sid.inpe.br/mtc-m19/2012/10.29.15.49-TDI

UMA ARQUITETURA DEFINIDA PARA USO DA LINGUAGEM PLUTO EM PROCEDIMENTOS DE TESTE APLICADOS A UM COMPUTADOR DE BORDO PARA SATÉLITES

Thiago Duarte Pereira

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Maurício Gonçalves Vieira Ferreira, aprovada em 17 de dezembro de 2012.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3CT6Q8S>

INPE
São José dos Campos
2012

PUBLICADO POR:

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

sid.inpe.br/mtc-m19/2012/10.29.15.49-TDI

UMA ARQUITETURA DEFINIDA PARA USO DA LINGUAGEM PLUTO EM PROCEDIMENTOS DE TESTE APLICADOS A UM COMPUTADOR DE BORDO PARA SATÉLITES

Thiago Duarte Pereira

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Maurício Gonçalves Vieira Ferreira, aprovada em 17 de dezembro de 2012.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3CT6Q8S>

INPE
São José dos Campos
2012

Dados Internacionais de Catalogação na Publicação (CIP)

Pereira, Thiago Duarte.

P414a Uma arquitetura definida para uso da linguagem pluto em procedimentos de teste aplicados a um computador de bordo para satélites / Thiago Duarte Pereira. – São José dos Campos : INPE, 2012.

xxx + 137 p. ; (sid.inpe.br/mtc-m19/2012/10.29.15.49-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012.

Orientadores : Dr. Maurício Gonçalves Vieira Ferreira.

1. procedimentos de teste. 2. sistemas aeroespaciais. 3. computador de bordo. 4. linguagens de procedimentos. 5. arquitetura de softwares I.Título.

CDU 629.7:004.415.53

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

Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais

Dra. Ana Maria Ambrosio

Presidente / INPE / São José dos Campos - SP

Dr. Mauricio Gonçalves Vieira Ferreira

Orientador(a) / INPE / SJCampos - SP

Dr. Ronaldo Arias

Membro da Banca / INPE / São José dos Campos - SP

Dr. Hassan Ahmad Sidaoui

Convidado(a) / MECTRON / SJCampos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(x) unanimidade

Aluno (a): ***Thiago Duarte Pereira***

São José dos Campos, 17 de Dezembro de 2012

Ao realizar mais de dois mil experimentos até descobrir o filamento exato para o funcionamento da lâmpada incandescente, *Thomas Edison* ao ser interrogado por um jornalista sobre como se sentia ao ter fracassado mais de duas mil vezes, ele simplesmente respondeu: "*_Não fracassei. Eu simplesmente descobri várias alternativas de como não se fazer uma lâmpada. Eu precisei de apenas uma para acendê-la." (Thomas A. Edison, inventor americano)*

"*Conhecer não é suficiente, é preciso aplicar. Desejar não é suficiente, é preciso fazer." (Johann Wolfgang von Goethe, escritor e filósofo alemão)*

Dedico aos que colaboraram direta ou indiretamente,
especialmente à minha família.

AGRADECIMENTOS

Deixei muita gente querida para trás: minha família. A compreensão e incentivo de muitos foi fundamental. Com coragem e esperança, fui à luta. Sou inteiramente grato aos meus amigos e professores Marcelo Galvão e Terezinha Faria. Eles mostraram o caminho!

Agradeço meu orientador, o Doutor Mauricio Gonçalves Vieira Ferreira, por ter me orientado com muitos incentivos, entusiasmo, e muita serenidade. Excepcional! Muito obrigado Doutor Mauricio! A você e aos demais professores, sempre dispostos a transmitir o conhecimento.

Quero agradecer a todos os companheiros de trabalho, integrantes do Grupo SUBORD do INPE, especialmente os Doutores Fabrício e Ronaldo. O apoio em vários aspectos foi muito importante! E àqueles que não mediram esforços, junto aos que também se manifestaram para contribuir com sugestões. Além do apoio financeiro, propiciado pela FINEP via Projeto SIA.

Aos amigos de longa e nova data (amigos de vários cantos do país) e aos que fazem parte dela, transmito a vocês o mais puro e sincero sentimento de gratidão. “Não declaro nomes, porque são vários.” Muitos até já deram continuidade em suas vidas, deixando boas lembranças. Obrigado pela compreensão. Para esses e aos que continuam em sintonia, dentro ou fora do INPE, muito obrigado!

A todos vocês, que acompanharam de perto ou de longe, ou até vivenciaram juntos, declaro convictamente, que as ocorrências e obstáculos em paralelo, são meios que proporcionam oportunidades para o crescimento, principalmente daqueles que buscam o conhecimento e não medem esforços para chegar onde desejam. Tudo são oportunidades, e para mim, essa etapa é apenas o começo. Um abraço a todos!

RESUMO

Testar as funcionalidades de um computador de supervisão de bordo, do inglês *On-Board Data Handling* (OBDH), e controle de atitude e órbita, do inglês *Attitude and Orbit Control* (AOC), para ser embarcado em plataformas orbitais desenvolvidas para satélites é uma atividade de alta complexidade. A disponibilidade de uma linguagem padrão para uso no preparo e execução de procedimentos de teste pode facilitar o gerenciamento dos testes. A cooperação européia responsável pela criação de padrões voltados para a área espacial, chamada *European Cooperation for Space Standardization* (ECSS), especificou uma linguagem com objetivo de padronizar o preparo e a execução de procedimentos de teste e de operação de sistemas e subsistemas espaciais, denominada *Procedure Language for Users in Test and Operations* (PLUTO). A utilização dessa linguagem requer a existência de alguns recursos, classificados essenciais para o preparo e a execução de tais procedimentos. Portanto, a presente dissertação aborda quais são esses recursos e apresenta a proposta de uma arquitetura computacional para gerenciar o preparo e a execução de procedimentos de teste durante o desenvolvimento de sistemas para aplicações espaciais. Para exercitar essa arquitetura e demonstrar como se utiliza a PLUTO, um protótipo, denominado *Spacecraft Test Procedures System* (STEPS) foi desenvolvido a partir de um estudo de caso, contextualizado no algoritmo de controle da atitude de uma plataforma orbital estabilizada em três eixos. Esse algoritmo é gerenciado pelo software de vôo de um computador de bordo de plataformas orbitais desenvolvidas para satélites.

A DEFINED ARCHITECTURE TO PLUTO LANGUAGE USAGE ON TESTS PROCEDURES APPLIED TO A SATELLITE'S ON-BOARD COMPUTER

ABSTRACT

The functional tests of an on-board computer embedded in orbital platforms are deemed to be highly complex tasks, especially when a standard language, appropriate for procedures preparation and execution is not available. Usually, test management is done under the availability of a language to standardize the preparation and execution of the procedures and facilitates test activities performed in the development during the space mission's life-cycle. The European Cooperation for Space Standardization (ECSS), in order to standardize the tests procedures preparation and executions and systems and subsystems operation, has proposed a Language called Procedure Language for Users in Test and Operations (PLUTO) filled out with some resources restrains essential for its usage. This master's thesis shows what these resources are and presents a proposed architecture to manage the PLUTO tests procedures preparation and executions during the systems and subsystems development for space applications. A Spacecraft Test Procedures System (STEPS) prototype was developed to present the language and architecture usage. A case study was defined in the context of an attitude control algorithm for a three-axis stabilization orbital platform. This algorithm is managed by the on board computer flight software designed to satellites orbital platforms.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

ACDH	*Attitude Control and Data Handling*
ACS	*Attitude Control System*
ADM-AEOLUS	*Atmospheric Dynamics Mission - Aeolus*
ADVS	Algoritmo de Determinação do Vetor Sol
AIT	*Assembly, Integration and Test*
AOCS	*Attitude and Orbit Control Subsystem*
ASI	*Agenzia Spaziale Italiana*
CAST	*Association for Science and Technology*
CBERS	*China-Brazil Earth Resources Satellite*
CDR	*Critical Design Review*
CRI	*Computer Resources International*
CSA	*Canadian Space Agency*
DCTA	Departamento de Ciência e Tecnologia Aeroespacial
DEA	Divisão de Eletrônica Aeroespacial
DSV	*Domain Specific View*
ECSS	*European Cooperation for Space Standardization*
EGSE	*Electrical Ground Support Equipment*

ESA	*European Space Agency*
ESTEC	*European Space Research and Technology Centre*
ETDP	*Exploration Technology Development Program*
ETE	Engenharia e Tecnologia Espaciais
ETOL	*European Test Operations Language*
FAR	*Flight Acceptance Review*
FUSE	*Far Ultraviolet Spectroscopic Explorer*
GMV	*GMV Space Systems, Inc.*
GOCE	*Gravity Field and Steady-State Ocean Circulation Explorer*
GPM-Br	*Global Precipitation Measurement Brazil*
GPS	*Global Positioning System*
GUI	*Graphical User Interface*
ICS	*Interface & Control Systems, Inc.*
IDE	*Integrated Development Environment*
IEC	*International Electrotechnical Commission*
IHC	Interface Homem Computador
INPE	Instituto Nacional de Pesquisas Espaciais
ISO	*International Organization for Standardization*
ISS	*International Space Station*

LIT	Laboratório de Integração e Teste
MAPSAR	*Multi-Application Purpose with Synthetic Aperture Radar*
MATIS	*Mission Automation System*
MCS	*Mission Control System*
MCT	Ministério de Ciência e Tecnologia
MDR	*Mission Definition Review*
NASA	*National Aeronautics and Space Administration*
NIS	*Network Interface System*
OBDH	*On-Board Data Handling*
P&D	Pesquisa & Desenvolvimento
PDR	*Preliminary Design Review*
PEB	Programa Espacial Brasileiro
PMM	Plataforma Multimissão
PLUTO	*Procedure Language for Users in Test and Operations*
PRL	*Procedure Representation Language*
PRR	*Preliminary Requirements Review*
PUS	*Packet Utilization Standard*
QR	*Qualification Review*
RADARSAT	*Radar Satellite*

RCP	*Rich Client Platform*
RD	*Report Data*
ROSAT	*Röntgensatellit*
SCL	*Spacecraft Command Language*
SCOS	*Satellite Control and Operation System*
SE	*System Element*
SES	*Société Européenne des Satellites*
SIA	Sistemas Inerciais para Aplicações Aeroespaciais
SMF	*Services Management Framework*
SPELL	*Satellite Procedure Execution Language and Library*
SRR	*System Requirements Review*
SSM	*Space System Model*
STEPS	*Spacecraft Test Procedures System*
STOL	*Satellite Test and Operations Language*
SUBORD	Supervisão de Bordo
SUT	*System Under Test*
TC	Telecomando
TCL	*Tool Command Language*
Tk	*Toolkit*

TM	Telemetria
UML	*Unified Modeling Language*
VEGA	*Vettore Europeo di Generazione Avanzata*
VLS	Veículo Lançador de Satélites
XML	*eXtensible Markup Language*

SUMÁRIO

Pág.

1 INTRODUÇÃO

A Coordenação de Engenharia e Tecnologias Espaciais (ETE) do Instituto Nacional de Pesquisas Espaciais (INPE) trabalha no desenvolvimento de produtos e soluções aplicáveis ao setor espacial para cumprimento de objetivos estabelecidos no quadro das missões espaciais brasileiras. Tais missões são vinculadas ao Programa Espacial Brasileiro (PEB), que está focado na busca por benefícios à sociedade e no aumento da autonomia do Brasil nesse setor.

Com o propósito de alcançar pleno domínio em tecnologias espaciais, o INPE vem trabalhando em uma série de projetos voltados para o campo da tecnologia de satélites (INPE, 2011). De acordo com (LARSON; WERTZ, 1999), a plataforma orbital de um satélite é composta por vários subsistemas que desempenham diversas funcionalidades. Cada subsistema é formado por diferentes equipamentos, necessários durante a execução de suas funcionalidades, para o cumprimento dos objetivos do satélite no espaço.

Um Subsistema de Supervisão de Bordo, por exemplo, mantêm como principal equipamento o computador de bordo e suas interfaces de comunicação com os demais subsistemas. Outro subsistema é o de Controle de Atitude e Órbita, que para manter o satélite em sua devida órbita e manter seu apontamento com precisão, necessita de um conjunto de equipamentos de aquisição de dados de referência para o controle de seu posicionamento e de um conjunto de equipamentos para atuação nas manobras realizadas durante sua operação.

Dentre exemplos de equipamentos de aquisição de dados encontram-se o sistema de posicionamento global, o *Global Positioning System* (GPS); os sensores de captação de luz solar (sensor solar); os sensores utilizados no processo de reconhecimento posicional de estrelas (sensor de estrelas), etc. Os equipamentos de atuação estão diretamente ligados à realização das manobras do satélite no espaço, como rodas de reação e propulsores.

Cada equipamento costuma ser desenvolvido separadamente por diferentes equipes, onde cada equipe é composta de um conjunto de profissionais qualificados na área específica do subsistema. Dessa forma, gerenciar as atividades realizadas durante todo o ciclo de vida do satélite, de modo que resista tanto aos efeitos do lançamento, quanto à operação no ambiente espacial, requer esforço coletivo em grande escala de diversas áreas do conhecimento, nas diversas fases do ciclo de vida da missão espacial.

Durante as fases de desenvolvimento, conjuntos de procedimentos de teste são executados em vários níveis de abstração com o propósito de validar o funcionamento do satélite em nível de sistema, subsistema e equipamento. Níveis esses, que vão desde o desenvolvimento unitário de cada subsistema, onde seus equipamentos devem ser testados e integrados, até o nível da integração final do satélite como um todo, ocorrida na fase denominada Montagem Integração e Teste, do inglês *Assembly Integration and Test* (AIT).

Para efeito de simplificação do texto, o termo ‘procedimento’ é utilizado para descrever procedimentos de testes e de operação de sistemas, de subsistemas e de equipamentos espaciais. O Glossário, pág. 111, apresenta a definição de alguns termos utilizados na presente dissertação.

Embora sejam aplicados em etapas e contextos diferentes, os procedimentos de teste têm como objetivo comum validar seu funcionamento e encontrar potenciais erros de projeto, de fabricação ou de integração. Uma forma de facilitar a geração e aplicação dos procedimentos de teste em várias etapas e em diversos níveis é por meio da sua padronização. Um único procedimento, se descrito de maneira padronizada, pode ser aplicado em diferentes cenários, por diversas vezes e até mesmo em diferentes missões.

Uma alternativa que pode ser aplicada nesse tipo de padronização é o uso de uma linguagem comum. Agências espaciais internacionais, como a Agência Espacial Européia, do inglês *European Space Agency* (ESA), bem como empresas do ramo espacial, por exemplo, como a *Astrium*, a *Rhea Group* e a

Vitrociset, também ligadas à ESA, estão explorando potenciais linguagens e criando soluções tecnológicas para suporte à descrição e execução automatizada de seus procedimentos.

No Brasil, o Laboratório de Integração e Teste (LIT) do INPE, desde a década de 1980 utiliza uma linguagem chamada *European Test Operations Language* (ETOL), para descrição e execução de procedimentos de teste nas etapas de AIT do satélite *China-Brazil Earth Resources Satellite* (CBERS).

A ETOL foi desenvolvida pela ESA em 1970, (MELTON; HUBSCHER, *et al*., 1996), e apresenta algumas limitações decorrentes do avanço tecnológico dos últimos 40 anos. Portanto, é recomendável que o INPE atualize o formato padrão de criação e execução de seus procedimentos de teste aplicados em atividades de AIT já nas fases de desenvolvimento dos equipamentos dos subsistemas de seus satélites.

Cada grupo do INPE ou empresa responsável pelos diferentes equipamentos do satélite adota seu próprio recurso de testes, do tipo *ad-hoc*, ou seja, não utilizam e nem seguem algum formato padronizado. Se padronizados e aplicados desde as fases de desenvolvimento individual de cada subsistema ou equipamento, eles podem ser reutilizados nas fases de AIT, nos planos de operações da missão em vôo, e até mesmo em missões posteriores.

Outro fato é que a padronização também permite o desenvolvimento de soluções computacionais para que tais procedimentos sejam executados automaticamente.

Sendo executados automaticamente, equipamentos e subsistemas que dependem do tempo para inicializar, agendar, sincronizar ou finalizar uma tarefa (sistemas de tempo real) poderão ser testados com mais facilidade, maior precisão e maior rapidez, além de diminuir a possibilidade de erros provenientes de recursos humanos durante a aplicação dos testes.

A cooperação européia responsável pela criação de padrões voltados para a área espacial, em inglês, chamada de *European Cooperation for Space Standardization* (ECSS), foi criada em 1993 com objetivo de definir um conjunto comum de padrões de engenharia, qualidade e gestão para uso em organizações vinculadas ao espaço, permitindo assim a redução de custos, aumento na produtividade e também uma melhor cooperação entre instituições e empresas.

Alguns padrões da ECSS vêm sendo adotados em alguns projetos do INPE, cujo objetivo é a busca pela melhoria em seus processos de gestão e atividades de engenharia, onde o foco está na organização do desenvolvimento e qualificação de seus produtos e soluções.

Além dos padrões que já vem sendo utilizados pelo INPE, existe também um padrão de engenharia, com especificações de uma linguagem formal, que pode ser usada para a descrição padronizada e execução automatizada de procedimentos de teste e operação de satélites.

O uso dessa linguagem depende da existência de um modelo do sistema que será testado ou operado, por isso, a ECSS também disponibiliza outro padrão que especifica uma forma de definir e estruturar os dados desse modelo.

As especificações dessa linguagem são de caráter público e a mesma pode ser usada por qualquer instituição ou empresa do setor espacial para construção de soluções tecnológicas de apoio à aplicação de procedimentos tanto de teste quanto de operação de satélites.

1.1. Motivação

Os testes do satélite ou de seus subsistemas são realizados por meio do equipamento comumente denominado *Electrical Ground Support Equipment* (EGSE). O EGSE é um equipamento desenvolvido com propósito de realizar testes funcionais, elétricos e de desempenho em sistemas, subsistemas e

equipamentos de uma plataforma orbital, como o computador de supervisão de bordo, equipamento pertencente ao Subsistema de Supervisão de Bordo.

A preparação e execução dos procedimentos de teste são gerenciadas por intermédio do EGSE, que na maioria dos casos conta com apoio de algum sistema computacional para facilitar o preparo e o monitoramento da execução de tais procedimentos.

A utilização de uma linguagem permite a descrição padronizada e a execução automatizada desses procedimentos em todas as fases de desenvolvimento do satélite. Atualmente, o Grupo de Supervisão de Bordo (SUBORD) da Divisão de Eletrônica Aeroespacial (DEA) do INPE, por exemplo, não utiliza uma linguagem padrão para especificação dos procedimentos de teste de seus equipamentos.

Para atender a essa necessidade, existe um padrão disponibilizado pela ECSS que pode ser utilizado, chamado de Linguagem de Procedimento de Teste e Operações, do inglês *Test and Operations Procedure Language* (ECSS, 2008), que especifica uma Linguagem de Procedimento para Usuários em Teste e Operações, que em inglês é denominada *Procedure Language for Users in Test and Operations* (PLUTO).

Essa linguagem permite a padronização da descrição e execução automatizada de procedimentos de teste e de operação. No entanto, para sua adoção é necessário a existência de um modelo do Sistema sob Teste, do inglês *System Under Test* (SUT), e a existência de um interpretador.

O padrão chamado Sistemas de Solo e Operações – Definição dos Dados de Monitoramento e Controle, em inglês chamado de *Ground Systems and Operations - Monitoring and Control Data Definition* (ECSS, 2003), especifica uma variedade de tipos e estruturas de dados para uso na definição desse modelo.

O padrão também especifica uma forma de manter os dados estruturados, de maneira hierárquica, para refletir seu real funcionamento. O nome atribuído a essa estrutura é Modelo do Sistema Espacial, que do inglês é denominada *Space System Model* (SSM). Para utilizar os conceitos do SSM é necessário desenvolver um mecanismo computacional que possibilite gerenciar seu preparo e sua utilização junto à PLUTO.

A PLUTO é interpretada, mas até o presente momento não existe um interpretador completo disponível para uso. A ESA tem interesse pela linguagem e possui um interpretador, mas ainda é limitado. O mesmo encontra-se em desenvolvimento e se apresenta incompleto para o atendimento a uma missão. O fato de já possuir uma estrutura de apoio ao desenvolvimento de suas missões, faz com que a ESA mantenha a PLUTO para fins experimentais.

Contudo, para adotar essa linguagem como padrão de preparo e execução de procedimentos é necessário que se tenha uma estrutura bem definida, capaz de manter os seguintes mecanismos:

1. Um interpretador da PLUTO;
2. Um gerenciador do modelo (para criação e execução) do sistema ou equipamento que será testado;
3. Ferramentas computacionais para apoio ao preparo e monitoramento da execução; e
4. Interfaces de comunicação entre o interpretador, o gerenciador do modelo em execução e o SUT.

Uma das motivações é que a partir da definição de uma arquitetura composta por esses mecanismos será possível desenvolver um sistema computacional para o preparo padronizado e a execução automatizada dos procedimentos de teste.

Se os procedimentos de teste forem padronizados por meio da PLUTO, será possível reduzir esforço, tempo e investimentos, que hoje são destinados ao desenvolvimento de diferentes ambientes e ferramentas computacionais, que na maioria dos casos são utilizados em uma única missão.

Por sua vez, um ambiente que permita a padronização dessas atividades pode ser reutilizado nos diferentes cenários e até mesmo em diferentes missões que adotarem o mesmo padrão de linguagem, pois os procedimentos poderão ser executados durante todo o ciclo de desenvolvimento dos equipamentos.

Outro fato motivador é que a execução automática de procedimentos de teste permite maior precisão e rapidez na aplicação e análise dos resultados dos testes. A execução manual dos procedimentos de testes de sistemas de tempo real pode ser inapropriada, devido ao tempo de estímulo e resposta do SUT, que pode variar em até frações de segundos.

1.2. Objetivo

O objetivo da presente dissertação é definir uma arquitetura que possua os recursos necessários para o desenvolvimento de um sistema computacional capaz de permitir o preparo e a execução automática de procedimentos de teste, utilizando a PLUTO como padrão de linguagem, para um computador de bordo desenvolvido para satélites.

Essa arquitetura deve ser capaz de:

- Permitir a descrição de procedimentos na sintaxe da linguagem;
- Permitir a modelagem do SSM em conformidade com o padrão ECSS-E-ST-70-31C, (ECSS, 2003);
- Interpretar os procedimentos para que sejam executados automaticamente;

- Gerenciar os dados do SSM de modo que possam ser acessados pelo interpretador durante a execução;
- Monitorar e analisar resultados da execução dos procedimentos.

1.3. Metodologia

A Figura 1.1 ilustra a metodologia da pesquisa realizada na presente dissertação.

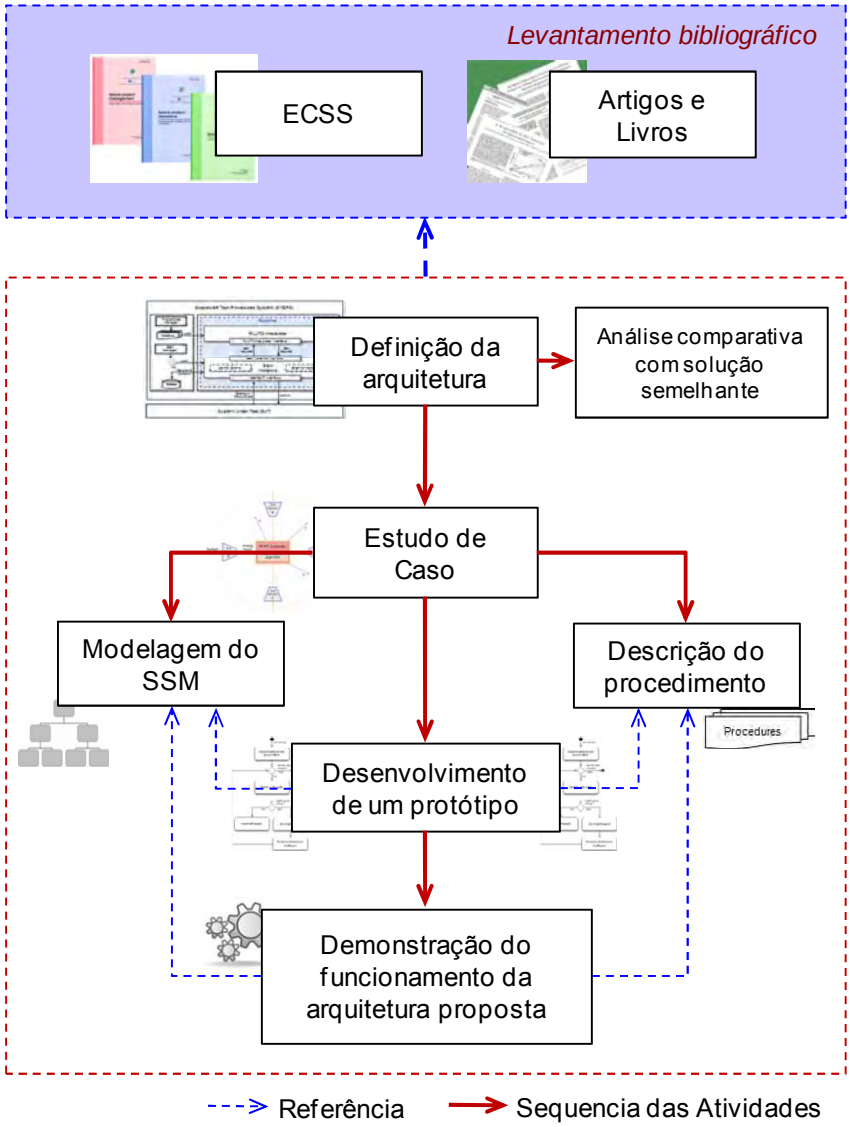

Figura 1.1 - Metodologia da Pesquisa

A presente dissertação é iniciada com o levantamento bibliográfico, a fim de se chegar ao devido embasamento teórico. Fazem parte do levantamento bibliográfico, padrões especificados pela ECSS e trabalhos correlatos encontrados na literatura. O levantamento realizado permitiu o estudo das características da PLUTO, do SSM e das tecnologias atuais existentes na presente área.

