todas as interfaces gráficas do protótipo. O pacote é composto pelas classes *FormType*, *FormState*, *FormExpression*, *FormAction*, *FormFunction*, *FormDomain* e *FormProblem* que estendem de *CDAAIPSForm* que por sua vez implementa a interface *IFormCDAAIPS*. A Figura 5.5 apresenta a estrutura deste pacote.

Na classe *CDAAIPSForm* tem-se a implementação do padrão *Observer* que através do método *updateForm* atualiza todos formulários contidos no mapa *formsCDAAIPS* invocando o método *update*, com informações representadas pelo formulário em questão. Um exemplo seria: ao adicionar um novo tipo no formulário *FormType* todos os formulários que de alguma forma necessitam

apresentar os tipos definidos recebem a atualização e assim através de sua implementação de *update* apresentam as informações atualizadas.

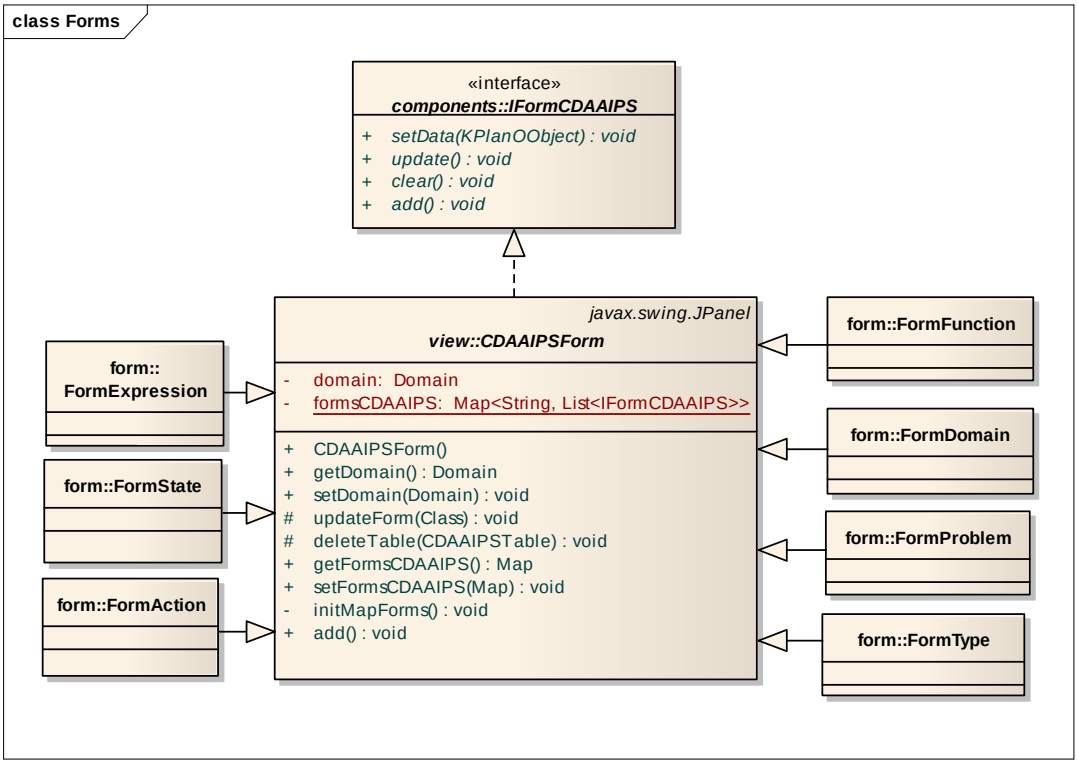

Figura 5.5 - Diagrama de classes do pacote br.inpe.crc.sacs,kplanoo.view

5.2.6. Especialização do protótipo

Uma importante característica do protótipo desenvolvido é a possibilidade de especialização do mesmo, de forma que possa ser usado para gerar código em outras formas de representação existentes. Para isso, os seguintes passos são necessários:

- Mapeamento dos conceitos e relacionamentos do meta-modelo para a forma de representação escolhida.

- Criação de uma classe que implemente a interface IDAOKPlanOO e implemente os métodos descritos nesta.

- Definição desta nova classe na coleção contida na classe FacadeKPlanOO, está definição pode ser feita via código ou através de um arquivo de configuração o que flexibilizaria ainda mais a inserção de novas especializações.

5.2.7. Inclusão de restrições de integridade no protótipo

O protótipo desenvolvido garante a consistência de um modelo de domínio ou problema através de sua construção. Porém esta consistência é feita baseada nas regras estabelecidas pelo *KPlanOO*, caso haja a necessidade da inserção de novas consistências ou até a inserção de regras de negócio, basta seguir os seguintes passo:

- Criar uma classe que implemente a interface *IValidator* e implemente o método execute.

- Incluir esta nova classe na coleção contida na classe *FacadeKPlanOO*.

5.2.8. Uso do protótipo

Apresenta-se nesta seção o detalhamento da estrutura geral do protótipo e serão mostradas as suas funcionalidades específicas por meio da visualização das interfaces que o compõem.

A tela inicial do protótipo é composta por duas abas de opções, são elas: *Planner* e *Persist*, além do botão *Exit*. A Figura 5.6 apresenta a tela principal do protótipo.

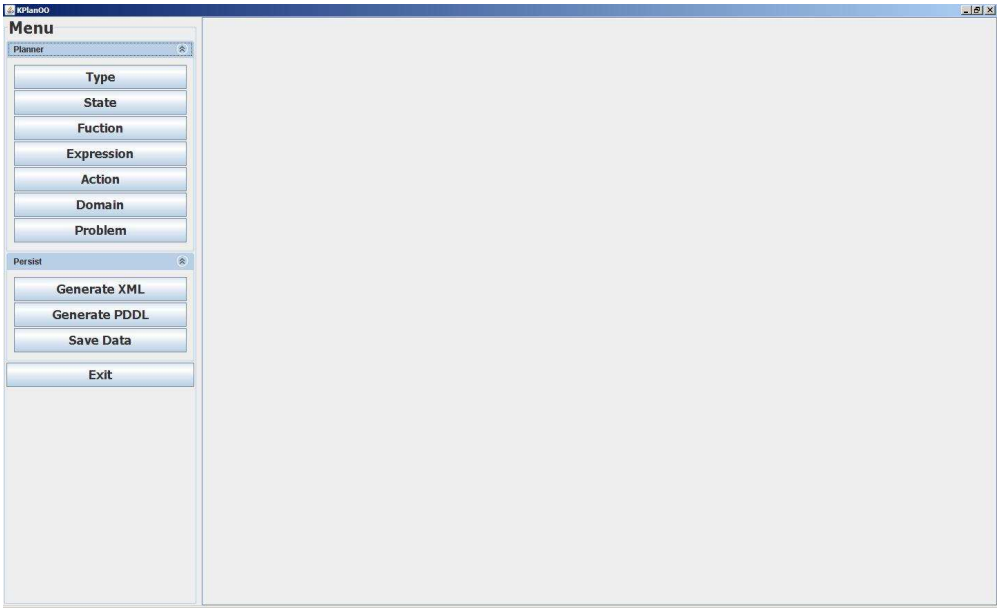

Figura 5.6 - Tela principal do protótipo

A aba *Planner* é composta por sete opções, são elas: *Type*, *State*, *Function*, *Expression*, *Action*, *Domain* e *Problem*. Cada opção possibilita criar, alterar, consultar e excluir definições para os respectivos conceitos representados no *KPlanOO*.

Na aba *Persist* são disponibilizadas opções de geração de código caso a especificação esteja com o meta-modelo e suas restrições, em XML e PDDL através das opções *Generate* XML e *Generate* PDDL respectivamente. Também é disponibilizada através da opção Save Data a possibilidade de salvar em uma base local ou remota conforme a implementação utilizada.

5.2.9. Utilizando as funcionalidades do protótipo

Para se definir um novo Domínio e um Problema de planejamento utilizando o protótipo é necessário primeiro definir todas as suas propriedades e dependências. Tudo que é necessário ser definido está disponibilizado nas opções da aba *Planner*. Ao optar por uma destas opções; o protótipo disponibiliza algumas funcionalidades específicas para o tratamento dos

conceitos em questão. Essas funcionalidades são a criação, consulta alteração e exclusão. As próximas seções detalham cada uma dessas funcionalidades.

5.2.9.1. Criação de Types

A tela inicial do protótipo permite a criação de tipos para um Domínio acionando o botão *Type* da aba *Planner*. Caso esse seja acionado, será apresentada a tela onde poderá ser especificado um nome e uma descrição para o tipo a ser criado, conforme a Figura 5.7. Após a especificação do tipo e o acionamento do botão *Add* este tipo será incluído em uma tabela na mesma tela, que contém todos os tipos já especificados.

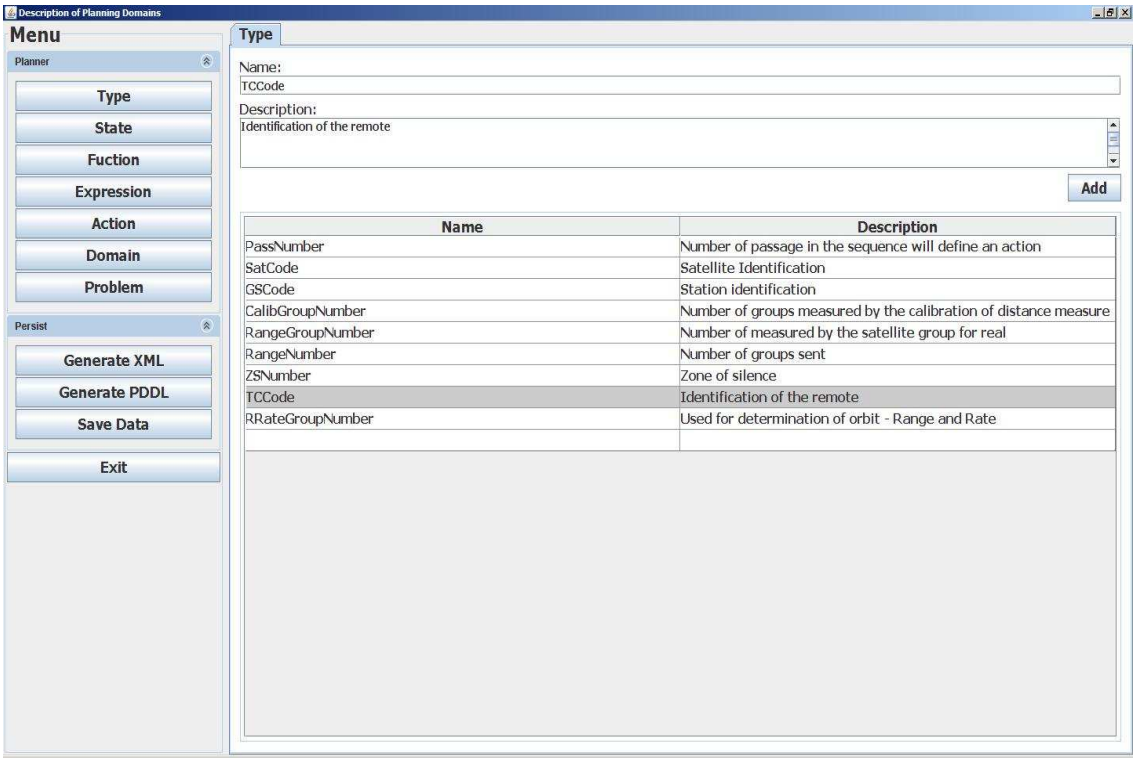

Figura 5.7 - Tela de especificação de tipos

5.2.9.2. Definindo States

Através do botão *State* é possível a definição dos estados de um domínio. Ao acionar o botão é apresentada a tela onde poderá ser especificada uma

definição e os parâmetros referentes aquele estado. Tais parâmetros terão que ser de um tipo já definido. A especificação do tipo será feita através de uma caixa de seleção que conterá todos os tipos definidos. A Figura 5.8 apresenta a tela de definição de estados de um domínio. Após a especificação do estado e o acionamento do botão *Add* o estado será incluído em uma tabela na mesma tela, que conterá todos os estados já especificados.

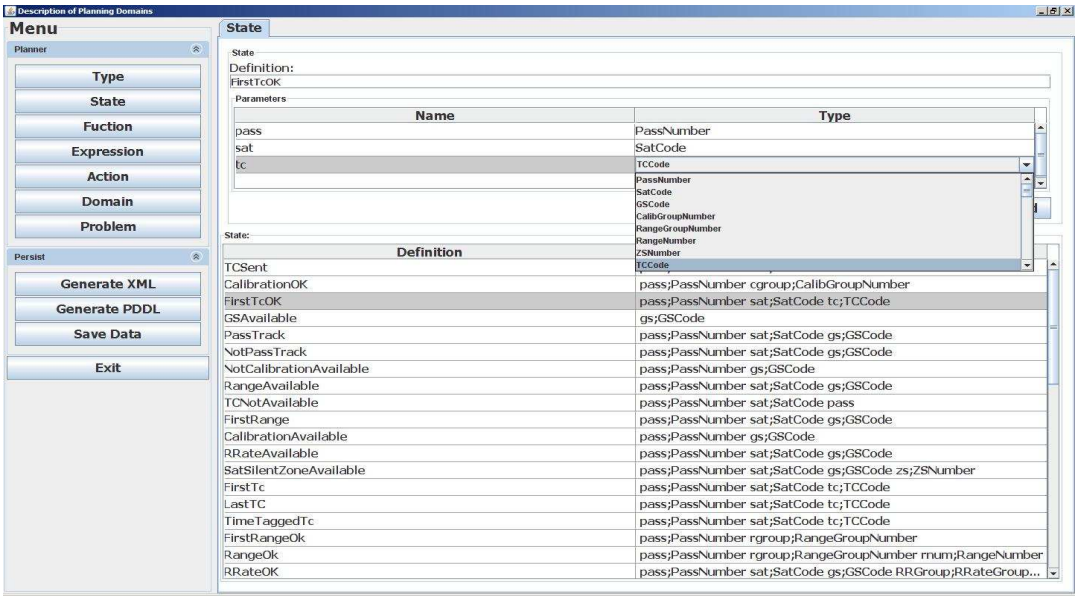

Figura 5.8 - Tela de especificação de estados

5.2.9.3. Criando Functions

O botão *Function* disponibiliza a tela para definição de funções de um domínio. A tela de *Function* segue a mesma estrutura descrita para a definição de states. Isto pode ser observado na Figura 5.9.

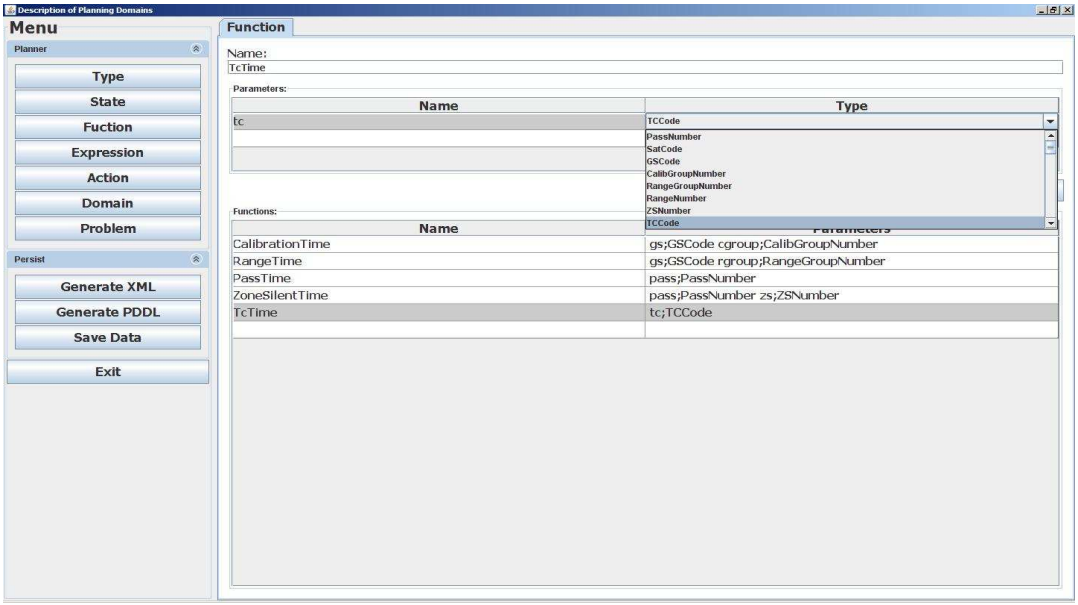

Figura 5.9 - Tela de especificação de funções

5.2.9.4. Criando expressões de avaliação

Para a definição de expressões de avaliação o botão disponibilizado é o *Expression*. Ao acionar este botão é apresentada uma tela onde poderá ser especificado um conjunto de expressões, sendo possível especificar um nome, um símbolo, uma descrição e um conjunto de parâmetros. Ao definir tais parâmetros a expressão se torna aplicável somente a elementos do tipo destes parâmetros. Na Figura 5.10 é apresentada a tela para definição de expressões de avaliação.

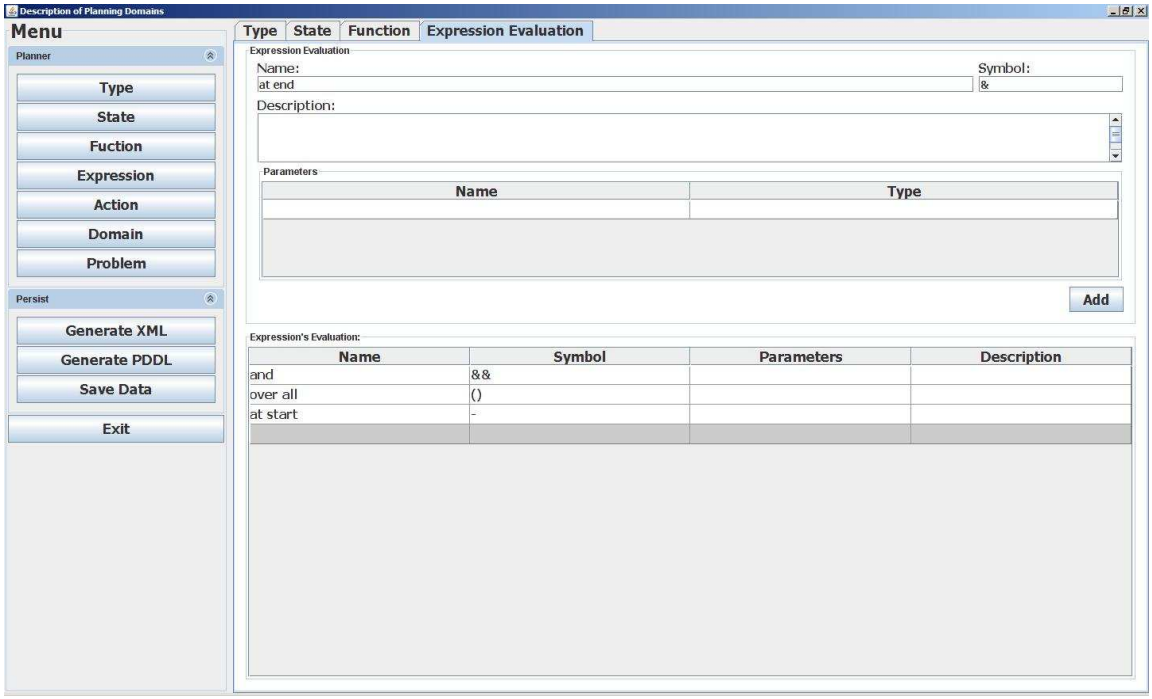

Figura 5.10 - Tela de especificação de expressões de avaliação

5.2.9.5. Especificando ações

A definição de ações consiste em definir um conceito complexo que envolve inúmeras restrições e vários relacionamentos, o acionamento do botão *Action* disponibiliza a tela apresentada pela Figura 5.11.

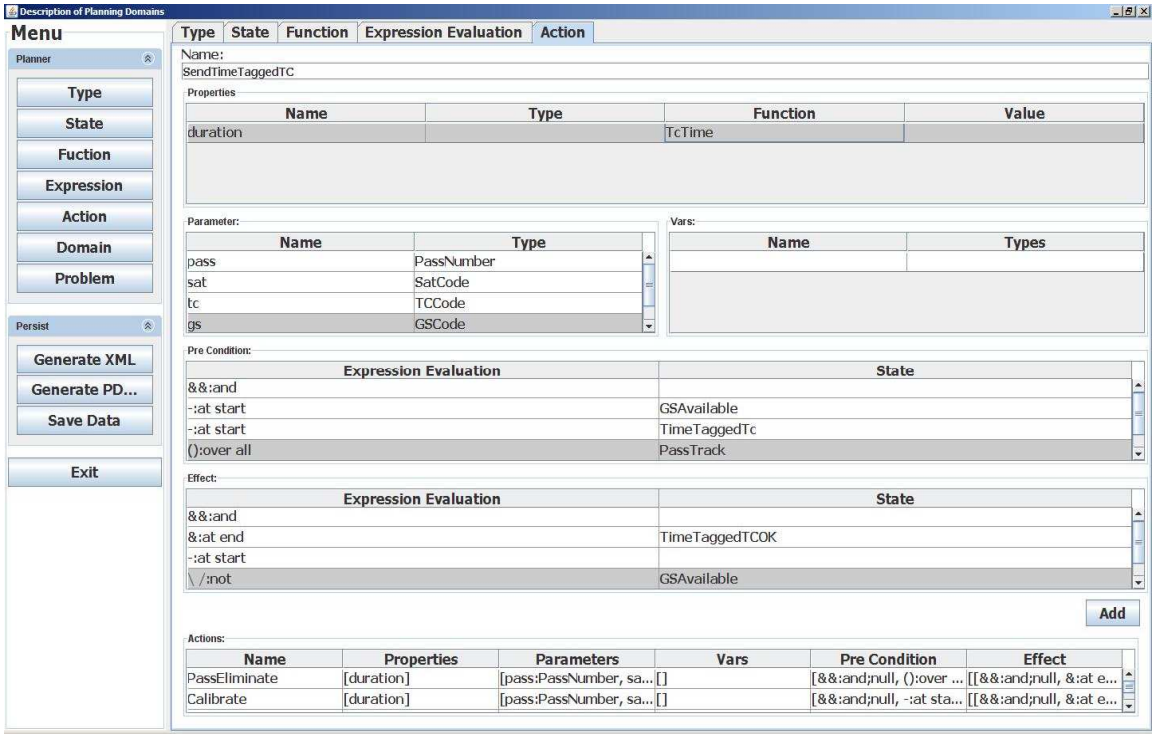

Figura 5.11 - Tela de especificação de ações

A seguir é descrita a estrutura da tela:

1. Um campo para a definição do nome da ação;

2. Uma tabela para definição de propriedades da ação, onde será definido o nome da propriedade seu tipo ou uma função associada. Essa propriedade pode ser iniciada com um valor caso seja simples, ou definir um valor para uma função associada.

3. É disponibilizada uma tabela para a inserção de variáveis, onde terá que ser definido o nome e um tipo para estas.

4. A tabela de pré-condições conterá os estados necessários para a execução da ação, estes estados podem ser concatenados com expressões de avaliação.

5. Definir-se-á o efeito da ação da mesma forma de uma pré-condição, estabelecendo o cenário de estados após a execução de uma ação.

5.2.9.6. Definindo um domínio

Através da ativação do botão *Domain* é apresentada a tela que possibilita a definição de um domínio; esta tela é estruturada conforme a imagem Figura 5.12. A tela contém um campo para uma definição textual do domínio, uma tabela para a inclusão dos tipos que farão parte do domínio. Os estados do domínio são definidos também em uma tabela onde além do estado são apresentados seus . Da mesma forma são definidas as funções e ações de um domínio.Todos os conceitos são preenchidos através de listas de seleção, que são apresentadas quando ocorrer um *click* em uma célula da coluna *Name* em Funções e Ações, na coluna *State* ao definir-se os Estados e em*Type* para a definição de tipos.