O fato de a área espacial contar com a presença de outros tipos de linguagens destinadas a procedimentos de testes possibilitou a realização de uma pesquisa com o propósito de apresentar as características e formas de disponibilização das linguagens de testes mais utilizadas na área espacial. As fases do ciclo de vida de uma missão espacial que possibilitam o uso de alguma linguagem também são apresentadas.

Com o embasamento teórico, a arquitetura, proposta na Seção 1.2, é definida para o desenvolvimento do sistema computacional definido como STEPS. Um estudo sobre a estrutura de preparo e automatização da execução de procedimentos da ESA é apresentado com objetivo de fazer uma análise comparativa entre o seu sistema computacional usado para a execução automática de procedimentos, do inglês *Automatic Schedule Execution* (ASE) e a arquitetura proposta na presente dissertação.

Como forma de exercitar e demonstrar o funcionamento da arquitetura proposta, um protótipo é desenvolvido a partir de um estudo de caso. O estudo de caso está inserido no contexto de uma funcionalidade de um computador de bordo, desenvolvido pelo Grupo SUBORD nas instalações da Divisão de Eletrônica Aeroespacial (DEA) do INPE para ser embarcado em satélites. Esse computador faz parte do projeto chamado Sistemas Inerciais para Aplicações Aeroespaciais (SIA).

O modelo SSM e o procedimento de teste na sintaxe da PLUTO são criados a partir do estudo de caso. Ambos foram criados para demonstrar como um procedimento deve ser descrito e associado ao SSM e quais os motivos da

necessidade de um modelo. Em seguida, o protótipo é desenvolvido para a demonstração do funcionamento da arquitetura.

1.4. Organização do Trabalho

O Capítulo 2 mostra em quais atividades e fases do ciclo de vida da missão espacial um padrão de linguagem pode ser utilizado. Esse capítulo também apresenta as características das linguagens mais predominantes em atividades espaciais no mundo.

As características da PLUTO, do SSM e os motivos da necessidade de um modelo do sistema, subsistema ou equipamento para uso dessa linguagem são apresentadas no Capítulo 3.

O Capítulo 4 apresenta a estrutura de preparo e automatização da execução de procedimentos da ESA e o que está sendo feito para que a ESA tenha uma estrutura unificada.

A arquitetura definida para o desenvolvimento do STEPS é apresentada no Capítulo 5. O desenvolvimento, a utilização e demonstração do protótipo do STEPS junto ao estudo de caso são apresentados no Capítulo 6.

Para finalizar, no Capítulo 7 são apresentadas as principais contribuições, trabalhos publicados durante a pesquisa, os trabalhos futuros que poderão dar continuidade à pesquisa sobre a PLUTO e considerações finais.

2 LINGUAGENS ESPECIFICADAS PARA TESTES E OPERAÇÃO DE SATÉLITES EM MISSÕES ESPACIAIS

Durante as fases do ciclo de vida de uma missão espacial, várias atividades são realizadas para garantir que o satélite execute suas funcionalidades de maneira que possa atingir os objetivos da missão. Agências internacionais e empresas que atuam na área espacial vêm desenvolvendo linguagens para descrição e execução dos procedimentos aplicados a essas atividades.

Na presente dissertação, o termo 'atividades espaciais', refere-se às atividades de teste e operação de satélites. No entanto, é importante ressaltar que a presente dissertação enfatiza as atividades de teste. As atividades de operação também serão colocadas em alguns trechos para expor o potencial de abrangência da PLUTO.

Esse Capítulo sumariza uma pesquisa realizada sobre as características das linguagens mais utilizadas na área espacial. São também apresentadas as fases e atividades do ciclo de vida de uma missão espacial que viabilizam e/ou requerem o emprego de alguma dessas linguagens.

2.1. Uso de Linguagens de Especificação de testes durante o Ciclo de Vida de uma Missão Espacial

Existem diferentes técnicas e padrões aplicáveis em atividades espaciais que contribuem para a validação e operação de satélites e espaçonaves. No contexto dos padrões da ECSS, além dos padrões de engenharia, existem conjuntos de padrões que também são aplicáveis aos processos de gestão do ciclo de vida da missão.

Alguns projetos espaciais do INPE são desenvolvidos a partir das recomendações especificadas por esses padrões. A organização do ciclo de vida de suas missões é feita a partir da metodologia apresentada no padrão *Space project management – Project planning and implementation* presente em

ECSS-M-ST-10C, (ECSS, 2009). Esse padrão destaca seis grupos de atividades distribuídas em sete fases: 0, A, B, C, D, E e F, onde uma atividade ou conjunto de atividades é atribuído para cada fase. A descrição dos objetivos e atividades de cada fase é apresentada na ilustração da Figura 2.1.

As linguagens especificadas para uso em atividades espaciais que são executadas por meio de procedimentos, como as de teste ou de operação, podem ser utilizadas durante as fases: C e D, produção, montagem, integração e teste; e E, as operações do satélite.

As atividades desempenhadas no decorrer dessas três fases viabilizam o uso de algum padrão de linguagem de preparo e execução de procedimentos de testes e operação. Dentre essas atividades, destacam-se: Desenvolvimento e validação dos procedimentos de teste; Testes dos subsistemas e seus equipamentos; Integração e testes do sistema; Desenvolvimento e verificação dos procedimentos de voo; etc.

A fase E abrange as atividades de lançamento e operação. Essa deve manter um plano de voo, que dependendo da missão pode ser complexo, por isso necessita de recurso ágil e que facilite a descrição das operações. Fazendo uso de um padrão de linguagem desde as fases iniciais do ciclo de vida da missão, proporcionará agilidade, pelo fato de que grande parte dos procedimentos criados durante as atividades de teste poderão ser reutilizadas na operação do satélite.

Esse mesmo padrão também recomenda uma série de revisões formais para análises e verificações durante o processo de desenvolvimento do satélite ou de algum subsistema do mesmo no decorrer do ciclo de vida da missão. Ao finalizar determinada etapa, a equipe de desenvolvimento prosseguirá para a(s) etapa(s) posteriores caso a respectiva etapa atender aos requisitos definidos para ela.

Fases 0/A		Fases B/C	Fases C/D		Fase E
Concepção da missão e projetos compatíveis com a carga útil	Definição do projeto da carga útil e da espaçonave	Refinamento e verificação do projeto	Produção, montagem, integração e teste		Operação da espaçonave
• Definição dos objetivos da missão • Definição de uma *baseline* da missão e alternativas • Análise dos requisitos iniciais • Documentação	• Análise dos requisitos da carga útil • Definição de conceitos alternativos para a carga útil • Análise dos requisitos da trajetória/órbita • Documentação padronizada	• Refinamento e verificação de projeto do sistema • Desenvolvimento e verificação do sistema e especificação dos equipamentos • Projeto funcional e verificação da performance • Projeto das interfaces • Orçamento	• Subcontratação da manufatura de componentes • Projeto detalhado dos componentes e layout do sistema • Desenvolvimento e teste do EGSE • Desenvolvimento e verificação do software de bordo • Desenvolvimento e validação dos procedimentos de teste • Testes dos subsistemas e seus equipamentos	• Verificação de software • Integração e teste do sistema • Validação da performance operacional e funcional • Desenvolvimento e verificação dos procedimentos de voo	• Validação do segmento solo • Treinamento dos operadores • Lançamento • Calibração da carga útil • Avaliação da performance • Suporte à resolução de problemas

Figura 2.1 - Atividades realizadas durante as fases do ciclo de vida de uma missão espacial

Fonte: adaptada de Eickhoff (2012)

A Figura 2.2 ilustra o agrupamento das atividades em função de cada fase e apresenta o momento recomendado para cada revisão durante o desenvolvimento de cada atividade durante o ciclo de vida da missão (ECSS, 2009).

As revisões recomendadas pela ECSS são: Revisão da definição da missão, do inglês *Mission Definition Review* (MDR); Revisão dos requisitos preliminares, do inglês *Preliminary Requirements Review* (PRR); Revisão dos requisitos de sistema, do inglês *System Requirements Review* (SRR); Revisão preliminar do projeto, do inglês *Preliminary Design Review* (PDR); Revisão crítica do projeto, do inglês *Critical Design Review* (CDR); Revisão da qualificação, do inglês *Qualification Review* (QR); e Revisão da aceitação para vôo, do inglês *Flight Acceptance Review* (FAR).

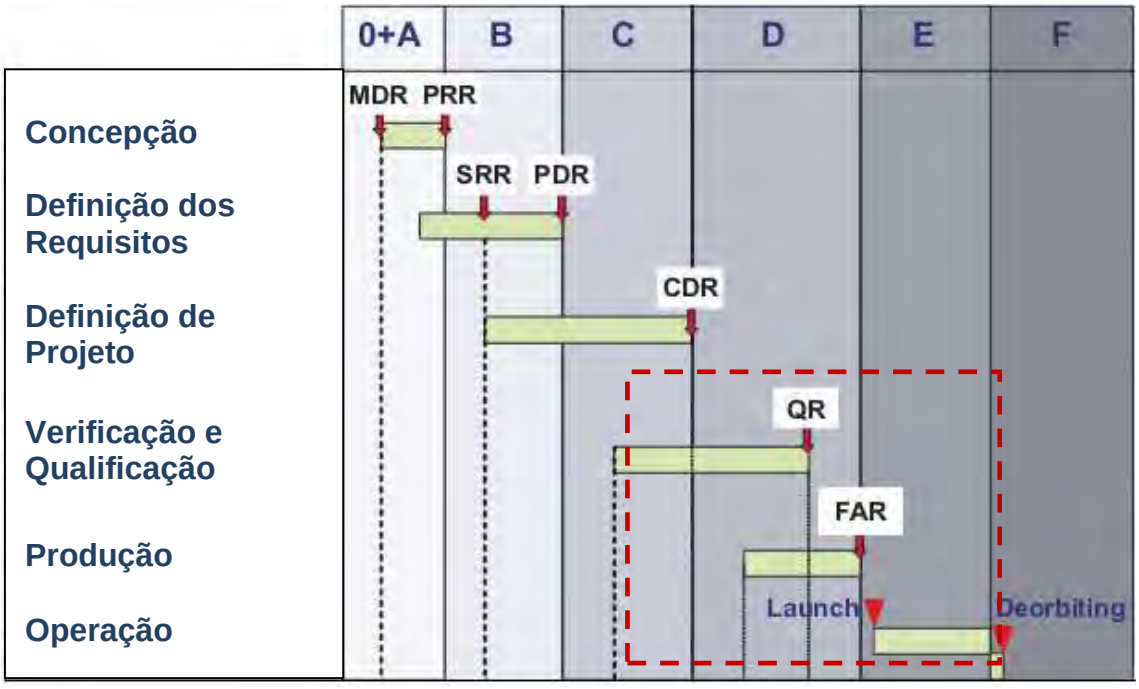

Figura 2.2 – Fases e revisões do ciclo de vida de uma missão espacial que possibilitam o uso de alguma linguagem de especificação de procedimentos de teste e operação.

Fonte: adaptada de (ECSS, 2009)

O mesmo padrão também recomenda o desenvolvimento de diferentes modelos do sistema espacial, como Modelo de Engenharia, Modelo de Qualificação e Modelo de Voo. Esses modelos são definidos para garantia da qualidade e confiabilidade do subsistema ou equipamento. As linguagens especificadas para procedimentos também podem ser utilizadas como recurso de apoio nas atividades de validação ou operação desses modelos.

2.2. Linguagens usadas em Procedimentos de Atividades Espaciais

A abordagem descrita nessa Seção está relacionada com linguagens que podem ser aplicadas, tanto para atividades de teste quanto para atividades de operação.

Há, de fato, vários benefícios para o aumento da utilização de alguma linguagem nessas atividades, pois agências e empresas podem aderir entre uma variedade de linguagens. Existem alguns fatores que podem influenciar na escolha da linguagem mais apropriada, como por exemplo, a maneira de disponibilização, que pode se distinguir, por exemplo, entre pública e proprietária.

2.2.1. European Test Operations Language (ETOL)

A ESA desenvolveu a *European Test and Operations Language* (ETOL), lançada em 1970, como principal recurso de auxílio às atividades de AIT e pré-lançamento de seus satélites. Essa linguagem é de gênero comercial e é utilizada por outros órgãos e empresas da área espacial.

O sistema de execução de procedimentos descritos com a ETOL cobre as seguintes funcionalidades: monitoramento dos dados de Telemetria (TM), monitoramento da execução de Telecomando (TC), execução sequencial de teste automático e exibição de *logs* para acompanhamento do *status* da execução do procedimento.

A ETOL foi utilizada nos seguintes projetos da área espacial:

- A sonda interplanetária *Ulysses*, lançada em 1990 e desenvolvida pela *National Aeronautics and Space Administration* (NASA) e ESA;

- A sonda *Giotto*, desenvolvida pela ESA com propósito de pesquisar o cometa Halley de perto, com lançamento realizado em 1985; e

- O telescópio alemão *Röntgensatellit* (ROSAT), lançado em 1990 e operado até 1999; e

- Tem sido utilizada na fase de AIT dos satélites do programa *China-Brazil Earth Resources Satellite* (CBERS);

Uma cooperação entre a *European Space Research and Technology Centre* (ESTEC) da ESA e o INPE foi estabelecida em meados de 1980, com objetivo da transferência de conhecimento. Nessa época, as atividades foram realizadas nas instalações da *Data Handling Division* da ESTEC e por meio da experiência adquirida em processos, metodologias de desenvolvimento e ferramentas aplicadas pela ESA, o INPE decidiu adotar o padrão da linguagem ETOL.

O contrato de aquisição de sua licença foi fechado com a empresa europeia *Computer Resources International* A/S (CRI), vinculada à ESA (MELTON; HUBSCHER, *et al*., 1996).

A ETOL está sendo usada em atividades de AIT dos satélites do programa CBERS desde 1994. O programa CBERS, iniciado em 1988, é uma cooperação entre o INPE e a *China Association for Science and Technology* (CAST) da China, para o desenvolvimento de uma série de satélites de monitoramento dos recursos naturais (sensoriamento remoto) do meio ambiente terrestre (WANG; CUI, *et al*., 2011).

Da maneira que foi projetada, a ETOL não possibilita reutilização dos procedimentos em outras missões do mesmo gênero, por isso a ESA, em

cooperação com a indústria espacial europeia, está trabalhando na construção de novas tecnologias.

Além disso, a evolução dos sistemas espaciais e o aumento das atividades de testes e operações de vôo também motivam a busca por novos desafios com o objetivo de garantir a qualidade do produto e para redução de custo nos programas espaciais.

2.2.2. Satellite Procedure Execution Language and Library (SPELL)

A *Satellite Procedure Execution Language and Library* (SPELL) foi desenvolvida pela *Société Européenne des Satellites* (SES) com o apoio da *GMV Space Systems, Inc.* (GMV). Seu desenvolvimento foi baseado em 25 anos de experiência da SES com as atividades de operação de satélites.

Essa linguagem é mantida sob domínio *open-source*, juntamente com um conjunto de componentes de software para suporte na preparação e execução dos procedimentos e vem sendo usada desde 2009 na automatização de procedimentos de voo de satélites geoestacionários.

Dentre esses componentes de software, encontram-se:

- Um ambiente para execução dos procedimentos em tempo-real com interfaces de comunicação com diferentes equipamentos de solo;
- Uma *Graphical User Interface* (GUI) baseada em *Eclipse Rich Client Platform* (*Eclipse* RCP) para monitoramento da execução dos procedimentos;
- Uma biblioteca da SPELL com extensão à sintaxe da linguagem *Python*;
- Um ambiente de desenvolvimento dos procedimentos, também baseado em *Eclipse* RCP; e
- Um mecanismo para controle integrado de revisão dos procedimentos.

Python foi disponibilizada em 1991. É uma linguagem interpretada e de alto nível. Todas as regras aplicáveis à linguagem *Python* são aplicáveis à SPELL, a diferença entre ambas encontra-se na existência de funcionalidades na linguagem SPELL, disponíveis em sua biblioteca, e que foram desenvolvidas para permitir interação com sistemas de controle de solo.

A linguagem SPELL é resultado de uma iniciativa voltada à busca da padronização de procedimentos e se mantém em constantes melhorias. Essa linguagem foi originalmente concebida e projetada para ser aplicada a procedimentos de voo (MORELLI; BOULEAU, 2010).

2.2.3. Satellite Test and Operations Language (STOL)

A linguagem *Satellite Test and Operations Language* (STOL), proposta em meados de 1970, é de alto nível, interpretada e mantida sob domínio proprietário pela NASA. Essa linguagem permite a comunicação com o satélite por meio de procedimentos tanto de teste quanto de operação. Sua sintaxe também é semelhante à sintaxe da linguagem *Python*.

Como o desenvolvimento do interpretador da linguagem STOL se deu a partir da linguagem *Python*, os engenheiros escreveram procedimentos na sintaxe *Python* em formato de comentários, e no decorrer do desenvolvimento as instruções do procedimento, que estavam comentados, foram meios de orientação para a implementação das rotinas de interpretação dos procedimentos em STOL. Os comentários eram a referência durante a simulação da execução de um procedimento STOL no decorrer de seu desenvolvimento (HAUCK; FINNIGAN, 2003).

2.2.4. Tool Command Language (TCL)

A *Tool Command Language* (TCL), criada nos anos 90, é uma linguagem especificada para permitir o desenvolvimento de aplicações com extensão às suas características e funcionalidades. A mesma provê facilidades de

programação genérica que são úteis para uma variedade de aplicações (OUSTERHOUT, 1993).

A NASA utiliza a TCL em suas atividades de teste e a ESA a utiliza em seu *Satellite Control and Operation System 2000* (SCOS-2000) para o preparo e execução de procedimentos operacionais de seus satélites. O funcionamento do SCOS-2000 é apresentado na Seção 4.2.

A TCL pode ser embarcada na aplicação. Seu interpretador foi desenvolvido como uma biblioteca de procedimentos na linguagem C, que pode ser incorporada dentro dessas aplicações. Cada aplicação pode estender as características do TCL *core*, com comandos adicionais específicos daquela aplicação.

Durante o desenvolvimento de uma aplicação, o interpretador da TCL permite a inclusão de novos comandos. A nova aplicação poderá manter seus comandos próprios, além dos providos pelo TCL *core*. O desenvolvedor define sua aplicação por meio de uma estrutura na linguagem C e em seguida adiciona os novos comandos na sintaxe da linguagem TCL dentro da aplicação.

A TCL vem acompanhada do *toolkit* denominado Tk. A junção da TCL e Tk provê um sistema de programação para desenvolvedores e usuários de aplicações com uma *Graphical User Interface* (GUI) para o Sistema Operacional *Microsoft Windows*.

2.2.5. Spacecraft Command Language (SCL)

A *Spacecraft Command Language* (SCL) foi desenvolvida pela empresa chamada *Interface & Control Systems, Inc*. (ICS) para permitir a descrição, teste e operação de sistemas de controle para aplicação em todo ciclo de vida da missão (MIMS, 2008).

Essa linguagem tem sido utilizada em diversas missões e programas, como a missão *Clementine* e o programa *Far Ultraviolet Spectroscopic Explorer* (FUSE), ambos concebidos pela NASA.

2.2.6. Procedure Representation Language (PRL)

A NASA realiza a concepção de missões tripuladas. Isso requer que seus procedimentos de voo sejam rigorosamente padronizados. Essa mesma rigorosidade, na maioria dos casos, também se estende às missões não-tripuladas. A vantagem disso é que seus procedimentos garantem a qualidade mesmo nas missões não-tripuladas.

Seus procedimentos definem cuidadosamente o que deve ser feito e quais as medidas a serem tomadas entre uma variedade de objetivos. Entre esses objetivos, encontram-se alteração dos modos de operação dos equipamentos, inicialização e finalização da operação dos componentes ou subsistemas e realização de várias atividades envolvendo também a intervenção de recursos humanos durante o percurso da espaçonave no espaço.

Os procedimentos podem incluir sequências de comandos normais para alteração do estado do equipamento a bordo de acordo com algum dado de *housekeeping* ou mudança de rotina na lógica do comportamento da execução de algum procedimento.

Segundo (LARSON; WERTZ, 1999) dados de *housekeeping* refletem o status do sistema espacial, tanto plataforma quanto carga-útil e são gerados pelo próprio sistema e enviados para solo. Esses dados são usados para controle e monitoramento do sistema.

Em determinadas situações, os procedimentos de voo das missões tripuladas também são executados manualmente. Para atender esse requisito, a NASA desenvolveu e mantém em constantes atualizações, a linguagem chamada

Procedure Representation Language (PRL) (MUSLINER; PELICAN; SCHLETTE, 2009).

A PRL é uma linguagem desenvolvida por meio do programa *NASA's Exploration Technology Development Program* (ETDP). É uma linguagem com notação baseada no formato *eXtensible Markup Language* (XML) e possui comandos bem estabelecidos (CONNORS; MUNOZ; *et al*., 2009).

Com essa linguagem é possível fazer a especificação de procedimentos de operação, permitindo que o nível de autonomia seja ajustável. Com ela, os modos de operação a serem executados manualmente podem ser automatizados.

2.2.7. Procedure Language for Users in Test and Operations (PLUTO)

A PLUTO é especificada pela ECSS em (ECSS, 2008) e pode ser aplicada em atividades de teste ou de operações de voo para satélites e espaçonaves. Suas especificações foram iniciadas em meados de 1990 a partir de experiências adquiridas e acumuladas pela ESA e empresas prestadoras de serviços, aliadas à ESA, durante o desenvolvimento de soluções a partir de linguagens do tipo *ad-hoc*.

A PLUTO é interpretada e sua sintaxe é similar a linguagens de programação estruturada de alto nível. Sua utilização permite que se descreva o comportamento de um sistema ou equipamento da área espacial, esteja ele sob teste ou em operação.

Procedimentos descritos com a PLUTO podem abranger grande parte do conhecimento da missão de maneira que tal conhecimento se mantenha encapsulado. Os dados utilizados durante a execução do procedimento são mantidos dentro de um modelo do sistema, possibilitando que o procedimento seja descrito de maneira mais simples e compreensível (KOLLER, 2010).

Esse é um padrão de linguagem que está sendo usado por empresas em projetos espaciais da ESA. Existem ferramentas e ambientes computacionais que estão sendo desenvolvidos para criação padronizada e execução de procedimentos tanto de teste quanto de operação (ADAMSON; BARGELLINI, *et al*., 2010), (CROCE; SIMONIC, 2008), (FRITZ; ROESER, *et al*., 2010) e (PEARSON; TRIFIN, *et al*., 2012).

2.3. Dados Relativos à Aquisição e Utilização das Linguagens

A Tabela 2.1 mostra uma compilação de informações relativas a aquisição e utilização das linguagens apresentadas nas Seções anteriores.

As licenças de aquisição são classificadas entre os domínios: público e proprietário. Algumas linguagens mantêm também o código-fonte do sistema executor sob gênero *open-source* e isso pode ser um benefício para alguns casos. A informação referente aos sistemas de código *open-source* é válida para as linguagens públicas.

Sistemas de gênero público podem apresentar necessidades de adaptações para se adequarem a determinado problema. Dessa forma, é necessário avaliar o esforço que possivelmente poderá ser aplicado para tais adaptações.

A Tabela 2.1 também mostra detalhes sobre a sintaxe da linguagem para aquelas que foram criadas a partir de outra linguagem, como é o caso das linguagens definidas a partir da *Python*. A mesma tabela também mostra a área de aplicação de cada linguagem à fase de desenvolvimento da missão espacial.