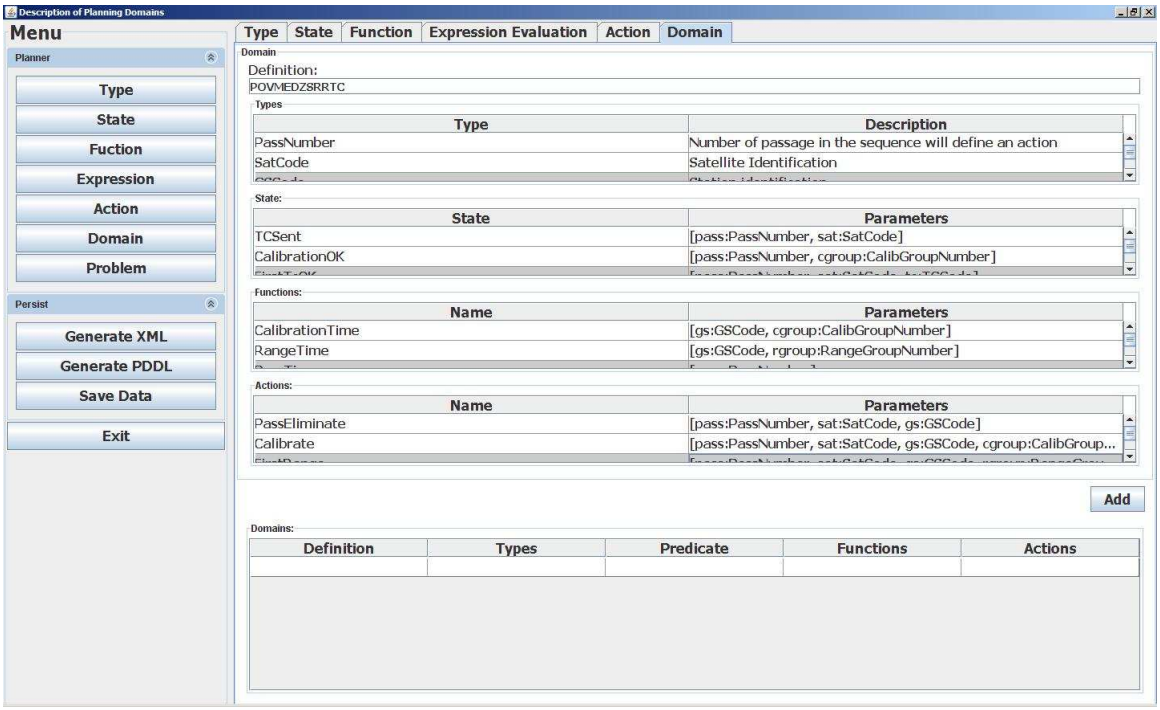

Figura 5.12 - Tela de definição de domínios

5.2.9.7. Declaração de problemas

Após a definição de ao menos um domínio é possível declarar um problema acionando o botão *Problem* que irá apresentar a tela mostrada na Figura 5.13. Nesta tela é possível fazer a definição textual do problema, associar este a um domínio através de uma caixa de seleção e em uma tabela define-se objetos com seus respectivos tipos.

Na definição do estado inicial é possível declarar funções a serem executadas no início do processo de planejamento, passando como parâmetros para estas funções, os objetos definidos anteriormente e pré-definindo valores para estas. Então é necessário definir a situação do estado inicial, através dos estados criados para o domínio em questão. Com isso é possível, por exemplo, definir um evento exógeno utilizando expressões e valores para os estados. Os parâmetros que serão definidos para os estados também serão baseados nos objetos definidos no problema em questão.

Por fim é feita a declaração da meta a ser alcançada pelo planejador; esta meta também é definida através de um cenário onde um conjunto de estados deve ser estabelecido.

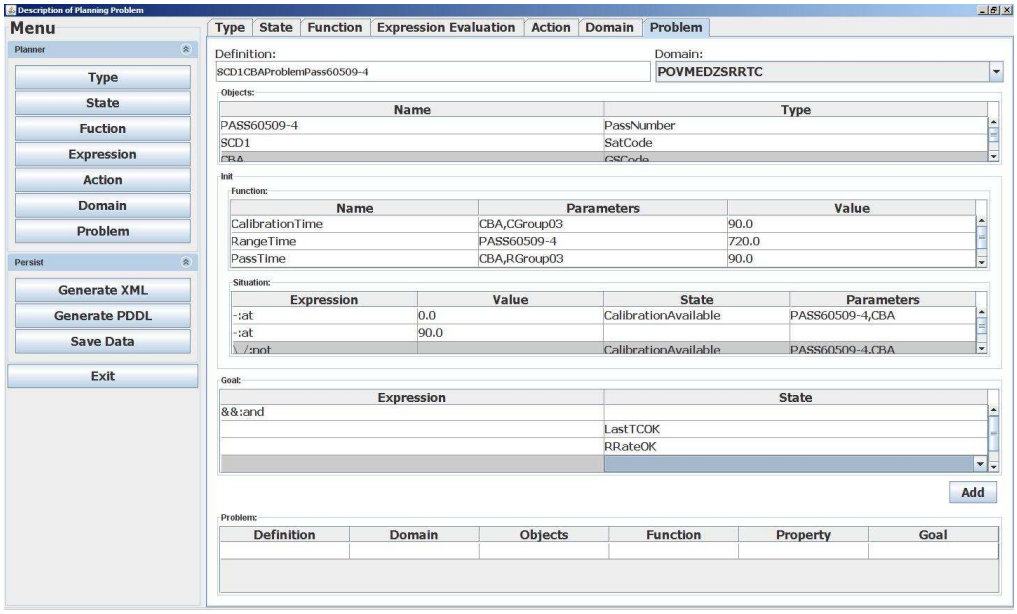

Figura 5.13 - Tela de declaração de problemas

Todas as tabelas utilizadas no protótipo têm uma funcionalidade para edição, através da seleção de um item correspondente; os dados são disponibilizados para edição e visualização nos respectivos campos de entrada. A exclusão de dados é feita através da seleção de um item e acionamento da tecla *Delete*.

5.3. Um XML *Schema* para descrições de domínios e problemas de planejamento em KPlanOO

O XKPS apresentado neste trabalho é um XML *Schema* que define uma linguagem baseada em XML para a descrição de problemas de planejamento automático. Este *schema,* está baseado no *KPlanOO*, porém, tira partido de XML para facilitar não só a melhor comprensão de documentos de especificação de domínios e problemas de planejamento, mas também a sua validação e planejamento.

O uso de XML é, atualmente, generalizado e existem diversas ferramentas (por exemplo: XML:parser (XML:PARSER, 2009), saxon (SAXON, 2009), xerces (XERCES, 2009), XML:simple (XML:SIMPLE, 2009)) e tecnologias associadas (por ex.: *Document Object Model* (DOM) (DOM, 2009), XML *Schema Definition*

Language (XSDL) (XML SCHEMA, 2009), *Extensible Stylesheet Language* (XSL) (XSL, 2009)) que simplificam e auxiliam significativamente o desenvolvimento de aplicações que processam documentos XML.

O uso de XML é vantajoso não apenas para o programador, mas também para o projetista que escreve e mantém as regras de um domínio. De fato, o recurso de XML torna particularmente clara a delimitação dos componentes, facilitando a sua legibilidade. Facilita ainda a validação de documentos usando ferramentas apropriadas de uso muito divulgado. Como mencionado a cima, tirando partido de XSDL, estas ferramentas, existentes, permitem a validação não só da sintaxe, mas também de outras restrições como a integridade referencial. Embora a validação possibilitada por XML não elimine a existência de erros na especificação dos domínios e problemas, contribui para a eliminação de muitos deles e consequentemente para uma gestão mais eficaz dos domínios de planejamento.

5.3.1. Regras usadas na concepção do XKPS

O *KPlanOO* é um modelo relativamente extenso, pois apesar ter somente 18 classes e duas interfaces conta com dezenas de relacionamentos gerando uma vasta gama de possibilidades. Assim, para garantir consistência e uniformidade entre domínios e problemas de planejamento especificados através de linguagens OO, que implemente o *KPlanOO* e domínios e problemas de planejamento especificados usando XKPS, adotou-se um conjunto de regras para realizar o mapeamento dos elementos que compõe o *KPlanOO* em elementos sintáticos do esquema XML, que enumera-se:

- **Regra 1**: Mapear objetos KPlanOO em elementos XML.

- **Regra 2**: Mapear atributos KPlanOO em elementos XML.

- **Regra 3**: Mapear tipos primitivos de KPlanOO em tipos derivados por restrição de tipos primitivos XSD.

- **Regra 4**: Especificar a multiplicidade/obrigatoriedade de um objeto ou atributo do KPlanOO recorrendo aos atributos minOccurs e maxOccurs de XSDL.

- **Regra 5**: Mapear atributos chave dos objetos do KPlanOO em elementos XML com um atributo *name* definido como chave para elemento raiz do XSD.

- **Regra 6**: Usar referências-chave (elemento XSDL keyref) na declaração dos elementos.

- **Regra 7**: Designar elementos XML pelos nomes dos elementos de KPlanOO (classes, atributos, tipos primitivos) correspondentes.

As Regras 1 a 4 afetam a estrutura propriamente dita dos documentos XML de especificação de domínios e problemas de planejamento, estabeleceu-se que tal estrutura como sendo a que um documento XML especificando domínios e problemas de planejamento deveria ter.

As Regras 5 e 6 foram determinadas não pela estrutura desejada para os documentos XML de especificação de políticas de segurança, mas pelo objetivo de explorar as facilidades de validação de XML na detecção de erros na especificação de políticas de segurança usando o XKPS.

A Regra 7 é essencial para facilitar a compreensão de domínios e problemas de planejamento expressos usando a linguagem definida por XKPS. O KPlanOO usa um conjunto de nomes com uma semântica comum na área de

planejamento. A adoção de nomes diferentes poderia tornar mais difícil a compreensão do domínio especificado.

5.3.2. XKPS: Um XML Schema para KPLANOO

Apresentado o processo usado para desenvolver um XML *Schema* baseado na *KPlanOO*, apresentamos a estrutura base deste *schema*, evidenciando os aspectos que considera-se mais importantes. Por uma questão de limitação de espaço não será apresentado o *schema* completo.

Na Figura 5.14 está representada a estrutura geral do *schema*. A Figura 5.15 apresenta a estrutura de *Action* do *schema* e, por último, na Figura 5.16 está representada a estrutura de *Problem* do *schema*.

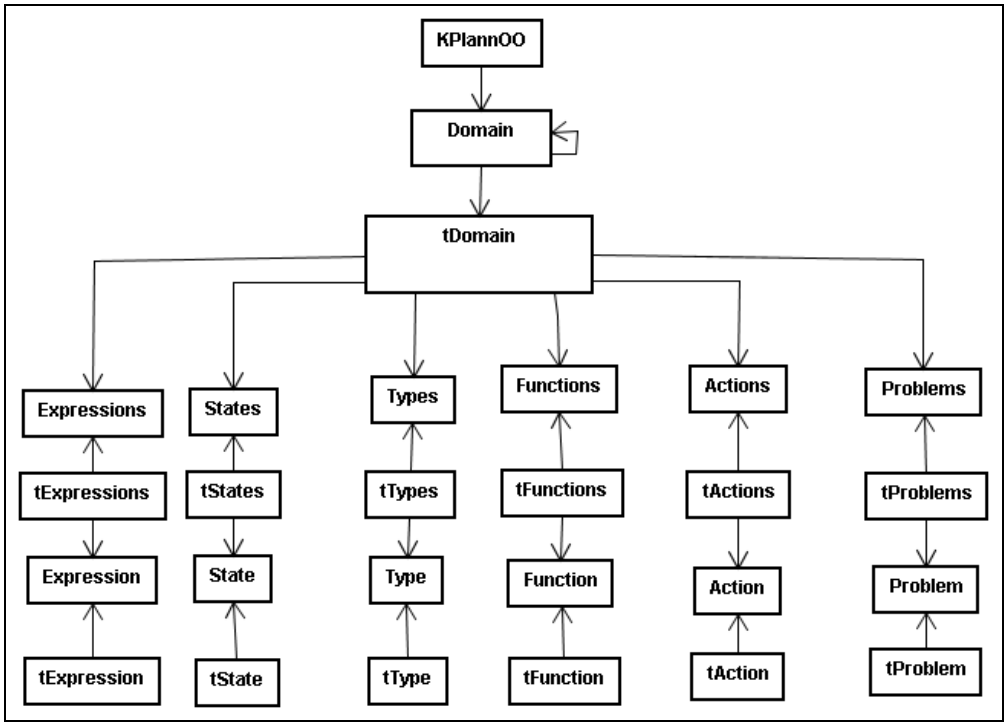

Figura 5.14 - Estrutura geral do XML Schema

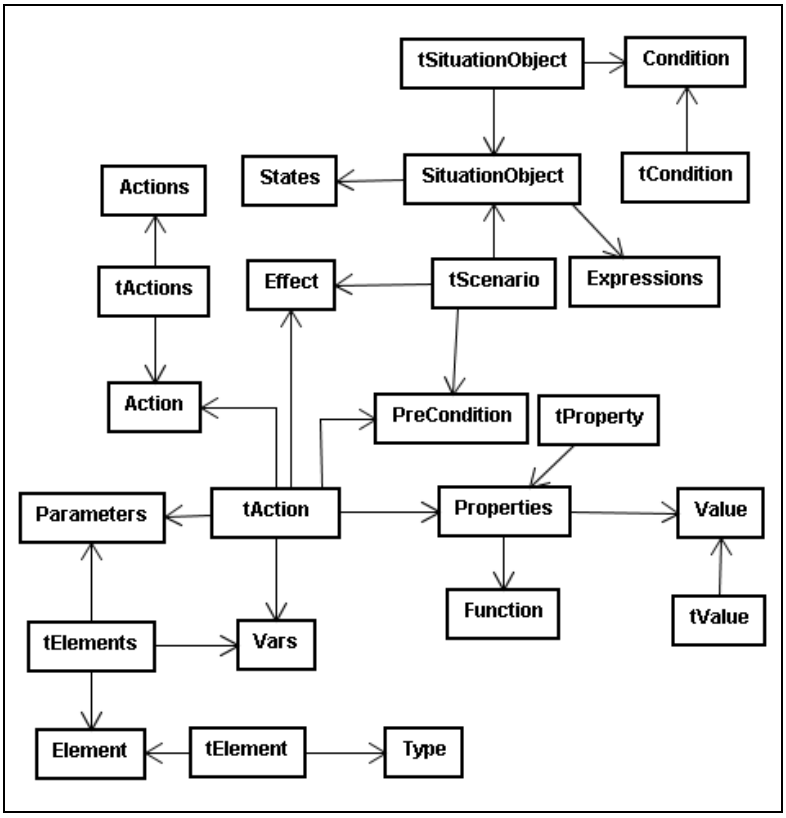

Figura 5.15 - Estrutura para Action do XML Schema

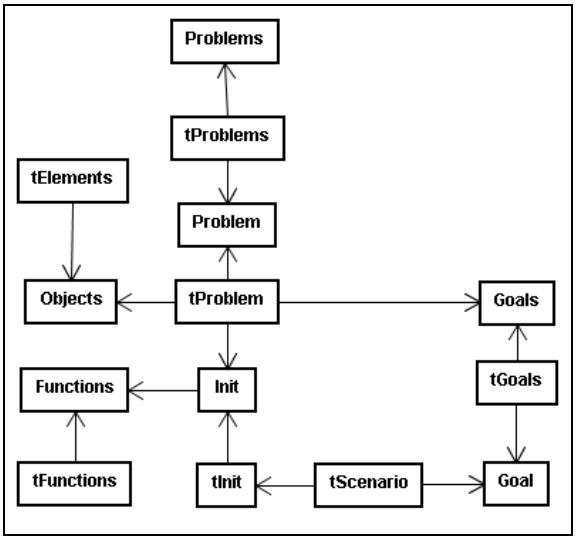

Figura 5.16 - Estrutura para problem do XML Schema

Como ilustrado pela Figura 5.14, o elemento raiz contém um conjunto de referências para outros elementos, cada um dos quais associados a um tipo. Na Figura 5.15 e Figura 5.16 temos uma estrutura bem parecida com a

estrutura do elemento raiz, porém com tipos inerentes a *actions* e *problems* respectivamente.

Faz-se agora a descrição em maior detalhe de cada um dos componentes deste XML *Schema,* começando no topo, no elemento raiz, e descendo progressivamente. O elemento raiz do XKPS foi definido como uma sequência de elementos que representam as diferentes classes definidas no *KPlanOO*. Denominamos este elemento de *KPlanOO*, uma vez que é usado para agrupar as informações necessárias à especificação de domínios de planejamento automático. A definição do objeto *KPlanOO* inclui a definição da sequência dos elementos correspondentes aos objetos *KPlanOO*, bem como a declaração das chaves necessárias para garantir a unicidade desses objetos, como se pode observar na Figura 5.17.

Figura 5.17 - Representação parcial do elemento raiz (*KPlanOO*) do XKPS

A definição deste elemento consiste em declarar o tipo e um conjunto de referências-chave (elemento `keyref` do XSD). Estas últimas garantem que qualquer objeto referenciado está declarado no documento XML, evitando deste modo referências quebradas (*dangling references*).

Nas linhas 7 a 11, está definido o elemento *Domain* através do tipo *tDomain* que pode ser observado na Figura 5.18, em *tDomain* está definido os vários elementos que correspondem aos objetos definidos pelo *KPlanOO*, aplicando a regra 5 de forma a permitir a verificação automática de multiplicidade e/ou obrigatoriedade dos objetos.

```
36      <xsd:complexType name="tDomain">
37          <xsd:sequence>
38              <xsd:element name="definition" type="xsd:string"
39                  minOccurs="1" maxOccurs="1" />
40              <xsd:element name="types" type="kpo:tTypes" maxOccurs="1"
41                  minOccurs="1" />
42
43              <xsd:element name="states" type="kpo:tStates"
44                  maxOccurs="1" minOccurs="1" />
45
46              <xsd:element name="functions" type="kpo:tFunctions"
47                  maxOccurs="1" minOccurs="1" />
48
49              <xsd:element name="actions" type="kpo:tActions"
50                  maxOccurs="1" minOccurs="1" />
51          </xsd:sequence>
52      </xsd:complexType>
```

Figura 5.18 - Representação do tipo Domain do XKPS

Note-se que estes elementos são declarados através de referência, de forma a estruturar o *schema* de uma forma mais modular.

Assim para cada classe de objetos *KPlanOO* é definido um elemento XML em nível global. Foram então definidos tipos específicos para as classes do *KPlanOO* e tipos que representam coleções destes. Observamos na Figura 5.19 a definição de *Type* nas linhas 57 a 60 e sua coleção (*tTypes*) entre as linhas 49 a 55.

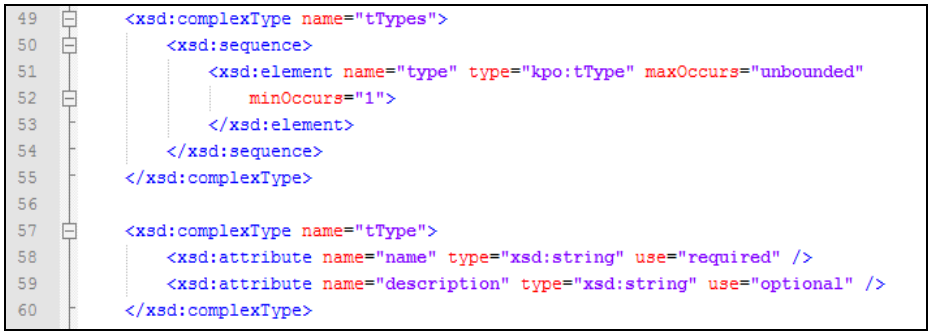

Figura 5.19 - Especificação de Type e sua coleção no XKPS

Em *tDomain*, apresentado na Figura 5.18, a sequência de elementos corresponde a definição de relacionamentos do *KPlanOO*, de acordo com a regra 2. Note-se ainda o uso dos atributos XSD `minOccurs` e `maxOccurs`, de acordo com a regra 4.

Foi ainda criado um conjunto de tipos de dados primitivos que permitem especificar as características de vários atributos quase sempre presentes na especificação de políticas de segurança. Estes elementos permitem facilitar e reduzir a complexidade na implementação do XML *Schema*, aumentando a sua modularidade. A Tabela 5.1 resume alguns dos tipos de dados primitivos e a sua função.

Tabela 5.1 - Lista de alguns tipos de dados primitivos

Tipo de Dados	Função
Type	Define um tipo
Element	Define um elemento
Argument	Define argumentos
Value	Define valores

5.4. Considerações

Este capítulo visou possibilitar um entendimento das ferramentas aplicadas na construção do protótipo. Com esse estudo, foi possível a definição do processo de consistência de modelos e geração de código entre o meta-modelo e formas de persistência de dados e descrição de domínios de planejamento. A grande

vantagem desse processo é a possibilidade de extensão do protótipo, permitindo a geração de código nas plataformas de implementação mapeadas.

Foi também apresentado um *schema* XSD para o *KPlanOO*, preservando a sua semântica, mas tirando partido de XML para reduzir a possibilidade de erros na especificação de domínios e problemas de planejamento, um aspecto fundamental para garantir a eficácia de planejadores. O uso de XML permite o recurso a um conjunto de ferramentas de difusão generalizada para edição e validação de XML, reduzindo o número de erros na edição de domínios e problemas de planejamento.

No capítulo seguinte são apresentados três exemplos de domínios e problemas de planejamento de aplicação do meta-modelo. Além disso, é feita uma análise dos benefícios da utilização do modelo. Por fim, é apresentada uma especialização do *KPlanOO* feita para servir como descrição para o domínio de simulação de um satélite, promovendo a experimentação das metas-propostas como reuso, especialização do meta-modelo e flexibilidade na definição de domínios reais.

Também foi elaborado uma proposta de arquitetura para demonstrar o uso do *KPlanOO* com os demais ciclos em um projeto de planejamento, e como este pode se integrar com os demais processos, em um contexto mais amplo de um sistema de controle de satélites.

6 ESTUDO DE CASO E ANÁLISE DOS RESULTADOS OBTIDOS

Apresentar-se-á neste capítulo resultados de modelagem de domínios de planejamento automático realizados durante o desenvolvimento do presente trabalho com o objetivo de verificar os conceitos do meta-modelo proposto. De fato, foram modelados diversos domínios clássicos de planejamento durante a elaboração deste trabalho utilizando os processos e os conceitos do *KPlanOO*. Muitos destes domínios guiaram o desenvolvimento do protótipo e do XML *Schema* apresentado no Capítulo 5.

Entre os domínios modelados, aqueles de maior destaque foram selecionados e são apresentados neste capítulo. Primeiro será apresentada a modelagem de um dos domínios mais conhecidos na área do Planejamento Automático em IA, o chamado Mundo de Blocos. Em seguida é apresentado o modelo do domínio de rastreio e controle de satélite do INPE, mostrando a capacidade do meta-modelo de modelar diferentes domínios de planejamento automático em especial os domínios da área espacial, característica essa que resultou em uma publicação no SpaceOps 2010. (SILVA et al., 2010)

Será apresentado também uma extensão do *KPlanOO* feita para servir como descrição para o domínio de simulação de um satélite, promovendo a experimentação das metas propostas. Foi também estendido o protótipo criado para realizar a captação dos dados para a descrição do domínio de simulação de um satélite.

Por fim será apresentado neste capitulo uma proposta de arquitetura elaborada para monstrar o uso do *KPlanOO* com os demais ciclos em um projeto de planejamento e como este pode se integrar com os demais processos, em um contexto mais amplo de um sistema de controle de satélite.

6.1. Domínio mundo de blocos

O problema do mundo dos blocos é um problema clássico da área de IA. Este domínio clássico vem sendo utilizando até hoje para ilustrar os principais conceitos teóricos e práticos sobre sistemas de planejamento. Para o problema, assume-se uma mesa com blocos identificados de forma única, empilhados uns sobre os outros num determinado número de colunas. O objetivo do problema é que um braço robótico rearranje os blocos de forma a se obter determinada configuração. Para atingir este objetivo, as seguintes regras devem ser respeitadas:

1. Deve-se mover apenas um bloco por vez;

2. Podem ser movidos apenas blocos que não estejam sob nenhum outro bloco.

A Figura 6.1 ilustra o domínio do Mundo de Blocos que será modelado no próximo tópico.

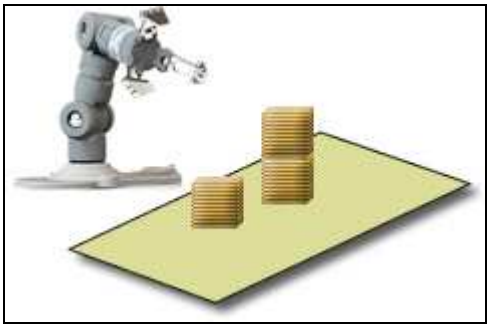

Figura 6.1 - Ilustração do mundo de blocos

6.1.1. Modelando o domínio

Baseado nas características do Mundo de Blocos Clássico acima pode-se dizer que os principais elementos deste domínio são os blocos, o braços e a mesa, sendo que o braço é o agente do domínio e os blocos e a mesa são os recursos do mesmo.

Podemos definir um tipo *block* para este domínio e os estados *on*, *ontable*, *clear*, *handempty* e *holding*. Na Figura 6.2 é mostrado o diagrama de objetos desta definição utilizando as entidades do *KPlanOO*.