Tabela 2.1 - Dados relativos a aquisição e utilização das linguagens

	Disponibilidade						
Linguagem	**Especificação da Linguagem**	**Sistema para Execução**	**Open Source**	**Runtime**	**Similaridades com a Sintaxe**	**Fase(s) do Ciclo de Vida da Missão**	**Participação em Missões**
ETOL	Proprietária	Proprietária	não	Compilada	C	AIT e Operação	*Ulysses*; *Giotto*; ROSAT; CBERS.
PLUTO	Pública	Proprietária	não	Interpretada	Visual Basic	AIT e Operação	RADARSAT-2; VEGA.
SPELL	Pública	Pública	sim	Compilada	Python	Operação	*Ariane* 5.
STOL	Proprietária	Proprietária	não	Interpretada	Python	AIT e Operação	*Messenger*; *MetOp* A.
TCL	Pública	Pública	sim	Interpretada	C	AIT e Operação	Estação Espacial Internacional, a *International Space Station* (ISS).
PRL	Proprietária	Proprietária	não	Compilada	XML	Operação de Missões Tripuladas	Ônibus Espacial, *Space Shuttle*, da NASA.

Os sistemas existentes para o gerenciamento da execução de procedimentos descritos com a PLUTO continuam sendo desenvolvidos inteiramente por empresas ligadas à ESA. Esses são sistemas de domínio proprietário e não existe outro tipo de licença para uso.

O INPE já vem utilizando padrões da ECSS no desenvolvimento de suas atividades espaciais. Com a adoção de uma linguagem como a PLUTO, o Instituto continuará mantendo suas missões alinhadas aos padrões vigentes da área espacial, em especial aos padrões da ECSS. Portanto, a PLUTO será apresentada com mais detalhes no Capítulo a seguir.

3 A PLUTO E O MODELO DO SISTEMA ESPACIAL

No presente Capítulo são apresentadas as características de um *Space System Model* (SSM), (ECSS, 2003), e a estrutura de um procedimento descrito com a PLUTO, (ECSS, 2008). Além disso, é feita uma explicação do porque um procedimento descrito com a PLUTO deve fazer referência ao SSM, como descrito na Seção 1.1.

3.1. O Space System Model (SSM)

A utilização da PLUTO para descrição de procedimentos em atividades espaciais requer a existência de um modelo de dados do sistema, do subsistema ou do equipamento que será testado ou operado. A ECSS disponibiliza o padrão *Ground Systems and Operations*, disponível em (ECSS, 2003), que pode ser usado para auxiliar a modelagem e estruturação dos dados do modelo.

A ECSS especifica o formato lógico de um modelo que deve ser mantido em estrutura hierárquica, de modo que possibilite a representação real do funcionamento do sistema, subsistema ou equipamento. A essa estrutura é dado o nome de *Space System Model* (SSM). O SSM consiste da decomposição hierárquica de diversos tipos de dados, que podem ser classificados entre diferentes tipos de objetos.

Os tipos de objetos de um SSM foram definidos para manter a organização entre os dados de sua estrutura. De acordo com o padrão especificado em (ECSS, 2003), cada objeto possui um objetivo pai e podem ser classificados entre os seguintes tipos:

- Elemento de Sistema, do inglês *System Element* (SE);
- Atividade, do inglês *Activity*, que pode ser executada pelo SE;

- Dado de Relato, do inglês *Report Data* (RD), que reflete o estado de funcionamento dos objetos do tipo SE; e

- Evento, do inglês *Event*, que pode ser manipulado para controle e monitoramento desses SE's.

Os objetos de um SSM devem ser definidos de modo que possam ser utilizados para controle e monitoramento do subsistema ou equipamento que está sob teste ou operação. A Figura 3.1 ilustra o exemplo de uma decomposição hierárquica composta de objetos, junto aos seus tipos, para um Subsistema de Controle de Atitude e Órbita, do inglês o *Attitude and Orbit Control Subsystem* (AOCS).

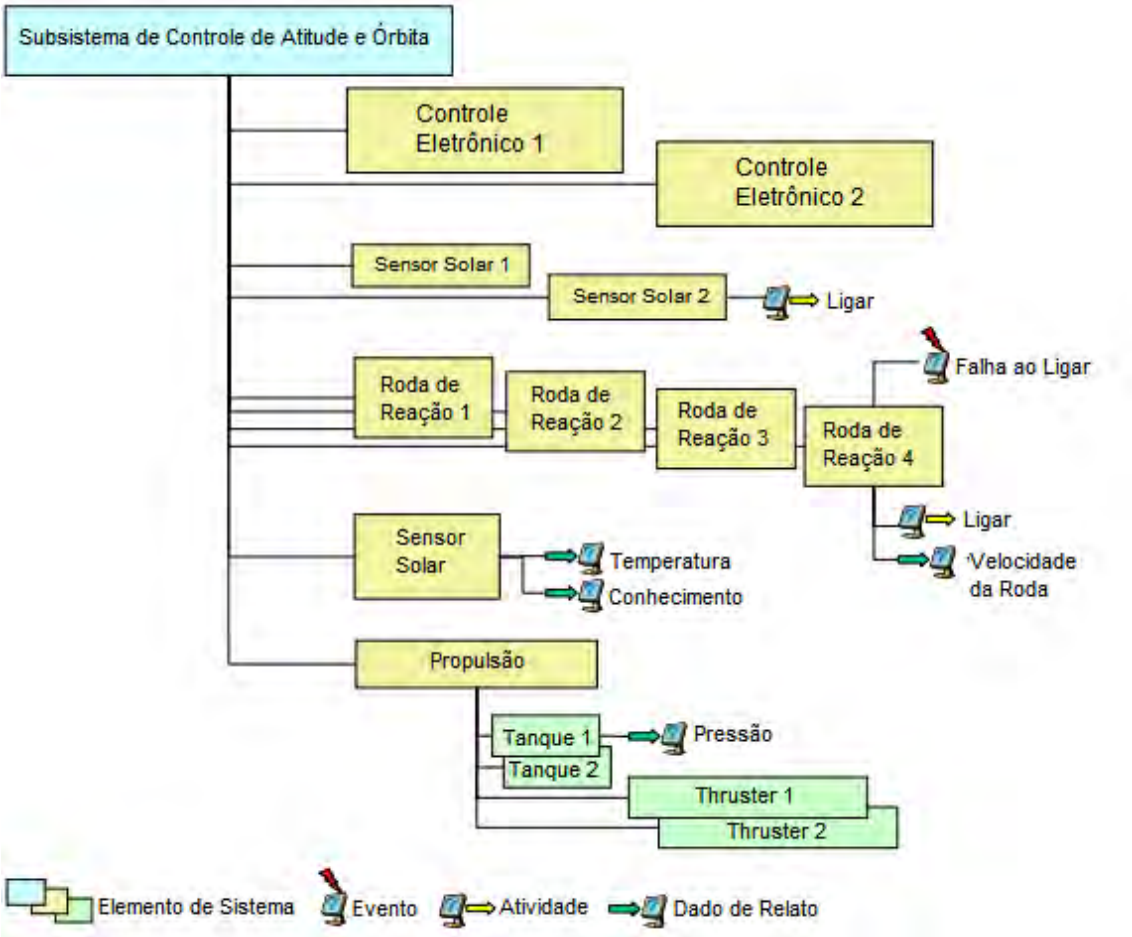

Figura 3.1 - Exemplo da decomposição hierárquica de um SSM para um AOCS

Fonte: ECSS (2003)

Os Elementos de Sistema correspondem aos objetos resultantes da decomposição funcional, do nível mais alto até o mais baixo. A esses elementos estão associados os Dados de Relato, as Atividades e os Eventos.

Os Dados de Relato são as informações que um Elemento de Sistema gera, independentemente de como esta informação é usada. Esses dados podem compreender medidas que refletem o estado do elemento associado ou saída de outro Elemento de Sistema.

O controle e monitoramento do sistema espacial ficam a cargo das Atividades. Uma atividade pode ser definida por meio de um Telecomando (TC) ou uma Telemetria (TM), um procedimento de teste ou operação, ou qualquer outro tipo de comando ou conjunto de comandos que seja específico para uma dada funcionalidade do sistema.

Os Eventos podem estar associados aos Elementos de Sistema, aos Dados de Relato e às Atividades. Um Evento é a ocorrência de uma condição ou conjunto de condições que podem ocorrer durante a execução de um teste ou operação e podem ser usados para iniciar a execução de uma Atividade de controle ou monitoramento do SUT.

A Figura 3.2 ilustra a representação de um Elemento de Sistema e suas Atividades, Dados de Relato associados, bem como dos demais dados que são usados para gerenciamento de sua configuração, sua função e requisitos daquele Elemento de Sistema. Um Elemento de Sistema também pode compreender outros objetos do tipo Elemento de Sistema.

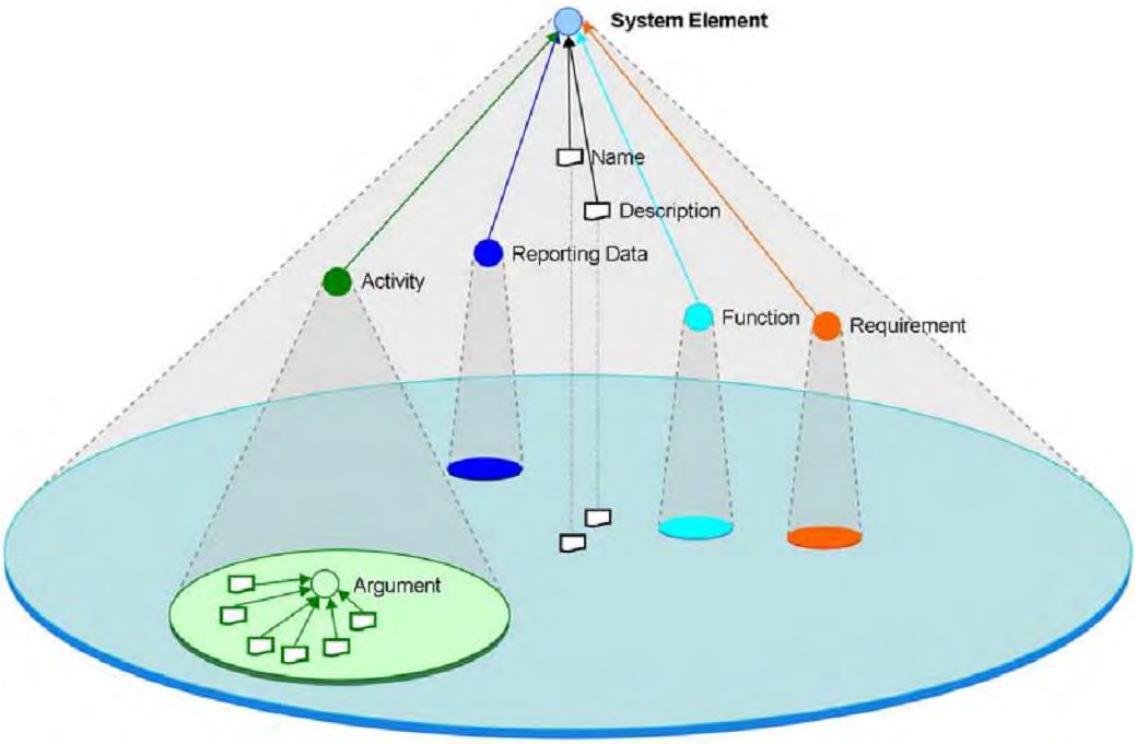

Figura 3.2 - Um objeto do tipo System Element e seus dados associados

Fonte: ECSS (2011)

O modelo de um sistema representado hierarquicamente pode abranger todos os seus equipamentos, componentes e subsistemas.

Devido ao grande número de dados e funcionalidades, torna-se difícil descrever todo o comportamento do satélite por meio do SSM, nesse caso, faz-se necessária a aplicação do conceito de Visão Específica de Domínio, que do inglês é denominada *Domain Specific View* (DSV). As características de uma DSV também são especificadas em (ECSS, 2003).

Uma DSV é composta por um subconjunto de funcionalidades do sistema, necessário para determinada aplicação. Esse mecanismo pode ser usado durante a modelagem individual de equipamentos e subsistemas, onde somente os dados de interesse estarão inseridos no SSM.

Um SSM, além de poder ser usado na aplicação de procedimentos de teste ou de operação, também traz outros benefícios, como a geração de históricos de dados, mantendo experiências adquiridas e permitindo a geração de conhecimento, que pode ser aplicado no desenvolvimento de equipamentos do mesmo gênero.

Ao longo do ciclo de vida de um sistema espacial, fornecedores entregam produtos aos seus clientes, onde cada produto consiste de componentes de hardware, software, ou ambos, e estão associados a uma documentação que contém todo conhecimento gerado durante as fases de desenvolvimento, integração, testes e operação. O SSM pode facilitar o intercâmbio das informações entre clientes e fornecedores e ser usado para gerenciar todo esse conhecimento.

3.2. Análise de Dados para a Modelagem do SSM

A modelagem de um SSM não é uma tarefa trivial, mesmo que seja apenas uma DSV do sistema que será testado ou operado. O presente trabalho propõe uma maneira de dar início à análise dos possíveis dados de composição da estrutura do SSM.

A Tabela 3.1 mostra alguns atributos que podem ser utilizados para análise desses dados. Esses atributos são úteis para definição inicial dos dados de um SSM.

Tabela 3.1 - Atributos de análise de dados para modelagem de um SSM

Id do Objeto	Nome	Tipo do Objeto	Descrição	Referência a um Objeto
<id_1>	<nome 1>	<tipo 1>	<descrição 1>	<nenhum>
<id_2>	<nome 2>	<tipo 2>	<descrição 2>	<id_1>
<id_3>	<nome 3>	<tipo 3>	<descrição 3>	<id_2>
...	...	...	...	...
<id_N>	<nome N>	<tipo N>	<descrição N>	<id_N>

- O campo ‘Id do Objeto’ é usado para identificação de cada objeto;
- O campo ‘Nome’ representa o nome do objeto que será referenciado dentro do procedimento;
- O campo ‘Tipo de Objeto’ classifica o tipo do objeto entre Elemento de Sistema, Dado de Relato, Atividade ou Evento;
- O campo ‘Descrição’ pode ser usado para descrever o objetivo do objeto; e
- O campo ‘Referência a um Objeto’ deve ser usado para indicar o objeto pai.

Como regra geral, todos os objetos de um SSM possuem um objeto referência, exceto o objeto que se mantém como raiz da estrutura. Um exemplo de objeto que se mantém na raiz da estrutura poderia ser um objeto do tipo Elemento de Sistema, cujo nome seria o do próprio satélite ou de um subsistema modelado como uma DSV.

3.3. A Estrutura de um Procedimento descrito com a PLUTO

A terminologia usada na definição da sintaxe da PLUTO é compatível com a *International Organization for Standardization* (ISO) e a *International Electrotechnical Commission* (IEC), por meio da norma ISO/IEC 14977. A sintaxe da linguagem é um conjunto de regras, coletivamente conhecidas como gramática. Suas palavras reservadas são *case-sensitive* ECSS (2008).

A estrutura de um procedimento PLUTO é ilustrada na Figura 3.3.

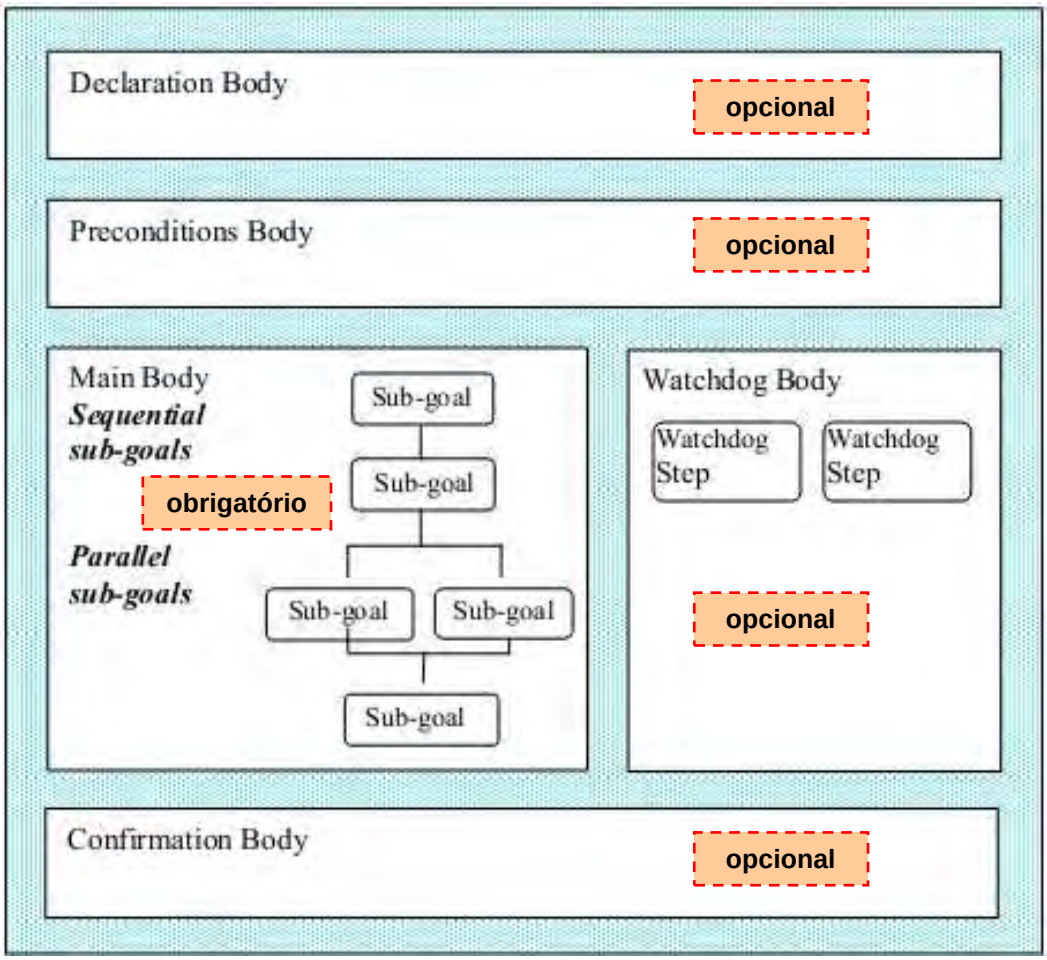

Figura 3.3 - Estrutura de um procedimento descrito com a PLUTO

Fonte: adaptado de ECSS (2008)

De acordo com (ECSS, 2008), a estrutura de um procedimento descrito com a PLUTO é classificada em diferentes conjuntos de instruções. Tais conjuntos são classificados entre: um corpo de declaração, o *Declaration Body*; um corpo de precondições, o *Preconditions Body*; um corpo principal, o *Main Body*; um corpo usado para medidas de contingência, o *Watchdog Body*; e um corpo de confirmação, o *Confirmation Body*.

Cada conjunto é destinado ao cumprimento de um determinado objetivo. O uso de cada grupo de instruções é classificado entre obrigatório e opcional. Apenas o *Main Body* é obrigatório, porque mantêm as principais instruções, necessárias para o procedimento, ECSS (2008).

- **O *Declaration Body***

  No *Declaration Body* o analista poderá declarar eventos a serem detectados durante a execução do procedimento. Os eventos que são declarados em nível de procedimento podem ser acessados somente pelo *Watchdog Body*. Além disso, o *Declaration Body* pode também ser utilizado para a declaração de variáveis globais.

- **O *Preconditions Body***

  As instruções do *Preconditions Body* impõem as condições que definem se um procedimento pode ser iniciado. As condições que podem ser estabelecidas para que um procedimento possa ser iniciado são:

  - esperar um exato momento (um tempo absoluto);
  - esperar por um dado intervalo de tempo;
  - esperar até que uma determinada condição seja verdadeira;
  - esperar até que determinado equipamento esteja ligado;

- iniciar se o resultado de uma expressão lógica (condição Booleana) for verdadeiro;

- ou solicitar a interação do analista (como tomar uma decisão);

- **O *Main Body***

O *Main Body* de um procedimento é obrigatório, e pode ser composto de uma sequência de instruções, como por exemplo:

- iniciar em paralelo.

  Na sintaxe da linguagem, tem-se: ***initiate in parallel***;

- iniciar e confirmar passo.

  Na sintaxe da linguagem, tem-se: ***initiate and confirm step***;

- iniciar e confirmar atividade.

  Na sintaxe da linguagem, tem-se: ***initiate and confirm activity***;

- iniciar atividade.

  Na sintaxe da linguagem, tem-se: ***initiate activity***;

- informar operador.

  Na sintaxe da linguagem, tem-se: ***inform user***;

- registrar log.

  Na sintaxe da linguagem, tem-se: ***log***;

A instrução '*initiate in parallel'* é usada para habilitar a execução de passos, também chamados de '*steps'*, e atividades em paralelo. A instrução '*initiate in parallel'* assume um dos seguintes comportamentos:

- a execução do conjunto de *steps* ou conjunto de atividades em paralelo será finalizada quando qualquer um dos *steps* ou atividades forem finalizados;

- ou a execução será finalizada quando todos os *steps* ou atividades forem finalizados.

As atividades solicitadas para execução, como na instrução '*initiate activity'*, são objetos do tipo *Activity*, declarados no modelo SSM e apresentados na Seção 3.1.

As instruções que solicitam confirmação devem relatar o status durante a execução e no término da execução, em nível de procedimento, em nível de *step* e em nível de atividade. Ao término da execução de cada nível, o status pode assumir '*confirmed'*, '*not confirmed'* ou '*aborted*'.

O *Main Body* também consiste de uma seqüência de passos do procedimento, os chamados *steps*. Os *steps* podem ser um comando ou conjunto de comandos, uma solicitação de ação do operador ou mesmo uma chamada de outro(s) procedimento(s). A estrutura de um *step* é apresentada na Seção 3.3.1.

- **O *Watchdog Body***

As instruções do *Watchdog Body* são os *steps* do *watchdog*. Cada *step* do *watchdog* monitora a ocorrência de uma condição de contingência e realiza ações corretivas. O propósito do *Watchdog Body* é detectar eventos em nível de sistema, como por exemplo, anomalias em bordo. O *watchdog* pode suspender ou abortar o procedimento se ocorrer alguma anomalia (falha) durante a execução.

As situações de contingência monitoradas pelos *steps* do *watchdog* são independentes umas das outras. A razão para isso é que se um *watchdog* detectar alguma anomalia, ele suspende, se necessário, a

execução do *Main Body* correspondente. Os outros *watchdogs* que estão engatilhados continuam monitorando seus respectivos *steps* declarados no *Main Body*.

- **O *Confirmation Body***

  Finalmente, o *Confirmation Body* consiste de instruções para a leitura do equipamento sob teste ou em operação, para determinar se todos os passos do *Main Body* foram executados corretamente.

A Figura 3.4 ilustra um exemplo da estrutura de um procedimento descrito com a PLUTO. O mesmo apresenta um *Declaration Body*, composto de variáveis e seus tipos; um *Precondition Body*, com a definição das precondições; e um *Main Body* com algumas instruções. Os tipos de variáveis podem ser definidos a partir das *Engineering Units*, que são definidas em (ECSS, 2003).

```
procedure <name>
declare
        variable type var_1 := <value> engineering unit;
        variable type var_2 := <value> engineering unit;
        ...
        variable type var_N := <value> engineering unit;
end declare
preconditions
        precondition 1;
        precondition 2;
        ...
        precondition N;
end preconditions
main
        initiate and confirm step <step name>
                declare
                        variable type var_r;
                        variable type var_s;
                end declare;
                main
                        var_r := var_1;
                        var_s := var_2;
                        var_1 := ref_ssm_object_2;
                        inform user 'message';
                        ...
                end
        end step
        initiate ref_ssm_object_1;
        var_2 := var_1 * <value>;
        initiate and confirm ref_ssm_object_3;
        ...
        initiate ref_ssm_object_N;
end main
end procedure
```

Figura 3.4 – Exemplo da estrutura de um procedimento descrito com a PLUTO

3.3.1. A Estrutura do Step de um Procedimento

Um *step* deve ter um nome único e consiste de um *Step Declaration Body*; um *Step Preconditions Body,* obrigatório para um *step* de *Watchdog*; um *Step Main Body*; um *Step Watchdog Body* e um *Step Confirmation Body*.

A estrutura de um *step* é idêntica à estrutura de um procedimento, mas algumas instruções apresentam suas peculiaridades, como eventos e variáveis, que ao serem declarados se destinam ao uso somente local.

O *Step Main Body*, por exemplo, é composto das mesmas instruções declaradas no *Main Body* em nível de procedimento, mas em adicional também permite acrescentar a elaboração de estruturas condicionais e de repetição, como:

- expressão ***if-then-else***
- expressão ***in case***
- expressão ***while-do***
- expressão ***for-to-do***
- expressão ***repeat-until***

A PLUTO permite a construção de condições lógicas, rotinas em loop, operações aritméticas, interação com o operador durante a execução, agendamento da execução, chamada de outros procedimentos dentro de um único procedimento e tratamento de diversos tipos de dados.