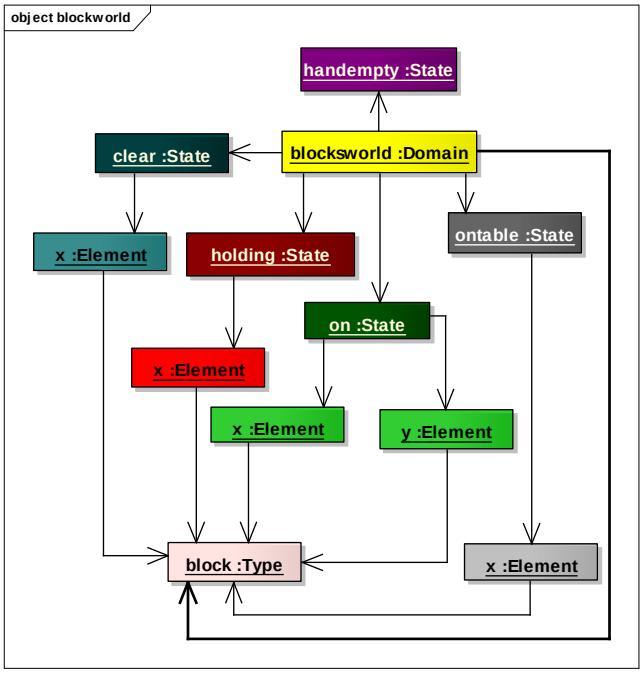

Figura 6.2 - Diagrama de objetos da definição do tipo block e dos estados de domínio de mundo de blocos.

Nessa versão do domínio do Mundo dos Blocos o braço robótico é capaz de executar quatro ações, estas podem ser observadas na Tabela 6.1.

Tabela 6.1 - Ações que possíveis de serem executadas pelo braço robótico

pick-up (?x - block)	o agente pega um bloco de cima da mesa e o segura
put-down (?x - block)	o agente coloca o bloco que está segurando em cima da mesa
stack (?x - block ?y - block)	o agente coloca o bloco que está segurando em cima de outro bloco
unstack (?x - block ?y - block)	o agente pega um bloco de cima de outro bloco e o segura

Na Figura 6.3 pode-se observar o diagrama de objetos que contempla a ação *pick-up*, onde foi criado um objeto *Action* para esta ação que tem como

parâmetro um objeto de *Element* nomeado como x do tipo *block*. Este parâmetro é referenciado tanto pelos estados que fazem parte da pré-condição para execução da ação como para o efeito que a mesma irá gerar.

Temos como pré-condição da ação *pick-up* que *x* deve estar livre (*clear*), deve estar sobre a mesa (*ontable*) e que o braço deve estar vazio. Como efeito está ação irá criar o seguinte cenário: *x* não estará mais na mesa nem livre, o braço passa estar ocupado segurando *x*.

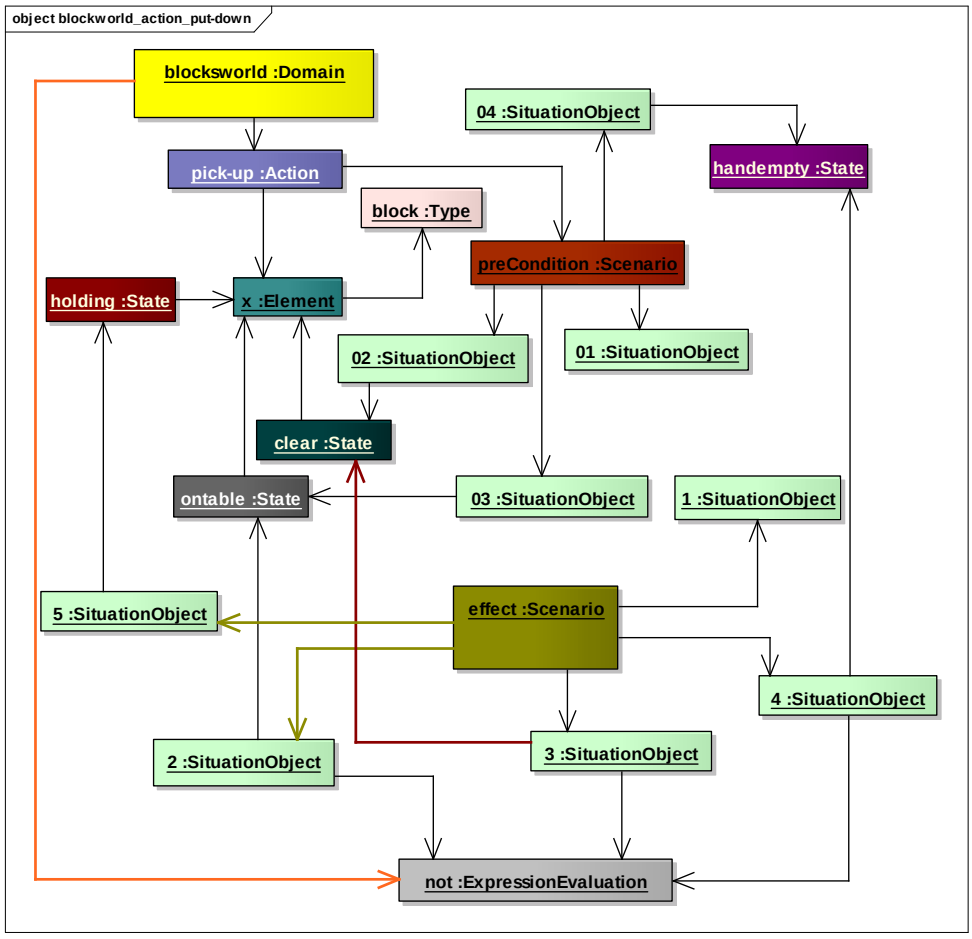

Figura 6.3 - Diagrama de objetos da definição da ação pick-up

Para completar o modelo do domínio é necessário ainda representar as demais ações que são descritas na Tabela 6.1. Podemos utilizar o Diagrama de Objetos para representar estes objetos no modelo, mas para este estudo de caso, iremos apresentar as demais ações através de uma descrição XML

baseada no XKPS. Na Figura 6.4 observamos a descrição XML para ação *put-down*.

```
1  <kpo:action name="put-down">
2      <kpo:parameters>
3          <kpo:element name="x" type="block"/>
4      </kpo:parameters>
5      <kpo:precondition>
6          <kpo:scenario>
7              <kpo:situations>
8                  <kpo:situationobj>
9                      <state definition="holding">
10                         <kpo:parameters>
11                             <kpo:element name="x"/>
12                         </kpo:parameters>
13                     </state>
14                 </kpo:situationobj>
15             <kpo:situations>
16         </kpo:scenario>
17     </kpo:precondition>
```

Figura 6.4 - Descrição XML para ação put-down

Nesta ação também tem-se: como parâmetro, um objeto de *Element* nomeado como x do tipo *block*, como pré-condição o cenário feito apenas do estado do braço segurando o objeto passado como parâmetro; e como efeito o cenário em que o braço não está mais segurando o objeto *x,* ficando livre, e o objeto x está sobre a mesa e livre.

Na Figura 6.5 é apresentada a descrição XML para ação *stack* com sua pré-condição. Esta ação diferentemente das anteriores tem como parâmetro dois elementos do tipo *block* representados pelos objetos *x* e *y.* A pré-condição para a execução de *stack* é o cenário em que x e *y* não podem ser iguais, o braço estar segurando o objeto *x* e o objeto *y* está livre. Entre as linhas 9 e 19 da Figura 6.5 temos a especificação da pré-condição que diz que *x* e *y* não podem ser iguais, que foi modelada se beneficiando da flexibilidade do KPlanOO, especificando um *SituationObject* que possui um *SituationFather* composto por uma *ExmpressionEvaluation* que representa uma negação, além do *SituationFather* este *SituationObject* possui uma *ExpressionEvaluation* representando igualdade entre os dois parâmetros definidos na ação.

```
<kpo:action name="stack">
    <kpo:parameters>
        <kpo:element name="x" type="block"/>
            <kpo:element name="y" type="block"/>
        </kpo:parameters>
    <kpo:precondition>
        <kpo:scenario>
            <kpo:situations>
                <kpo:situationobj>
                    <kpo:situationFather>
                        <expressionEvaluation name="not"/>
                    </kpo:situationFather>
                    <expressionEvaluation symbol="=">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                                <kpo:element name="y" type="block"/>
                        </kpo:parameters>
                    </expressionEvaluation>
                </kpo:situationobj>
                <kpo:situationobj>
                    <state definition="holding">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                    </kpo:situationobj>
                        <kpo:situationobj>
                            <state definition="clear">
                                <kpo:parameters>
                                    <kpo:element name="y"/>
                                </kpo:parameters>
                            </state>
                        </kpo:situationobj>
                    </kpo:situations>
        </kpo:scenario>
    </kpo:precondition>
```

Figura 6.5 - Descrição XML para ação *stack* com sua pré-condição

Após a execução desta ação teremos como efeito o cenário onde o braço não estará mais segurando *x*, *y* deixará de estar livre, o braço estara vazio e *x* estará sobre *y*, na Figura 6.6 observamos a descrição XML do efeito e as *tags* de finalização da ação *stack*.

Figura 6.6 - Descrição XML do efeito da ação stack e fechamento de tags abertas na Figura 6.5

Por último descreve-se a ação *unstack* onde também temos dois blocos como parâmetros, e o cenário de pré-condição é que os dois blocos que são representados pelos objetos *x* e *y* não sejam iguais, que *x* esteja sobre *y* e esteja livre e também que o braço esteja vazio. A descrição XML desta pré-condição pode ser observada na Figura 6.7. O efeito desta ação é a negação do efeito da ação *stack*, sendo assim não se expõe este efeito, já que observando a Figura 6.6 pode-se conferir o cenário, sendo que para *unstack*, nos *situationobj* onde tem `<expressionEvaluation name="not"/>` não existe e nos que em *stack* não possuí a *expressionEvaluation* foi adicionado. No Apêndice A

encontra-se toda a definição em XML do domínio do Mundo de Blocos baseada no XKPS.

```
<kpo:precondition>
    <kpo:scenario>
        <kpo:situations>
            <kpo:situationobj>
                <kpo:situationFather>
                    <expressionEvaluation name="not"/>
                </kpo:situationFather>
                <expressionEvaluation symbol="="/>
                    <kpo:parameters>
                        <kpo:element name="x"/>
                        <kpo:element name="y" type="block"/>
                    </kpo:parameters>
                <expressionEvaluation>
            </kpo:situationobj>
            <kpo:situationobj>
                <state definition="on">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                        <kpo:element name="y"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>
            <kpo:situationobj>
                <state definition="clear">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>
            <kpo:situationobj>
                <state definition="handempty"/>
            </kpo:situationobj>
        <kpo:situations>
    </kpo:scenario>
</kpo:precondition>
```

Figura 6.7 - Descrição XML da pré-condição da ação unstack

6.1.2. Modelando o problema do mundo de blocos

Neste estudo de caso do domínio Mundo de Blocos, para demonstrar como é possível modelar de problemas utilizando o *KPlanOO,* modelou-se quatro blocos (A, B, C e D) como objetos a serem manipulados, estes estarão inicialmente dispostos todos em cima da mesa e livres, consequentemente o braço robótico estará vazio, como ilustra a Figura 6.8.

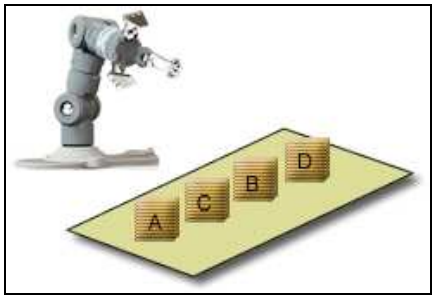

Figura 6.8 - Ilustração do cenário inicial para o problema do Mundo de Blocos

Como objetivo deseja-se que o bloco D esteja sobre o C, o C sobre o B e este esteja sobre o A, como é observado na Figura 6.9.

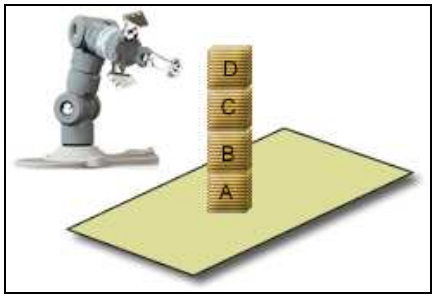

Figura 6.9 - Ilustração do estado objetivo para problema do Mundo de Blocos

Na Figura 6.10 é apresentado o Diagrama de Objetos com a definição dos objetos para o problema do Mundo de Blocos. Já na Figura 6.11, é mostrado a modelagem do estado inicial do problema.

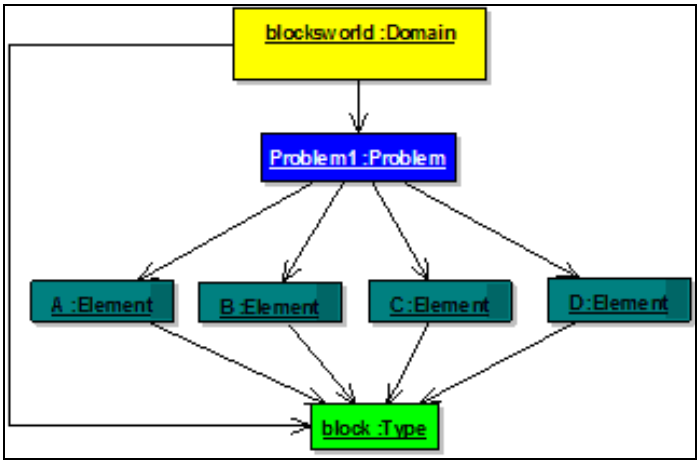

Figura 6.10 - Diagrama de objetos com a definição dos objetos para o problema do mundo de blocos.

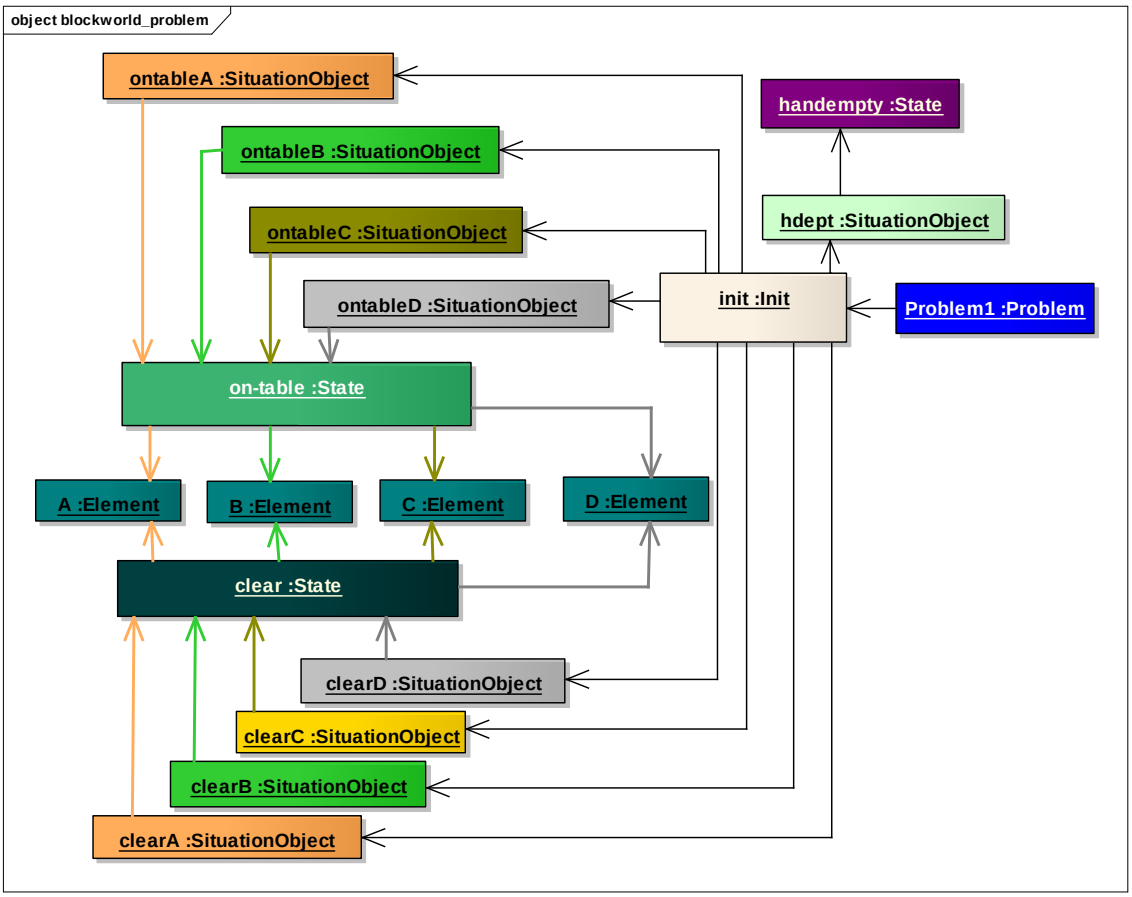

Figura 6.11 - Diagrama de objetos com a definição do cenário inicial do problema do mundo de blocos

Na definição do estado inicial foi criada uma instância da classe *Init* que é uma especialização de cenário. Tendo referência para nove objetos do tipo *SituationObject* sendo quatro para definir os estados onde os objetos estão sobre a mesa, quatro que define que estes estão livres e um que estabelece que o braço está vazio.

Para o cenário objetivo, criamos uma instância da classe *Goal* que por sua vez terá três referências para objetos da classe *SituationObject*, cada um deste objetos irá referenciar o objeto que representa o estado *on* tendo como parâmetros os respectivos blocos que o enunciado propõe. Na Figura 6.12 é apresentado o Diagrama de Objetos com o cenário objetivo para o problema do Mundo de Blocos, os números nas setas de associação entre o objeto *on* e os

objetos que representam os blocos é para definir a ordem dos parâmetros. Sendo assim temos definido que o bloco D está sobre o C, C está sobre B e B sobre A.

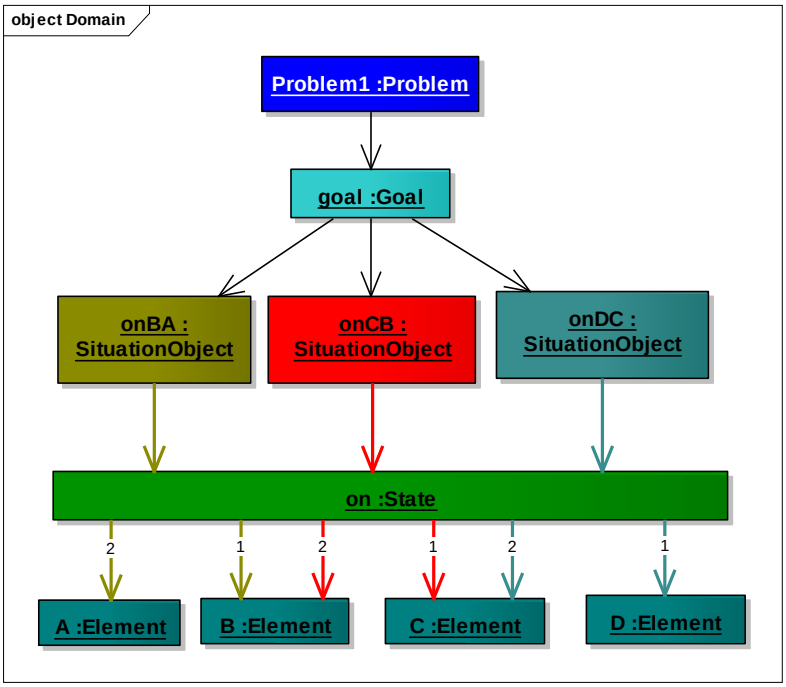

Figura 6.12 - Diagrama de objetos com a definição do cenário objetivo do problema do mundo de blocos

6.2. Domínio de rastreio de satélites do INPE

Para este domínio foi utilizado a especificação feita por Gonçalves (GONÇALVES, 2006), pois este domínio foi dentre todos os pesquisados, aquele que exigiu mais recursos e abstrações do meta-modelo. Estass exigências foram determinantes na condução de grande parte do desenvolvimento do KPlanOO, pois se mostraram como necessidades de outros domínios da área espacial. Mesmo outros domínios da área espacial como o de Satélite para obter Imagem de Fenômenos Espaciais utilizado na IPC não tinha restrições e necessidades como as do domínio em questão.

Em sua definição, Cardoso (GONÇALVES, 2006) estabeleceu uma ontologia do domínio definindo um vocabulário para este. A definição desse vocabulário foi feita orientada para os elementos existentes na estrutura da linguagem PDDL que inclui a definição de todos os tipos, ações, funções e predicados existentes no domínio. Como no desenvolvimento do *KPlanOO* a premissa da generalização foi sempre trabalhada e perseguida tendo se baseado também, como já dito, em conceitos de meta-ontologia, o domínio descrito por Cardoso é contemplado em sua totalidade pelo meta-modelo.

Na ontologia do domínio de rastreio para o grupo de satélites (SCD1, SCD2 e CBERS2) estudado definiu-se alguns tipos para o domínio: Código da Passagem, Código do Satélite, Código da Estação Terrena, Código do Telecomando e Identificação da Zona de Silêncio. Foram também definidas as funções para o cálculo de duração das seguintes ações: calibração, transmissão de telecomando, execução de medidas de velocidade, execução de medidas de distância e zona de silêncio. Alguns estados definidos para o domínio foram definidos com base em categorias, como pode ser observado abaixo:

- **Estados iniciais do ambiente**: são, aqueles predicados que informam o estado do ambiente para iniciar a preparação do plano de vôo. Como exemplo podem-se citar alguns predicados definidos que informam a situação inicial do ambiente da passagem a ser rastreada: ET disponível para rastreio (GSAvailable), a passagem deve ser rastreada (PassTrack) e a medida de distância deve ser executada para a passagem (RangeAvailable);

- **Estados intermediários**: são, aqueles estados que ocorrem durante o planejamento cujos não são mapeados no arquivo de problemas. Para exemplificar pode-se citar o predicado de calibração executada (CalibrationOK), que é um estado atingido após a execução da ação

calibrar (Calibrate), necessário para a ação Executar Medida de Distância (Range);

- **Eventos exógenos**: são, aqueles eventos que passam a ser verdadeiros ou falsos em um instante de tempo determinado independentemente das ações que o planejador escolhe para executar. Este tipo foi bastante utilizado na modelagem do domínio de rastreio de satélite. O evento que informa a ocorrência de zona de silêncio durante o rastreio de uma passagem (SatSilentZoneAvailable) é um exemplo típico deste tipo de evento, ou seja, este evento informa a janela de tempo em que ocorre uma zona de silêncio, durante a qual nenhuma ação de comunicação entre o satélite e a estação terrena pode ser executada;

- **Estados objetivos**, são, aqueles predicados que podem fazer parte do estado objetivo informando o que se quer atingir com o plano a ser gerado. O objetivo do plano pode ser composto de sub-objetivos, isto é, muitas vezes não é possível estabelecer um único predicado objetivo para a geração do plano e o objetivo passa a ser uma conjunção de predicados. Alguns exemplos do estudo de caso são: medida de distância executada (RangeOK) e passagem eliminada (PassEliminated) (GONÇALVES, 2006).

Após a definição da ontologia, foi realizada a codificação do arquivo de domínio na linguagem PDDL selecionada, onde todos os termos definidos na ontologia do domínio são escritos de acordo com a estrutura da linguagem. Nesse trabalho realizamos a especificação deste domínio de várias formas. As Figuras 5.6 até 5.12 utilizadas do capitulo 5 para demonstrar o uso do protótipo proposto, foram geradas a partir da especificação deste domínio.

Algumas ações definidas neste domínio necessitam de uma duração, que é o tempo necessário para a execução da mesma (*duration*). Ela deve ser definida como um valor fixo ou um valor variável que é obtido por uma função. Esta duração é modelada no *KPlanOO* através da coleção de propriedades que a classe *Activity* (Super Classe de *Action*) tem. Cada propriedade pode ter um valor(*Value*) ou uma função, como foi observado no capítulo 4 e ilustrado pela Figura 4.11. Vale ressaltar que o KPlanOO nesse caso flexibiliza ainda mais tendo possibilitando que seja definido um intervalo de valores através do auto-relacionamento existente na classe *Value*. O valor retornado pela função deve ser definido no arquivo de problemas.

A Figura 6.13 apresenta a tela para o cadastro de Ações do protótipo criado com os dados da Ação *SendTC* preenchidos.

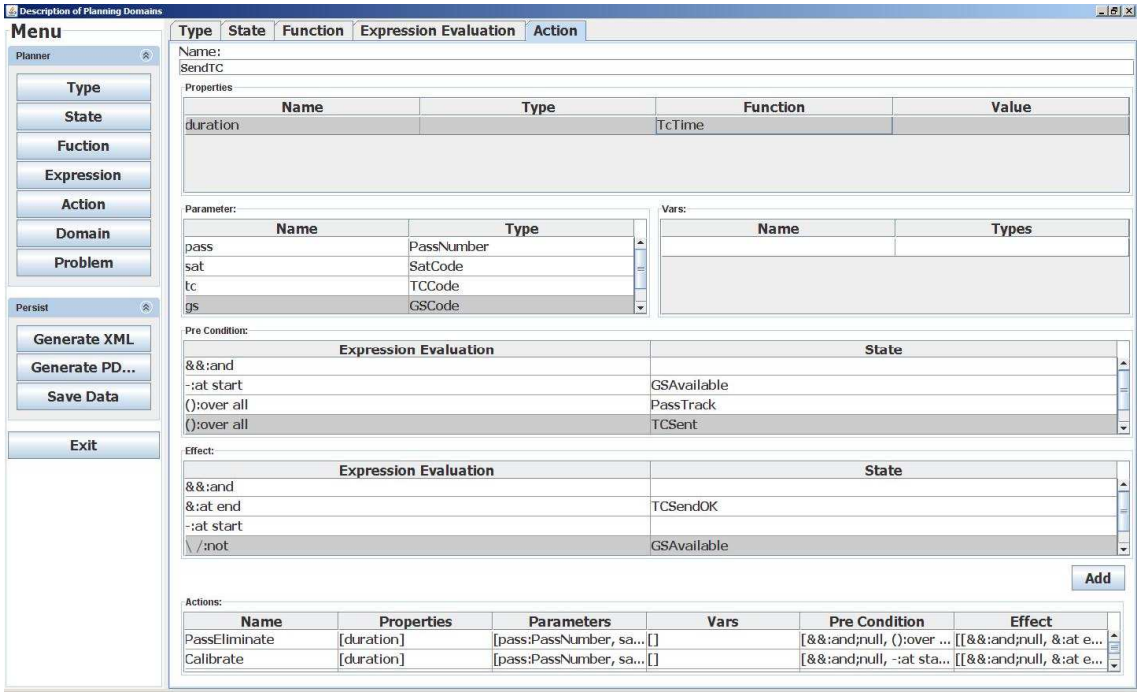

Figura 6.13 - Tela de Definição de ações com dados da ação SendTc preenchidos

Observamos nessa tela que a propriedade *duration* tem o valor definido através de uma função. A Figura 6.14 apresenta um trecho da descrição XML para a ação *SendTC*, onde podemos observar a definição da propriedade *duration*.

```
<kpo:action name="SendTC">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="cgroup" type="CalibGroupNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="TcTime">
                    <kpo:parameters>
                        <kpo:element name="tc" type="TCCode"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>
```

Figura 6.14 - Trecho da descrição XML da ação SendTC

Na Figura 6.15 é apresentado o Diagrama de componentes que demonstra todas as ações do domínio, por questão de espaço são apresentados apenas os objetos de cada ação, a especificação de cada uma das ações em XML pode ser vista no Apêndice B.

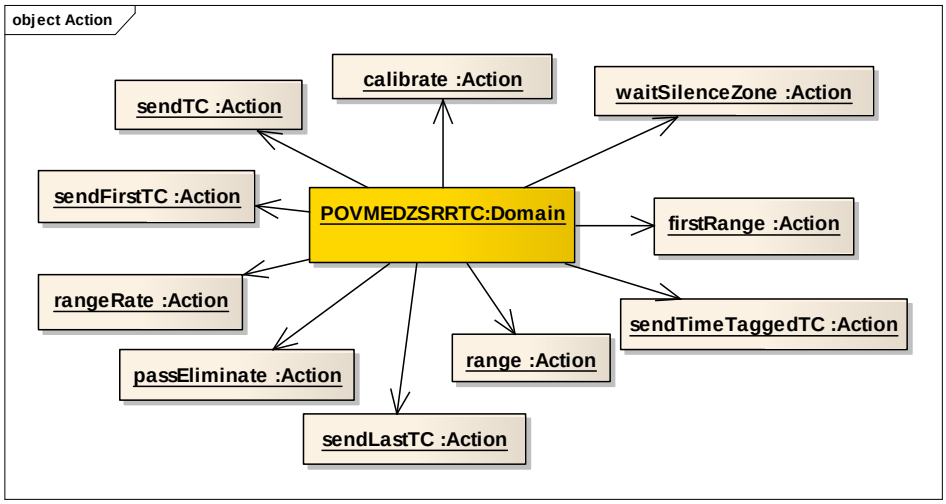

Figura 6.15 - Diagrama de objetos com as ações do domínio de rastreio de satélites do INPE.

Na Figura 6.16, são apresentadas as definições dos tipos e das funções do domínio de rastreio de satélites do INPE.

```
<domain>
    <kpo:definition>POVMEDZSRRTC</kpo:definition>
    <kpo:types class="list">
        <kpo:type name="PassNumber"/>
        <kpo:type name="SatCode"/>
        <kpo:type name="GSCode"/>
        <kpo:type name="CalibGroupNumber"/>
        <kpo:type name="RangeGroupNumber"/>
        <kpo:type name="RangeNumber"/>
        <kpo:type name="ZSNumber"/>
        <kpo:type name="TCCode"/>
        <kpo:type name="RRateGroupNumber"/>
    </kpo:types>

    <kpo:functions>
        <kpo:function name="CalibrationTime">
            <kpo:parameters>
                <kpo:element name="gs" type="GSCode"/>
                <kpo:element name="cgroup" type="CalibGroupNumber"/>
            </kpo:parameters>
        </kpo:function>

        <kpo:function name="RangeTime">
            <kpo:parameters>
                <kpo:element name="gs" type="GSCode"/>
                <kpo:element name="rgroup" type="RangeGroupNumber"/>
            </kpo:parameters>
        </kpo:function>

        <kpo:function name="PassTime">
            <kpo:parameters>
                <kpo:element name="pass" type="PassNumber"/>
            </kpo:parameters>
        </kpo:function>

        <kpo:function name="ZoneSilentTime">
            <kpo:parameters>
                <kpo:element name="pass" type="PassNumber"/>
                <kpo:element name="zs" type=    "ZSNumber"/>
            </kpo:parameters>
        </kpo:function>

        <kpo:function name="TcTime">
            <kpo:parameters>
                <kpo:element name="tc" type="TCCode"/>
            </kpo:parameters>
        </kpo:function>
    </kpo:functions>
```

Figura 6.16 - Trecho da descrição XML do domínio de rastreio de satélites do INPE com a definição de tipos e funções

Todo o restante da definição segue a mesma estrutura já apresentada anteriormente, a definição de todo o domínio pode ser vista no Apêndice B.

6.2.1. Modelando o problema de rastreio de satélites

No estudo de caso do domínio de Rastreio de Satélites do INPE, iremos também modelar um problema especificado em Cardoso (GONÇALVES, 2006). Neste problema modelamos 16 objetos sendo; um para representar o número da passagem do satélite (PASS60509-4), um para o código do satélite (SCD1), um para o código da estação terrena (CBA), sete para os códigos de telecomandos (TC137, TC118, TC138, TC140, TC150), um para identificação de zona de silêncio (ZS01) e mais sete para controle de envio e calibração. Na Figura 6.17 é apresentado o trecho da descrição do problema onde são definidos os objetos envolvidos. Na Figura 6.18 é apresentado, na tela do protótipo, o local onde estes objetos podem ser preenchidos, destacado em vermelho.

```
<KplanOO>
    <problem>
        <definition>SCD1CBAProblemPass60509-4</definition>
        <domain>
            <kpo:definition>POVMEDZSRRTC</kpo:definition>
        </domain>
        <objects>
            <kpo:element name="PASS60509-4" type="PassNumber"/>
            <kpo:element name="SCD1" type="SatCode"/>
            <kpo:element name="CBA" type="GSCode"/>
            <kpo:element name="CGroup03" type="CalibGroupNumber"/>
            <kpo:element name="RGroup03" type="RangeGroupNumber"/>
            <kpo:element name="Range02" type="RangeNumber"/>
            <kpo:element name="Range03" type="RangeNumber"/>
            <kpo:element name="Range04" type="RangeNumber"/>
            <kpo:element name="Range05" type="RangeNumber"/>
            <kpo:element name="ZS01" type="ZSNumber"/>
            <kpo:element name="RRGroup07" type="RRateGroupNumber"/>
            <kpo:element name="TC137" type="TCCode"/>
            <kpo:element name="TC118" type="TCCode"/>
            <kpo:element name="TC138" type="TCCode"/>
            <kpo:element name="TT140" type="TCCode"/>
            <kpo:element name="TT150" type="TCCode"/>
        </objects>
```

Figura 6.17 - Trecho da descrição XML do problema para o domínio de rastreio de satélites com a definição dos objetos envolvidos no problema

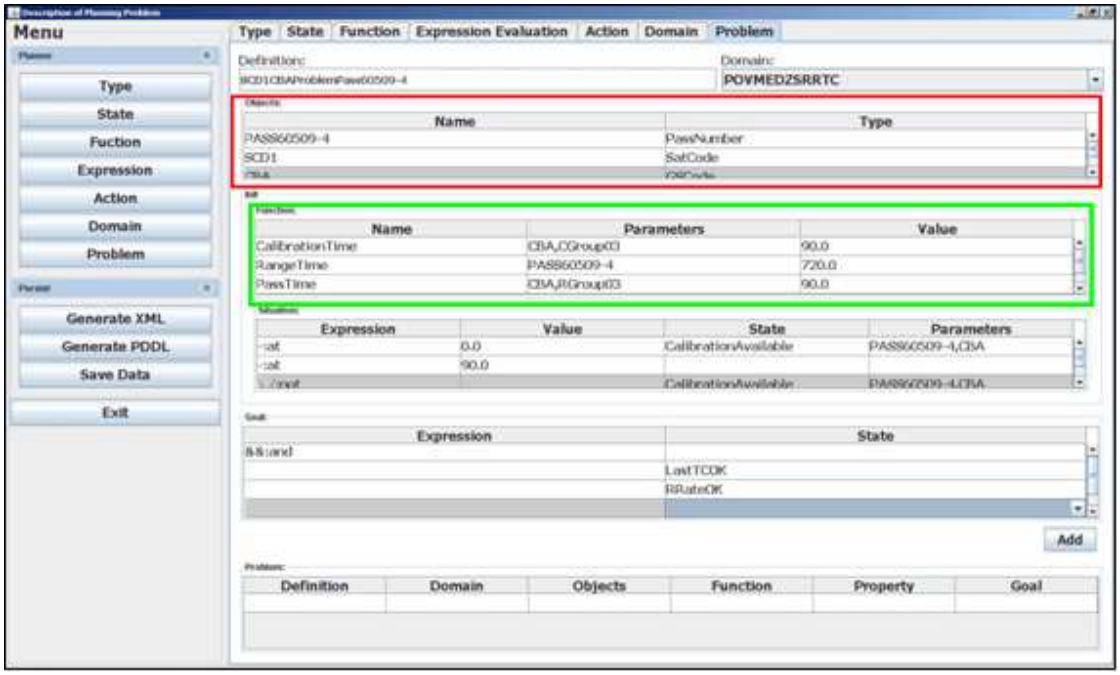

Figura 6.18 - Tela do protótipo para definição de problemas com destaques

Nesta descrição de problema é definido que algumas funções deverão ser executadas no início do processo de planejamento para preparação do ambiente. Na definição do domínio somente foram declaradas as funções do domínio, e na declaração do problema foram declaradas quais delas serão utilizadas, quais objetos serão utilizados como parâmetro desta e seu valores de retorno. Pode-se observar esta declaração no destaque em verde na Figura 6.18 e no trecho XML apresentado pela Figura 6.19.

```
<init>
    <functions>
        <kpo:function name="CalibrationTime">
            <kpo:parameters>
                <kpo:element name="CBA" type="GSCode"/>
                <kpo:element name="CGroup03" type="CalibGroupNumber"/>
            </kpo:parameters>

            <kpo:properties>
                <kpo:property value="30.0"/>
            </kpo:properties>
        </kpo:function>

        <kpo:function name="PassTime">
            <kpo:parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
            </kpo:parameters>

            <kpo:properties>
                <kpo:property value="1052.0"/>
            </kpo:properties>
        </kpo:function>
```

Figura 6.19 - Trecho da descrição XML com definição de funções a serem executadas no início do processo de planejamento.

Para o estado inicial foi definido um cenário para o ambiente onde a estação terrena CBA está disponível para rastreio da passagem definida pelo objeto PASS60509-4 e a medida de distância para o satélite definido pelo objeto SCD1 também está disponível.

O estado inicial também foi composto da definição de estados para eventos exógenos com tempos pré-determinados para cada passagem. Assim os estados que representam estes eventos são: Calibração (*CalibrationAvailable*) com tempo 0.0, Primeira Medida de Distância (*FirstRange*) com tempo entre 480.0 e 570.0, envio do primeiro telecomando (*FirstTc*) que deve ocorrer em um tempo entre 390.0 e 392.0, envio de último telecomando (*LastTC*) onde o tempo definido fica entre 1050.0 e 1052.0, medida de velocidade que tem o intervalo(*RRateAvailable*) de 420.0 e 421.0 para ocorrer e por fim a definição de Zona de Silêncio que deverá ocorrer entre 870.0 e 900.0.

Diferente da definição na PDDL onde seria necessário a declaração destes estados (predicados em PDDL) por duas vezes fazendo uma negação do

predicado em questão para definir o fim do intervalo, no *KPlanOO* utilizamos o recurso de intervalo de valores disponibilizado pela classe *Value*. O uso deste recurso pode ser observado no trecho da descrição XML apresentado pela Figura 6.20, onde é definida uma expressão de avaliação “at” um valor base (480.0) e seu limitador. Vale lembrar que podemos ter várias declarações de *rangeValue* para *Value*.

```
<kpo:situationobj>
    <expressionEvaluation name="at">
        <toReturn>
            <value>
                <amount>"480.0"</amount>
                <rangeValue>"570.0"</rangeValue>
            </value>
            <rangeValue>
        </toReturn>

    <stateObject definition="FirstRange">
        <parameters>
            <kpo:element name="PASS60509-4" type="PassNumber"/>
            <kpo:element name="SCD1" type="SatCode"/>
            <kpo:element name="CBA" type="GSCode"/>
        </parameters>
    </stateObject>
</kpo:situationobj>
```

Figura 6.20 - Trecho da descrição XML com definição de intervalo de tempo representado pela expressão “at”.

Como objetivo, este problema estabelece que na passagem definida como objeto (PASS60509-4) deverá ter sido enviado para o satélite definido pelo seu código (SCD1), como último telecomando, o telecomando representado pelo objeto TC138 e que todos os envios estabelecidos para a passagem deverão ter tido sucesso e por fim ser estabelecida zona de silencio entre a passagem, o satélite e a estação terrena. Na Figura 6.21 é apresentada esta definição do objetivo através da descrição XML. As definições recolhidas são as demais declarações de sucesso dos envios (*RRateOk*).

```
<kpo:goal>
    <kpo:situationobjs>
        <kpo:situationobj>
            <stateObject definition="LastTCOK">
                <parameters>
                    <kpo:element name="PASS60509-4" type="PassNumber"/>
                    <kpo:element name="SCD1" type="SatCode"/>
                    <kpo:element name="TC138" type="TCCode"/>
                </parameters>
            </stateObject>
        </kpo:situationobj>
        <kpo:situationobj>
            <stateObject definition="RRateOK">
                <parameters>
                    <kpo:element name="PASS60509-4" type="PassNumber"/>
                    <kpo:element name="SCD1" type="SatCode"/>
                    <kpo:element name="CBA" type="GSCode"/>
                    <kpo:element name="RRGroup07" type="RRateGroupNumber"/>
                </parameters>
            </stateObject>
        </kpo:situationobj>
        <kpo:situationobj>
        <kpo:situationobj>
        <kpo:situationobj>
        <kpo:situationobj>
        <kpo:situationobj>
            <stateObject definition="SilentZone">
                <parameters>
                    <kpo:element name="PASS60509-4" type="PassNumber"/>
                    <kpo:element name="SCD1" type="SatCode"/>
                    <kpo:element name="CBA" type="GSCode"/>
                    <kpo:element name="ZS01" type="ZSNumber"/>
                </parameters>
            </stateObject>
        </kpo:situationobj>
    </kpo:situationobjs>
</kpo:goal>
```

Figura 6.21 - Definição do objetivo para o problema de rastreio de satélites do INPE

Toda a definição do problema para o domínio de Rastreio de Satélites do INPE encontra-se no Apêndice C.

6.3. KPlanOO estendido para descrever um domínio de simulação de satélites

Tominaga (2010) propõe a verificação de planos de operações em vôo por meio de um simulador de satélites, buscando respostas para questões inerentes a deficiências encontradas em outros simuladores quanto à sua capacidade de adequação do modelo a situações reais de operação. A solução encontrada consiste numa ferramenta de inteligência artificial, associada a um modelo codificado na forma de base de regras.

No referido trabalho foi construído um protótipo do simulador em três etapas. Conforme seu funcionamento proposto são as etapas:

- A busca por um modelo configurável capaz de representar o conhecimento a respeito do satélite levou à elaboração de estruturas de dados constituídos por regras de controle e parâmetros de simulação divididos em estados e eventos, com a inclusão posterior de curvas de funções. O XML foi adotado para estes arquivos por ser de fácil manipulação na ausência de um editor específico.

- A criação de um núcleo de processamento capaz de processar os arquivos XML e gerar um histórico de estados internos em formato CSV, compatível com o sistema de telemetrias de satélites em uso no CRC.

- Construção do sistema de configuração do modelo, que se encontra em pleno desenvolvimento, incorporando conceitos que surgiram a partir de estudos ligados a outros trabalhos.

Durante reuniões freqüentes do grupo de pesquisa que vem trabalhando na automatização de satélites no CRC, foi vislumbrado a possibilidade do *KPlanOO* servir como a estrutura de dados citada na primeira etapa da construção do simulador.

De fato como o *KPlanOO* foi criado procurando possibilitar a generalização dos conceitos envolvidos no planejamento automático visando principalmente a área espacial, o meta-modelo proposto mostrou ser capaz de representar o conceitos de regras de controle e parâmetros de simulação divididos em estados e eventos, envolvidos no trabalho de simulação de satélites.

Foi então realizada a especialização de duas classes do meta-modelo proposto, para dar um sentido semântico maior no contexto da simulação de satélites, como é o caso da classe Simulator que especializa *Domain* e também para incluir características especificas necessárias para a simulação como a classe *SimulatorExpression* que é uma subclasse de *ExpressionEvaluation*.

Além das especializações citadas foram incluídas classes para representar uma fila de eventos (*EventQueue*), curvas de funções e calibração (*Curves*) e entradas e saídas para as curvas de funções (*Point*).

As regras são definidas através da coleção de ações definida na classe *Domain* herdada por Simulator. Na Figura 6.22 observa-se parte do modelo com as devidas especializações para descrever um domínio de simulação de satélites.

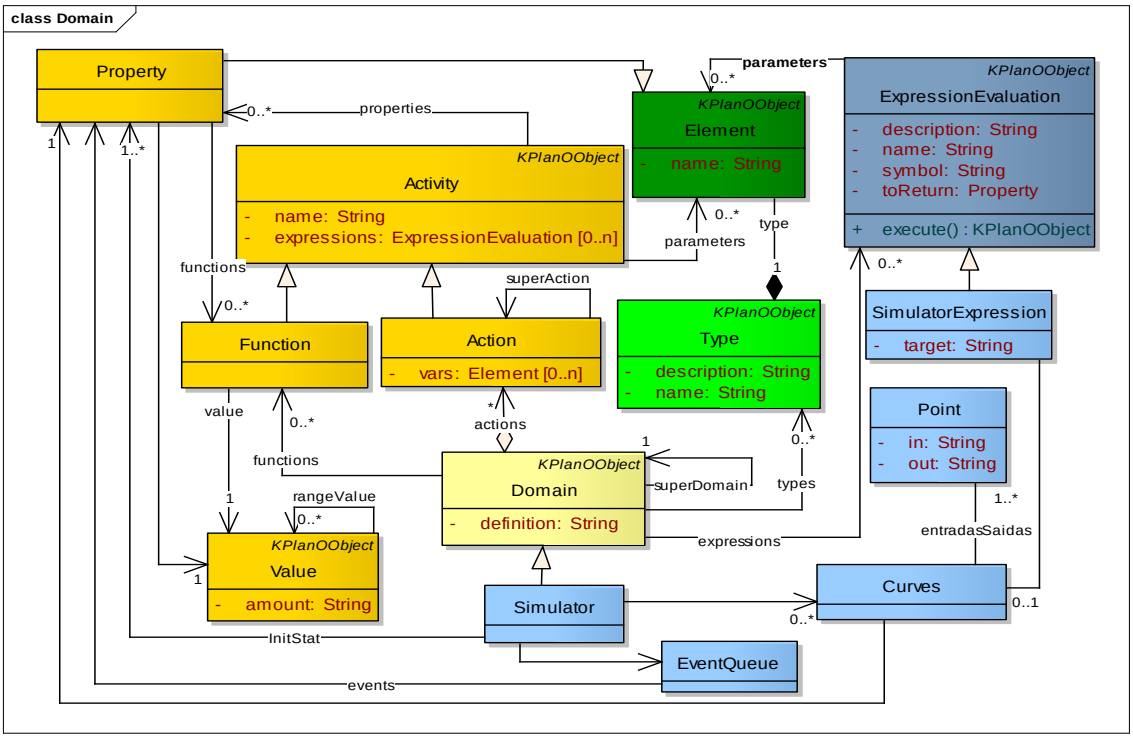

Figura 6.22 - Parte do KPlanOO com a especialização para descrever um domínio de simulação de satélites

6.3.1. Modelando um exemplo simples

Um exemplo ilustrativo de aplicação do simulador consiste num 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 et al., 2008) (TOMINAGA et al., 2009) .

Para este domínio especificou-se dois tipos de dados, um chamado de eventos e outro de estado (Este poderia ter utilizado a abstração *State* do *KPlanOO*). Foram então especificados diversos parâmetros do domínio que são configurados no início da simulação utilizando a coleção de propriedades *initSat*. Foi modelado as filas de eventos que também formam uma coleção de propriedades. Para todos os parâmetros foram configurados os respectivos tipos e valores conforme descrito por Tominaga (TOMINAGA, 2010). Foram especificadas quatro curvas com suas respectivas entradas e saídas (*Point*). São elas: *funcTEST*, *HIGH*, *MED* e *LOW*. Por fim foram criadas as regras *RNEXT*, *RHOT* e *RSTOP*. Ressalta-se que diferentemente dos domínios de planejamento automático mostrados neste trabalho as ações aqui utilizadas para a descrição de regras consideram da possibilidade de existir mais de um efeito. Toda a descrição deste domínio pode ser vista no Apêndice D.

6.3.2. Especialização do protótipo para o domínio de simulação de satélites

Seguindo as características de extensão apresentadas na seção 5.2.6, foi desenvolvida uma interface gráfica para permitir o cadastro de estado inicial, regras de controle e curvas de funções, utilizados pelo simulador. Foi adicionada na tela inicial mostrada na Figura 5.6, uma aba chamada *Simulator* contendo três opções. São elas: *Parameter*, *Curves* e *Rules*. Na Figura 6.23 podemos observar a tela inicial com a aba Simulator.

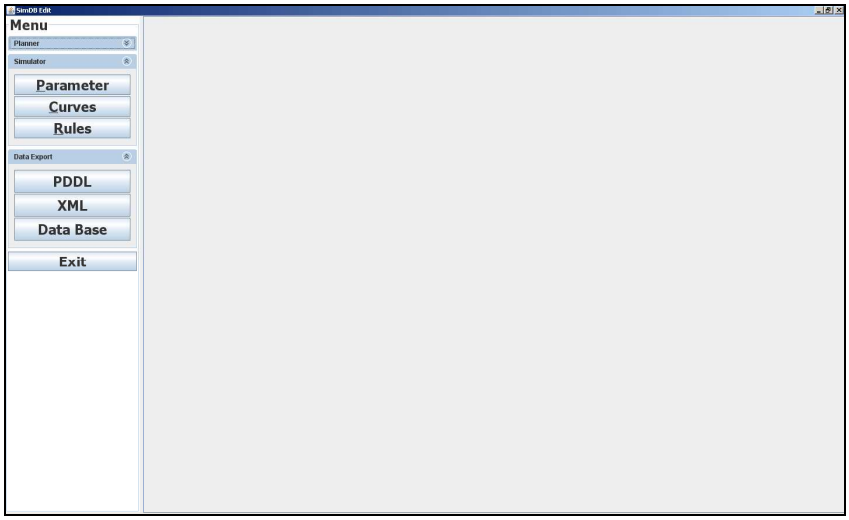

Figura 6.23 - Tela principal do protótipo especializada para o simulator

Para simular um satélite é necessário alimentar descrever o domínio com dados de uma fila de eventos obtido de planos de operação de vôo já pré-processados, dos valores dos estados internos do satélite e das regras de controle. (KONO et al., 2010) As próximas seções detalham as opções disponibilizadas para a descrição do domínio de simulação de satélites.