Dentre os diversos tipos de dados, existem os tipos pré-definidos em (ECSS, 2008), que são: *Boolean*; *Enumerated*; *Signed Integer*; *Unsigned Integer*; *Real*; *String*; e *Time*. Para informações mais detalhadas sobre a PLUTO, consultar *Test and Operations Procedure Language*. Este é o padrão ECSS-E-ST-70-32C, apresentado em (ECSS, 2008).

3.4. Relação entre a PLUTO e o SSM

Os padrões que especificam a PLUTO e o SSM são especificados para que os procedimentos possam fazer referência aos objetos do modelo. O

procedimento descrito com a PLUTO deve fazer referência ao SSM para que durante sua execução possa ler ou escrever valores em seus objetos.

Um procedimento descrito com a PLUTO é composto de instruções que possibilitam o acesso a esses objetos para a leitura ou escrita de dados. A Figura 3.5 ilustra a forma com que esses objetos podem ser referenciados por meio do procedimento.

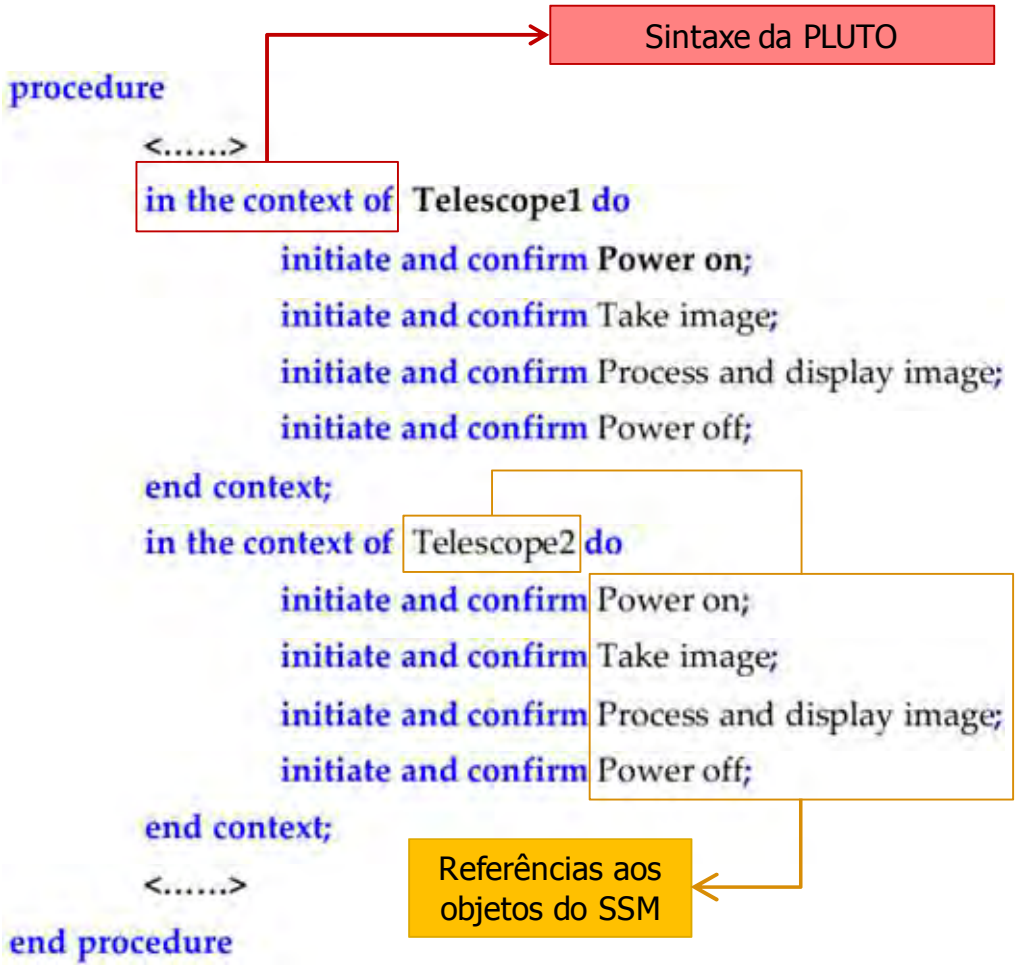

Figura 3.5 – Exemplo de um procedimento que faz referência a objetos de um SSM

Fonte: adaptação de ECSS (2008)

Conjuntos de Telecomandos (TCs) e Telemetrias (TMs) são encapsulados junto a esses objetos, cada qual com seus respectivos dados de aquisição (Dados de Relato e Eventos) e dados de atuação (Atividades).

Os dados de aquisição são obtidos por meio das TMs recebidas do SUT e os dados de atuação são carregados no TC e enviados ao SUT.

Esse tipo de estrutura proporciona transparência na descrição do procedimento, pois não existe a necessidade de definir os comandos de envio e recepção no corpo do procedimento, pois estarão estruturados no SSM.

Além disso, manter os objetos desse modelo em estrutura hierárquica facilita a rastreabilidade dos objetos durante a execução.

O trabalho de (CROCE; SIMONIC, 2008) utiliza esses dois padrões em procedimentos de atividades de teste e descreve que uma forma de facilitar a comunicação do procedimento com os dados do modelo é manter o modelo entre o procedimento e o SUT.

Dessa maneira o SSM se comunica de um lado com o SUT para obter o seu status durante a execução e do outro lado com o sistema de execução dos procedimentos para receber as solicitações das instruções da linguagem que foram descritas no procedimento. A Figura 3.6 ilustra o resultado do trabalho de (CROCE; SIMONIC, 2008).

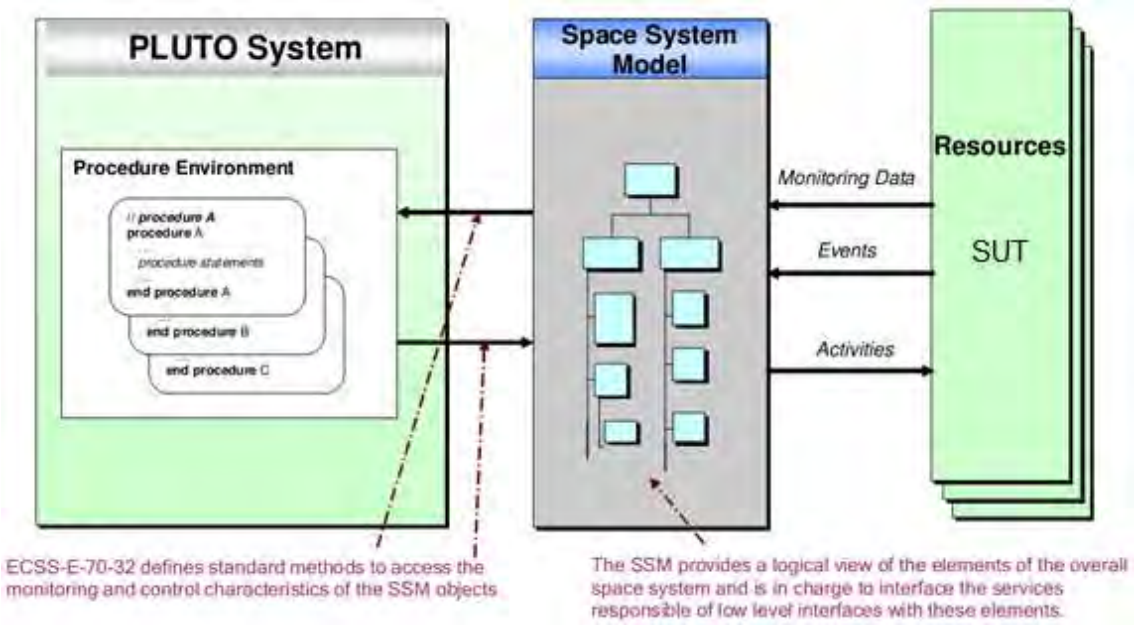

Figura 3.6 – Exemplo de arquitetura que mantém a relação entre um sistema executor de procedimentos com a PLUTO e um SSM apropriado que se comunica com seu respectivo SUT

Fonte: adaptação de Croce, *et al*. (2008)

3.5. Aplicação da PLUTO em Missões Espaciais

A ESA está explorando a PLUTO em algumas de suas missões espaciais:

- ***Precursore IperSpettrale della Missione Applicativa*** **(PRISMA)**: Satélite de observação da terra com lançado em 2010. A missão adota PLUTO como linguagem padrão. Os procedimentos são gerados em XML e em seguida traduzidos para a sintaxe da PLUTO (KARLSSON; AHLGREN, *et al*., 2012) e (AHLGREN; KARLSSON, *et al*., 2012).

- ***BepiColombo***: Missão de exploração planetária, com lançamento previsto para 2015 e chegada ao planeta Mercúrio em 2021. A PLUTO está fazendo parte das atividades de teste e os procedimentos também serão aproveitados na operação, (SCHWAB; EILENBERGER; BORG, 2012).

- ***Atmospheric Dynamics Mission* (ADM-AEOLUS)**: O autor (ADAMSON; BARGELLINI, *et al*., 2010) descreve que a PLUTO está presente na execução das operações de voo do satélite ADM-AEOLUS. Os procedimentos descritos por meio dessa linguagem são carregados a bordo do satélite e agendados para execução, de modo que o satélite tenha contato com solo uma vez a cada cinco dias.

- ***Radar Satellite* (RADARSAT) - 2**: (SEYMOUR, 2004) e (CROCE; SIMONIC, 2008) afirmam que essa linguagem foi utilizada no lançamento e atualmente está sendo usada nas atividades de operação da missão RADARSAT-2. O satélite RADARSAT-2 é resultado do trabalho da *Canadian Space Agency* (CSA), que atualmente é membro na ESA.

- ***Vettore Europeo di Generazione Avanzata* (VEGA)**: A PLUTO também foi utilizada em atividades de teste durante o desenvolvimento do veículo lançador VEGA. O EGSE desse veículo executou os procedimentos de teste na sintaxe da linguagem. Seu desenvolvimento foi realizado pela *Agenzia Spaziale Italiana* (ASI) (CROCE; SIMONIC, 2008).

- ***Gravity Field and Steady-State Ocean Circulation Explorer* (GOCE)**: O satélite GOCE faz parte da família das missões de observação da terra da ESA. A linguagem foi utilizada nos procedimentos de teste e foram executados automaticamente (KOLLER, 2010).

Dando continuidade, o próximo Capítulo, Capítulo 4, apresenta um panorama das tecnologias (sistemas de software) que utilizam a PLUTO e que pertencem à estrutura de preparo e execução de procedimentos da ESA. O motivo dessa abordagem é apresentar, no âmbito do “estado da arte”, a relação entre as tecnologias existentes e a arquitetura do STEPS.

4 USO DA PLUTO EM SISTEMAS DE PREPARO E EXECUÇÃO DE PROCEDIMENTOS EM ATIVIDADES ESPACIAIS DA ESA

Os trabalhos de (WALSH; PECCHIOLI, *et al*., 2012), (SCHWAB; EILENBERGER; BORG, 2012), (PEARSON; TRIFIN, *et al*., 2012), (LANNES; PISSIAS, *et al*., 2012), (KARLSSON; AHLGREN, *et al*., 2012), (HEINEN; REID; PEARSON, 2012), (LORETUCCI; NISIO, *et al*., 2008), (FRITZ; ROESER, *et al*., 2010), (KOLLER, 2010), (AHLGREN; KARLSSON, *et al*., 2012) e (REID; RENSON, *et al*., 2012) mostram que nos últimos anos a ESA vem investindo no desenvolvimento de tecnologias, de maneira padronizada, para se chegar a um ambiente único, capaz de atender as atividades de desenvolvimento, manufatura e operação de satélites e espaçonaves em diferentes missões.

O presente Capítulo aborda a estrutura de preparo e automatização dos procedimentos da ESA, junto aos seus mecanismos e ferramentas computacionais de apoio. Tal abordagem apresenta como essa estrutura encontra-se atualmente e como a ESA está trabalhando para que tal estrutura torne-se um sistema único, para uso em todas as suas futuras missões. O objetivo desse sistema é apoiar a automatização dos procedimentos da ESA.

Ainda neste Capítulo, é feita uma correlação entre as características da arquitetura do STEPS e as características de um sistema que faz parte da estrutura da ESA.

4.1. Um Sistema Único para o Preparo e a Execução de Procedimentos das Futuras Missões Espaciais da ESA

A ESA pretende adaptar sua estrutura de preparo e execução de procedimentos já existente, de modo que fique genérica, para que também possa controlar e monitorar satélites e espaçonaves durante todas as fases do desenvolvimento (AIT e pré-lançamento) e das operações (pós-lançamento) para todos os tipos de missões (WALSH; PECCHIOLI, *et al*., 2012).

Existem inúmeros sistemas computacionais de apoio às atividades de teste e operação da ESA que são desenvolvidos para uma única missão, e nem sempre são reaproveitados. Com o desenvolvimento de tecnologias genéricas, reutilizáveis em diversas missões provê um conjunto de benefícios, como facilidades no intercâmbio de informações entre as organizações, redução de tempo, custo, aumento da produtividade e maior confiabilidade durante a execução das atividades.

As tecnologias podem ser classificadas como genéricas se forem desenvolvidas para serem aplicadas em processos, ou procedimentos padronizados. A ESA definiu um plano que conta com o desenvolvimento de um conjunto de tecnologias que permitem a sistematização e padronização das atividades de AIT e operação. Muitas dessas tecnologias estão em desenvolvimento e as já existentes estão sendo adaptadas para se tornarem genéricas.

Esse plano de desenvolvimento é resultado de um conhecimento que foi adquirido no desenvolvimento das diferentes missões estabelecidas durante as últimas décadas.

O primeiro passo foi dado entre meados de 2009 a 2010, quando foi discutida na ESA a possibilidade de prover o desenvolvimento do chamado *European Ground Systems – Common Core* (EGS-CC), que atualmente encontra-se na fase de concepção.

Para dar início ao desenvolvimento do EGS-CC, a ESA solicitou a colaboração de um grande número de participantes, incluindo a *Astrium Satellites*; *Astrium Space Transportation*; *Thales Alenia Space* da França e Itália e a *Orbitale Hochtechnologie Bremen* (OHB).

As agências espaciais *Centre National d'Études Spatiales* (CNES), da França, e a *Deutsches Zentrum für Luft – und Raumfahrt* (DLR), da Alemanha, também sinalizaram interesse e ofereceram participação nesse projeto.

Segundo as afirmações de (WALSH; PECCHIOLI, *et al*., 2012), o EGS-CC é composto de sessões que agrupam, sistematicamente, componentes formados de dados e mecanismos para atender a diferentes instâncias, como instâncias Operacionais, Simulação ou Testes. Diferentes sessões podem ser instanciadas em paralelo, especialmente pelo ambiente de AIT, permitindo a geração de conjuntos independentes de resultados.

O EGS-CC está sendo projetado para ser um núcleo central, mais conhecido como *Kernel*, que poderá ser utilizado em diferentes aplicações. Essas aplicações poderão ser desenvolvidas de modo que possam utilizar todo o recurso do *Kernel* do EGS-CC.

Os integrantes da equipe de desenvolvimento desse projeto classificam-no como um projeto ambicioso (WALSH; PECCHIOLI, *et al*., 2012), pelo fato de apresentar as seguintes características:

- O atendimento a todas as fases e tipos de missões;
- Uma arquitetura orientada a serviço, *Service Oriented Architecture* (SOA), com características modulares;
- Funcionalidades genéricas e adaptáveis;
- Implementação em camadas;
- Interfaces de comunicação padronizadas;
- Alta performance;
- Escalabilidade;

Para desenvolver um sistema com essas características, uma série de padrões da ECSS, estão sendo utilizados com objetivo de padronizar dados, processos de desenvolvimento, execução das atividades, ferramentas de apoio e mecanismos necessários para suprir as necessidades das missões.

Dentre os padrões utilizados para tais fins, encontram-se os padrões destinados para:

- Protocolo de comunicação via TC e TM, os ECSS-E-ST-50-03 e 04;
- Padronização de pacotes em nível de aplicação, como o *Packet Utilisation Standard* (PUS), ECSS-E- 70-41A;
- Definição e estruturação dos dados para controle e monitoramento de sistema, subsistema ou equipamento, como o padrão ECSS-E-ST-70-31C, (ECSS, 2003), o mesmo padrão utilizado no presente trabalho;
- Padronização dos procedimentos por meio de linguagens, como o ECSS-E-ST-70-32C, (ECSS, 2008), o mesmo padrão de linguagem utilizado no presente trabalho, a PLUTO; e
- Padronização dos procedimentos destinados ao controle de bordo, como o ECSS-E-ST-70-01.

No contexto da padronização de dados, (HEINEN; REID; PEARSON, 2012) afirmam que um banco de dados único e centralizado é o primeiro passo para se chegar a uma arquitetura genérica. Isso está sendo feito com a modelagem de um banco de dados a partir das especificações do padrão ECSS-E-ST-70-31C, (ECSS, 2003), o mesmo que especifica o SSM.

A mesma afirmação também é feita no trabalho de (WALSH; PECCHIOLI, *et al*., 2012), onde é anunciado que a capacidade de modelar os dados de um sistema espacial completo é essencial para se manter o EGS-CC. O padrão que especifica o SSM permite abranger todos os dados de uma missão espacial, incluindo todos os segmentos.

A Figura 4.1 apresenta uma ilustração obtida em (WALSH; PECCHIOLI, *et al*., 2012) para facilitar o entendimento das características hierárquicas de um SSM e mostrar sua importância para o EGS-CC.

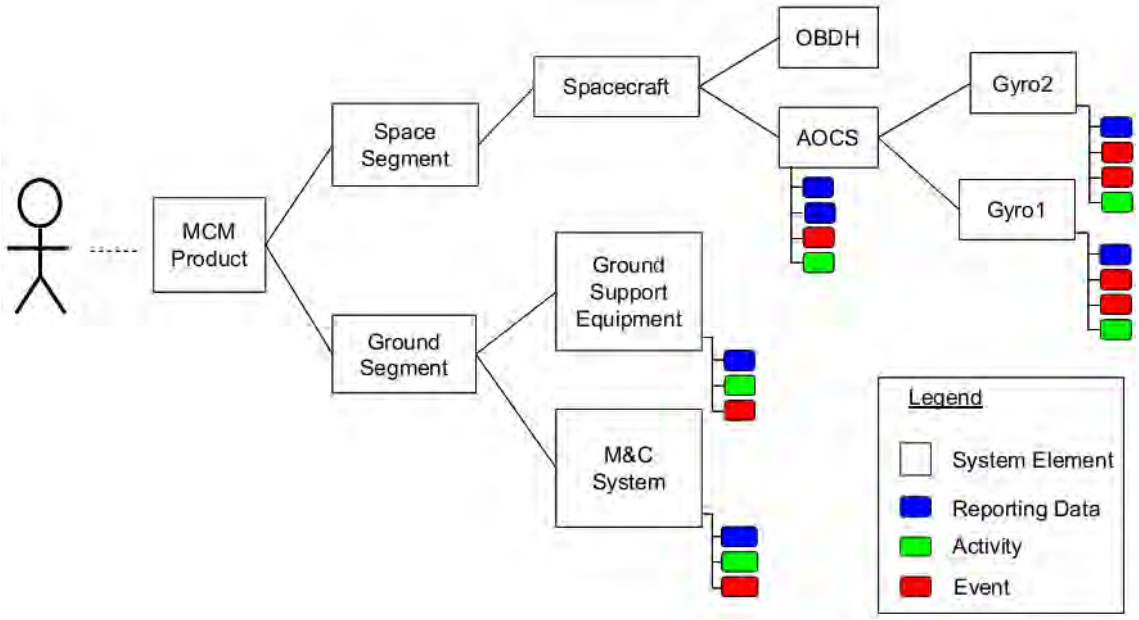

Figura 4.1 - Uma ilustração que mostra o uso do padrão do SSM na modelagem de um banco de dados único, como parte do projeto do EGS-CC, para ser usado em diversos tipos de missões da ESA.

Fonte: Walsh *et al*. (2012)

Espera-se que a finalização dessa etapa do projeto, incluindo os requisitos especificados em projeto detalhado, desenvolvimento, integrações e toda a validação do ambiente, seja concluída até meados de 2013. Com sua finalização, a ESA, juntamente com seus integrantes colaboradores, pretende desenvolver um EGSE/MCS completo, de maneira que utilize o *Kernel* do EGS-CC. Isso já está sendo feito com adaptações nas tecnologias já existentes e com a concepção de novas tecnologias.

Como o objetivo da ESA é generalizar e unificar as atividades e processos de desenvolvimento de suas missões espaciais, em paralelo ao EGS-CC, encontra-se também o desenvolvimento do chamado *Ground Station Monitoring and Control system* (GSMC), para atender as atividades de suas estações terrenas (LANNES; PISSIAS, *et al*., 2012).

Para abranger todos os processos do ciclo de vida das missões, as funcionalidades das estações terrenas, que são ligadas ao segmento solo, também serão padronizadas. Isso é geograficamente necessário, uma vez que a ESA mantém diversos centros de manufatura e controle espalhados por todo território europeu, sendo o intercâmbio de informações extremamente necessário.

4.2. Um Sistema de Gerenciamento e Preparação de Procedimentos em Missões Espaciais da ESA

Um dos sistemas mais importantes, que já faz parte da estrutura de automatização da ESA e que continuará fazendo parte da nova estruturação é o Sistema de Informação de Operações e Manufatura, do inglês *Manufacturing and Operations Information System* (MOIS). Esse sistema está sendo adaptado para que futuramente utilize o *kernel* do EGS-CC e se alinhe ao objetivo do desenvolvimento de um EGSE/MCS completo.

O MOIS foi projetado para a preparação, execução automática e manual de procedimentos de teste e operações de voo. Já foi utilizado como ferramenta de apoio no desenvolvimento de cerca de quase 80 satélites e espaçonaves, RHEA (2012).

O MOIS se mantém como principal recurso para tais atividades e está sendo muito utilizado pelo *European Space Operations Centre* (ESOC), agências e empresas do ramo espacial europeu (HEINEN; REID; PEARSON, 2012), (SCHWAB; EILENBERGER; BORG, 2012) e (FRITZ; ROESER, *et al*., 2010).

Seu desenvolvimento está sob a responsabilidade da *Rhea Group*, que conta com a colaboração de profissionais, do quadro de recursos humanos da ESA, com conhecimento adquirido no decorrer dos últimos anos. Participam desse projeto, engenheiros de operações de voo, engenheiros de AIT e desenvolvedores das tecnologias do segmento solo.

O MOIS é genérico, flexível e permite o uso de qualquer linguagem, dentre elas PLUTO, Elisa, STOL, CGS, tcl/TOPE, etc. Com toda essa flexibilidade, o sistema ainda é escalável, permitindo o incremento de novas funcionalidades sem grandes impactos (REID; RENSON, *et al*., 2012).

A Figura 4.2 ilustra os elementos que compõem a arquitetura do MOIS.

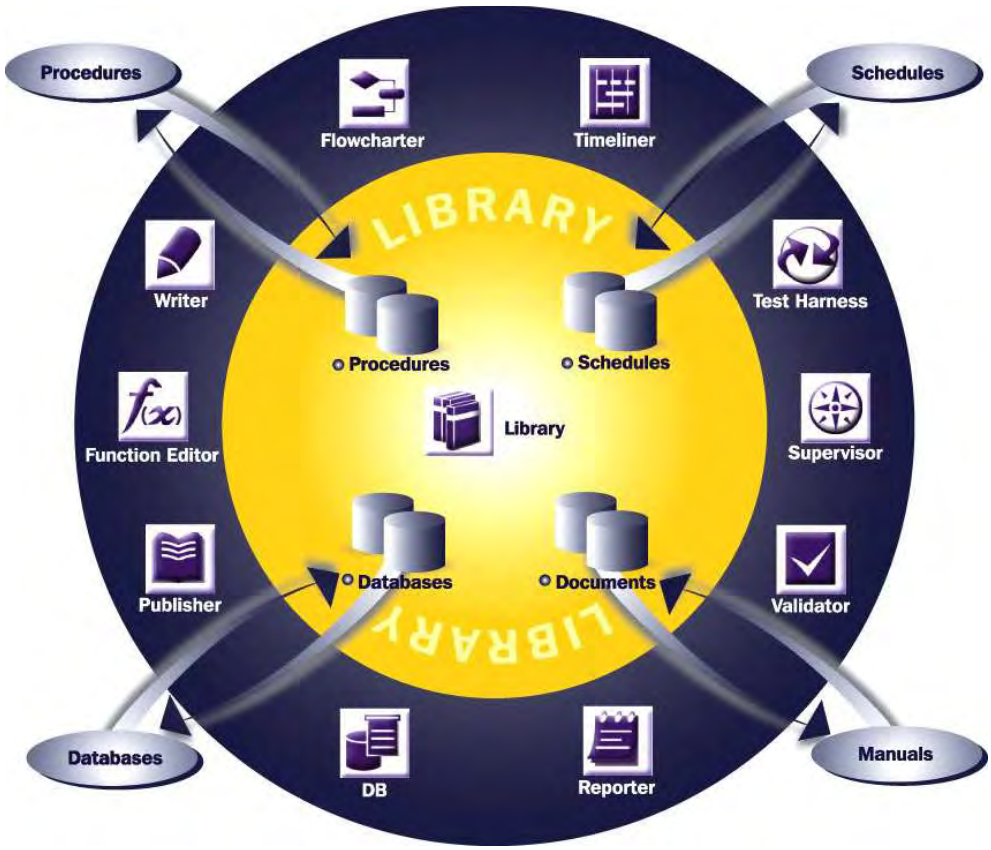

Figura 4.2 - Visão geral dos componentes da arquitetura do MOIS

Fonte: RHEA (2012)

O MOIS é composto por um conjunto de ferramentas que foram projetadas para permitir o preparo, gerenciamento e monitoramento da execução dos procedimentos. São elas:

- ***Writer*** – permite o desenvolvimento e criação dos procedimentos, com os *steps* do procedimento associados aos requisitos do Plano de Teste, de maneira tabular ou gráfica (via *Flowcharter*).

- ***Flowcharter*** – permite a criação, edição e visualização da estrutura dos *steps* do procedimento de maneira gráfica.

- ***Timeliner*** – fornece uma fonte de TC e TM, substituindo a necessidade de ligar a um sistema de controle e algum simulador para validação inicial do procedimento.

- ***Test Harness*** – permite ao usuário testar e validar funcionalmente a lógica do procedimento. Com uso dessa ferramenta, os recursos do sistema de controle e simulação podem ser otimizados durante o processo de validação.

- ***Supervisor*** – controla e monitora os executores do MOIS. Vários executores podem ser executados em paralelo, conectados a diferentes sistemas de controle, simuladores ou EGSEs. O *Supervisor* permite iniciar e parar os executores, gerenciar a conexão, definir o tempo de referencia para sua execução e prover uma visão geral dos procedimentos que estão executando e fazendo uso de todos os executores. Também permite controlar as conexões CORBA ao SCOS-2000, Simuladores e EGSEs.

- ***Validator*** – o objetivo do *Validator* é testar o procedimento e registrar os resultados. Geralmente, o procedimento é validado com uso de um simulador. Uma vez tendo o procedimento validado, o *Validator* exibe o status do procedimento, informando se o mesmo está pronto para uso sob o sistema real.

- ***Reporter*** – gera relatos para apoio às análises na resolução de problemas, acompanhamento do status do procedimento e consistência dos dados, tanto no preparo quanto durante a execução.

- **DB** – esse é o acrônimo de *Data Base*. Sua atribuição é controlar a criação ou realizar a importação de dados de outras bases. Como ilustrado na Figura 4.2, seriam importações de bases de dados de diferentes missões. Essa ferramenta mantém interface com o SCOS-2000.

- ***Publisher*** – utilizada para gerar e publicar toda a documentação referente aos procedimentos criados. Inclui o controle da configuração e de versão dos documentos. Tais documentos fornecem um histórico do amadurecimento dos procedimentos. Muito utilizada para aquisição de conhecimento durante a gestão da execução dos procedimentos.

- ***Function Editor*** – possui uma *Interface Homem Computador* (IHC) amigável para criar e manter funções e diretivas para uso durante a descrição dos procedimentos.

- ***Schedules*** – permite o agendamento da execução dos procedimentos durante as operações do satélite.

- ***Library*** – gerencia de forma integrada todos os dados da missão, incluindo procedimentos agendados ou não, bases de dados e documentos, para todas as ferramentas do MOIS.

Para o MOIS, a ESA adotou o uso do padrão ECSS-E-ST-70-31C, (ECSS, 2003) para o estabelecimento de uma base de dados unificada, capaz de manter todos os dados da missão. Isso mostra que o MOIS também foi projetado na mesma linha de padronização do EGS-CC.

É por esse motivo que o MOIS continuará fazendo parte da nova infraestrutura de automatização das atividades espaciais da ESA, para tanto, ainda serão

necessárias algumas adaptações. A Figura 4.3 ilustra os componentes do MOIS arranjados fisicamente com a presença do SSM para unificação e gestão dos dados.

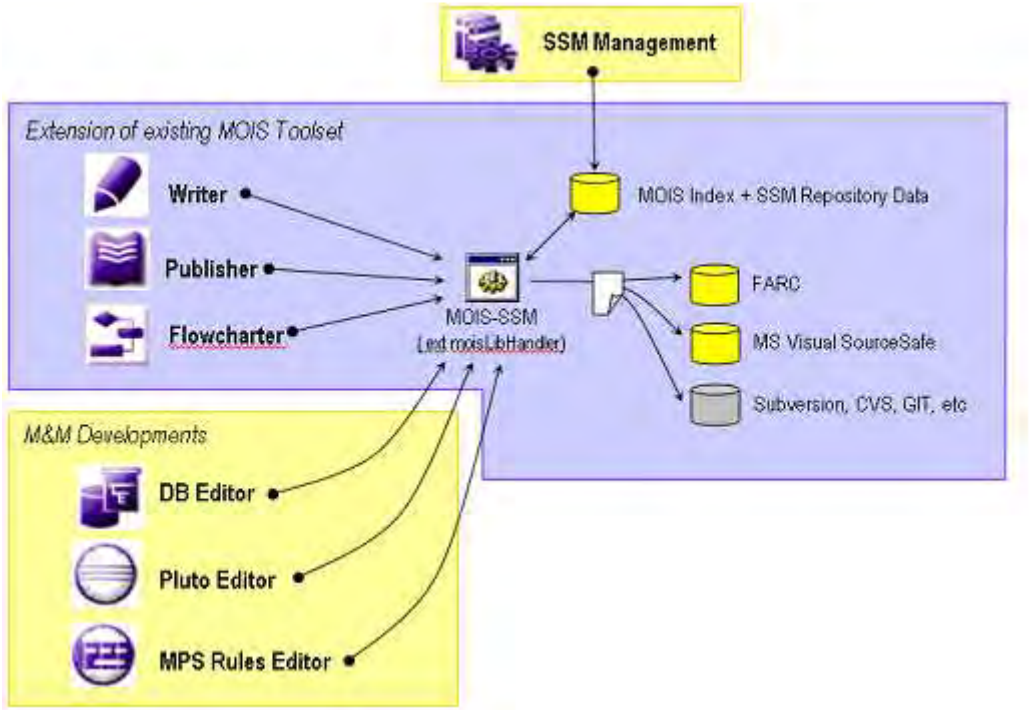

Figura 4.3 - O SSM presente no meio da estrutura e componentes de preparo dos procedimentos por meio do MOIS.

Fonte: Heinen, *et al*. (2012)

A partir de um banco de dados com as características do SSM, a ESA percebeu que poderia ir mais além. Outra iniciativa voltada para a execução dos procedimentos, teve como resultado a comodidade de escolha entre um conjunto de linguagens. Desse modo, o MOIS mantém um conjunto de executores, disponíveis para atender às diferentes linguagens (EICKHOFF, 2012).

O profissional de AIT ou de operação, por exemplo, pode editar graficamente o procedimento utilizando o MOIS e em seguida convertê-lo para qualquer tipo de linguagem. O trabalho de (SCHWAB; EILENBERGER; BORG, 2012)

apresenta esse recurso e disponibiliza uma ilustração, Figura 4.4, que facilita o entendimento desse processo.

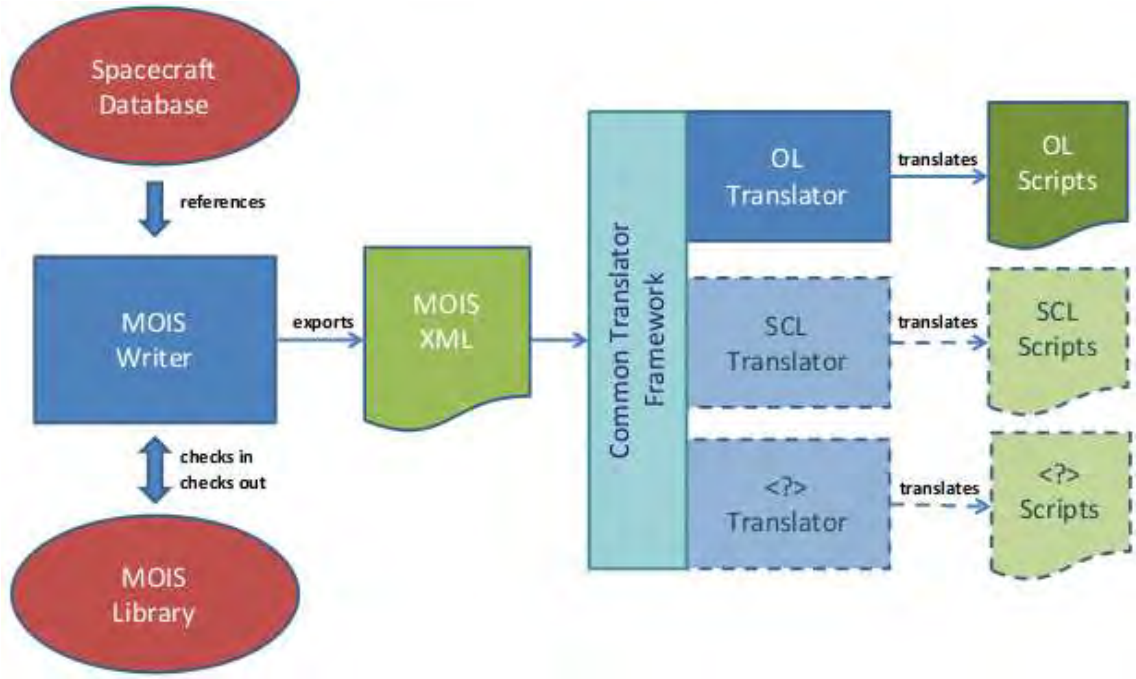

Figura 4.4 - Processo de conversão dos procedimentos preparados via MOIS para qualquer tipo de linguagem.

Fonte: Schwab, *et al*. (2012)

Ao ser editado, o procedimento pode ser exportado para o formato MOIS XML e em seguida traduzido para a sintaxe da respectiva linguagem, por meio do *Common Translator Framework*. Esse *framework* mantém um dicionário com a sintaxe das linguagens que o MOIS permite utilizar, dentre elas a PLUTO.

A Figura 4.5 ilustra a utilização do MOIS no cenário de preparação dos procedimentos via ferramenta *Flowcharter*.

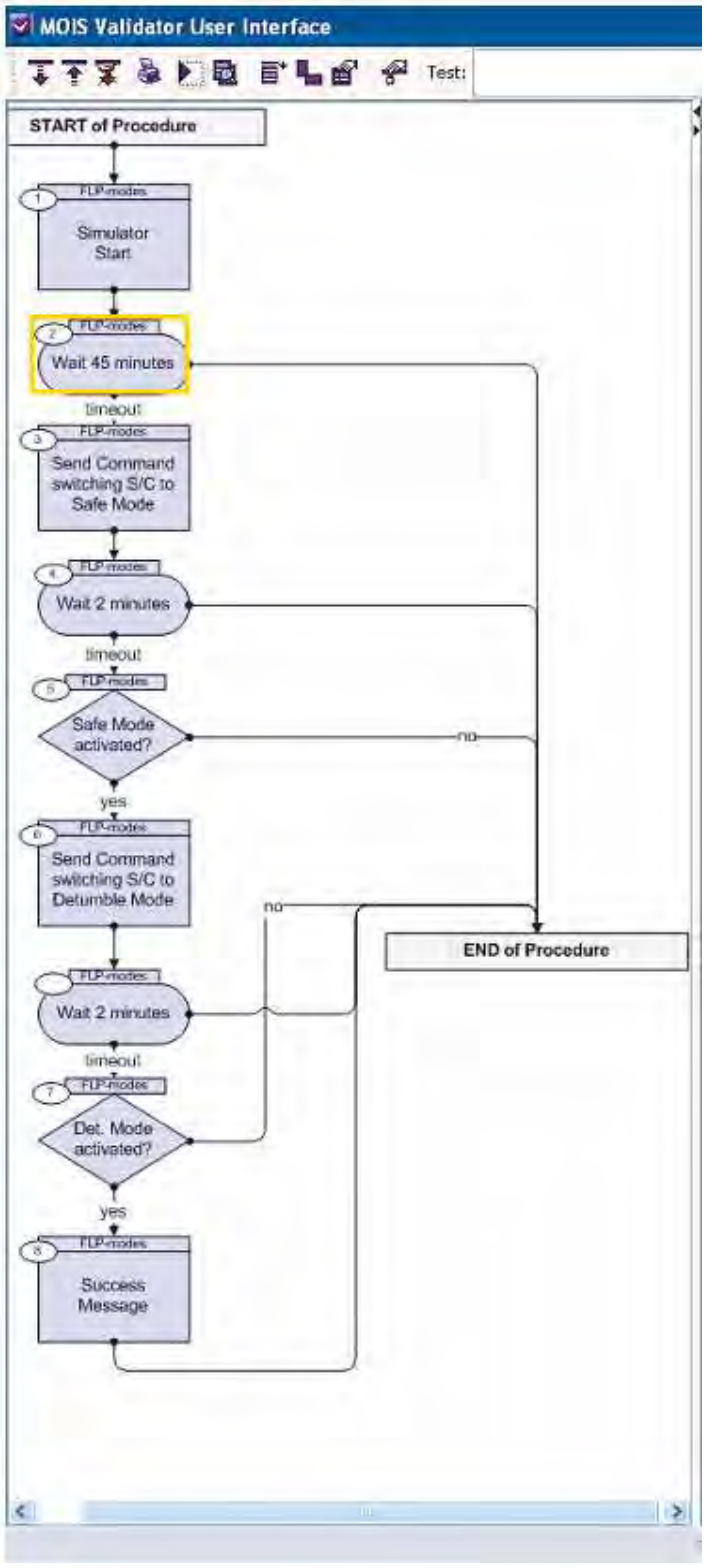

Figura 4.5 – Preparação de procedimentos via Flowcharter do MOIS.

Fonte: adaptada de Eickhoff (2012)

Até o presente momento, foram apresentados apenas o MOIS e a maneira com que os procedimentos são preparados e exportados para a respectiva linguagem de interesse. A partir de agora será apresentada a maneira com que tais procedimentos são executados automaticamente.

Durante a execução, é necessário que o sistema executor tenha acesso aos dados da missão, como dados de AIT ou dados do plano de voo da espaçonave para que o objetivo estabelecido no procedimento seja atingido, portanto, esses dados precisam ser bem estruturados.

Para esclarecer quais são e como esses dados estão sendo mantidos atualmente pela ESA, vamos começar pelo Sistema de Controle da Missão, do inglês *Mission Control System* (MCS) do Centro de Operações Espaciais Européias, o chamado *European Space Operations Centre* (ESOC).

O ESOC é um dos principais centros operacionais de satélites da ESA e utiliza um MCS visando suporte à comunicação entre os Segmentos Solo e Espacial, como planejamento da missão; verificação, transmissão e recepção de TCs e TMs; distribuição e análise de dados; gerenciamento do banco de dados de TCs e TMs; e gerenciamento do software de bordo.

O principal MCS da ESA é o chamado SCOS-2000. Esse sistema mantém os dados da missão em formato de serviços. O mesmo segue as especificações do padrão *Packet Utilization Standard* (PUS) para controle e gerenciamento da comunicação e dos serviços executados pelo satélite. Com a expansão de seu desenvolvimento decorrente dos últimos anos, a disponibilidade de seus serviços foi se tornando cada vez mais complexa.

Para amenizar a complexidade, foi desenvolvida mais uma camada, com o objetivo de disponibilizar os serviços do SCOS-2000, de maneira que pudessem ser reutilizáveis por qualquer outra aplicação. Essa camada é chamada de *Services Management Framework* (SMF), (KOLLER, 2010), (ADAMSON; BARGELLINI, *et al*., 2010) e (LORETUCCI; NISIO, *et al*., 2008).

O SMF foi criado para disponibilizar os serviços do MCS de maneira genérica, para serem utilizados por diferentes sistemas de software, como os sistemas de execução dos procedimentos de teste ou de operação. As atividades e dados, oferecidos pelo SCOS-2000 são disponibilizados via serviços SMF. O SMF foi estabelecido por meio do padrão ECSS-E-ST-70-31C, especificado em (ECSS, 2003). Esse é o mesmo padrão que especifica a estrutura de um SSM, o qual está sendo utilizado no presente trabalho, como descrito na Seção 3.1.

Com o desenvolvimento do SMF, a automatização da execução dos procedimentos tornou-se mais acessível. Isso permite o desenvolvimento de diversas ferramentas, além das já existentes no MOIS.

4.3. Unificação das Ferramentas de Preparo e Execução de Procedimentos em Missões Espaciais da ESA

O objetivo da ESA é unificar todos os elementos para o preparo e a execução de procedimentos, de modo que se comuniquem fazendo parte de um único sistema. Esse é o motivo da iniciativa voltada ao desenvolvimento do EGS-CC. A ideia é que as ferramentas de automatização dos procedimentos passem a utilizar a base de dados SSM para o preparo e o *Kernel* do EGS-CC para a execução dos procedimentos.

Gerenciar diversas missões em paralelo requer também atenção às atividades específicas, que podem variar de missão para missão. Nesse caso, torna-se viável o desenvolvimento de soluções específicas para o atendimento a essas atividades.

A Figura 4.6 sintetiza o relacionamento entre tais elementos. Alguns encontram-se em desenvolvimento, como o caso do EGS-CC e do *SSM Database*. Essa Figura foi gerada na presente dissertação a partir da pesquisa bibliográfica realizada.

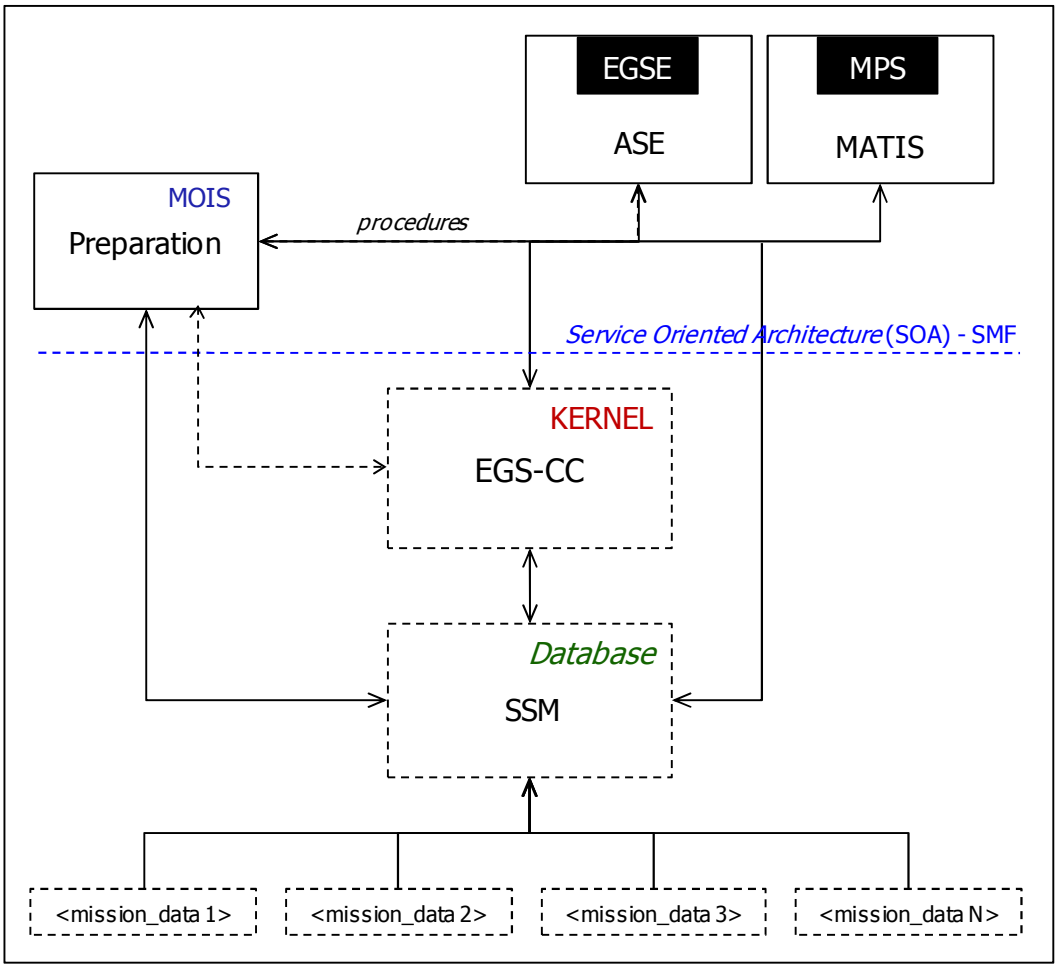

Figura 4.6 – Uma síntese dos principais elementos da estrutura de preparo e execução de procedimentos de teste e operações em missões da ESA.

Pretende-se que o MOIS seja o principal gerenciador de preparo dos procedimentos para casos que não são específicos, visto que para esses casos, os procedimentos poderão ser exportados para reedição em outras soluções.

Dois exemplos de soluções, desenvolvidas para atender esses casos e automatizar a execução de procedimentos são os chamados *Mission Automation System* (MATIS) para operação e o *Automatic Schedule Execution* (ASE) para testes. O MATIS está inserido no contexto do *Mission Planning System* (MPS) e o ASE está inserido no contexto do EGSE.

Ambos mantêm estrutura independente, ou seja, possuem autonomia suficiente para atender as tarefas de preparo e execução dos procedimentos, a partir dos mecanismos necessários para interpretação da PLUTO, por exemplo, e também aceitam procedimentos de entrada, preparados via MOIS e dados da aplicação, modelados via SSM.

4.3.1. O Mission Automation System (MATIS)

O MATIS lê, interpreta e executa automaticamente procedimentos descritos via MOIS. Além disso, permite que o procedimento seja visualizado e editado, mesmo que isso já tenha sido feito via MOIS, possibilitando ajustes para atividades específicas da missão. Durante a execução, o MATIS utiliza os serviços do MCS SCOS-2000 via SMF para interagir com o sistema real ou simulador (ADAMSON; BARGELLINI, *et al*., 2010).

Além do MATIS, outros sistemas também podem utilizar os serviços fornecidos pelo SCOS-2000.

A Figura 4.7 ilustra os componentes da estrutura do MATIS, incluindo também a comunicação entre eles. Onde está sendo citado MCS, leia-se SCOS-2000. O MATIS recebe de entrada os procedimentos em formato XML MOIS.

O autor mostra que os serviços operacionais do MCS SCOS-2000 e dos serviços de rede *Network Interface System* (NIS) são fornecidos pelo SMF de maneira genérica. O NIS também é usado pelo SMF para monitorar os links de comunicação entre as estações terrenas.

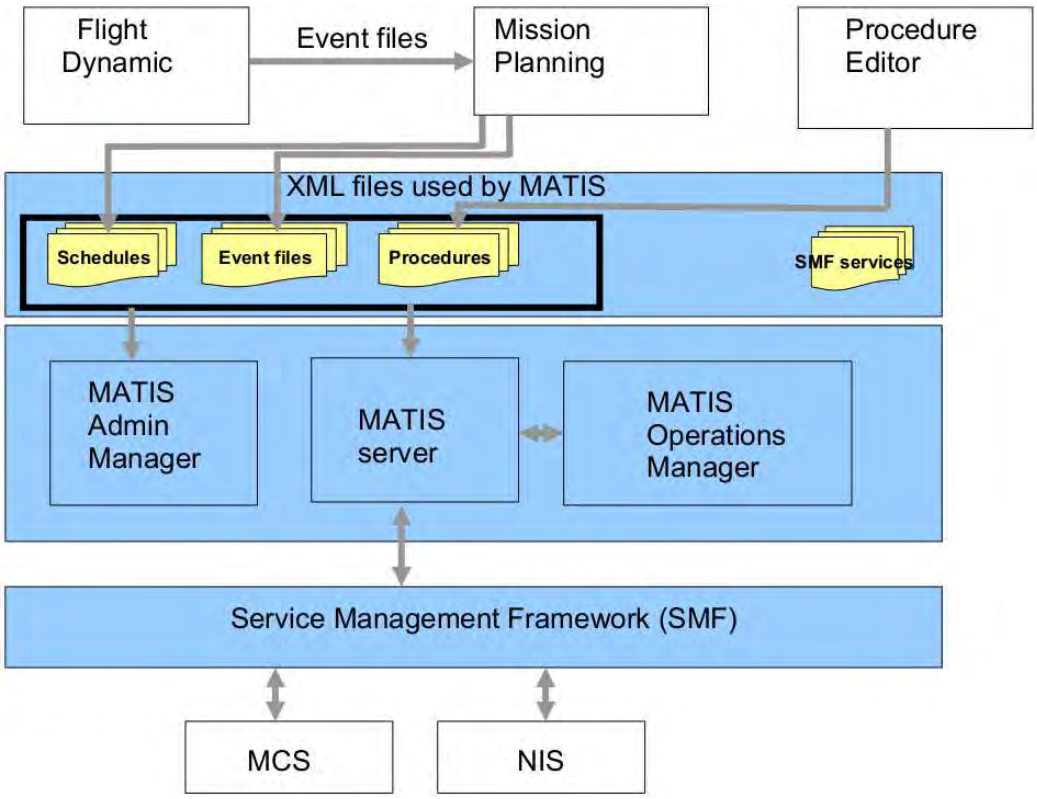

Figura 4.7 – Os componentes do MATIS e da camada SMF para disponibilização dos serviços providos pelo MCS SCOS-2000.