6.3.3. Definindo parâmetros

Ao selecionar a opção *Parameter*, a tela apresentada na Figura 6.24 é disponibilizada. Nesta tela é possível adicionar um novo parâmetro clicando no botão *Ins* para adicionar uma nova linha na tabela de cadastro logo abaixo. Nesta tabela a primeira coluna é para o registro de um identificador para o parâmetro (*Identifier*), e a segunda para o tipo (estado ou evento). É possível excluir um parâmetro clicando no botão *Del*, para editá-lo é só navegar até o registro e realizar a alteração.

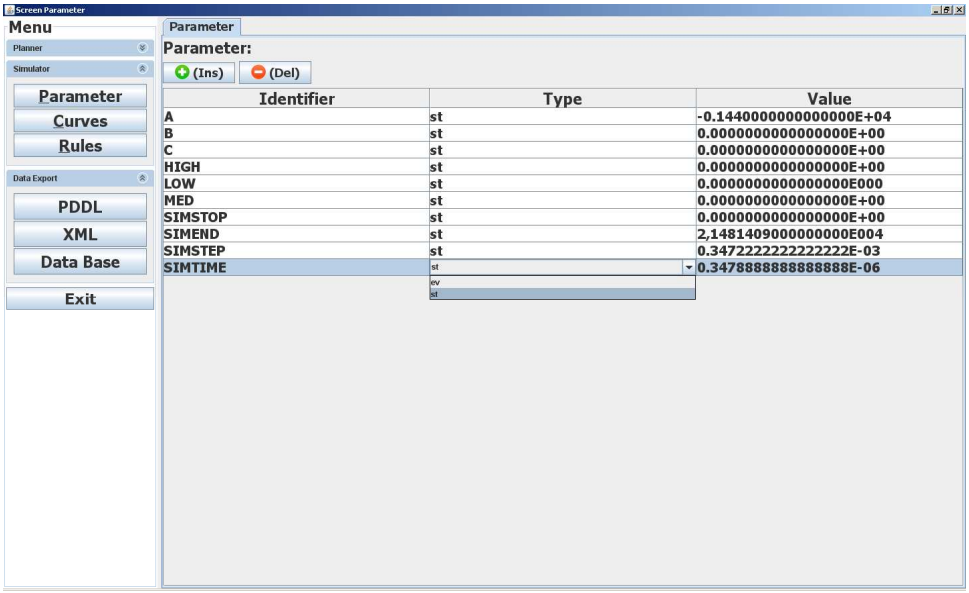

Figura 6.24 - Tela de especificação de parâmetros

6.3.4. Criando curvas

Para criar curvas deverá ser selecionada a opção *Curves* na tela principal que disponibiliza a tela de edição de Curvas de Funções, apresentada na Figura 6.25. Para inserir, alterar e excluir pontos de entrada e saída das curvas de funções o funcionamento é o mesmo descrito na seção anterior.

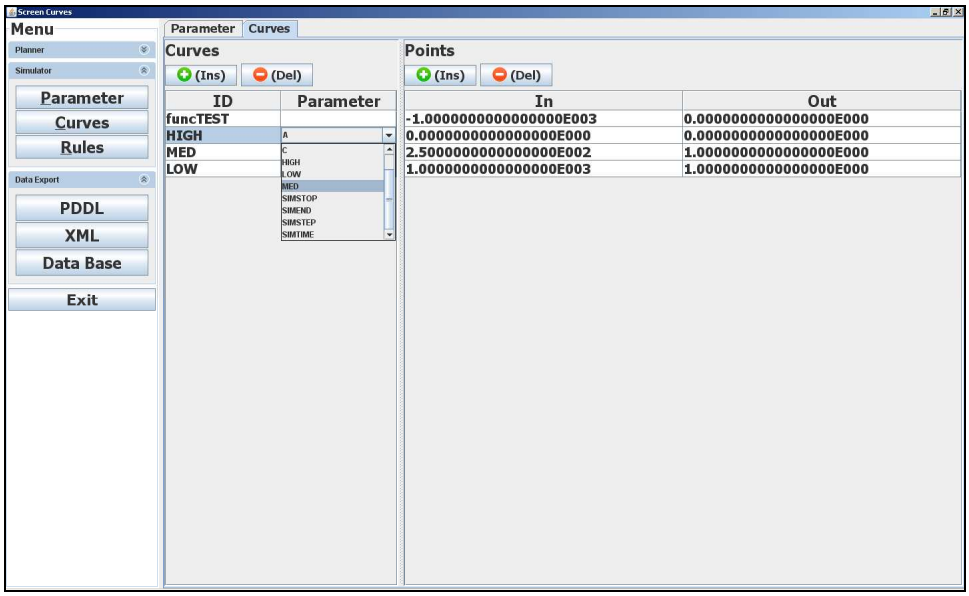

Figura 6.25 - Tela de especificação de curvas de funções

6.3.5. Definindo regras

A definição de regras se dará a partir da seleção da opção *Rules* que disponibilizará a tela para a manutenção das mesmas. A Figura 6.26 mostra esta tela, para inserir, alterar e excluir pontos de entrada e saída das curvas de funções. O funcionamento é o mesmo descrito para a edição de parâmetros. Abaixo da tabela que irá conter as regras será possível adicionar a pré-condição (*Condition*) e efeitos (*Effect*) através das opções *Insert* e *Delete*. Ao selecionar a opção *Insert* será habilitado o editor de regras que possibilitará a definição das pré-condições e efeitos através de uma sintaxe própria do simulador.

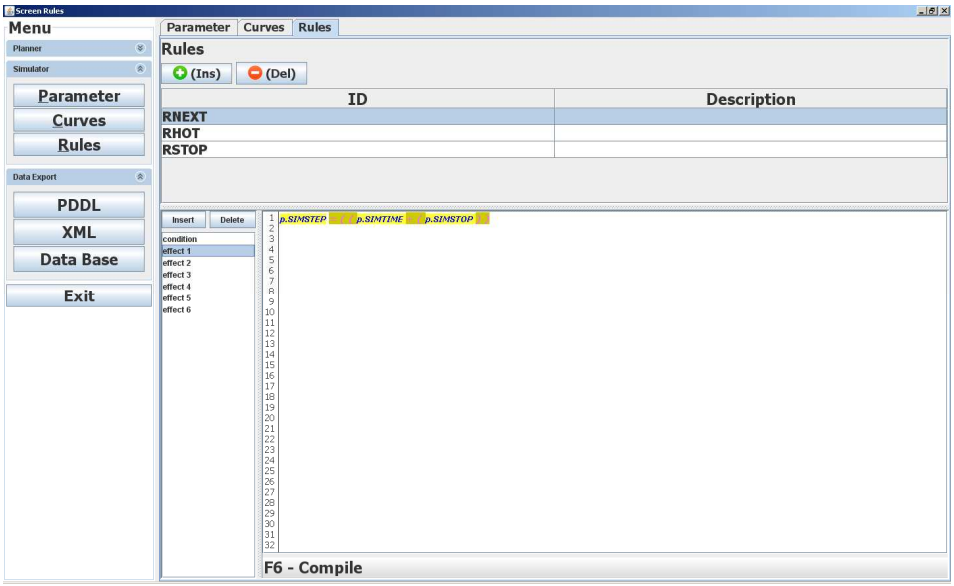

Figura 6.26 - Tela de especificação de curvas de regras

6.4. Uma arquitetura integrada para projetos de planejamento de operações espaciais

A Figura 6.27 mostra a arquitetura proposta para o novo sistema de planejamento de operações de satélites, segundo o projeto de automação

atualmente em desenvolvimento no CRC (SOUZA, 2010) (KUCINSKIS, 2010) (KONO, 2010) (TOMINAGA, 2010) (WINANDY, 2010).

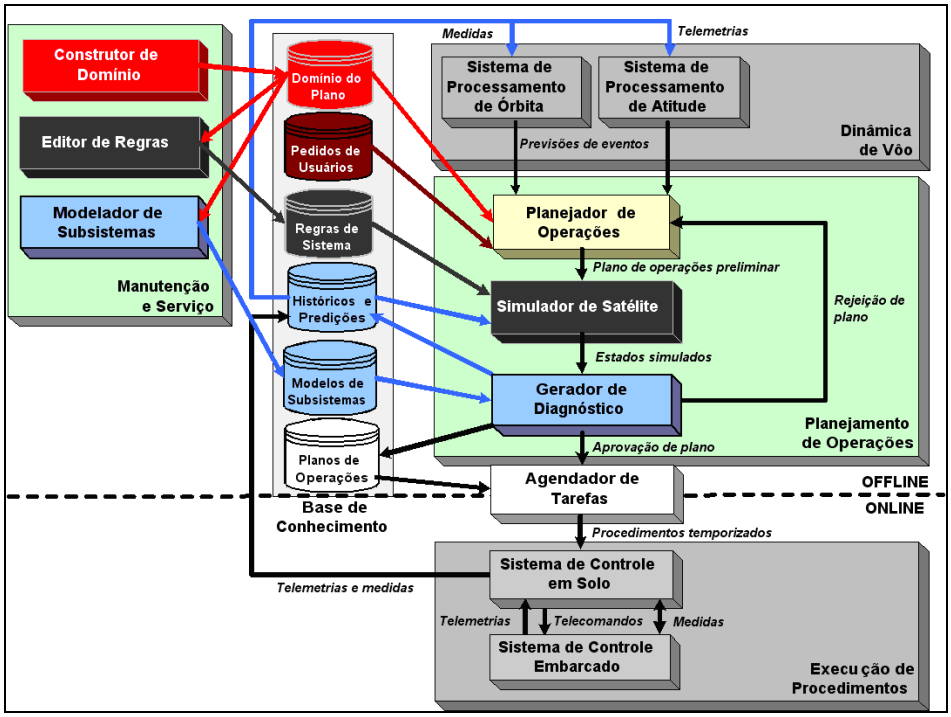

Figura 6.27 - Arquitetura do novo sistema para planejamento autônomo de operações

Na arquitetura apresentada, a camada Base de Conhecimento tem uma base chamada Domínio do Plano, que representa as informações representadas pelo *KPlanOO* que servirá tanto para alimentar o editor de regras como mostrado na seção anterior, quanto como domínio para um módulo Modelador de Subsistemas e obviamente para o Planejador de Operações.

Nas camadas Manutenção e Serviço é apresentado um construtor de domínios, indicado pela cor vermelha, que irá ser uma interface para captação de dados dos domínios de planejamento baseados no *KPlanOO*, que neste trabalho foi apresentado o um protótipo de interfaces gráficas ou arquivos XML´s baseados no XKPS.

Conforme indica a Figura 6.27, um dos objetivos do projeto de integração é modelar o domínio de planejamento de operações, através do *KPlanOO*, incluindo aspectos de modelos de sistemas utilizados pelo simulador de satélite e pelo gerador de diagnóstico. Isso possibilita o compartilhamento de informações entre as diferentes bases de conhecimento. Em termos práticos do ponto de vista do simulador de satélite, a reconfiguração do modelo poderia ser feita através de realimentação de telemetrias obtidas do satélite e resultados de diagnósticos. Como um dos objetivos do projeto é a automação do sistema de planejamento, a sua implementação poderia significar a eliminação da necessidade de configuração manual de regras, ficando esta tarefa a cargo de agentes inteligentes responsáveis pela manutenção do sistema.

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1. Conclusões

Neste trabalho foi apresentado um meta-modelo, nomeado como *KPlanOO,* como ferramenta para o processo de modelagem de domínios e problemas de planejamento automático. Este meta-modelo atende os requisitos de modelagem destes domínios baseado em conceitos de áreas como Engenharia do Conhecimento, Análise de Domínios, Ontologia, Orientação Objetos, visando facilitar o trabalho do projetista com tais domínios.

O presente trabalho apresentou também um protótipo para captação de dados para o *KPlanOO*, implementado em Java, construído de forma a possibilitar alterações, extensões e novas implementações. Foi proposto um XML *Schema*, identificado como XKPS, baseado no *KPlanOO,* para descrição de domínios e problemas de planejamento em XML, facilitando a compreensibilidade de documentos de especificação destes domínios, mas também a sua validação e planejamento.

Muitos pesquisadores utilizam a PDDL como linguagem inicial para a representação dos domínios de planejamento, mas esta linguagem além de impor algumas restrições vem se mostrando de difícil compreensão, tornando o trabalho de modelagem muito exaustivo e muitas vezes inviável para domínios com níveis elevados de complexidade, ou seja, domínio reais de planejamento. Diante disso, este trabalho propos o uso de uma abordagem de modelagem mais amigável para auxiliar o projetista na realização de processos disciplinados de modelagem e análise de domínios reais de planejamento.

Do estudo realizado, as maiores contribuições são:

a) a independência do meta-modelo com diferentes abordagens de desenvolvimento e plataformas de implementação, possibilitando criar

modelos de planejamento a partir de qualquer das abordagens estudadas;

b) mapear os modelos de planejamento criados para os conceitos e relacionamentos do meta-modelo e traduzir esses de maneira automática para código fonte de alguma das plataformas de implementação estudadas utilizando a arquitetura proposta no protótipo criado;

c) a aplicação do modelo de restrições de integridade, garantindo a consistência dos modelos com o meta-modelo proposto;

d) e o protótipo extensível, que possibilita a geração de código em diferentes plataformas de implementação mediante os passos apresentados na seção 5.2.6.

Os estudos de casos apresentados neste trabalho mostram o potencial do meta-modelo, nos processos de modelagem de domínios de planejamento. De fato, após realizar todo o processo de modelagem o projetista possui um modelo mais claro baseado nos conceitos inerentes a área de planejamento automático com grandes possibilidades de reutilização (devido, entre outros fatores, a abordagem orientada a objetos). Com os modelos resultantes o projetista consegue entender e verificar o domínio sem profundos conhecimentos em uma linguagem especifica.

Além da ferramenta desenvolvida, o presente trabalho gerou produções acadêmicas relevantes para a área de pesquisa de Planejamento Automático e principalmente para a área da Engenharia do Conhecimento e Ontologia. As publicação, tanto internacionais quanto nacionais, resultantes deste trabalho (SILVA, R. R, et. al, 2008) (SILVA, R. R, et. al, 2009) (SILVA, R. R, et. al, 2009-A) (SILVA, R. R, et. al, 2010) (KONO et al., 2010), demonstram o interesse,

pelo mesmo, da comunidade de pesquisa na área espacial e de engenharia de conhecimento.

Este trabalho traz para a comunidade da área de planejamento automático e para a área espacial um primeiro estudo da aplicação da união de técnicas de OO, Ontologias, Engenharia do Conhecimento e Análise de Domínios à descrição de domínios de planejamento automático. Com isso ele abre uma porta para a aplicação de outras técnicas e abordagens no processo de planejamento de atividades, em atividades de controle e simulação de satélites.

Durante todo o desenvolvimento dessa dissertação foram realizadas várias pesquisas bibliograficas, não tendo sido encontrado na literatura analisada nenhum trabalho que reunisse informações sobre os meta-modelos genéricos para representação de domínios e problemas de planejamento e também nenhuma forma de representação que contemplasse todas as necessidades da área espacial. Todos os dados encontrados vieram de artigos e livros descrevendo abordagens específicas, contendo no máximo comparações pontuais com uma ou duas outras formas de representação.

Para os pesquisadores da área de planejamento automático na área espacial, a proposta desse trabalho traz uma nova forma de representar propriedades, estados, eventos e consumo de recursos, adequada para uso no segmento espacial.

Para a comunidade de planejamento automático, este trabalho representa uma contribuição alternativa e flexível para a representação de problemas de planejamento, O *KPlanOO* possui uma série de características bastante diferentes do que é comumente visto em sistemas baseados nos conceitos da STRIPS e da PDDL, o que possibilita que a representação do conhecimento seja menos tediosa, restrita e trabalhosa para domínios desta área.

Na Figura 7.1, é apresentado uma linha de tempo com as principais formas de representação do conhecimento para domínios de planejamento automático. O *KPlanOO* pode ser inserido nesta linha, sendo visto como a primeira iniciativa de se representar domínios de planejamento automático através de um meta-modelo OO que supre as atuais necessidades de representação de domínios da área espacial.

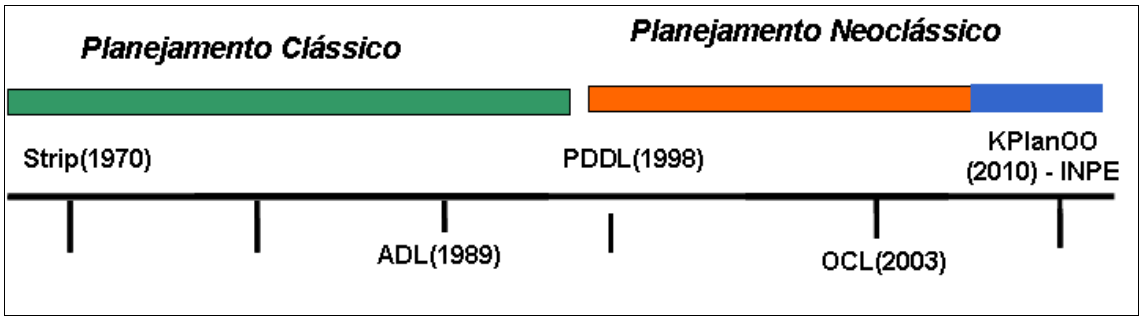

Figura 7.1 - Linha de tempo com principais formas de representações do conhecimento para domínios de planejamento automático

7.1.1. Pontos desfavoráveis

Neste trabalho não foi levado em consideração por questão de escopo questões como: balanceamento de informação, propriedades para escalonamento no processo de planejamento, semântica operacional das entidades, processo de levantamento de requisitos e mapeamento dos mesmos para o modelo proposto. Também não foi criado delimitador de representações do modelo para outras linguagens.

No próximo tópico são discutidos algumas possibilidades de trabalhos futuros, evidenciados durante a elaboração dos conceitos apresentados neste trabalho, que podem de alguma forma potencializar o presente projeto, bem como sua linha de pesquisa.

7.2. Trabalhos futuros

Acredita-se que os trabalhos que poderão ser desenvolvidos futuramente podem agregar valor ao trabalho apresentado e, conseqüentemente, potencializar a área de pesquisa de Planejamento Automático aplicado a área espacial não só no cenário nacional, mas também no cenário internacional.

Alguns trabalhos que poderão ser desenvolvidos como seqüência ao aqui apresentado são:

1. Estudo de outros diagramas da UML para identificar como estes podem contribuir no processo de modelagem dos domínios de planejamento;

2. Realização de estudos de casos com diferentes domínios onde os níveis de complexidade possam ser maiores se comparados com os domínios aqui apresentados, para conceber a evolução do meta-modelo com novas abstrações, relaxamento ou aumento de restrições;

3. A construção de uma sintaxe que represente o meta-modelo.

4. Uma especialização do modelo para diferentes áreas, nomeando as entidades com nomes específicos de cada área.

5. Refatoração do protótipo para utilizar termos específicos de diferentes áreas.

6. A integração do protótipo criado com um planejador automático. Neste caso o modelo representado com o *KPlanOO* deveria ser transformado em código PDDL pois os planejadores mais clássicos hoje trabalham com essa representação.

7. O desenvolvimento de um planejador orientado a objetos que manipulasse diretamente as entidades do meta-modelo proposto.

8. Desenvolvimento de um processo de levantamento de requisitos e mapeamento semântico para o *KPlanOO*.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBERTS L. K. YMIR: a sharable ontology for the formal representation of engineering design knowledge. In: GERO J.S.; TYUGU E. **Formal design methods for computer-aided design**. Amsterdam: Elsevier, 1994. p. 3-32. (Ifip Transactions B-Applications in Technology).

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

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

BACCHUS, F. The power of modeling - a response to PDDL2.1 - Commentary. **Journal of Artificial Intelligence Research,** v. 20, p. 125-132, 2003.

BARRET, A.; WELD, D. **Partial order planning**: evaluation possible efficiency gains. Washington: Dept. of Computer Science and Engineering, University of Washington, 1992. Technical Report 92-05-01.

BATEMAN, J. A.; MAGNINI, B., RINALDI, F. The generalized italian, german, english upper mode., In: WORKSHOP COMPARISON OF IMPLEMENTED ONTOLOGIES, 1994. Amsterdam, Nederland. **Proceedings...** Amsterdam: N.J.I. Mars, 1994.

BLUM, A.; FURST, M. Fast planning through planning graph analysis. **Artificial Intelligence**, v. 90, p. 281-300, 1997.

BODDY, M. Imperfection match: PDDL2.1 and real applications, **Journal of Artificial Intelligence Research** v. 20, p. 133-137, 2003.

BONET, B.; GEFFNER, H. Planning as heuristic search: new results. In: EUROPEAN CONFERENCE ON PLANNING - ECP'99, 1999.
Durham, UK., 1999. **Proceedings...** Durham: Springer, 1999.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling language user guide**. Massachusetts: Addison-Wesley, 1999.

BOORSTIN, D.J. **Os investigadores** – a história da permanente busca do homem para compreender o seu mundo. Rio de Janeiro: Civilização Brasileira, 2003.

CARDOSO, L. S.; FERREIRA, M. G. V.; ORLANDO, V.; BIANCHO, A. C. Aplicação da tecnologia de agentes de planejamento em operações de satélites. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 7TH (SBAI), 2005, São Luiz, Maranhão.**Anais...** 2005. p. 07. Papel. (INPE-14070-PRE/9239). Disponível em: <http://urlib.net/sid.inpe.br/mtc-m16@80/2006/08.21.20.19>. Acesso em: 13 abr. 2010.

CARNIELLO, A. **Uma arquitetura multi-agente de planejamento de controle de satélites**. 2008. 220 p. (INPE-15666-TDI/1442). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 2009. Disponível em: <HTTP://URLIB.NET/SID.INPE.BR/MTC-M18@80/2008/11.17.15.15>. Acesso em: 05 jun. 2010.

CHAPMAN, D. Planning for conjunctive goals. **Artificial Intelligence**, v. 32, n.1 p. 333-377, 1987.

Clancey W.J. The knowledge level reinterpreted - modelling socio-technical systems. **International Journal of Intelligent Systems**, v. 8, n.1, p. 33-49. 1993

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

D'SOUZA, F.D.; WILLS, A.C. **Object, components, and frameworks with uml** – the catalysis aApproach. Addison-Wesley, United States of America. 1999.

DAHLGREN, K. A linguistic ontology. **Int. J. Hum.-Comput. Stud**. v. 43, n. 5-6, p. 809-818, 1995.

Document Object Model (DOM). **W3C.** Disponível em: http://www.w3.org/TR/DOM/ . Acesso em 15/12/2009.

DOUGLASS, B. P. **Real-time uml:** developing efficient objects for embedded systems. [S.l.]: Addison-Wesley, 1998.

EDELKAMP, S.; HOFFMANN J. **PDDL 2.2:** The language for classical part of the 4th International Planning Competition. Freiburg: Fachbereich Informatik and Institut für Informatik, 2004. Technical Report.

ERNST, G.; NEWELL, A. **GPS:** a case study in generality and problem solving. New York: Academic Press. 1969.

EROL, K.; HENDLER, J.; NAU, D.S. HTN planning: complexity and expressivity. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE – AAAI-94, 1994, Seattle. **Proceedings…** Seattle: AAAI, 19994. p. 1123-1128.

FALBO R. A. **Integração de conhecimento em um ambiente de engenharia de software.** 1998. Tese (Doutorado em Informática) - Engenharia de Sistemas e Computação, COPPE, Universidade Federal do Rio de Janeiro.

FARIA, C, G. **Uma técnica para a aquisição e construção de modelos de domínio e usuários baseados em ontologias para a engenharia de domínio multiagente**. Dissertação (Mestrado) – Centro Universitário do Maranhão, 2000.

FEBER, J. **Multi-agent systems an introduction to distributed artificial intelligence**. Oxford: Addison-Wesley, London, p. 528. 1999.

FIKES, R. E.; NILSSON N. J.; STRIPS: A new approach to the application of theorem proving to problem solving. **Artificial Intelligence**, v. 2 p.189-208. 1971

FONSECA, F.; EGENHOFER, M.; BORGES, K. Ontologias e interoperabilidade semântica entre SIGs. In: WORKSHOP BRASILEIRO EM GEOINFORMÁTICA - GEOINFO, 2., São Paulo. **Anais…** São Paulo: SBC, 2000

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.

FUKUNAGA, A.; RABIDEAU, G.; CHIEN, S.; YAN, D. Towards an application framework for automated planning and scheduling. In: INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE ROBOTICS AND AUTOMATION IN SPACE, 1997, Tokyo, Japan. **Proceedings…** Tokyo: [s.n.], July 1997.