Fonte: Loretucci (2008)

O MATIS é classificado como um sistema de gênero proprietário, na coluna 'Sistema para Execução' da Tabela 2.1 do Capítulo 2.

4.3.2. O Automatic Schedule Execution System (ASE)

O ASE permite descrever, validar e executar procedimentos de teste descritos na sintaxe da PLUTO. O sistema mantém uma *runtime* para a execução do SSM, o chamado *Runtime Space System Model* (RT-SSM).

Esse componente instancia os dados do modelo e durante a execução permite interação com o mundo externo, que nesse caso resume-se em diferentes equipamentos. Isso é feito por meio do conceito de *plugins* (CROCE; SIMONIC, 2008). A Figura 4.8 ilustra a interface gráfica do ASE.

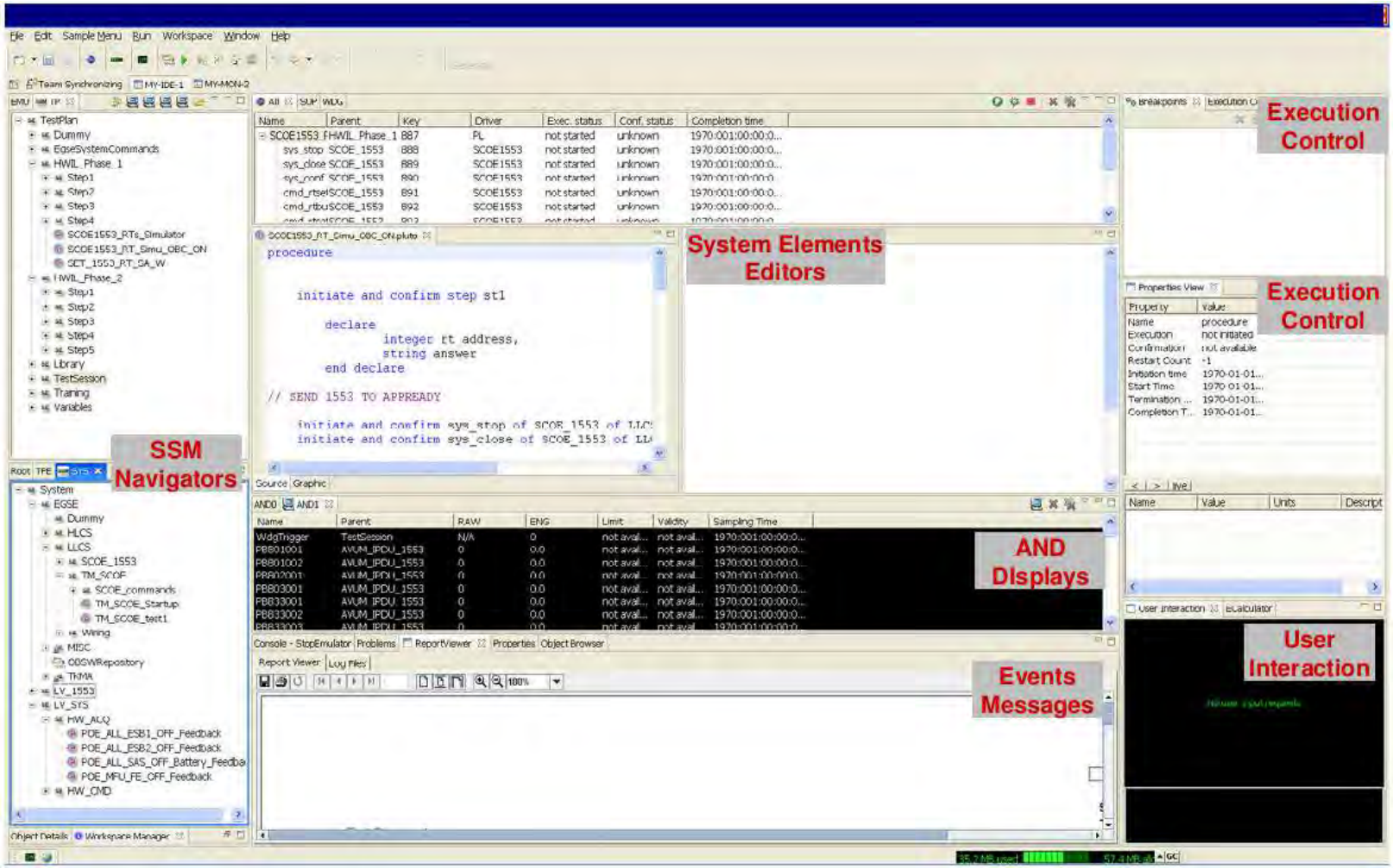

Figura 4.8 - Interface gráfica do ASE.

Fonte: Croce, *et al*. (2008)

Esse sistema é semelhante ao STEPS. O mesmo mantém uma estrutura de gerenciamento do SSM, um *Kernel* para a interpretação dos procedimentos e um conjunto de ferramentas que permitem o preparo e monitoramento da execução. A Figura 4.9 apresenta a arquitetura do ASE.

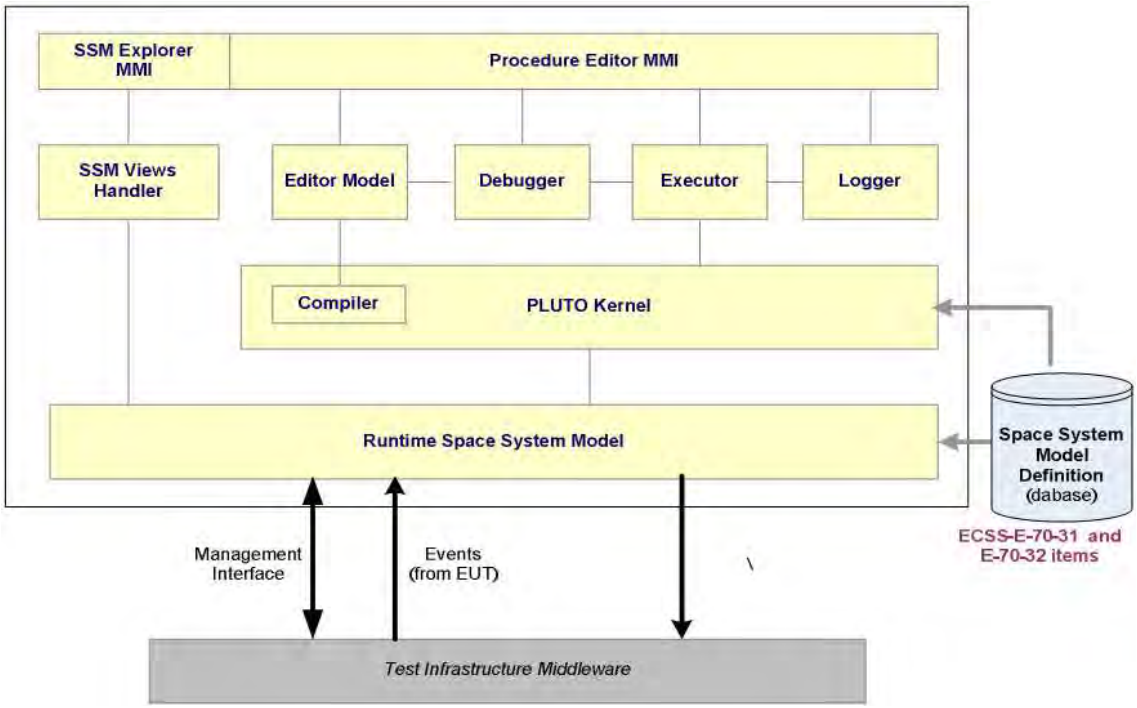

Figura 4.9 - Arquitetura do ASE.

Fonte: Croce, *et al*. (2008)

A partir das informações apresentadas no Capítulo 3, como as características da PLUTO, do SSM e a relação entre ambos, e no Capítulo 4, como os sistemas e ambientes computacionais pertencentes à estrutura de preparo e execução de procedimentos da ESA, foi possível definir a arquitetura estabelecida como objetivo da presente dissertação. A mesma é apresentada no Capítulo a seguir.

5 UMA ARQUITETURA COMPUTACIONAL PARA USO DA PLUTO EM ATIVIDADES ESPACIAIS DO INPE

O presente Capítulo apresenta a arquitetura do *Spacecraft Test Procedures System* (STEPS) com a descrição dos elementos e recursos necessários para o preparo e a execução de procedimentos descritos com a PLUTO, bem como de suas características modulares e das interfaces de comunicação entre os módulos da arquitetura e o *System Under Test* (SUT).

5.1. A Arquitetura do Spacecraft Test Procedures System (STEPS)

A arquitetura do STEPS possui características modulares. A mesma foi definida para se comunicar com um sistema, um subsistema ou um equipamento sob teste, o SUT. A Figura 5.1 ilustra a arquitetura do STEPS.

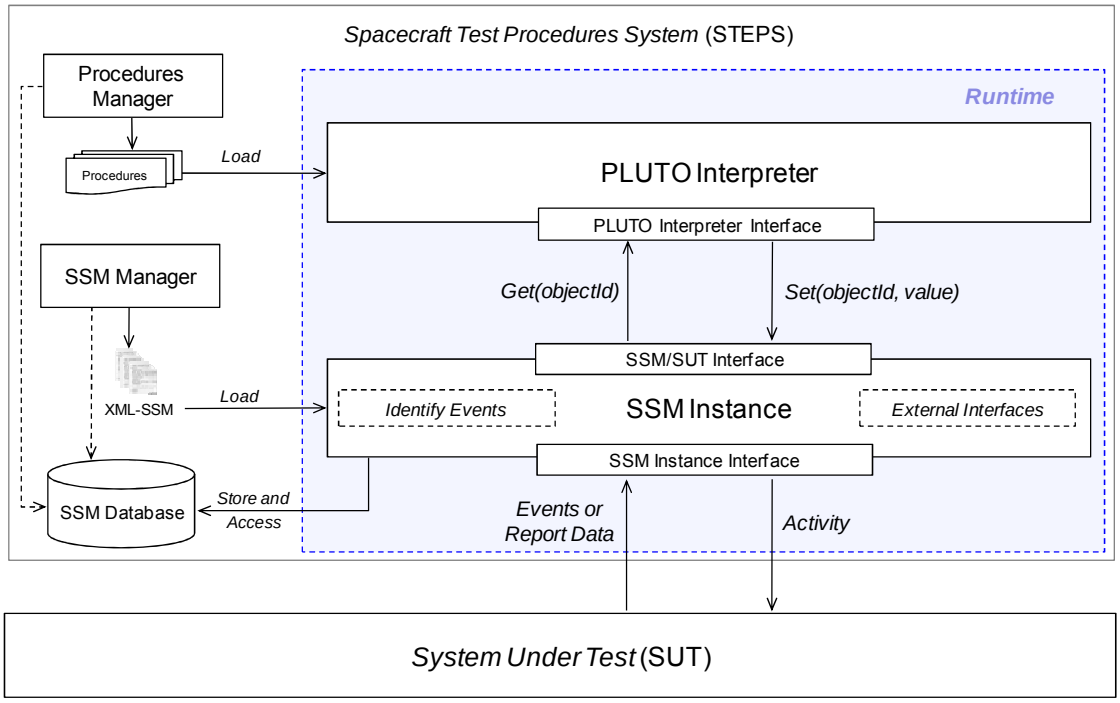

Figura 5.1 - Arquitetura do STEPS

Os módulos que compõem a arquitetura do STEPS foram definidos a partir dos padrões da ECSS e de trabalhos correlatos. Tais módulos são: *Procedures*

Manager; *SSM Manager*; *PLUTO Interpreter*; *SSM Instance*, composto pelos componentes *Identify Events* e *External Interfaces*; e o *SSM Database*.

Essa arquitetura permite o preparo, a execução e o monitoramento da execução de procedimentos de testes descritos com a PLUTO. A direção do apontamento das setas representa a direção do fluxo dos dados.

Os próximos subitens detalham os módulos da arquitetura proposta.

5.1.1. Módulos para o Preparo do Modelo e dos Procedimentos

A preparação dos procedimentos conta com o apoio dos módulos *Procedures Manager* e *SSM Manager*. Ambos foram definidos para auxílio ao usuário na descrição do procedimento, na modelagem do SSM e no monitoramento da execução. Cada ferramenta pode ser desenvolvida com uma GUI, para evitar que erros sejam introduzidos no preparo dos procedimentos. A Figura 5.2 ilustra os módulos definidos para uso no preparo dos procedimentos e do modelo.

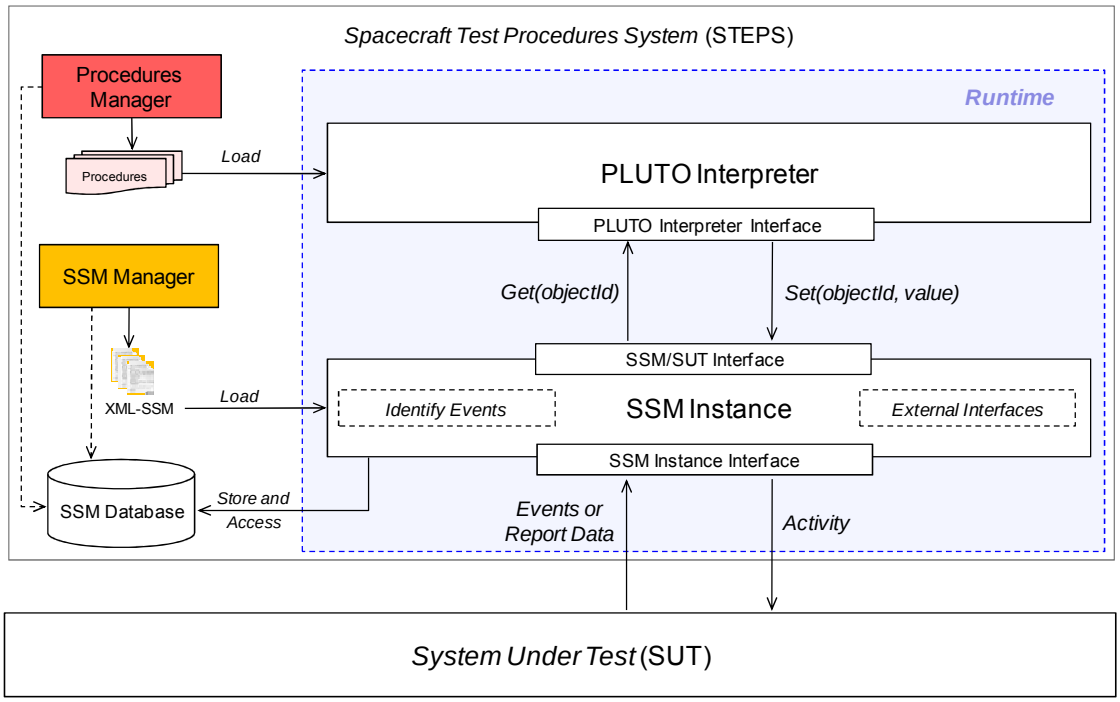

Figura 5.2 - Módulos para o Preparo do Modelo e dos Procedimentos

Ambos geram dois conjuntos de dados para que os módulos da *Runtime* possam usá-los na execução. A *Runtime* mantém os módulos que se encarregam de interpretar o procedimento e gerenciar o modelo durante a execução. Esses dois conjuntos de dados podem ser exportados para arquivos com extensão (*.pluto) para manter o procedimento e extensão (*.ssm) para manter o modelo.

O procedimento editado por meio do *Procedures Manager* deve ser cuidadosamente validado para manter a sintaxe correta da linguagem e não permitir qualquer referência inválida aos objetos do SSM. No *Main Body*, objetos do modelo podem ser referenciados dentro ou fora dos *steps*. Nesse caso, o nome ou caminho do respectivo objeto deve ser enviado ao modelo durante a execução da respectiva instrução.

A estrutura e dados do modelo são registrados em formato XML ao serem exportados via *SSM Manager*. A XML está sendo utilizada porque apresenta características semelhantes às características especificadas para um SSM, como a hierarquia dos dados.

A Figura 5.3 ilustra a estrutura de um SSM escrito em formato XML.

```
<System_Element id = "value" name = "value" object_type = "value" type = "value">

        <System_Element id = "value" name = "value" object_type = "value" type = "value">
        </System_Element>
                ...
        <Report_Data id = "value" name = "value" object_type = "value" type = "value">
        </Report_Data>
                ...
        <Activity id = "value" name = "value" object_type = "value" type = "value">
        </Activity>
                ...
        <Event id = "value" name = "value" object_type = "value" type = "value">
        </Event>
                ...
</System_Element>
```

Figura 5.3 - Estrutura padrão de um SSM em formato XML

Um SSM também é composto de nós que são arranjados hierarquicamente e cada nó do formato XML representa um objeto SSM do tipo *System Element*, *Report Data*, *Activity* ou *Event*. Cada nó do formato XML também permite a inclusão de propriedades para o registro das características de cada objeto. Mais detalhes a respeito das características foram apresentados na Seção 3.1.

5.1.2. Módulos para Execução dos Procedimentos

Após a preparação do procedimento e do modelo, a *Runtime* pode ser disparada para iniciar a execução. O conteúdo dos dois arquivos, (*.ssm) e (*.pluto), são lidos e usados como dados de entrada pela estrutura de gerenciamento do modelo em memória, o *SSM Instance*, e pelo interpretador, o *PLUTO Interpreter*.

A Figura 5.4 destaca os módulos definidos para a execução dos procedimentos.

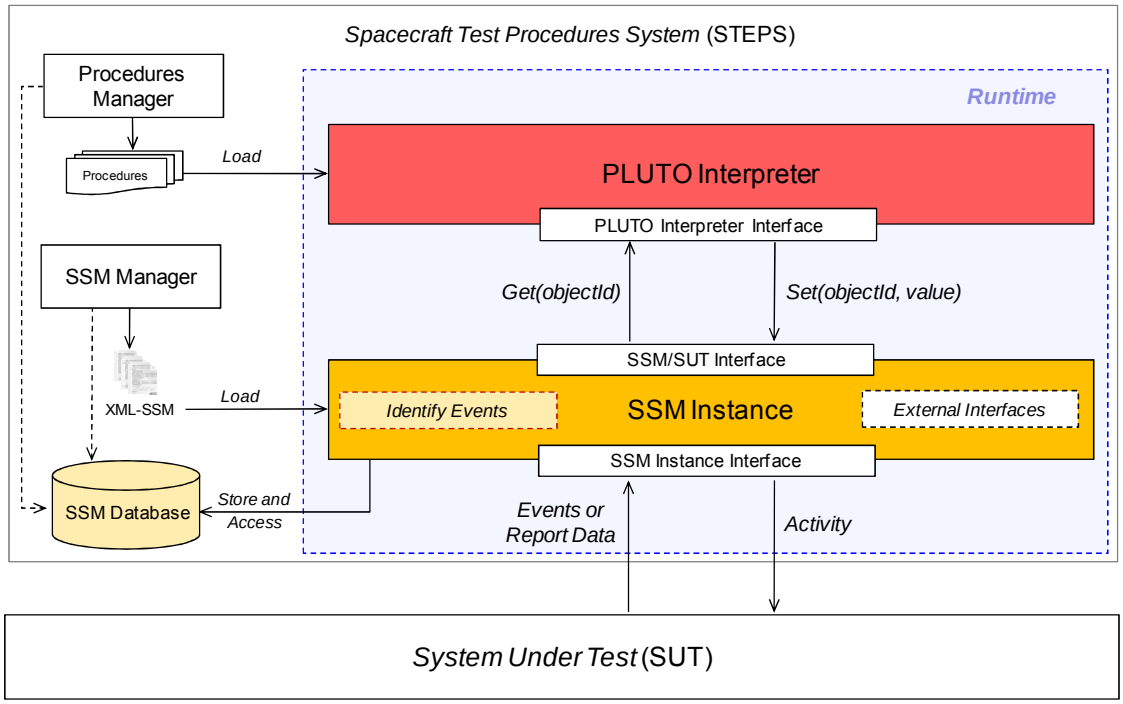

Figura 5.4 - Módulos para Execução dos Procedimentos

A necessidade de comunicação entre os módulos *PLUTO Interpreter* e *SSM Instance* é que durante a execução, o interpretador necessita acessar os dados dos objetos do modelo, por isso esses objetos devem estar acessíveis ao interpretador. A Seção 3.4 justifica em detalhes os motivos da necessidade dessa comunicação.

O controle do SSM deve ser em tempo de execução e o mesmo deve ser gerenciado de modo que seus objetos e dados sejam facilmente rastreados. Para manter os dados do modelo acessíveis durante a execução, o módulo *SSM Instance* cria uma estrutura computacional, com as mesmas características hierárquicas presentes na estrutura do modelo.

Essa estrutura computacional é alocada em memória e destina-se ao carregamento dos dados do modelo. Uma vez carregados para tal estrutura, os objetos já poderão ser acessados pelo *PLUTO Interpreter*.

O módulo *PLUTO Interpreter* contém as rotinas computacionais que se encarregam de interpretar e executar os procedimentos. Além da interpretação e execução, esse módulo também contém uma interface para comunicação com o módulo *SSM Instance* para ter acesso aos objetos do modelo.

O *SSM Instance* se comunica de um lado com o SUT e do outro com o *PLUTO Interpreter*. A estrutura do modelo recebe as solicitações do interpretador para executá-las e registra os dados recebidos do SUT em seus respectivos objetos. Tais dados podem ser respostas geradas durante a execução de alguma *Activity*, onde pode haver a troca de comandos ou somente relatos gerados automaticamente pelo SUT, como dados de *housekeeping* ou eventos. Os dados de *housekeeping*, na maioria dos casos são gerados periodicamente.

O termo '*Instance'* foi atribuído ao módulo *SSM Instance* porque o modelo é carregado em uma estrutura computacional instanciada em memória. Esse

termo refere-se à instância do SSM em memória para que seus dados sejam gerenciados durante a execução do procedimento.

O componente *Identify Events* do módulo *SSM Instance* é usado na configuração da identificação de eventos. Os eventos definidos por meio desse componente são classificados como eventos customizados e representam a ocorrência de uma condição ou um conjunto de condições, inseridas no contexto das funcionalidades e *Report Data* do modelo.

As condições ou conjunto de condições podem ser estabelecidas pelo usuário durante a configuração desses eventos e podem ser geradas a partir dos dados originados como *Report Data*. Existem também os eventos não customizados, que são eventos padrões, gerados pelo próprio SUT.

O elemento *SSM Database* registra as ocorrências provenientes da execução dos procedimentos. Ao se atingir um número considerável de registros, poderá ser possível trabalhar com os dados para geração de conhecimento. O conhecimento adquirido pode ser aplicado no desenvolvimento de outros equipamentos do mesmo gênero.

Além de poder ser usada para o registro dos resultados da execução dos procedimentos, o *SSM Database* também mantém o registro de toda a estrutura do SSM, incluindo os dados da missão. O *SSM Database* também pode ser usado pelas ferramentas de preparo dos procedimentos.

5.1.3. Interfaces de Comunicação entre os Módulos da Arquitetura

As interfaces de comunicação gerenciam o fluxo de dados e são as responsáveis pela comunicação entre *PLUTO Interpreter*, *SSM Instance* e SUT. O *PLUTO Interpreter* se comunica com o *SSM Instance* por meio de solicitações do tipo *Get* e *Set*:

- **Solicitação *Set*:** A solicitação do tipo *Set* é enviada ao *SSM Instance* para solicitar a execução de alguma funcionalidade, da qual se resume em um objeto do tipo *Activity*; e

- **Solicitação *Get*:** A solicitação do tipo *Get* é enviada ao *SSM Instance* para solicitar a leitura do valor de algum objeto que seja do tipo *Report Data* ou *Event*.