GAMMA E.; HELM R.; JOHNSON R.; VLISSIDES J. **Design patterns:** Elements of reusable object-oriented software, 1.ed. Oregon: Addison-Wesley, 1995. ISBN 0-201-63361-2

GEFFNER, H. PDDL2.1: Representation vs. Computation, **Journal of Artificial Intelligence Research** v. 20, p.139-140, 2003.

GEREVINI, A.; LONG, D. **Plan constraints and preferences in pddl3 – the language of fifth international planning competition.** Brecia: Department of Electronics for Automation, University of Brescia, Italy, Aug., 2005. Technical Report.

GHALLAB M.; NAU D.; TRAVERSO. P. **Automated planning**: theory and practice. California: Morgan Kaufmann. 2004.

GIL, Y.; VELOSO, M.; CHIEN, S.A.; MCDERMOTT, D.; NAU, D. Planning and learning: on to real applications. In: FALL SYMPOSIUM, 1994, Menlo Park. **Papers…** Melo Park: AAAI Press, American Association for Artificial Intelligence. Papers from 1994 AAAI Fall Symposium, number FS-94-01. 1995. ISBN 0-929280-75-X.

GÓMEZ-PÉREZ, A.; BENJAMINS, V.R**.** Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI-99). WORKSHOP ON ONTOLOGIES AND PROBLEM-SOLVING METHODS (KRR5), 16., 1999, Stockholm, Sweden. **Proceedings…** Stockolm: [s.n.], 1999.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ, M.; VICENTE, A. J. Towards a method to conceptualize domain ontologies. In: ECAI'96 - WORKSHOP ON ONTOLOGICAL ENGINEERING, 1996, Budapest. **Proceedings...** Budapest: [s.n.], 1996.

GÓMEZ-SANZ, J.J. **Meta-modelos para el desarrollo de sistemas multi-agente**. Disponível no site [http://grasia.fdi.ucm.es/SP/publicaciones/index.html] Acesso em 01/2010.

GONÇALVES, L. S. C. **Aplicação da tecnologia de agentes de planejamento em operações de satélites**. 2006. 167 p. (INPE-14092-TDI/1075). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 2006. Disponível em: <HTTP://URLIB.NET/SID.INPE.BR/MTC-M13@80/2006/05.04.17.36>. Acesso em: 05 jun. 2010.

GREEN, C. Application of theorem proving to problem solving. In: international joint conference on artificial intelligence - IJCAI-69, 1., 1969, Menlo Park, CA. **Proceedings…** Menlo Park, CA: [s.n.], 1969. p.219-239.

GRUBER, T.R. Toward principles for the design of ontologies used for knowledge sharing. **International Journal of Human-Computer Studies (IJHCS)**, v. 43, n. 5/6, p. 907 928, 1995.

GRUNINGER M. **Integrated ontologies for enterprise modelling**. Disponível em: http://www.ie.utoronto.ca/EIL/tove/ontoTOC.html . Acesso em 15 de dez. 2009.

GUARINO, N. Understanding, building and using ontologies. **Int. Journal Human-Computer Studies**, v. 45, n. 2/3. 1997.

GUARINO, N. Formal ontologies and information systems. In: INTERNATIONAL CONFERENCE (FOIS), 1., 1998, Trento, Itália.

Proceedings... Trento: IOS Press, 1998. Disponível em: http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/FOIS98.pdf.

GUIZZARDI, G. **Desenvolvimento para e com reuso:** um estudo de caso no domínio de vídeo sob demanda, 2000. Tese Doutorado em em Ciência da Computação, Universidade Federal do Espirito Santo.

GUIZZARD G. **Uma Abordagem metodológica de desenvolvimento para e com reuso, baseada em ontologias formais de domínio**. Dissertação de Mestrado. Universidade Federal do Espírito Santo, 2000.

HELMERT, M. A planning heuristic based on causal graph analysis. international conference on automated planning and scheduling (ICAPS-04),14., 2004, Whistler, Canada. **Proceedings...** Canada: Morgan Kaufmann. 2004. p.161-170.

HOFFMANN, J.; NEBEL, B. The FF planning system: fast plan generation through heuristic search. **Journal of Artificial Intelligence Research**, v.14, p. 253-302, 2001.

HOFFMANN, J. Extending FF to numerical state variables. In: EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (ECAI-02), 15., 2002, Lyon. **Proceedings…** Lyon, France: [s.n.], 2002. p. 571-575.

HURLBLUT, R. **A survey on approaches for describing and formalizing use-cases.** 1997. Disponível na Internet em HTTP://WWW.IIT.EDU/~RHURLBUT/XPT-TR-97-03.HTML,. Acesso em 15/12/2009.

JOSLIN, D.; POLLACK, M. E. Is 'early commitment' in plan generation ever a good idea? In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 13., 1996, Portland. **Proceedings…** Menlo Park: AAAI, 1996. p. 1188-1193.

KAUTZ, H.; SELMAN, B. Pushing the envelope: planning, propositional logic, and stochastic search. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 13., 1996, Portland. **Proceedings…** Menlo Park: AAAI, 1996. 1194-1201.

KAUTZ, H.; SELMAN, B. Unifying SAT-based and graph-based planners. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE - IJCAI-99, 15.,1999, Stockholm, Sweden. **Proceedings…** Stockholm: [s.n.], 1999.

KHATIB, L.; FRANK, J.; SMITH, D.; MORRIS, R.; DUNGAN, J.; **Interleaved Observation Execution and Rescheduling on Earth Observing Systems.** In Proc. of the ICAPS-03 Workshop on Plan Execution, Trento, Italy, 2003.

KLEIN, M. Combining and relating ontologies: An analysis of problems and solutions. In: WORKSHOP ON ONTOLOGIES AND INFORMATION SHARING AT THE 17TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2001, Seattle, USA. **Proceedings…**Seattle: [s.n], p. 53-62.

KONO, Y.; FERREIRA, M. G. V.; SARAIVA JÚNIOR. F. E. G.; PEREIRA, L. M.; **Strategies for implementation of an automated planning system.** SPACEOPS 2010, Huntsville, USA: AIAA, April 2010.

KOWALSKI, R. A.; SERGOT, M. A logic-based calculus of events. **New Generation Computing**. v.4 p. 67-95. 1969

KUCINSKIS, F. N. **Alocação dinâmica de recursos computacionais para experimentos científicos com replanejamento automatizado a bordo de satélites**. 2007. 165 p. (INPE-14798-TDI/1241). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 2007. Disponível em: <HTTP://URLIB.NET/SID.INPE.BR/MTC-M17@80/2007/04.23.11.42>. Acesso em: 05 jun. 2010.

KUCINSKIS, F. M.; FERREIRA, M. G. V. Taking the ECSS autonomy concepts one step further. In: SPACEOPS 2010, 2010, Huntsville. **Proceedings…** Huntsville: [s.n], 2010.

LENAT, D. B. CYC: a large-scale investment in knowledge infrastructure. **Communications of the ACM**, v. 38, n. 11, p. 33-38, 1995.

LIU, D.; MCCLUSKEY, T. L. **The OCL language manual, version 1.2**. Huddersfield: University of Huddersfield (UK), Department of Computing and Mathematical Sciences, 2000. Technical Report,

MANGAN, M. A. S.; MURTA, L. G. P.; SOUZA, J. M.; WERNER, C. M. L., Modelos de domínio e ontologias: uma comparação através de um estudo de caso prático em hidrologia. In: INTERNATIONAL SYMPOSIUM ON KNOWLEDGE MANAGEMENT/DOCUMENT MANAGEMENT, 4., 2001, Curitiba, PR, Brasil. **Anais…** Curitiba: [s.n.], 2001. p.149-172.

MCALLESTER, D.; ROSENBLITT, D. **systematic nonlinear planning,** New York: American Association for Artificial Intelligence – AAI, 1991.

MCCLUSKEY T. L.; PORTEOUS, J. M. Two complementary techniques in knowledge compilation for planning. In: WORKSHOP ON KNOWLEDGE COMPILATION AND SPEEDUP LEARNING, ML'93, 1993, **Proceedings…** Massechussettes: University Press , 1993.

MCCLUSKEY T. L.; PORTEOUS, L. M.; NAIK, J.; TAYLOR, C. N.; JONES S.; An requirements capture method and its use in air traffic control application. **Software-Practice and Experience**, v. 25 p. 47-41. 1995.

MCCLUSKEY, T. L. Knowledge engineering: issues for the ai planning community. In: AIPS-2002 WORKSHOP ON KNOWLEDGE ENGINEERING TOOLS AND TECHNIQUES FOR AI PLANNING, 2002, Toulouse, France. **Proceedings…** Toulouse: [s.n], 2002.

MCCLUSKEY, T. L. PDDL2.1: a language with a purpose?,In: ICAPS 03 – Workshop on PDDL, 2003, Trento. **Proceedings…** Trento: [s.n], 2003.

MCCLUSKEY, T. L.; ALER, R.; BORRAJO, D.; HASLUM, P.; JARVIS, P.; I.REFANIDIS ; SCHOLZ, U. **Knowledge engineering for planning roadmap.** 2003, Available on: http://scom.hud.ac.uk/planet/.

MCDERMOTT, D.; HENDLER, J. Planning: what it is, what it could be - an introduction to the special issue on planning and scheduling**. Artificial Intelligence**, v. 76 p.1-16. 1995.

MCDERMOTT, D. **The PDDL Planning Domain Definition Language**. The AIPS-98 Planning Competition Committe. [S.l.],1998.

MCDERMOTT, D. PDDL2.1 - The Art of the possible?, commentary on fox and long. **Journal of Artificial Intelligence Research,** v. 20, p.145 – 148, 2003.

MILLER, G. A. WORDNET: a on-line lexical database. **International Journal of Lexicography**, n. 3/4, 1990.

S. MINTON. **Learning effective search control knowledge:** an explanation-based approach. PhD thesis, Carnegie-Mellon University, Pittsburgh, PA, 1988.

MUSCETTOLA, N.; NAYAK, P. P.; PELL, B.; and WILLIAMS, B. C. Remote agent: to boldly go where no ai system has gone before. **Artificial Intelligence**, v.103. n.1-2. p. 5-48. 1998.

MYERS, K.; WILKINS, D. **The act-editor user's guide:** a manual for version2.2. Menlo Park: SRI International, Artificial Intelligence Center. 1997.

NEIGHBORS J. **Software construction using components.** Tese (Doutorado em ciência da Computação) - Universidade da Califórnia, Irvine, EUA, 1981.

NEWELL, A.; SIMON, H. **Human problem solving.** Eglewood Cliffs, NJ: Prentice-Hall. 1972.

NOY, N.F.; MCGUINNESS, D.L. **Ontology development 101:** a guide to creating your first ontology. Stanford: Stanford Knowledge Systems Laboratory, (Relatório Técnico KSL-01-05). 2001.

OBJECT MANAGEMENT GROUP (OMG). **Unified modeling language specification**: version 2.0, URL http://www.omg.org/uml. 2009.

ORLANDO, V.; KUGA, H. K. Os satélites SCD1 e SCD2 da Missão Espacial Completa Brasileira - MECB. In: Winter, Prado, O. C.; A. F. B. A. (Orgs.). **A conquista do espaço** - Do Sputnik à Missão Centenário. 1 ed. São Paulo: Editora Livraria da Física, 2007, v. 1, p. 151-176.

PEDNAULT, E. ADL: Exploring the middle ground between STRIPS and situation calculus. In: INT. CONF. ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING, 1., 1989, Toronto. **Proceedings…** Toronto: [s.n], 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. **Proceedings…** Boston, MA: Morgan Kaufmann, 1992. p. 103-114.

RUMBAUGH, J.R.; BLAHA, M.R.; LORENSEN, W. **Object-oriented modeling and design.** Upper Saddle River: Prentice-Hall, 1991.

RUSSEL, S.; NORVING, P. First-Order Logic. In:____ (Ed.). **Artificial intelligence:** a modern approach, New Jersey: Prentice-Hall, 1995. Chapter 8, p. 285-315

SACERDOTI, E. Planning in a hierarchy of abstraction spaces. **Artificial Intelligence**, v.5 p.115-135. 1974.

SACERDOTI, E. **A Structure for plans and behaviors**. New York: North Holland. 1977.

SAXON - **The XSLT and Xquery processor**. Available on: http://saxon.sourceforge.net/ , acessado em 15/12/2009.

SILVA, R. R. ; VIEIRA, M. F. G. ; VIJAYKUMAR, N. L. Descrevendo domínios e problemas de planejamento através de um modelo orientado a objetos. In: FÓRUM DE INTELIGÊNCIA ARTIFICIAL DA REGIÃO SUL, 4., 2008, Canoas, RS. **Anais...** Canoas: Unisinos, 2008.

SILVA, R. R. ; VIEIRA, M. F. G. ; VIJAYKUMAR, N. L. . Uma ontologia baseada em um meta-modelo orientado a objetos para descrição de domínios e problemas de planejamento da área espacial. In: Seminário de Pesquisa em

Ontologia do Brasil, 2., 2009, Rio de Janeiro. **Anais...** Rio de Janeiro: [s.n], 2009.

SILVA, R. R.; FERREIRA, M. G. V. ; VIJAYKUMAR, N. L. An object-oriented meta-model as ontology for describing domains and problems for planning space applications. In: SPACEOPS 2010, 2010, Huntsville. **Proceedings…** Hunstiville: [s.n.], 2010.

SILVA, R. P. **Avaliação de metodologias de análise e projeto orientadas a objetos voltadas ao desenvolvimento de aplicações, sob a ótica de sua utilização no desenvolvimento de frameworks orientado a objetos.** Porto Alegre: CPGCC da UFRGS, 1996.

SIMPSON, R.M,. MCCLUSKEY, T. L.; ZHAO, W.; AYLETT, R. S.; DONIAT C. GIPO: an integrated graphical tool to support knowledge engineering in AI Planning. In: EUROPEAN CONFERENCE ON PLANNING, 2001, Toledo, Spain. **Proceedings…** Toledo, Spain: [s.n], 2001.

SMITH, D. The case for durative actins: a commentary on PDDL 2.1. **Journal of Artificial Intelligence Research,** v. 20, p.149–154, 2003.

SOUZA, J. F.; SIQUEIRA, S. W. M.; AZEVEDO, L. G.; BAIAO, F.; LOPES, M.; SANTORO, F. M.; CAPPELLI, C.; NUNES, V.; MAGDALENO, A. **Gestão de ontologias**. Rio de Janeiro: Departamento de Informática Aplicada da UNIRIO 2008.

SOUZA, P. B.; FERREIRA, M. G. V.; SILVA, J. D. S.; AMBROSIO, A. M. Decision Support Tool for Prediction of Critical Data to the Satellite Integrity. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE, 9. (WORCAP), 2009, São José dos Campos. **Anais...** São José dos Campos: INPE, 2009. On-line. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m18@80/2010/05.28.16.58 <http://urlib.net/rep/sid.inpe.br/mtc-m18@80/2010/05.28.16.58?languagebutton=pt-BR>>. Acesso em: 11 jun. 2010.

SOWA, J. F. Sowa: top-level ontological categories. **Int. J. Hum.-Comput. Stud.** v.43, n.5-6. p. 669-685, 1995.

Sowa, J. F. **Relating templates to language and logic. I**n Pazienza p. 76-94, 1999.

STEFICK, M. Planning with constraints. **Artificial Intelligence**, v.16 p.111-140. 1981.

STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: principles and methods. **IEEE Transactions on Data and Knowledge Engineering,** v.25 p.161-97. 1998.

SwingX The enhanced Swing components. Disponível em: <http://www.swinglabs.org/. Acesso em: 15/15/2009.

TATE, A. Generalizing project networks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE - IJCAI-77, 5., Cambridge, MA, 1977. **Proceedings…** Cambridge: [s.n], 1977. p. 888-893.

TATE, A.; DRABBLE, B.; DALTON, J. **O-Plan:** a knowledge-based planner and its application to logistics. Edinburgh: University of Edinburgh, 1996.

TATE, A.; POLYAK, S. T.; Jarvis, P. **TF Method:** an initial framework for modeling and analyzing planning domains. Edinburg: University of Edinburgh. 1998. Technical Report

TKACH, D.; PUTTICK, R. **Object technology in application development**, [S.l.]: The Benjamin/Cummings, 1994.

TOMINAGA, J.; FERREIRA, M. G. V.; SILVA, J. D. S. A proposal for implementing automation in satellite control planning, In: SpaceOps 2008, Heidelberg, Germany. **Proceedings…** Heidelberg: AIAA, 2008.

TOMINAGA, J.; FERREIRA, M. G. V.; SILVA, J. D. S. An implementation of a satellite simulator as a fuzzy rule-based inference machine. In: WORCAP 2009, 2009, São José dos Campos, Brazil. **Proceedings...** São José dos Campos: INPE, 2009.

TOMINAGA, J. **Simulador de satélites para verificação de planos de operações em vôo**. 2010. xx + 152 p. (INPE--T/). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 2010. Disponível em: <HTTP://URLIB.NET/SID.INPE.BR/MTC-M19@80/2010/05.24.18.55>. Acesso em: 05 jun. 2010.

USCHOLD, M.; GRUNINGER M. Ontologies: principles, methods and applications. **The Knowledge Engineering Review**, v. 11, n. 2, p. 93-136, 1996.

VAQUERO, T. S. **itSIMPLE: Ambiente integrado de modelagem e análise de domínios de planejamento automático**. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

VAREJÃO F. M. **DORPA:** uma ontologia de design que integra requisitos, artefatos e processos. Tese (Doutorado em Informática) – Pontifícia Universidade Católica, Rio de Janeiro, 1999.

WAH, B. W.; CHEN Y. Subgoal partitioning and global search for solving temporal planning problems in mixed space. **Int'l J. of Artificial Intelligence Tools**, v. 13, n. 4, p. 767-790. 2004.

WILKINS, D. E. Representation in a domain-independent planner. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI-83), 8., 1983, Karlsruhe. **Proceedings…** Karlsruhe: [s.n], 1983.

WILKINS, D., Domain-independent planning: representation and plan generation, **Artificial Intelligence**, v.22 p. 269-301.1984.

WILKINS, D., Using the SIPE-2 Planning System: A Manual for SIPE-2, Version5.0, SRI International, Artificial Intelligence Center. 1999.

WINANDY, C –E. ; FERREIRA, M. G. V. Development process for applications of automated planning for satellites control. In: SPACEOPS 2010, 2010, HunsVille. **Proceedings…** Hunstville: [s.n.], 2010.

XERCES, http://xerces.apache.org/, acessado em 15/12/2009.

XML Schema, **W3C**. Available on: http://www.w3.org/xml/schema. Access on: 15/12/2009.

XML:**Parser.** Available on: http://search.cpan.org/dist/xml-parser/ Access on: 15/12/2009.

XML:**Simple**. Available on: http://search.cpan.org/dist/xml-simple/ Access on: 15/12/2009.

THE EXTENSIBLE STYLESHEET LANGUAGE FAMILY (XSL). **W3C**. Available on: http://www.w3.org/style/xsl/ Access om : 15/12/2009.

Xstream. Disponível em: <http://xstream.codehaus.org/>. Acesso em: 20/11/2009

YANG, Q.;TENENBERG, J. ABTWEAK: abstract a nonlinear, least commitment planner. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE - AAAI-90, 1990, Boston. **Proceedings…** Boston: [s.n.], 1990. p. 204-209.

ZLOT, F.; OLIVEIRA, K.M.; ROCHA. A.R. Modeling Task Knowledge to Support Software Development. In: INTERNATIONAL CONFERENCE ON SOFTWARE

ENGINEERING AND KNOWLEDGE ENGINEERING, SEKE'02, 14., Ischia,Itália, 2002. **Proceedings…** Ischia: ACM, 2002.

ZWEBEN, B.; DAVIS, M.; DAUN, E.; DEALE, M. Scheduling and rescheduling with iterative repair. **IEEE Transactions on Systems, Man, and Cybernetics**. v. 23, n. 6. 1993.

APÊNDICE A – DEFINIÇÃO EM XML DO DOMÍNIO DO MUNDO DE BLOCOS BASEADA NO XKPS

```
<?xml version="1.0" encoding="UTF-8"?>
<kpo:KPlannOO xmlns:kpo="http://www.rrochas.com/kplanoo/xkps"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.rrochas.com/kplanoo/xkps KPlanOOSchemaDomain.xsd">

    <kpo:domain>
        <kpo:definition>blocks</kpo:definition>

        <kpo:types>
            <kpo:type name="block" />
        </kpo:types>

        <kpo:states>
            <kpo:state definition="on">
                <kpo:parameters>
                    <kpo:element name="x" type="block"/>
                    <kpo:element name="y" type="block"/>
                </kpo:parameters>
            </kpo:state>

            <kpo:state definition="ontable">
                <kpo:parameters>
                    <kpo:element name="x" type="block"/>
                </kpo:parameters>
            </kpo:state>

            <kpo:state definition="clear">
                <kpo:parameters>
                    <kpo:element name="x" type="block"/>
                </kpo:parameters>
            </kpo:state>

            <kpo:state definition="handempty"/>

            <kpo:state definition="holding">
                <kpo:parameters>
                    <kpo:element name="x" type="block"/>
                </kpo:parameters>
            </kpo:state>
        </kpo:states>

        <kpo:actions>

            <kpo:action name="pick-up">
                <kpo:parameters>
                    <kpo:element name="x" type="block"/>
                </kpo:parameters>

                <kpo:precondition>
                    <kpo:scenario>
                        <kpo:situations>

                            <kpo:situationobj>
                                <state definition="clear">
                                    <kpo:parameters>
                                        <kpo:element name="x"/>
                                    </kpo:parameters>
                                </state>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <state definition="ontable">
                                    <kpo:parameters>
                                        <kpo:element name="x"/>
                                    </kpo:parameters>
```

```
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="handempty"/>
                </kpo:situationobj>
            </kpo:situations>

        </kpo:scenario>
    </kpo:precondition>

    <kpo:effect>
        <kpo:scenario>
            <kpo:situations>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <state definition="ontable">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <state definition="clear">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <state definition="handempty"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="holding">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

            </kpo:situations>
        </kpo:scenario>
    </kpo:effect>

</kpo:action>

<kpo:action name="put-down">
    <kpo:parameters>
        <kpo:element name="x" type="block"/>
    </kpo:parameters>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situations>
                <kpo:situationobj>
                    <state definition="holding">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>
            </kpo:situations>
        </kpo:scenario>
    </kpo:precondition>
```

```
    <kpo:effect>
        <kpo:scenario>
            <kpo:situations>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <state definition="holding">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="clear">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="handempty"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="ontable">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

            </kpo:situations>
        </kpo:scenario>
    </kpo:effect>

</kpo:action>

<kpo:action name="stack">
    <kpo:parameters>
        <kpo:element name="x" type="block"/>
        <kpo:element name="y" type="block"/>
    </kpo:parameters>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situations>

                <kpo:situationobj>
                    <kpo:situationFather>
                        <expressionEvaluation name="not"/>
                    </kpo:situationFather>

                    <expressionEvaluation symbol="=">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                            <kpo:element name="y" type="block"/>
                        </kpo:parameters>
                    </expressionEvaluation>

                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="holding">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>
```

```
                <kpo:situationobj>
                    <state definition="clear">
                        <kpo:parameters>
                            <kpo:element name="y"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

            </kpo:situations>
        </kpo:scenario>
    </kpo:precondition>

    <kpo:effect>
        <kpo:scenario>
            <kpo:situations>
                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <state definition="holding">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <state definition="clear">
                        <kpo:parameters>
                            <kpo:element name="y"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="handempty"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <state definition="on">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                            <kpo:element name="y"/>
                        </kpo:parameters>
                    </state>
                </kpo:situationobj>
            </kpo:situations>

        </kpo:scenario>
    </kpo:effect>

</kpo:action>

<kpo:action name="unstack">
    <kpo:parameters>
        <kpo:element name="x" type="block"/>
        <kpo:element name="y" type="block"/>
    </kpo:parameters>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situations>

                <kpo:situationobj>
                    <kpo:situationFather>
                        <expressionEvaluation name="not"/>
                    </kpo:situationFather>

                    <expressionEvaluation symbol="=">
                        <kpo:parameters>
                            <kpo:element name="x"/>
                            <kpo:element name="y" type="block"/>
```

```
                    </kpo:parameters>
                </expressionEvaluation>
            </kpo:situationobj>

            <kpo:situationobj>
                <state definition="on">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                        <kpo:element name="y"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>

            <kpo:situationobj>
                <state definition="clear">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>

            <kpo:situationobj>
                <state definition="handempty"/>
            </kpo:situationobj>

        </kpo:situations>
    </kpo:scenario>
</kpo:precondition>

<kpo:effect>
    <kpo:scenario>
        <kpo:situations>

            <kpo:situationobj>
                <state definition="holding">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>

            <kpo:situationobj>
                <state definition="clear">
                    <kpo:parameters>
                        <kpo:element name="y"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="not"/>
                <state definition="clear">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                    </kpo:parameters>
                </state>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="not"/>
                <state definition="handempty"/>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="not"/>
                <state definition="on">
                    <kpo:parameters>
                        <kpo:element name="x"/>
                        <kpo:element name="y"/>
                    </kpo:parameters>
                </state>
```

```
                        </kpo:situationobj>

                    </kpo:situations>
                </kpo:scenario>
            </kpo:effect>

        </kpo:action>
    </kpo:actions>
</kpo:domain>
</kpo:KPlannOO>
```

APÊNDICE B – DESCRIÇÃO XML DO DOMÍNIO DE RASTREIO DE SATÉLITES DO INPE

```
<?xml version="1.0" encoding="UTF-8"?>
<kpo:KPlannOO xmlns:kpo="http://www.rrochas.com/kplanoo/xkps"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.rrochas.com/kplanoo/xkps KPlanOOSchemaDomain.xsd">

    <domain>
        <kpo:definition>POVMEDZSRRTC</kpo:definition>
        <kpo:types class="list">
            <kpo:type name="PassNumber"/>
            <kpo:type name="SatCode"/>
            <kpo:type name="GSCode"/>
            <kpo:type name="CalibGroupNumber"/>
            <kpo:type name="RangeGroupNumber"/>
            <kpo:type name="RangeNumber"/>
            <kpo:type name="ZSNumber"/>
            <kpo:type name="TCCode"/>
            <kpo:type name="RRateGroupNumber"/>
        </kpo:types>

        <kpo:functions>
            <kpo:function name="CalibrationTime">
                <kpo:parameters>
                    <kpo:element name="gs" type="GSCode"/>
                    <kpo:element name="cgroup" type="CalibGroupNumber"/>
                </kpo:parameters>
            </kpo:function>

            <kpo:function name="RangeTime">
                <kpo:parameters>
                    <kpo:element name="gs" type="GSCode"/>
                    <kpo:element name="rgroup" type="RangeGroupNumber"/>
                </kpo:parameters>
            </kpo:function>

            <kpo:function name="PassTime">
                <kpo:parameters>
                    <kpo:element name="pass" type="PassNumber"/>
                </kpo:parameters>
            </kpo:function>

            <kpo:function name="ZoneSilentTime">
                <kpo:parameters>
                    <kpo:element name="pass" type="PassNumber"/>
                    <kpo:element name="zs" type=    "ZSNumber"/>
                </kpo:parameters>
            </kpo:function>

            <kpo:function name="TcTime">
                <kpo:parameters>
                    <kpo:element name="tc" type="TCCode"/>
                </kpo:parameters>
            </kpo:function>
        </kpo:functions>

        <kpo:states>

            <kpo:state definition="TCSent">
                <parameters>
                    <kpo:element name="pass" type="PassNumber"/>
                    <kpo:element name="sat" type="SatCode"/>
                </parameters>
            </kpo:state>

            <kpo:state definition="CalibrationOK">
                <parameters>
```

```
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="cgroup" type="CalibGroupNumber"/>
    </parameters>
</kpo:state>

<kpo:state definition="FirstTcOK">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode" />
    </parameters>
</kpo:state>

<kpo:state definition="GSAvailable">
    <parameters>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="PassTrack">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="NotPassTrack">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="NotCalibrationAvailable">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="RangeAvailable">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="TCNotAvailable">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="FirstRange">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="CalibrationAvailable">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>
```

```
<kpo:state definition="RRateAvailable">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="SatSilentZoneAvailable">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="zs" type="ZSNumber"/>
    </parameters>
</kpo:state>

<kpo:state definition="FirstTc">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="LastTC">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="TimeTaggedTc">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
    </parameters>
</kpo:state>

<kpo:state definition="FirstRangeOk">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="rgroup" type="RangeGroupNumber"/>
    </parameters>
</kpo:state>

<kpo:state definition="RangeOk">
    <parameters >
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="rgroup" type="RangeGroupNumber"/>
        <kpo:element name="rnum" type="RangeNumber"/>
    </parameters>
</kpo:state>

<kpo:state definition="RRateOK">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="RRGroup" type="RRateGroupNumber"/>
    </parameters>
</kpo:state>

<kpo:state definition="TimeTaggedTCOK">
    <parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
```

```
        </parameters>
    </kpo:state>

    <kpo:state definition="TCSendOK">
        <parameters>
            <kpo:element name="pass" type="PassNumber"/>
            <kpo:element name="sat" type="SatCode"/>
            <kpo:element name="tc" type="TCCode"/>
        </parameters>
    </kpo:state>

    <kpo:state definition="SilentZone">
        <parameters>
            <kpo:element name="pass" type="PassNumber"/>
            <kpo:element name="sat" type="SatCode"/>
            <kpo:element name="gs" type="GSCode"/>
            <kpo:element name="zs" type="ZSNumber"/>

        </parameters>
    </kpo:state>

    <kpo:state definition="PassEliminated">
        <parameters>
            <kpo:element name="pass" type="PassNumber"/>
            <kpo:element name="sat" type="SatCode"/>
            <kpo:element name="gs" type="GSCode"/>
        </parameters>
    </kpo:state>

    <kpo:state definition="LastTCOK">
        <parameters>
            <kpo:element name="pass" type="PassNumber"/>
            <kpo:element name="sat" type="SatCode"/>
            <kpo:element name="tc" type="TCCode"/>
        </parameters>
    </kpo:state>
</kpo:states>

<kpo:actions>
    <kpo:action name="PassEliminate">
        <parameters>
            <kpo:element name="pass" type="PassNumber"/>
            <kpo:element name="sat" type="SatCode"/>
            <kpo:element name="gs" type="GSCode"/>
        </parameters>

        <kpo:properties>
            <kpo:property name="duration">
                <kpo:functions>
                    <kpo:function name="PassTime">
                        <kpo:parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                        </kpo:parameters>
                    </kpo:function>
                </kpo:functions>
            </kpo:property>
        </kpo:properties>

        <kpo:precondition>
            <kpo:scenario>
                <kpo:scenario>
                    <kpo:situationobjs>

                        <kpo:situationobj>
                            <expressionEvaluation name="and"/>
                        </kpo:situationobj>

                        <kpo:situationobj>
                            <expressionEvaluation name="over all"/>

                            <stateObject definition="NotPassTrack">
```

```
                            <parameters>
                                <kpo:element name="pass" type="PassNumber"/>
                                <kpo:element name="sat" type="SatCode"/>
                                <kpo:element name="gs" type="GSCode"/>
                            </parameters>
                        </stateObject>
                    </kpo:situationobj>

                    <kpo:situationobj>
                        <expressionEvaluation name="at start"/>

                        <stateObject definition="GSAvailable">
                            <parameters>
                                <kpo:element name="gs" type="GSCode"/>
                            </parameters>
                        </stateObject>
                    </kpo:situationobj>

                </kpo:situationobjs>
            </kpo:scenario>
        </kpo:precondition>

        <kpo:effect>
            <kpo:scenario>
                <kpo:situationobjs>

                    <kpo:situationobj>
                        <expressionEvaluation name="and"/>
                    </kpo:situationobj>

                    <kpo:situationobj>
                        <expressionEvaluation name="at end"/>
                        <stateObject definition="PassEliminated">
                            <parameters>
                                <kpo:element name="pass" type="PassNumber"/>
                                <kpo:element name="sat" type="SatCode"/>
                                <kpo:element name="gs" type="GSCode"/>
                            </parameters>
                        </stateObject>
                    </kpo:situationobj>

                    <kpo:situationobj>
                        <expressionEvaluation name="not"/>


                        <stateObject definition="GSAvailable">
                            <parameters>
                                <kpo:element name="gs" type="GSCode"/>
                            </parameters>
                        </stateObject>

                        <situationFather>
                            <expressionEvaluation name="at start"/>
                        </situationFather>
                    </kpo:situationobj>

                    <kpo:situationobj>
                        <expressionEvaluation name="at end"/>

                        <stateObject definition="GSAvailable">
                            <parameters>
                                <kpo:element name="gs" type="GSCode"/>
                            </parameters>
                        </stateObject>
                    </kpo:situationobj>

                </kpo:situationobjs>
            </kpo:scenario>
        </kpo:effect>
    </kpo:action>
```

```
<kpo:action name="Calibrate">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="cgroup" type="CalibGroupNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="CalibrationTime">
                    <kpo:parameters>
                        <kpo:element name="gs" type="GSCode"/>
                        <kpo:element name="cgroup" type="CalibGroupNumber"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>

                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="CalibrationAvailable">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="PassTrack">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="RangeAvailable">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
```

```
                                    </parameters>
                                </stateObject>

                            </kpo:situationobj>
                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:precondition>

                <kpo:effect>
                    <kpo:scenario>
                        <kpo:situationobjs>

                            <kpo:situationobj>
                                <expressionEvaluation name="and"/>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="CalibrationOK">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="cgroup"
type="CalibGroupNumber"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="not"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="at start"/>
                                </situationFather>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="not"/>

                                <stateObject definition="CalibrationAvailable">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="at end"/>
                                </situationFather>

                            </kpo:situationobj>
                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:effect>
            </kpo:action>
```

```
<kpo:action name="FirstRange">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="rgroup" type="RangeGroupNumber"/>
        <kpo:element name="cgroup" type="CalibGroupNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="RangeTime">
                    <kpo:parameters>
                        <kpo:element name="gs" type="GSCode"/>
                        <kpo:element name="rgroup" type="RangeGroupNumber"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>
                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="FirstRange">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="PassTrack">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="RangeAvailable">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
```

```
                                    <kpo:element name="gs" type="GSCode"/>
                                </parameters>
                            </stateObject>
                        </kpo:situationobj>

                        <kpo:situationobj>
                            <stateObject definition="NotCalibrationAvailable">
                                <parameters>
                                    <kpo:element name="pass" type="PassNumber"/>
                                    <kpo:element name="gs" type="GSCode"/>
                                </parameters>
                            </stateObject>

                            <situationFather>
                                <expressionEvaluation name="or"/>

                                <stateObject definition="CalibrationOK">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="cgroup"
type="CalibGroupNumber"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="over all"/>
                                </situationFather>

                            </situationFather>
                        </kpo:situationobj>
                    </kpo:situationobjs>
                </kpo:scenario>
            </kpo:precondition>

            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobjs>

                        <kpo:situationobj>
                            <expressionEvaluation name="and"/>
                        </kpo:situationobj>

                        <kpo:situationobj>
                            <expressionEvaluation name="at end"/>

                            <stateObject definition="FirstRangeOk">
                                <parameters>
                                    <kpo:element name="pass" type="PassNumber"/>
                                    <kpo:element name="rgroup"
type="RangeGroupNumber"/>
                                </parameters>
                            </stateObject>
                        </kpo:situationobj>

                        <kpo:situationobj>
                            <expressionEvaluation name="at start"/>

                            <stateObject definition="GSAvailable">
                                <parameters>
                                    <kpo:element name="gs" type="GSCode"/>
                                </parameters>
                            </stateObject>

                            <situationFather>
                                <expressionEvaluation name="at start"/>
                            </situationFather>
                        </kpo:situationobj>

                        <kpo:situationobj>
                            <expressionEvaluation name="at end"/>
```

```
                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

            </kpo:situationobjs>
        </kpo:scenario>
    </kpo:effect>
</kpo:action>

<kpo:action name="Range">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="rgroup" type="RangeGroupNumber"/>
        <kpo:element name="rnum" type="RangeNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="RangeTime">
                    <kpo:parameters>
                        <kpo:element name="gs" type="GSCode"/>
                        <kpo:element name="rgroup" type="RangeGroupNumber"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>
                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="PassTrack">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="RangeAvailable">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
```

```
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="over all"/>

                                <stateObject definition="FirstRangeOk">
                                  <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                     <kpo:element name="rgroup" type="RangeGroupNumber"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:precondition>

                <kpo:effect>
                    <kpo:scenario>
                        <kpo:situationobjs>

                            <kpo:situationobj>
                                <expressionEvaluation name="and"/>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="RangeOk">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="rgroup"
type="RangeGroupNumber"/>
                                        <kpo:element name="rnum" type="RangeNumber"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="not"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="at start"/>
                                </situationFather>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:effect>
            </kpo:action>

            <kpo:action name="WaitSilentZone">
                <kpo:parameters>
```

```
    <kpo:element name="pass" type="PassNumber"/>
    <kpo:element name="sat" type="SatCode"/>
    <kpo:element name="gs" type="GSCode"/>
    <kpo:element name="zs" type="ZSNumber"/>
</kpo:parameters>

<kpo:properties>
    <kpo:property name="duration">
        <kpo:functions>
            <kpo:function name="ZoneSilentTime">
                <kpo:parameters>
                    <kpo:element name="pass" type="PassNumber"/>
                    <kpo:element name="zs" type="ZSNumber"/>
                </kpo:parameters>
            </kpo:function>
        </kpo:functions>
    </kpo:property>
</kpo:properties>

<kpo:precondition>
    <kpo:scenario>
        <kpo:situationobjs>

            <kpo:situationobj>
                <expressionEvaluation name="and"/>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at start"/>

                <stateObject definition="GSAvailable">
                    <parameters>
                        <kpo:element name="gs" type="GSCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at start"/>

                <stateObject definition="SatSilentZoneAvailable">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="gs" type="GSCode"/>
                        <kpo:element name="zs" type="ZSNumber"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

        </kpo:situationobjs>
    </kpo:scenario>
</kpo:precondition>

<kpo:effect>
    <kpo:scenario>
        <kpo:situationobjs>
            <kpo:situationobj>
                <expressionEvaluation name="and"/>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at end"/>

                <stateObject definition="SilentZone">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="gs" type="GSCode"/>
                        <kpo:element name="zs" type="ZSNumber"/>
                    </parameters>
```

```
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>
                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>

                    <situationFather>
                        <expressionEvaluation name="at start"/>
                    </situationFather>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at end"/>
                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

            </kpo:situationobjs>
        </kpo:scenario>
    </kpo:effect>
</kpo:action>

<kpo:action name="RangeRate">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="RRGroup" type="RRateGroupNumber"/>
    </kpo:parameters>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>

                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="RRateAvailable">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>
```

```
                                <stateObject definition="PassTrack">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="sat" type="SatCode"/>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>
                        </kpo:situationobjs>

                    </kpo:scenario>
                </kpo:precondition>
                <kpo:effect>

                    <kpo:scenario>
                        <kpo:situationobjs>
                            <kpo:situationobj>
                                <expressionEvaluation name="and"/>

                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="RRateOK">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="sat" type="SatCode"/>
                                        <kpo:element name="gs" type="GSCode"/>
                                        <kpo:element name="RRGroup"
type="RRateGroupNumber"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="not"/>

                                <stateObject definition="RRateAvailable">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="sat" type="SatCode"/>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="at end"/>
                                </situationFather>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="not"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="at start"/>
                                </situationFather>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
```

```
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

            </kpo:situationobjs>
        </kpo:scenario>
    </kpo:effect>
</kpo:action>

<kpo:action name="SendFirstTC">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>

        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="cgroup" type="CalibGroupNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="TcTime">
                    <kpo:parameters>
                        <kpo:element name="tc" type="TCCode"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>
                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="FirstTc">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="tc" type="TCCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="PassTrack">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>
```

```
                            <kpo:situationobj>
                                <expressionEvaluation name="over all"/>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <stateObject definition="NotCalibrationAvailable">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="or"/>

                                    <stateObject definition="CalibrationOK">
                                        <parameters>
                                            <kpo:element name="pass" type="PassNumber"/>
                                            <kpo:element name="cgroup"
type="CalibGroupNumber"/>
                                        </parameters>
                                    </stateObject>

                                    <situationFather>
                                        <expressionEvaluation name="at start"/>
                                    </situationFather>

                                </situationFather>
                            </kpo:situationobj>
                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:precondition>

                <kpo:effect>
                    <kpo:scenario>
                        <kpo:situationobjs>
                            <kpo:situationobj>
                                <expressionEvaluation name="and"/>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="FirstTcOK">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="sat" type="SatCode"/>
                                        <kpo:element name="tc" type="TCCode"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="TCSent">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="sat" type="SatCode"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
```

```
                    </stateObject>

                    <situationFather>
                        <expressionEvaluation name="at start"/>
                    </situationFather>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at end"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>

                    <situationFather>
                        <expressionEvaluation name="at start"/>
                    </situationFather>
                </kpo:situationobj>

            </kpo:situationobjs>
        </kpo:scenario>
    </kpo:effect>
</kpo:action>

<kpo:action name="SendLastTC">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="TcTime">
                    <kpo:parameters>
                        <kpo:element name="tc" type="TCCode"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>

                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="LastTC">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
```

```
                        <kpo:element name="tc" type="TCCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="over all"/>

                <stateObject definition="PassTrack">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="gs" type="GSCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="over all"/>

                <stateObject definition="TCSent">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

        </kpo:situationobjs>
    </kpo:scenario>
</kpo:precondition>

<kpo:effect>
    <kpo:scenario>
        <kpo:situationobjs>
            <kpo:situationobj>
                <expressionEvaluation name="and"/>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at end"/>

                <stateObject definition="LastTCOK">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="tc" type="TCCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="not"/>

                <stateObject definition="GSAvailable">
                    <parameters>
                        <kpo:element name="gs" type="GSCode"/>
                    </parameters>
                </stateObject>

                <situationFather>
                    <expressionEvaluation name="at start"/>
                </situationFather>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at end"/>

                <stateObject definition="GSAvailable">
                    <parameters>
                        <kpo:element name="gs" type="GSCode"/>
```

```
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>

                    <stateObject definition="TCSent">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                        </parameters>
                    </stateObject>

                    <situationFather>
                        <expressionEvaluation name="at end"/>
                    </situationFather>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="not"/>

                    <stateObject definition="PassTrack">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>

                    <situationFather>
                        <expressionEvaluation name="at end"/>
                    </situationFather>
                </kpo:situationobj>

            </kpo:situationobjs>
        </kpo:scenario>
    </kpo:effect>
</kpo:action>

<kpo:action name="SendTimeTaggedTC">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
        <kpo:element name="gs" type="GSCode"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="TcTime">
                    <kpo:parameters>
                        <kpo:element name="tc" type="TCCode"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>
                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>
                    <stateObject definition="GSAvailable">
                        <parameters>
```

```
                        <kpo:element name="gs" type="GSCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at start"/>

                <stateObject definition="TimeTaggedTc">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="tc" type="TCCode"/>

                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="over all"/>

                <stateObject definition="PassTrack">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="gs" type="GSCode"/>

                    </parameters>
                </stateObject>
            </kpo:situationobj>

        </kpo:situationobjs>
    </kpo:scenario>
</kpo:precondition>

<kpo:effect>
    <kpo:scenario>
        <kpo:situationobjs>

            <kpo:situationobj>
                <expressionEvaluation name="and"/>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at end"/>

                <stateObject definition="TimeTaggedTCOK">
                    <parameters>
                        <kpo:element name="pass" type="PassNumber"/>
                        <kpo:element name="sat" type="SatCode"/>
                        <kpo:element name="tc" type="TCCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="not"/>

                <stateObject definition="GSAvailable">
                    <parameters>
                        <kpo:element name="gs" type="GSCode"/>
                    </parameters>
                </stateObject>

                <situationFather>
                    <expressionEvaluation name="at start"/>
                </situationFather>
            </kpo:situationobj>

            <kpo:situationobj>
                <expressionEvaluation name="at end"/>
```

```
                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

            </kpo:situationobjs>
        </kpo:scenario>
    </kpo:effect>
</kpo:action>

<kpo:action name="SendTC">
    <kpo:parameters>
        <kpo:element name="pass" type="PassNumber"/>
        <kpo:element name="sat" type="SatCode"/>
        <kpo:element name="tc" type="TCCode"/>
        <kpo:element name="gs" type="GSCode"/>
        <kpo:element name="cgroup" type="CalibGroupNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property name="duration">
            <kpo:functions>
                <kpo:function name="TcTime">
                    <kpo:parameters>
                        <kpo:element name="tc" type="TCCode"/>
                    </kpo:parameters>
                </kpo:function>
            </kpo:functions>
        </kpo:property>
    </kpo:properties>

    <kpo:precondition>
        <kpo:scenario>
            <kpo:situationobjs>
                <kpo:situationobj>
                    <expressionEvaluation name="and"/>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at start"/>

                    <stateObject definition="GSAvailable">
                        <parameters>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="PassTrack">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                            <kpo:element name="gs" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="over all"/>

                    <stateObject definition="TCSent">
                        <parameters>
                            <kpo:element name="pass" type="PassNumber"/>
                            <kpo:element name="sat" type="SatCode"/>
                        </parameters>
```

```
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <stateObject definition="NotCalibrationAvailable">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="or"/>

                                    <stateObject definition="CalibrationOK">
                                        <parameters>
                                            <kpo:element name="pass" type="PassNumber"/>
                                            <kpo:element name="cgroup"
type="CalibGroupNumber"/>
                                        </parameters>
                                    </stateObject>

                                    <situationFather>
                                        <expressionEvaluation name="at start"/>
                                    </situationFather>

                                </situationFather>
                            </kpo:situationobj>
                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:precondition>

                <kpo:effect>
                    <kpo:scenario>
                        <kpo:situationobjs>
                            <kpo:situationobj>
                                <expressionEvaluation name="and"/>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="TCSendOK">
                                    <parameters>
                                        <kpo:element name="pass" type="PassNumber"/>
                                        <kpo:element name="sat" type="SatCode"/>
                                        <kpo:element name="tc" type="TCCode"/>
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="not"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
                                    </parameters>
                                </stateObject>

                                <situationFather>
                                    <expressionEvaluation name="at start"/>
                                </situationFather>
                            </kpo:situationobj>

                            <kpo:situationobj>
                                <expressionEvaluation name="at end"/>

                                <stateObject definition="GSAvailable">
                                    <parameters>
                                        <kpo:element name="gs" type="GSCode"/>
```

```
                                    </parameters>
                                </stateObject>
                            </kpo:situationobj>

                        </kpo:situationobjs>
                    </kpo:scenario>
                </kpo:effect>
            </kpo:action>
        </kpo:actions>
    </domain>
</kpo:KPlannOO>
```