A Figura 5.5 destaca as interfaces de comunicação entre os módulos da arquitetura e comunicação do STEPS com o SUT e equipamentos que interagem com o ambiente externo.

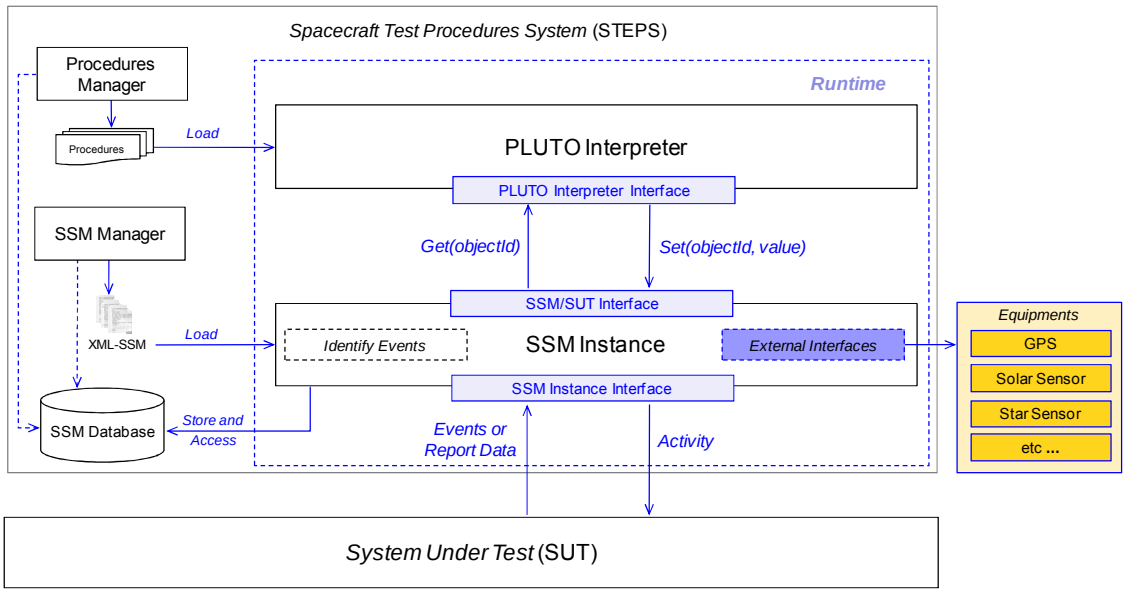

Figura 5.5 - Interfaces de comunicação entre os módulos da arquitetura, SUT e elementos do ambiente externo

O componente *External Interfaces* possibilita a comunicação do STEPS com equipamentos que interagem com o ambiente externo. Exemplo de alguns desses equipamentos: sensor solar, sensor de estrela, sensor GPS, etc. Esses são equipamentos de aquisição de dados referentes ao ambiente externo do satélite para auxílio à sua navegação durante o controle de atitude e órbita.

O *External Interfaces* permite maior flexibilidade ao usuário para configurar as interfaces de comunicação com esses equipamentos. O mesmo encontra-se no *SSM Instance*, pois durante a comunicação, o STEPS necessita do intercâmbio de dados com tais equipamentos.

O usuário, ao modelar o SSM, pode criar objetos do tipo *Activity*, por exemplo, e relacioná-los com o *External Interfaces*. Isso significa que esses equipamentos podem ser controlados por meio do procedimento.

Uma situação comum, por exemplo, é a execução de um procedimento de teste de um computador de bordo, o qual requer parâmetros de equipamentos do ambiente externo. Isso pode ser exemplificado a partir da atividade desempenhada por um sensor solar, cujo objetivo é enviar dados ao computador de bordo para que seja feito o controle de atitude da plataforma.

Atualmente, não existe recurso semelhante, muito menos sendo usado via procedimento PLUTO. Contudo, uma instrução foi definida, no presente trabalho, para acesso aos objetos do modelo que interagem com esses equipamentos, por meio do *External Interfaces*.

5.1.4. Definição de uma Instrução para Uso do External Interfaces

A comunicação entre o STEPS e os equipamentos do ambiente externo é realizada a partir da instrução chamada *Simulate Environment*, cujo objetivo é solicitar ao *SSM instance* a comunicação com tais equipamentos.

Durante sua execução, o módulo *PLUTO Interpreter* envia uma solicitação *Set* ao *SSM Instance* com a referência ao objeto do modelo que tenha sido modelado para fazer uso do *External Interface*. Essa solicitação realiza o rastreio do objeto para que a execução seja iniciada.

Durante a modelagem do SSM, o usuário é quem define o objeto do modelo que realiza essa funcionalidade, juntamente da interface de comunicação com o respectivo equipamento.

5.2. Relação entre o ASE e o STEPS

Dois sistemas apresentados nas Seções 4.3.1 e 4.3.2, MATIS e ASE, são os sistemas que encontram-se no mesmo nível de abrangência e aplicabilidade do STEPS.

Os objetivos do STEPS, MATIS e ASE são semelhantes, porém é mais adequado escolher o ASE para uma análise comparativa com o STEPS, pelo fato de também ser aplicado em procedimentos de teste. A Figura 5.6 descreve o nível de abrangência do STEPS em relação a toda estrutura de preparo e execução automática de procedimentos da ESA.

A Figura 5.7 apresenta as principais diferenças, presentes em ambas arquiteturas, e a Tabela 5.1 apresenta a correlação de equivalência entre os elementos da arquitetura do STEPS e os elementos do ASE.

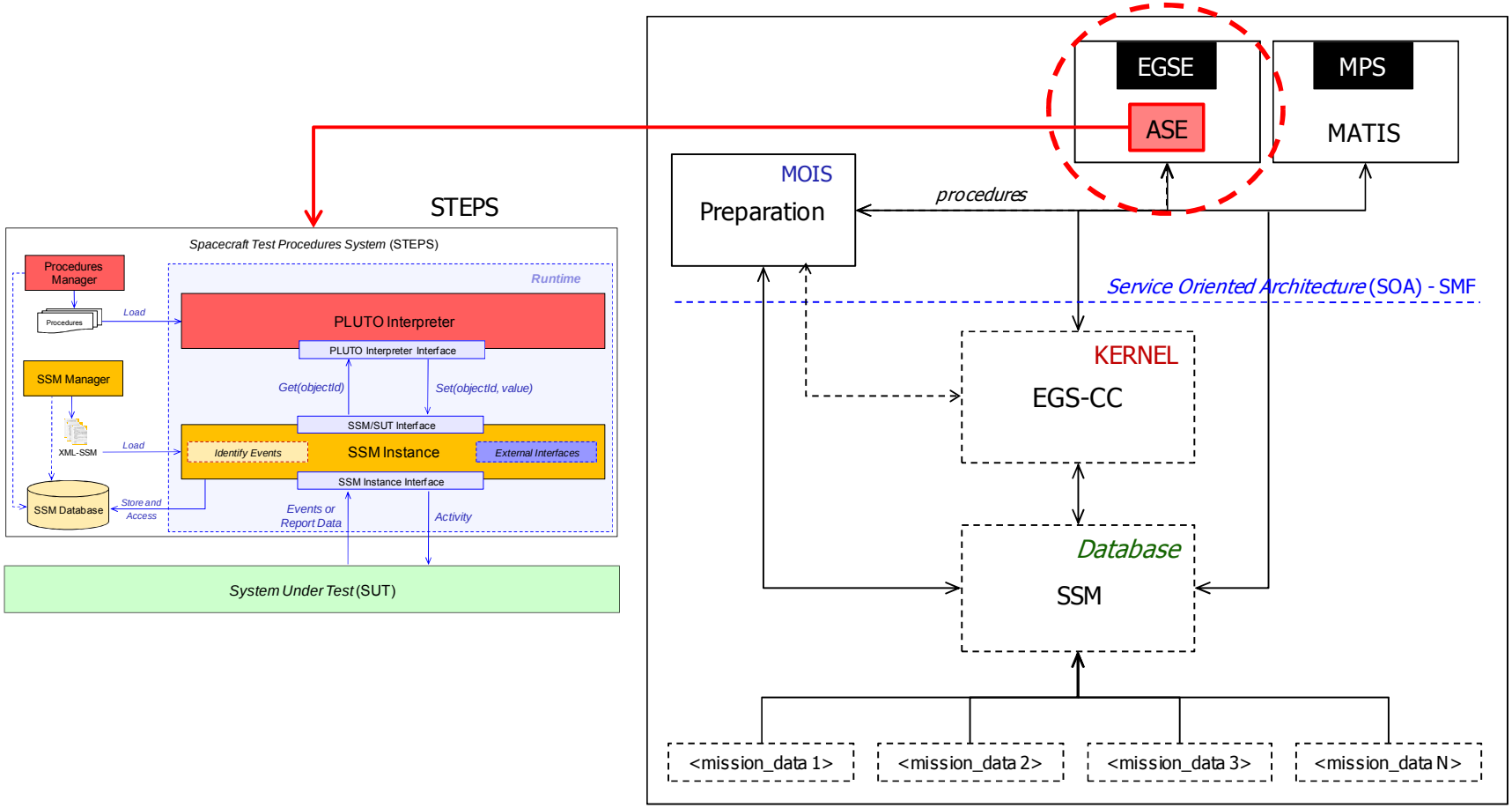

Figura 5.6 - Nível de abrangência do STEPS em relação a toda estrutura de preparo e execução automática de procedimentos da ESA

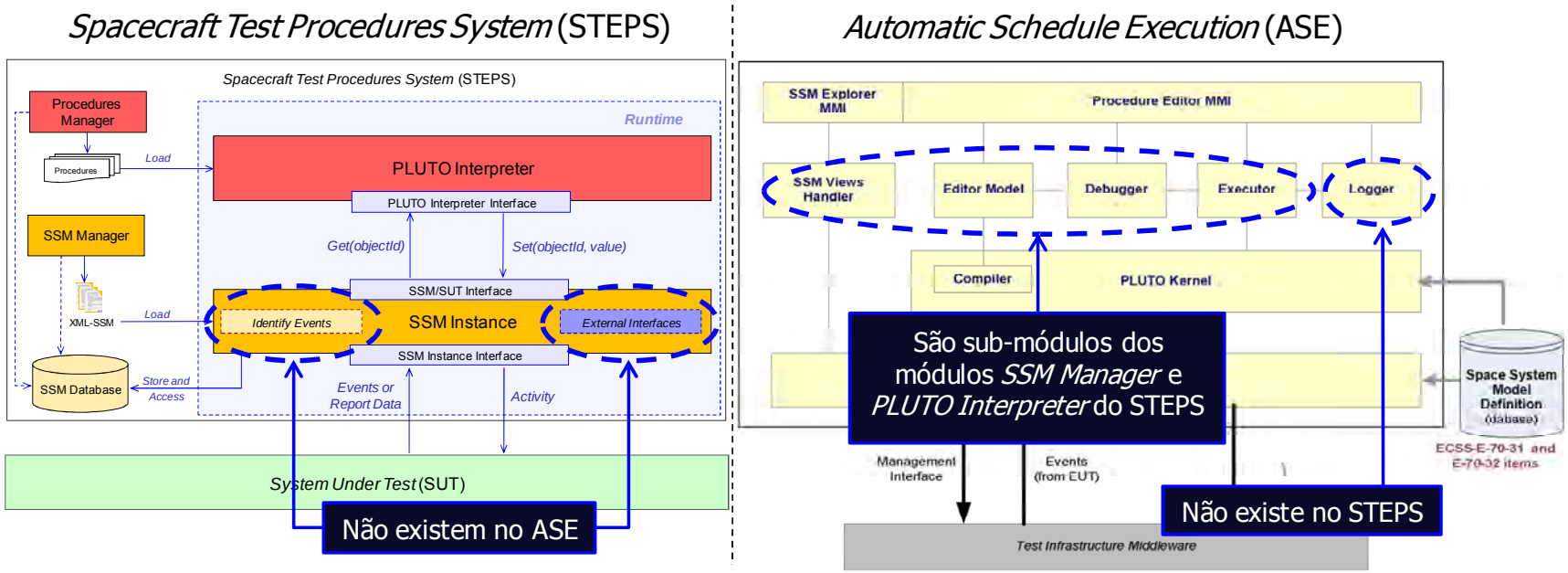

Figura 5.7 - Relação entre os diferenciais presentes nas arquiteturas do STEPS e do ASE

Tabela 5.1 – Relação entre os elementos da arquitetura do STEPS e elementos da arquitetura do ASE.

		ASE										
		SSM Explorer MMI	Procedure Editor MMI	SSM Views Handler	Editor Model	Debugger	Executor	Logger	Compiler	PLUTO Kernel	RT- SSM	SSM Database
STEPS	PLUTO Interpreter					X	X		X	X		
	SSM Instance										X	
	External Interfaces											
	Identify Events											
	Procedures Manager		X									
	SSM Manager	X		X	X							
	Database											X

A arquitetura do ASE, (CROCE; SIMONIC, 2008), não apresenta recurso semelhante ao *Identify Events* da arquitetura do STEPS. O ASE realiza o processamento dos eventos gerados unicamente pelo SUT e não permite gerenciar a definição de eventos customizados para eventuais testes. O STEPS permite essa flexibilidade junto ao usuário.

O ASE também não apresenta recurso semelhante ao *External Interfaces*. O ASE se comunica com equipamentos do ambiente externo, porém não com a flexibilidade fornecida pelo módulo *External Interface* do STEPS. Além disso, não permite gerenciar a comunicação com tais equipamentos por meio de procedimentos descritos em PLUTO (CROCE; SIMONIC, 2008).

Com o *External Interfaces*, o STEPS proporciona facilidades ao usuário para definir as interfaces de comunicação com equipamentos do meio externo, e isso pode ser também aplicado para equipamentos específicos, que variam de missão para missão. Essas interfaces podem ser criadas junto aos objetos do SSM durante sua modelagem. O fato de serem interligadas aos objetos do SSM permite que sejam acessadas por meio do procedimento.

Sob o ponto de vista da arquitetura do STEPS, não existe recurso semelhante ao chamado *Logger* da arquitetura do ASE. Está previsto a inclusão dessa característica em próximas versões do STEPS.

O próximo Capítulo mostra um protótipo do STEPS, desenvolvido para exercitar e validar a arquitetura proposta nesta dissertação. Um estudo de caso, definido para demonstração do funcionamento dessa arquitetura também será apresentado.

6 DESENVOLVIMENTO DE UM PROTÓTIPO E SUA UTILIZAÇÃO JUNTO A UM ESTUDO DE CASO

O protótipo do STEPS foi desenvolvido no presente trabalho e consta de alguns elementos da arquitetura, necessários para o preparo e a execução de um procedimento de teste com a PLUTO a partir do estudo de caso. O presente Capítulo mostra esses elementos e descreve o processo de preparo e execução desse procedimento, bem como o estudo de caso.

6.1. Módulos da Arquitetura

Os módulos da arquitetura do STEPS presentes no protótipo são:

- PLUTO Interpreter;
- SSM Instance;
- PLUTO Manager; e
- SSM Manager, incluindo o componente *External Interfaces*.

Os primeiros módulos que foram desenvolvidos são o *SSM Instance* e o *PLUTO Interpreter*. Eles foram desenvolvidos para a execução de um procedimento de teste que foi descrito durante o estudo de caso, apresentado na Seção 6.3.

Para executar tal procedimento, as atividades mostradas na Figura 6.1 e na Figura 6.2 são executadas pelo *PLUTO Interpreter* e pelo *SSM Instance*. A Figura 6.1 apresenta os passos realizados durante a interpretação e execução do procedimento e a Figura 6.2 apresenta as atividades realizadas para a gestão da estrutura e dos dados do SSM durante a execução. Os dois módulos foram criados separadamente e se comunicam por meio das interfaces.

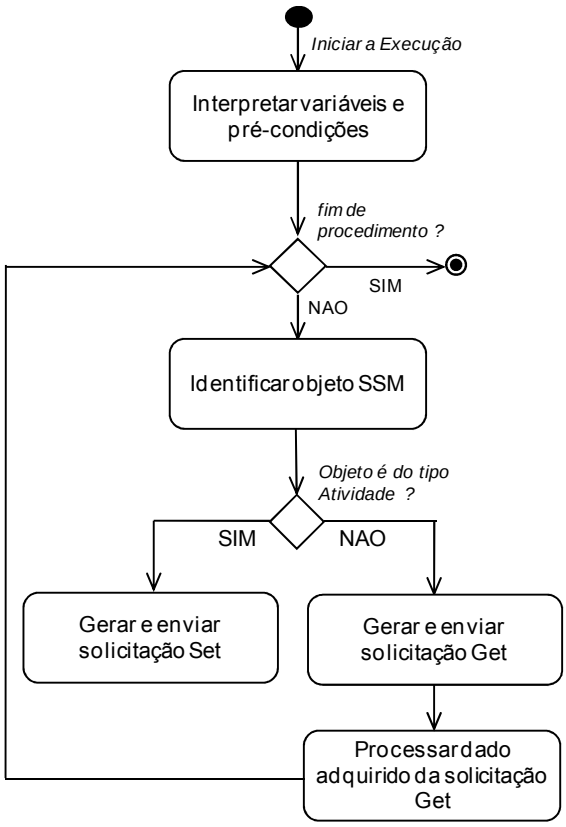

Figura 6.1 - Atividades executadas pelo PLUTO Interpreter

Para o *Main Body* do procedimento, tanto a interpretação quanto a execução são feitas linha por linha. A interpretação é feita por meio de um *Reader* que divide cada linha em partes independentes. Essas partes são chamadas de *tokens* (MAK, 2009).

Cada *token* é interpretado para identificação da respectiva instrução descrita no procedimento; identificação do *path* do objeto do modelo, se houver; e identificação de algum dado destinado para escrita no respectivo objeto do modelo.

A aplicação de um *Parser* é feita sob esse conjunto de *tokens*, com objetivo de validar as palavras reservadas da PLUTO; tipos de variáveis e dados; nome e *path* do objeto do SSM. Uma vez realizada essas duas tarefas, *Reader e*

Parser, o *PLUTO Interpreter* se encarrega de criar a solicitação adequada para cada instrução que referencia os objetos do SSM.

Se o objeto SSM, referenciado na respectiva instrução, for do tipo *Activity*, uma solicitação do tipo *Set* é criada, caso contrário, será criada uma do tipo *Get*. A solicitação do tipo *Get* é usada para leitura de dados em objetos do tipo *Report Data* e *Event*.

A referência sob algum objeto do modelo é feita a partir de seu nome de identificação (um rótulo), o mesmo nome inserido no campo 'Nome' da Tabela 3.1. O analista encarregado de testar o SUT deve ter conhecimento do objetivo de cada objeto. Isso significa que para modelar o SSM, deverá conhecer o sistema.

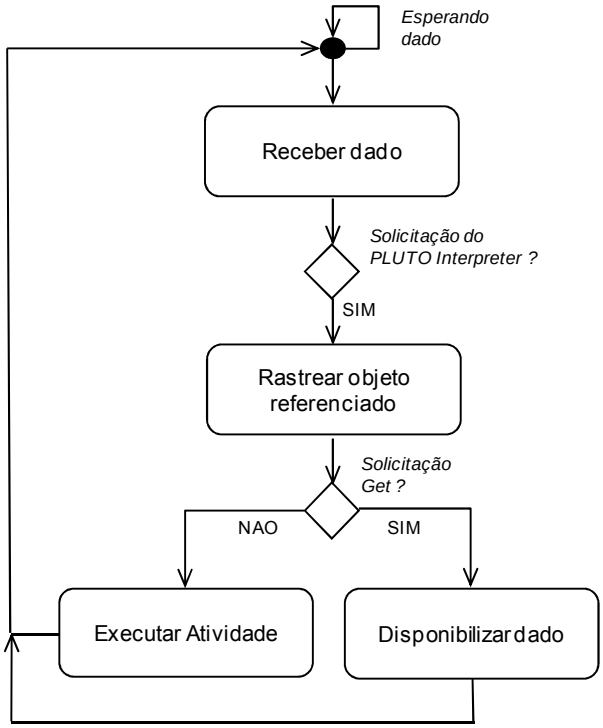

Figura 6.2 - Atividades executadas pelo SSM Instance

As solicitações *Get* e *Set* mantêm o seguinte formato: *Get(objectId)* e *Set(objectId, value)*. Ao receber uma solicitação, o *SSM Instance* a reconhece,

rastreia o objeto de interesse e executa a ação correspondente ao objetivo atribuído ao objeto referenciado.

O objetivo de cada objeto do modelo é atribuído durante a modelagem e está relacionado com cada tipo, *Report Data*, *Event* ou *Activity*. Cada tipo mantém um objetivo específico, seja para leitura e escrita de dados ou para execução das funcionalidades do SUT.

6.2. Interface Gráfica

O STEPS possui uma interface gráfica que permite o gerenciamento da preparação e execução dos procedimentos de teste. O protótipo dessa interface é ilustrada na Figura 6.3.

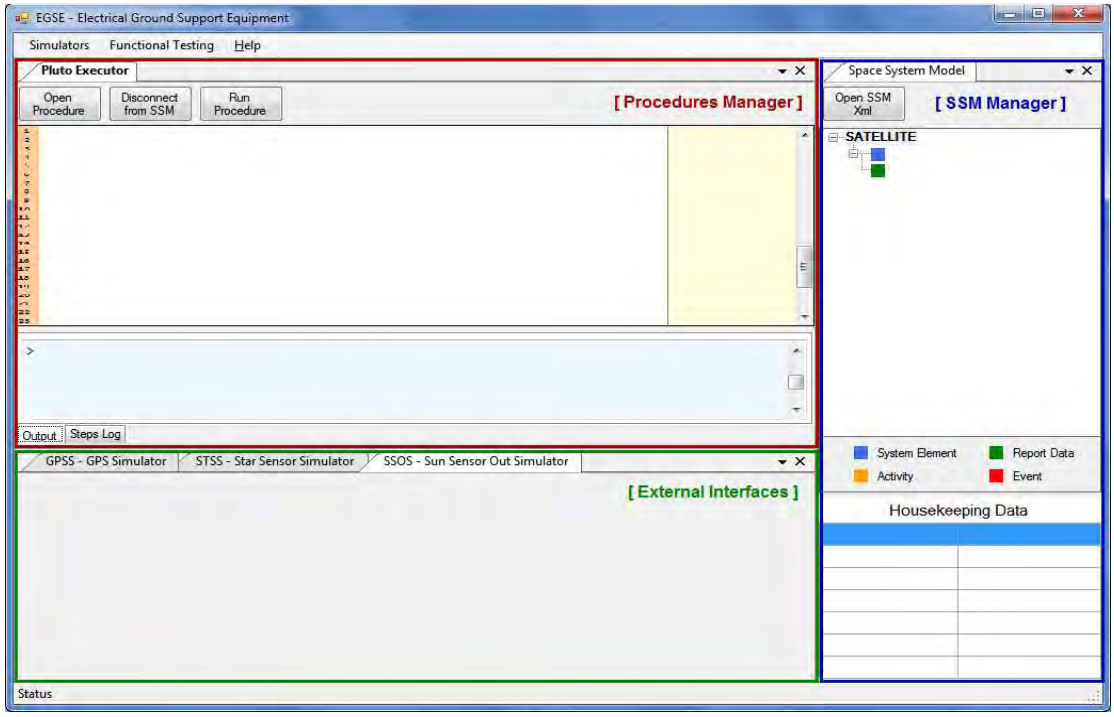

Figura 6.3 - Interface gráfica para o protótipo do STEPS

Essa interface compreende os módulos *Procedures Manager* e *SSM Manager*. Para monitorar a execução do procedimento, a área *Housekeeping Data* da interface foi destinada para exibição dos dados recebidos e armazenados pelo

SSM Instance. Esses são os dados que são gravados dentro da estrutura do modelo.

A comunicação do STEPS com equipamentos externos deve ser configurada na área identificada de *External Interfaces*. Essa área poderá receber diferentes interfaces para comunicação com diferentes equipamentos que não estejam vinculados diretamente no SUT. A mesma foi introduzida para atender aos casos onde o SUT necessita se comunicar com equipamentos externos.

O presente protótipo apresenta três tipos de *External Interfaces*. Essas interfaces compreendem três equipamentos do tipo sensores: um *Global Positioning System* (GPS), um *Sun Sensor* e um *Star Sensor*. Um *Sun Sensor* foi utilizado nesse trabalho. Esse sensor não está disponível para uso em estudo de caso, mas foi simulado para a aplicação do protótipo do STEPS.

A Figura 6.4 ilustra o protótipo da GUI do STEPS durante execução com a descrição de um trecho do procedimento criado, do SSM e dos dados provenientes do status de sua execução no estudo de caso.