APÊNDICE C – DESCRIÇÃO XML DO PROBLEMA PARA O DOMÍNIO DE RASTREIO DE SATÉLITES DO INPE

```
<?xml version="1.0" encoding="UTF-8"?>
<kpo:KPlannOO xmlns:kpo="http://www.rrochas.com/kplanoo/xkps"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.rrochas.com/kplanoo/xkps KPlanOOSchemaProblem.xsd
">

<problem>
        <definition>SCD1CBAProblemPass60509-4</definition>
        <domain>
            <kpo:definition>POVMEDZSRRTC</kpo:definition>
        </domain>
        <objects>
            <kpo:element name="PASS60509-4" type="PassNumber"/>
            <kpo:element name="SCD1" type="SatCode"/>
            <kpo:element name="CBA" type="GSCode"/>
            <kpo:element name="CGroup03" type="CalibGroupNumber"/>
            <kpo:element name="RGroup03" type="RangeGroupNumber"/>
            <kpo:element name="Range02" type="RangeNumber"/>
            <kpo:element name="Range03" type="RangeNumber"/>
            <kpo:element name="Range04" type="RangeNumber"/>
            <kpo:element name="Range05" type="RangeNumber"/>
            <kpo:element name="ZS01" type="ZSNumber"/>
            <kpo:element name="RRGroup07" type="RRateGroupNumber"/>
            <kpo:element name="TC137" type="TCCode"/>
            <kpo:element name="TC118" type="TCCode"/>
            <kpo:element name="TC138" type="TCCode"/>
            <kpo:element name="TT140" type="TCCode"/>
            <kpo:element name="TT150" type="TCCode"/>
        </objects>

        <init>
            <kpo:situationobjs>
                <kpo:situationobj>
                    <expressionEvaluation name="at" toReturn="0.0"/>

                    <stateObject definition="CalibrationAvailable">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="CBA" type="GSCode"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at" toReturn="90.0"/>

                    <situationFather>
                        <expressionEvaluation name="not"/>

                        <stateObject definition="CalibrationAvailable">
                            <parameters>
                                <kpo:element name="PASS60509-4" type="PassNumber"/>
                                <kpo:element name="CBA" type="GSCode"/>
                            </parameters>
                        </stateObject>
                    </situationFather>
                </kpo:situationobj>

                <kpo:situationobj>
                    <expressionEvaluation name="at" toReturn="480.0"/>

                    <stateObject definition="FirstRange">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="SCD1" type="SatCode"/>
```

```
            <kpo:element name="CBA" type="GSCode"/>
        </parameters>
    </stateObject>
</kpo:situationobj>

<kpo:situationobj>
    <expressionEvaluation name="at" toReturn="570.0"/>

    <situationFather>
        <expressionEvaluation name="not"/>

        <stateObject definition="FirstRange">
            <parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
                <kpo:element name="CBA" type="GSCode"/>
            </parameters>
        </stateObject>
    </situationFather>
</kpo:situationobj>

<kpo:situationobj>
    <expressionEvaluation name="at" toReturn="390.0"/>

    <stateObject definition="FirstTc">
        <parameters>
            <kpo:element name="PASS60509-4" type="PassNumber"/>
            <kpo:element name="SCD1" type="SatCode"/>
            <kpo:element name="TC137" type="TCCode"/>
        </parameters>
    </stateObject>
</kpo:situationobj>

<kpo:situationobj>
    <expressionEvaluation name="at" toReturn="392.0"/>

    <situationFather>
        <expressionEvaluation name="not"/>

        <stateObject definition="FirstTc">
            <parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
                <kpo:element name="TC137" type="TCCode"/>
            </parameters>
        </stateObject>
    </situationFather>
</kpo:situationobj>

<kpo:situationobj>
    <expressionEvaluation name="at" toReturn="1050.0"/>

    <stateObject definition="LastTC">
        <parameters>
            <kpo:element name="PASS60509-4" type="PassNumber"/>
            <kpo:element name="SCD1" type="SatCode"/>
            <kpo:element name="TC118" type="TCCode"/>
        </parameters>
    </stateObject>
</kpo:situationobj>

<kpo:situationobj>
    <expressionEvaluation name="at" toReturn="1052.0"/>

    <situationFather>
        <expressionEvaluation name="not"/>

        <stateObject definition="LastTC">
            <parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
```

```
                    <kpo:element name="TC118" type="TCCode"/>
                </parameters>
            </stateObject>
        </situationFather>
    </kpo:situationobj>

    <kpo:situationobj>
        <expressionEvaluation name="at" toReturn="420.0"/>

        <stateObject definition="RRateAvailable">
            <parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
                <kpo:element name="CBA" type="GSCode"/>
            </parameters>
        </stateObject>
    </kpo:situationobj>

    <kpo:situationobj>
        <expressionEvaluation name="at" toReturn="421.0"/>

        <situationFather>
            <expressionEvaluation name="not"/>

            <stateObject definition="RRateAvailable">
                <parameters>
                    <kpo:element name="PASS60509-4" type="PassNumber"/>
                    <kpo:element name="SCD1" type="SatCode"/>
                    <kpo:element name="CBA" type="GSCode"/>
                </parameters>
            </stateObject>
        </situationFather>
    </kpo:situationobj>

    <kpo:situationobj>
        <expressionEvaluation name="at" toReturn="870.0"/>

        <stateObject definition="SatSilentZoneAvailable">
            <parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
                <kpo:element name="CBA" type="GSCode"/>
                <kpo:element name="ZS01" type="ZSNumber"/>
            </parameters>
        </stateObject>
    </kpo:situationobj>

    <kpo:situationobj>
        <expressionEvaluation name="at" toReturn="900.0"/>

        <situationFather>
            <expressionEvaluation name="not"/>

            <stateObject definition="SatSilentZoneAvailable">
                <parameters>
                    <kpo:element name="PASS60509-4" type="PassNumber"/>
                    <kpo:element name="SCD1" type="SatCode"/>
                    <kpo:element name="CBA" type="GSCode"/>
                    <kpo:element name="ZS01" type="ZSNumber"/>
                </parameters>
            </stateObject>
        </situationFather>
    </kpo:situationobj>

</kpo:situationobjs>
<functions>
    <kpo:function name="CalibrationTime">
        <kpo:parameters>
            <kpo:element name="CBA" type="GSCode"/>
            <kpo:element name="CGroup03" type="CalibGroupNumber"/>
        </kpo:parameters>
```

```
    <kpo:properties>
        <kpo:property value="30.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="PassTime">
    <kpo:parameters>
        <kpo:element name="PASS60509-4" type="PassNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property value="1052.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="RangeTime">
    <kpo:parameters>
        <kpo:element name="CBA" type="GSCode"/>
        <kpo:element name="RGroup03" type="RangeGroupNumber"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property value="30.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="TcTime">
    <kpo:parameters>
        <kpo:element name="TC137" type="TCCode"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property value="72.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="TcTime">
    <kpo:parameters>
        <kpo:element name="TC118" type="TCCode"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property value="720.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="TcTime">
    <kpo:parameters>
        <kpo:element name="TC138" type="TCCode"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property value="720.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="TcTime">
    <kpo:parameters>
        <kpo:element name="TT140" type="TCCode"/>
    </kpo:parameters>

    <kpo:properties>
        <kpo:property value="2.0"/>
    </kpo:properties>
</kpo:function>

<kpo:function name="TcTime">
    <kpo:parameters>
        <kpo:element name="TT150" type="TCCode"/>
    </kpo:parameters>
```

```
            <kpo:properties>
                <kpo:property value="720.0"/>
            </kpo:properties>
        </kpo:function>

        <kpo:function name="ZoneSilentTime">
            <kpo:parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="ZS01" type="ZSNumber"/>
            </kpo:parameters>

            <kpo:properties>
                <kpo:property value="3.0"/>
            </kpo:properties>
        </kpo:function>

        <kpo:function name="GSAvailable">
            <kpo:parameters>
                <kpo:element name="CBA" type="GSCode"/>
            </kpo:parameters>
        </kpo:function>

        <kpo:function name="PassTrack">
            <kpo:parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
                <kpo:element name="CBA" type="GSCode"/>
            </kpo:parameters>
        </kpo:function>

        <kpo:function name="RangeAvailable">
            <kpo:parameters>
                <kpo:element name="PASS60509-4" type="PassNumber"/>
                <kpo:element name="SCD1" type="SatCode"/>
                <kpo:element name="CBA" type="GSCode"/>
            </kpo:parameters>
        </kpo:function>
    </functions>
</init>

<goals>
    <kpo:goal>
        <kpo:situationobjs>

            <kpo:situationobj>
                <expressionEvaluation name="and"/>
            </kpo:situationobj>

            <kpo:situationobj>
                <stateObject definition="LastTCOK">
                    <parameters>
                        <kpo:element name="PASS60509-4" type="PassNumber"/>
                        <kpo:element name="SCD1" type="SatCode"/>
                        <kpo:element name="TC138" type="TCCode"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
                <stateObject definition="RRateOK">
                    <parameters>
                        <kpo:element name="PASS60509-4" type="PassNumber"/>
                        <kpo:element name="SCD1" type="SatCode"/>
                        <kpo:element name="CBA" type="GSCode"/>
                        <kpo:element name="RRGroup07" type="RRateGroupNumber"/>
                    </parameters>
                </stateObject>
            </kpo:situationobj>

            <kpo:situationobj>
```

```
                    <stateObject definition="RRateOK">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="RGroup03" type="RangeGroupNumber"/>
                            <kpo:element name="Range02" type="RangeNumber"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <stateObject definition="RRateOK">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="RGroup03" type="RangeGroupNumber"/>
                            <kpo:element name="Range03" type="RangeNumber"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <stateObject definition="RRateOK">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="RGroup03" type="RangeGroupNumber"/>
                            <kpo:element name="Range04" type="RangeNumber"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <stateObject definition="RRateOK">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="RGroup03" type="RangeGroupNumber"/>
                            <kpo:element name="Range05" type="RangeNumber"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>

                <kpo:situationobj>
                    <stateObject definition="SilentZone">
                        <parameters>
                            <kpo:element name="PASS60509-4" type="PassNumber"/>
                            <kpo:element name="SCD1" type="SatCode"/>
                            <kpo:element name="CBA" type="GSCode"/>
                            <kpo:element name="ZS01" type="ZSNumber"/>
                        </parameters>
                    </stateObject>
                </kpo:situationobj>
            </kpo:situationobjs>
        </kpo:goal>
    </goals>
</problem>
</KplanOO>
```

APÊNDICE D – DESCRIÇÃO XML DO DOMÍNIO DE SIMULAÇÃO DE SATÉLITES

```
<?xml version="1.0" encoding="UTF-8"?>
<kpo:KPlannOO xmlns:kpo="http://www.rrochas.com/kplanoo/xkps"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.rrochas.com/kplanoo/xkps
KPlanOOSchemaSimulator.xsd ">

    <kpo:simulator>
        <kpo:types>
            <kpo:type name="ev" />
            <kpo:type name="st" />
        </kpo:types>
        <kpo:initstat>
            <kpo:property id="SIMTIME" type="st" value="0.2148040900000000E+05"/>
            <kpo:property id="SIMEND" type="st" value="0.2148140900000000E+05"/>
            <kpo:property id="SIMSTEP" type="st" value="0.3472222222222222E-03"/>
            <kpo:property id="SIMSTOP" type="ev" value="0.0000000000000000E+00"/>
            <kpo:property id="ECL" type="ev" value="0.0000000000000000E+00"/>
            <kpo:property id="SUN" type="ev" value="0.0000000000000000E+00"/>
            <kpo:property id="A" type="st" value="-0.1440000000000000E+04"/>
            <kpo:property id="B" type="st" value="0.0000000000000000E+00"/>
            <kpo:property id="C" type="st" value="0.0000000000000000E+00"/>
            <kpo:property id="HIGH" type="st" value="0.0000000000000000E+00"/>
            <kpo:property id="MED" type="st" value="0.0000000000000000E+00"/>
            <kpo:property id="LOW" type="st" value="0.0000000000000000E000"/>
            <kpo:property id="TEST" type="ev" value="1.0000000000000000E000"/>
            <kpo:property id="AAA" type="st" value="1.0000000000000000E000"/>
            <kpo:property id="BBB" type="st" value="2.0000000000000000E000"/>
            <kpo:property id="CCCX" type="st" value="-9.0000000000000000E-001"/>
            <kpo:property id="CCC" type="st" value="-8.8200000000000000E-001"/>
        </kpo:initstat>
        <kpo:eventQueue>
            <kpo:property id="ECL" type="ev" value="0.2151227395833334E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151229756944445E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151234375000000E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151236736111111E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151241319444445E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151243715277778E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151248298611111E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151250659722222E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151255277777778E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151257638888889E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151262256944444E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151264618055556E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151269201388889E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151271597222222E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151276180555555E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151278541666667E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151283159722223E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151285520833333E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151290138888889E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151292500000000E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151297083333333E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151299479166667E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151304062500000E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151306423611111E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151311041666667E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151313402777778E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151318020833334E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151320381944445E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151324965277778E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151327361111111E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151331944444445E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151334305555555E+05"/>
            <kpo:property id="ECL" type="ev" value="0.2151338923611111E+05"/>
            <kpo:property id="SUN" type="ev" value="0.2151341284722222E+05"/>
```

```
<kpo:property id="ECL" type="ev" value="0.2151345902777778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151348263888889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151352847222222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151355243055556E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151359826388889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151362187500000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151366805555556E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151369166666667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151373784722222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151376145833333E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151380729166667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151383125000000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151387708333333E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151390069444444E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151394687500000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151397048611111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151401666666667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151404027777778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151408611111111E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151411006944444E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151415590277778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151417986111111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151422569444444E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151424930555555E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151429548611111E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151431909722223E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151436527777778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151438888888889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151443472222222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151445868055556E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151450451388889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151452812500000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151457430555556E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151459791666667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151464409722222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151466770833333E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151471354166667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151473750000000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151478333333333E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151480694444444E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151485312500000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151487673611111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151492291666667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151494652777778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151499236111111E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151501631944445E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151506215277778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151508576388889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151513194444445E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151515555555556E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151520173611111E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151522534722222E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151527118055556E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151529513888889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151534097222222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151536458333333E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151541076388889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151543437500000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151548055555555E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151550416666667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151555000000000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151557395833333E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151561979166667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151564340277778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151568958333333E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151571319444444E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151575937500000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151578298611111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151582881944445E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151585277777778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151589861111111E+05"/>
```

```
<kpo:property id="SUN" type="ev" value="0.2151592222222222E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151596840277778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151599201388889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151603819444445E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151606180555556E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151610763888889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151613159722222E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151617743055556E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151620104166667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151624722222222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151627083333333E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151631701388889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151634062500000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151638680555555E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151641041666667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151645625000000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151647986111111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151652604166667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151654965277778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151659583333333E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151661944444444E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151666562500000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151668923611111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151673506944444E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151675902777778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151680486111111E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151682847222222E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151687465277778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151689826388889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151694444444445E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151696805555555E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151701388888889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151703784722222E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151708368055556E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151710729166667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151715347222222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151717708333334E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151722326388889E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151724687500000E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151729270833333E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151731666666667E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151736250000000E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151738611111111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151743229166667E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151745590277778E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151750208333333E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151752569444444E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151757152777778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151759548611111E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151764131944445E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151766493055556E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151771111111111E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151773472222222E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151778090277778E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151780451388889E+05"/>
<kpo:property id="ECL" type="ev" value="0.2151785034722222E+05"/>
<kpo:property id="SUN" type="ev" value="0.2151787430555555E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2150855243055556E+05"/>
<kpo:property id="TCF20" type="ev" value="0.2150855277777778E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2150959687500000E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151057188657407E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151057193287037E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151057197916667E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151058267361111E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151058271990741E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151058276620370E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151064254629629E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151064259259259E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151064263888889E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151064969907407E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151064974537037E+05"/>
```

```
<kpo:property id="TCGR18" type="ev" value="0.2151064979166667E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151154876157407E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151154880787037E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151154885416667E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151155836805556E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151155841435185E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151155846064815E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151161725694444E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151161730324074E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151161734953703E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151162723379629E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151162728009259E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151162732638889E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151252673611111E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151252678240741E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151252682870370E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151253313657407E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151253318287037E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151253322916667E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151259295138889E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151259299768518E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151259304398148E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151260386574074E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151260391203704E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151260395833333E+05"/>
<kpo:property id="TCF20" type="ev" value="0.2150959861111111E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2151057326388889E+05"/>
<kpo:property id="TCF20" type="ev" value="0.2151057361111111E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2151155000000000E+05"/>
<kpo:property id="TCF20" type="ev" value="0.2151155034722223E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2151259409722222E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151356927083334E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151356931712963E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151356936342593E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151358000000000E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151358004629630E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151358009259259E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151363962962963E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151363967592593E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151363972222222E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151364728009259E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151364732638889E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151364737268518E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151265481481481E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151265486111111E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151265490740741E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151265922453704E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151265927083333E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151265931712963E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151272215277778E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151272219907407E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151272224537037E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151273071759259E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151273076388889E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151273081018518E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151279173611111E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151279178240741E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151279182870371E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151279798611111E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151279803240741E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151279807870370E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151454623842593E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151454628472222E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151454633101852E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151455562500000E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151455567129630E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151455571759260E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151461451388889E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151461456018519E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151461460648148E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151462465277778E+05"/>
```

```
<kpo:property id="TCGR06" type="ev" value="0.2151462469907407E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151462474537037E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151369844907408E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151369849537037E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151369854166667E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151370659722223E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151370664351852E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151370668981482E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151376744212963E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151376748842592E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151376753472222E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151377513888889E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151377518518519E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151377523148148E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151467504629630E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151467509259260E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151467513888889E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151468193287037E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151468197916667E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151468202546296E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151474337962963E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151474342592592E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151474347222222E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151475179398148E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151475184027778E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151475188657407E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151481373842593E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151481378472222E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151481383101852E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151481766203704E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151481770833334E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151481775462963E+05"/>
<kpo:property id="TCF20" type="ev" value="0.2151259548611111E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2151357048611111E+05"/>
<kpo:property id="TCF20" type="ev" value="0.2151357083333334E+05"/>
<kpo:property id="TCF19" type="ev" value="0.2151461562500000E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151552447916667E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151552452546296E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151552457175926E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151553016203704E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151553020833333E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151553025462963E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151559030092593E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151559034722222E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151559039351852E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151560123842592E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151560128472222E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151560133101852E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151565237268518E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151565241898148E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151565246527778E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151565627314815E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151565631944445E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151565636574074E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151571951388889E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151571956018519E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151571960648148E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151572805555556E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151572810185185E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151572814814815E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151578901620370E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151578906250000E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151578910879630E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151579547453704E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151579552083333E+05"/>
<kpo:property id="TCGR18" type="ev" value="0.2151579556712963E+05"/>
<kpo:property id="TCGR17" type="ev" value="0.2151656667824074E+05"/>
<kpo:property id="TCGR05" type="ev" value="0.2151656672453704E+05"/>
<kpo:property id="TCS01" type="ev" value="0.2151656677083333E+05"/>
<kpo:property id="TCS02" type="ev" value="0.2151657732638889E+05"/>
<kpo:property id="TCGR06" type="ev" value="0.2151657737268518E+05"/>
```

```
    <kpo:property id="TCGR18" type="ev" value="0.2151657741898148E+05"/>
    <kpo:property id="TCGR17" type="ev" value="0.2151663677083333E+05"/>
    <kpo:property id="TCGR05" type="ev" value="0.2151663681712963E+05"/>
    <kpo:property id="TCS01" type="ev" value="0.2151663686342592E+05"/>
    <kpo:property id="TCS02" type="ev" value="0.2151664483796297E+05"/>
    <kpo:property id="TCGR06" type="ev" value="0.2151664488425926E+05"/>
    <kpo:property id="TCGR18" type="ev" value="0.2151664493055556E+05"/>
    <kpo:property id="TCGR17" type="ev" value="0.2151669583333334E+05"/>
    <kpo:property id="TCGR05" type="ev" value="0.2151669587962963E+05"/>
    <kpo:property id="TCS01" type="ev" value="0.2151669592592593E+05"/>
    <kpo:property id="TCS02" type="ev" value="0.2151670386574074E+05"/>
    <kpo:property id="TCGR06" type="ev" value="0.2151670391203704E+05"/>
    <kpo:property id="TCGR18" type="ev" value="0.2151670395833334E+05"/>
    <kpo:property id="TCGR17" type="ev" value="0.2151676475694445E+05"/>
    <kpo:property id="TCGR05" type="ev" value="0.2151676480324074E+05"/>
    <kpo:property id="TCS01" type="ev" value="0.2151676484953704E+05"/>
    <kpo:property id="TCS02" type="ev" value="0.2151677256944445E+05"/>
    <kpo:property id="TCGR06" type="ev" value="0.2151677261574074E+05"/>
    <kpo:property id="TCGR18" type="ev" value="0.2151677266203704E+05"/>
    <kpo:property id="TCGR17" type="ev" value="0.2151767247685185E+05"/>
    <kpo:property id="TCGR05" type="ev" value="0.2151767252314815E+05"/>
    <kpo:property id="TCS01" type="ev" value="0.2151767256944445E+05"/>
    <kpo:property id="TCS02" type="ev" value="0.2151767914351852E+05"/>
    <kpo:property id="TCGR06" type="ev" value="0.2151767918981482E+05"/>
    <kpo:property id="TCGR18" type="ev" value="0.2151767923611111E+05"/>
    <kpo:property id="TCGR17" type="ev" value="0.2151774072916666E+05"/>
    <kpo:property id="TCGR05" type="ev" value="0.2151774077546296E+05"/>
    <kpo:property id="TCS01" type="ev" value="0.2151774082175926E+05"/>
    <kpo:property id="TCS02" type="ev" value="0.2151774918981481E+05"/>
    <kpo:property id="TCGR06" type="ev" value="0.2151774923611111E+05"/>
    <kpo:property id="TCGR18" type="ev" value="0.2151774928240741E+05"/>
    <kpo:property id="TCGR17" type="ev" value="0.2151781093750000E+05"/>
    <kpo:property id="TCGR05" type="ev" value="0.2151781098379630E+05"/>
    <kpo:property id="TCS01" type="ev" value="0.2151781103009259E+05"/>
    <kpo:property id="TCS02" type="ev" value="0.2151781526620370E+05"/>
    <kpo:property id="TCGR06" type="ev" value="0.2151781531250000E+05"/>
    <kpo:property id="TCGR18" type="ev" value="0.2151781535879630E+05"/>
    <kpo:property id="TCF20" type="ev" value="0.2151461701388889E+05"/>
</kpo:eventQueue>
<kpo:curves>
    <kpo:curve id="funcTEST" property=" ">
        <kpo:point in="-1.0000000000000000E003" out="5.0000000000000000E001"/>
        <kpo:point in="-5.0000000000000000E002" out="2.0000000000000000E001"/>
        <kpo:point in="5.0000000000000000E002" out="2.0000000000000000E001"/>
        <kpo:point in="1.0000000000000000E003" out="5.0000000000000000E001"/>
    </kpo:curve>
    <kpo:curve id="HIGH" property="A">
        <kpo:point in="-1.0000000000000000E003" out="0.0000000000000000E000"/>
        <kpo:point in="0.0000000000000000E000" out="0.0000000000000000E000"/>
        <kpo:point in="2.5000000000000000E002" out="1.0000000000000000E000"/>
        <kpo:point in="1.0000000000000000E003" out="1.0000000000000000E000"/>
    </kpo:curve>
    <kpo:curve id="MED" property="A">
        <kpo:point in="-1.0000000000000000E003" out="0.0000000000000000E000"/>
        <kpo:point in="-2.5000000000000000E002" out="0.0000000000000000E000"/>
        <kpo:point in="0.0000000000000000E000" out="1.0000000000000000E000"/>
        <kpo:point in="2.5000000000000000E002" out="0.0000000000000000E000"/>
        <kpo:point in="1.0000000000000000E003" out="0.0000000000000000E000"/>
    </kpo:curve>
    <kpo:curve id="LOW" property="A">
        <kpo:point in="-1.0000000000000000E003" out="0.0000000000000000E000"/>
        <kpo:point in="0.0000000000000000E000" out="0.0000000000000000E000"/>
        <kpo:point in="2.5000000000000000E002" out="1.0000000000000000E000"/>
        <kpo:point in="1.0000000000000000E003" out="1.0000000000000000E000"/>
    </kpo:curve>
</kpo:curves>
<kpo:actions>
    <kpo:action name="RNEXT">
        <kpo:precondition>
            <kpo:scenario>
                <kpo:situationobjs>
```

```
                        <kpo:situationobj>
                            <kpo:simulatorExpression valueReturn="1" />
                        </kpo:situationobj>
                    </kpo:situationobjs>
                </kpo:scenario>
            </kpo:precondition>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="SIMSTEP" symbol="+">
                            <kpo:properties>
                                <kpo:property name="SIMTIME" />
                                <kpo:property name="SIMSTOP" />
                            </kpo:properties>
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="A" valueReturn="1"
symbol="+">
                            <kpo:properties>
                                <kpo:property name="A" />
                            </kpo:properties>
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="B">
                            <kpo:properties>
                                <kpo:property name="A" />
                            </kpo:properties>
                            <kpo:curves>
                                <kpo:curv name="funcTEST" />
                            </kpo:curves>
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="HIGH">
                            <kpo:properties>
                                <kpo:property name="A" />
                            </kpo:properties>
                            <kpo:curves>
                                <kpo:curv name="HIGH" />
                            </kpo:curves>
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="MED">
                            <kpo:properties>
                                <kpo:property name="A" />
                            </kpo:properties>
                            <kpo:curves>
                                <kpo:curv name="MED" />
                            </kpo:curves>
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
```

```
            </kpo:effect>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="LOW">
                            <kpo:properties>
                                <kpo:property name="A" />
                            </kpo:properties>
                            <kpo:curves>
                                <kpo:curv name="LOW" />
                            </kpo:curves>
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
        </kpo:action>
        <kpo:action name="RHOT">
            <kpo:precondition>
                <kpo:scenario>
                    <kpo:situationobjs>
                        <kpo:situationobj>
                            <kpo:simulatorExpression property="A" symbol="IS">
                                <kpo:curves>
                                    <kpo:curv name="HIGH" />
                                </kpo:curves>
                            </kpo:simulatorExpression>
                        </kpo:situationobj>
                    </kpo:situationobjs>
                </kpo:scenario>
            </kpo:precondition>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="C" valueReturn="-1">
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
        </kpo:action>
        <kpo:action name="RSTOP">
            <kpo:precondition>
                <kpo:scenario>
                    <kpo:situationobjs>
                        <kpo:situationobj>
                            <kpo:simulatorExpression symbol="GE">
                                <kpo:properties>
                                    <kpo:property name="SIMTIME" />
                                    <kpo:property name="SIMEND" />
                                </kpo:properties>
                            </kpo:simulatorExpression>
                        </kpo:situationobj>
                    </kpo:situationobjs>
                </kpo:scenario>
            </kpo:precondition>
            <kpo:effect>
                <kpo:scenario>
                    <kpo:situationobj>
                        <kpo:simulatorExpression target="SIMSTOP" valueReturn="1">
                        </kpo:simulatorExpression>
                    </kpo:situationobj>
                </kpo:scenario>
            </kpo:effect>
        </kpo:action>
    </kpo:actions>
</kpo:simulator>
</kpo:KPlannOO>
```