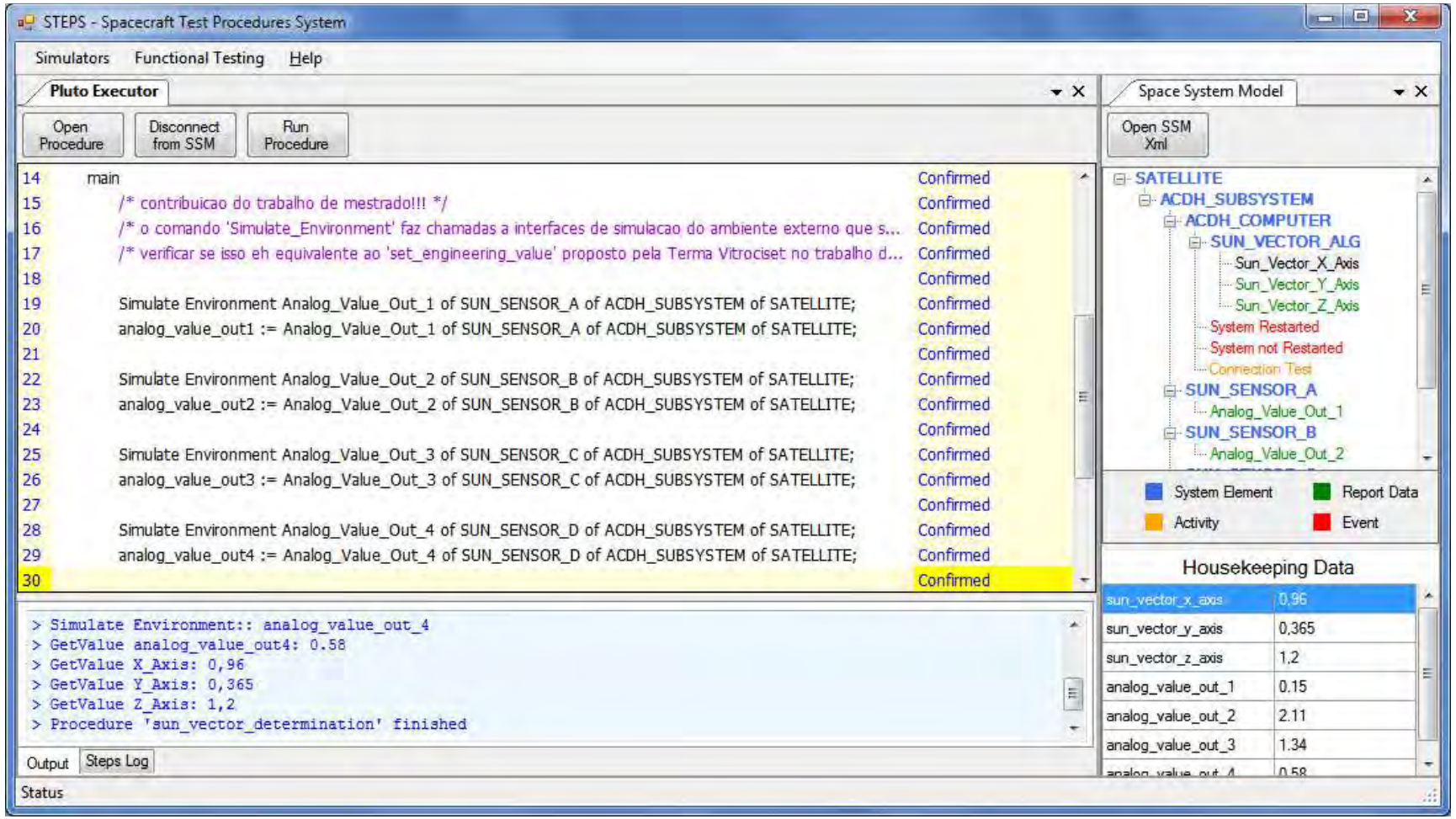

Figura 6.4 - Protótipo da GUI do STEPS durante a execução

6.3. O Estudo de Caso

O estudo de caso foi derivado do projeto de um computador de bordo que está sendo desenvolvido no INPE para controle de atitude e supervisão de dados a bordo de uma plataforma com estabilização em três eixos. O desenvolvimento desse computador é parte dos objetivos do projeto chamado Sistemas Inerciais para Aplicações Aeroespaciais (SIA).

O estudo de caso aborda uma funcionalidade do software de controle de atitude da plataforma orbital a ser executado no computador de bordo também desenvolvido no contexto do Projeto SIA.

6.3.1. O Projeto SIA

O Projeto SIA inclui participação do Departamento de Ciência e Tecnologia Aeroespacial (DCTA) e o Instituto Nacional de Pesquisas Espaciais (INPE), ambos vinculados ao Ministério de Ciência e Tecnologia (MCT).

Com propósito de melhorar a precisão no guiamento de veículos com navegação inercial, o Projeto SIA visa planejamento, fabricação, integração e qualificação de sensores, software e eletrônica embarcada para duas plataformas inerciais: uma para uso em satélites e outra para uso em Veículos Lançadores de Satélites (VLS) (CGEE, 2006) e (CGEE, 2010).

Em adição às necessidades de instrumentos de ponta para medições inerciais, encontra-se também o desenvolvimento de um sistema de controle de atitude, o *Attitude Control System* (ACS). O ACS utiliza um conjunto de algoritmos de controle para determinar o posicionamento da plataforma orbital durante sua navegação.

6.3.2. O Computador de ACDH

O computador de bordo da plataforma orbital do Projeto SIA será responsável pelas atividades de supervisão de bordo, *On-Board Data Handling* (OBDH) e

de controle de atitude e órbita de satélites, *Attitude and Orbit Control* (AOC). Com a junção dessas duas tarefas, *Attitude Control* e *Data Handling*, esse computador está sendo chamado de *Attitude Control and Data Handling* (ACDH) *Computer*, o Computador de ACDH.

Como parte do processo de validação do Computador de ACDH para uso na plataforma orbital é necessário a definição e aplicação de testes para garantir a confiabilidade desejada na aplicação. Testes elétricos, funcionais, de interface e desempenho são testes que serão aplicados ao Computador de ACDH em forma de procedimentos, por meio de um EGSE.

Um EGSE também está sendo desenvolvido pelo Grupo SUBORD via Projeto SIA, para ser o equipamento responsável pelo gerenciamento da execução dos procedimentos de teste do Computador de ACDH.

6.4. Especificação do Estudo de Caso

O Computador de ACDH possui interfaces de comunicação analógica com quatro sensores que medem a incidência de raios solares no satélite. Esses sensores são instalados em posições estratégicas, nas superfícies externas da plataforma, para potencializar o ganho na receptividade dos raios solares.

Cada sensor gera uma medida em *Volts*, que representa a intensidade de luz solar refletida na respectiva superfície da plataforma. Essas medidas são disponibilizadas para o Computador de ACDH via interfaces analógicas e permitem ter o conhecimento da posição atual do sol.

Para o presente estudo de caso, foram assumidos quatro parâmetros de saída gerados pelos quatro sensores solares.

Os quatro parâmetros de saída desses sensores são processados pelo Computador de ACDH e convertidos em parâmetros de entrada, coerentes com o formato aceito por um algoritmo chamado Algoritmo de Determinação do Vetor Sol (ADVS).

É importante enfatizar que o funcionamento interno desse algoritmo faz parte da área de Teoria de Controle. Esse não é o foco do presente estudo de caso. O que é tratado são as entradas e saídas, ou seja, o funcionamento interno desse algoritmo não faz parte do presente trabalho.

6.4.1. Algoritmo de Determinação do Vetor Sol (ADVS)

O Algoritmo de Determinação do Vetor Sol (ADVS) recebe os quatro parâmetros gerados pelos quatro sensores solares, um parâmetro de cada sensor, e calcula os valores dos vetores X, Y e Z.

Esses vetores são as coordenadas usadas como referência na realização de manobras durante o controle de atitude. Tais manobras são executadas para o posicionamento adequado da plataforma no espaço. A Figura 6.5 ilustra o cenário de funcionamento do ADVS.

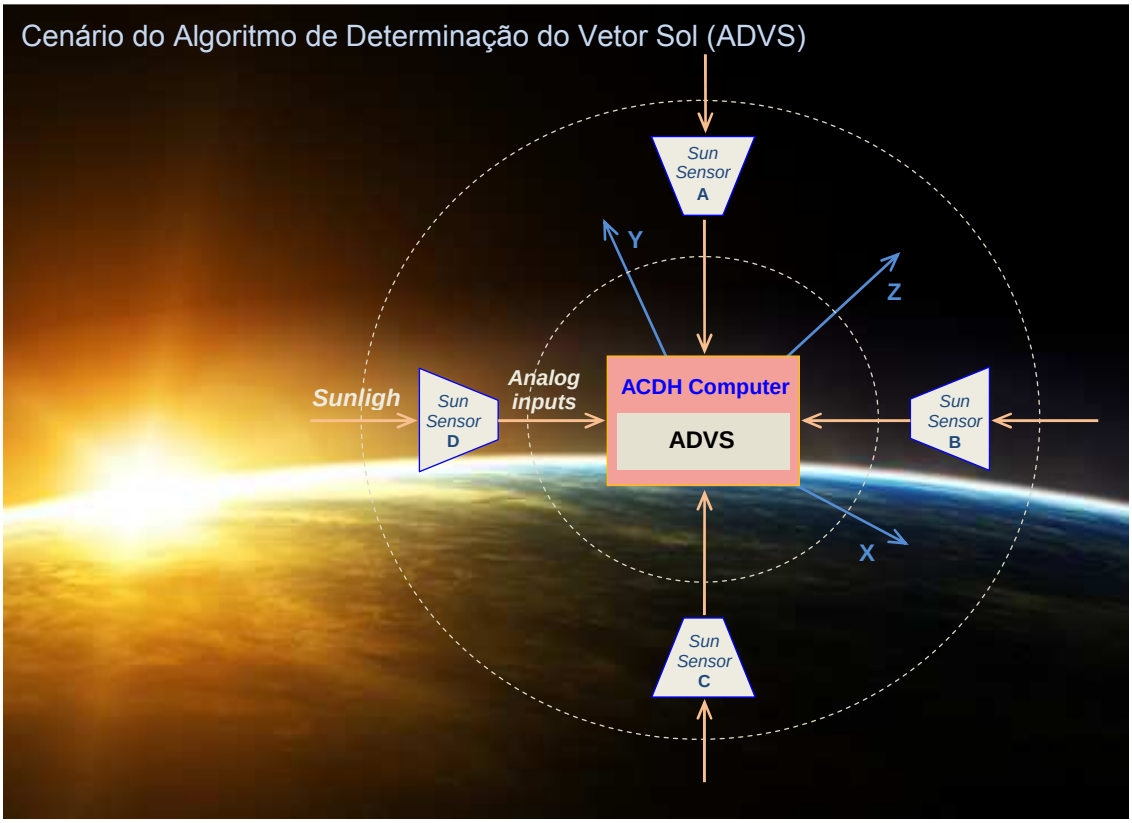

Figura 6.5 - Cenário do Algoritmo de Determinação do Vetor Sol (ADVS)

Em diversas situações é necessário conhecer a posição do sol para a realização de manobras específicas, como o posicionamento da plataforma de modo que os painéis solares sejam apontados em direção aos raios solares para geração de energia.

A execução do ADVS necessita do Computador de ACDH e dos quatro sensores solares, descritos na Seção 6.4. O presente cenário foi selecionado para a descrição de um procedimento de modo a facilitar o entendimento do uso da PLUTO.

Pode-se afirmar que o ADVS se adapta ao conceito definido pela ECSS em (ECSS, 2003), chamado de *Domain Specific View* (DSV). Uma DSV é composta por um subconjunto de funcionalidade do sistema, subsistema ou equipamento, necessário para determinada aplicação.

Esse conceito pode ser usado durante a modelagem individual de equipamentos e subsistemas, onde somente os dados de interesse estarão presentes no SSM. Mais detalhes sobre as características de uma DSV foram apresentadas na Seção 3.1.

6.5. Uso do Protótipo do STEPS no Estudo de Caso

Neste estudo de caso, um SSM foi modelado para representar o comportamento do ADVS e um procedimento foi descrito na sintaxe da PLUTO, para que juntos pudessem permitir a demonstração do funcionamento da arquitetura.

6.5.1. A Modelagem do SSM

O modelo SSM descreve o comportamento do ADVS, incluindo a estrutura de dados de origem (origem) e destino (saída) durante sua execução por meio do Computador de ACDH. Durante sua modelagem, um levantamento dos dados

relacionados foi realizado com auxílio da Tabela 3.1. Essa tabela foi utilizada para o registro dos dados do presente estudo de caso.

Entre os tipos de objetos pertencentes a esse modelo estão presentes os *System Elements* e *Report Data*. Objetos do tipo *Activitiy* e *Event* não foram definidos para o estudo de caso.

A Tabela A.1 do Apêndice A descreve as características dos dados do tipo *System Element*, pertencentes à decomposição funcional desse modelo. A mesma tabela informa também o tipo e a descrição de cada um deles.

A Tabela A.2 do Apêndice A apresenta as características de cada dado do tipo *Report Data*. Esses são os dados transmitidos dos sensores solares para o Computador de ACDH. Essa tabela também descreve suas características e quais SE's são seus objetos de referência.

As Tabelas A.1 e A.2 foram criadas a partir da Tabela 3.1, apresentada como recurso de apoio.

Os *Report Data* da Tabela A.2 com identificação RD1, RD2, RD3 e RD4 mantêm o registro dos valores gerados em cada sensor solar. Esses valores são enviados ao Computador de ACDH e usados como parâmetros de entrada na execução do ADVS. Os *Report Data* da Tabela A.2 com identificação RD5, RD6 e RD7 são os objetos que serão usados para registro dos parâmetros de saída do ADVS, que correspondem aos valores dos eixos X, Y e Z.

Esses dados foram gerados em formato XML, na mesma lógica hierárquica de um SSM e são apresentados no Apêndice A. As propriedades de cada nó mantêm registrados os dados de descrição e características de cada objeto. Essas propriedades foram definidas a partir dos tipos de dados e características que um objeto pode assumir.

Tais propriedades foram especificadas pelo padrão da ECSS para que durante a execução o *SSM Instance* possa distingui-los e processar seus dados de maneira adequada.

Um exemplo de propriedade é a chamada '*simulation_interface'*, que mantém registrado o parâmetro de chamada da interface que deverá ser configurada para comunicação do STEPS com equipamentos externos ao SUT. Outro exemplo de propriedade é a chamada '*type'*, que informa o tipo do objeto.

O formato XML armazena os dados estáticos da estrutura do modelo. O uso da XML facilita a leitura, o carregamento em memória e o rastreio dos objetos durante a execução. Os dados de objetos do tipo *Report Data e Event* estão sujeitos a atualizações em tempo de execução.

Para cada objeto do tipo *System Element*, *Report Data* e *Event*, existe uma variedade de tipos. De acordo com os dados da Tabela A.1, o objeto do tipo *System Element*, cujo nome é 'SUN_SENSOR_A' é um *System Element* do tipo '*sensor'*. Existem também outros tipos, como '*equipment'*, '*software module'* etc. Todos especificados pelo padrão ECSS-E-ST-70-31C.

6.5.2. A Descrição do Procedimento

A partir do procedimento PLUTO, descrito no Apêndice B, foi possível implementar rotinas computacionais no módulo *PLUTO Interpreter*. Durante a interpretação e execução de cada linha, o mecanismo de acesso aos objetos do modelo também foram implementados.

O comando *Simulate Environment* não encontra-se nas especificações da linguagem. Sua definição foi feita para acesso ao componente *External Interfaces* do *SSM Instance*. Em alguns casos, um procedimento requer comunicação com equipamentos que interagem com o ambiente externo. No presente trabalho, os equipamentos alvo são os sensores solares.

A interface de comunicação entre eles é definida no componente *External Interfaces* do módulo *SSM Instance*. Ao receber uma solicitação do tipo *Set(objectId, value)*, o *SSM Instance* identifica o respectivo objeto e inicializa a execução da *Activity* correspondente. Tal *Activity* tem acesso à interface definida no *External Interfaces* para comunicação com os sensores.

A solicitação *Set* que se encarrega de sinalizar ao modelo que a comunicação deve ser feita por meio do *External Interfaces* é gerada em função do comando *Simulate Environment* descrito no procedimento.

6.5.3. Preparação do Ambiente para Execução

O Computador de ACDH e os sensores não estão disponíveis atualmente para uso, por isso, a simulação desses elementos foi realizada por meio de software e são constituídas da seguinte maneira:

- **O Computador de ACDH:** para simulá-lo, uma rotina computacional foi criada para simular o ADVS embarcado em seu software de voo, o chamado ADVS *Simulator* (ADVSim); e

- **Os sensores solares:** para simular o funcionamento dos quatro sensores, uma rotina computacional foi criada, cujo nome é *Sun Sensor Simulator* (SSSim).

Esses simuladores foram criados para representar o funcionamento dos elementos que fazem parte do cenário de abrangência da utilização do STEPS sob o estudo de caso. Ambos mantêm a tarefa de envio e recepção de dados. Detalhes internos e específicos não foram implementados.

O SSSim foi criado para enviar dados, periodicamente, ao ADVSim, de modo a representar o mesmo comportamento dos sensores, cuja funcionalidade é o envio de dados ao Computador de ACDH. Esse simulador permite o envio dos quatro parâmetros de maneira individual ou em conjunto.

A execução desse simulador pode ser feita de forma manual ou automática, via procedimento PLUTO. No presente trabalho a execução está sendo feita via procedimento. O SSSim mantém uma interface gráfica para permitir o monitoramento dos dados enviados ao ADVSim. Já o ADVSim mantém em sua estrutura uma rotina que processa os quatro valores recebidos do SSSim e gera três valores de saída, os vetores X, Y e Z.

A Figura 6.6 ilustra o nível de abrangência da utilização do protótipo do STEPS no estudo de caso.

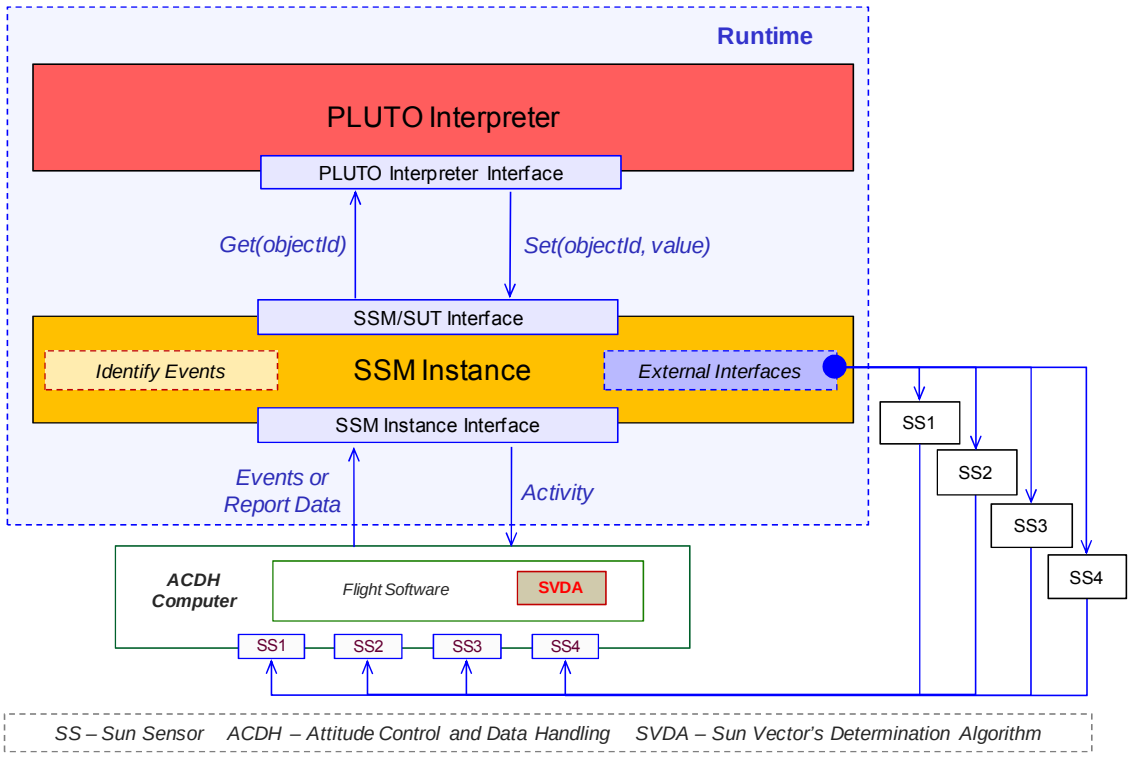

Figura 6.6 – Nível de abrangência com a utilização do protótipo do STEPS

O SSSim, ADVSim e STEPS são elementos independentes, no formato *executable* (*.exe), que se comunicam a partir de interfaces de comunicação separadas. Não apresentam dependência para representar o funcionamento real dos equipamentos do presente cenário.

A Figura 6.7 ilustra os elementos simulados e a Figura 6.8 ilustra a visão macro da utilização do protótipo desenvolvido para uso no estudo de caso.

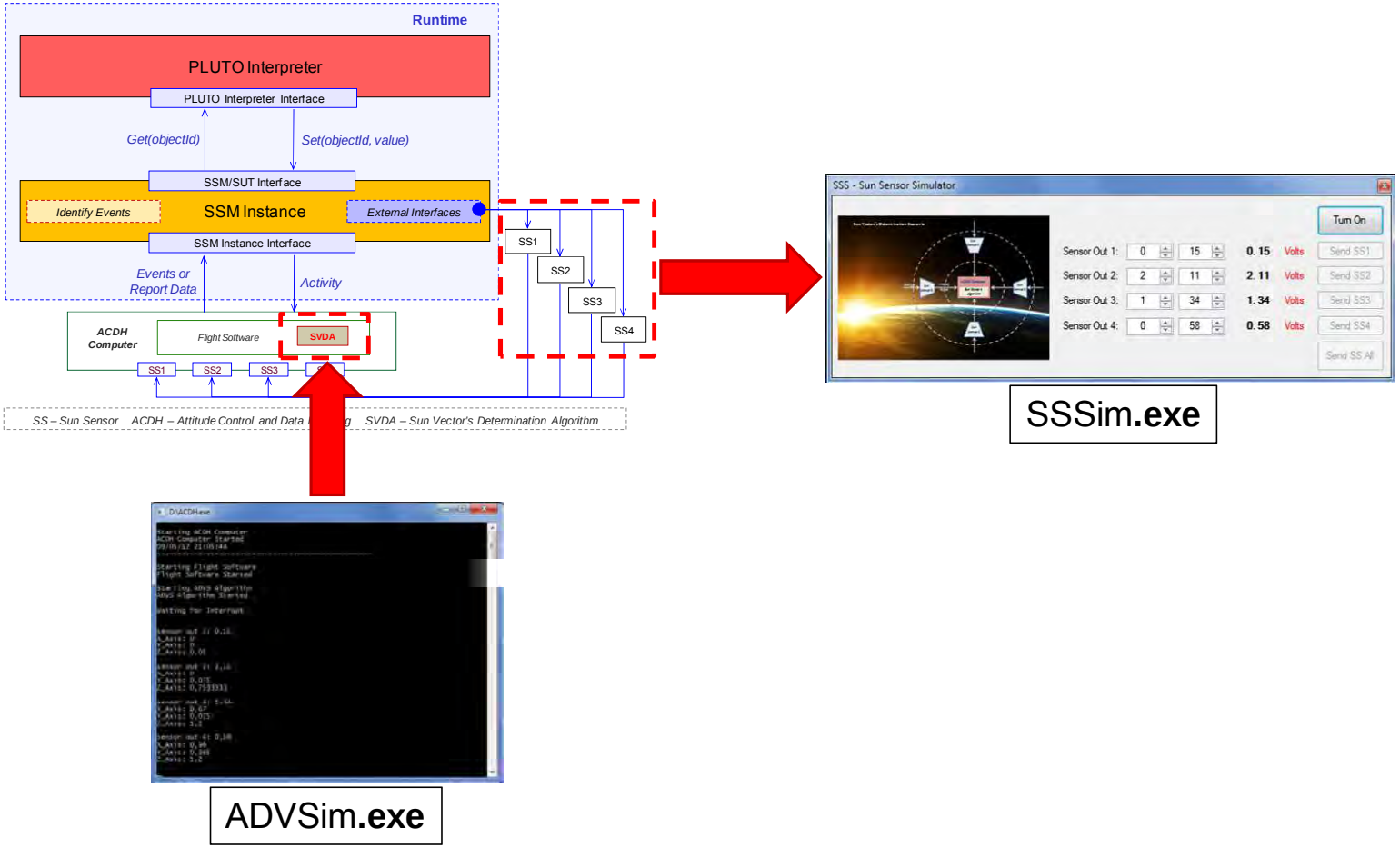

Figura 6.7 - Simulações realizadas para a utilização do protótipo do STEPS

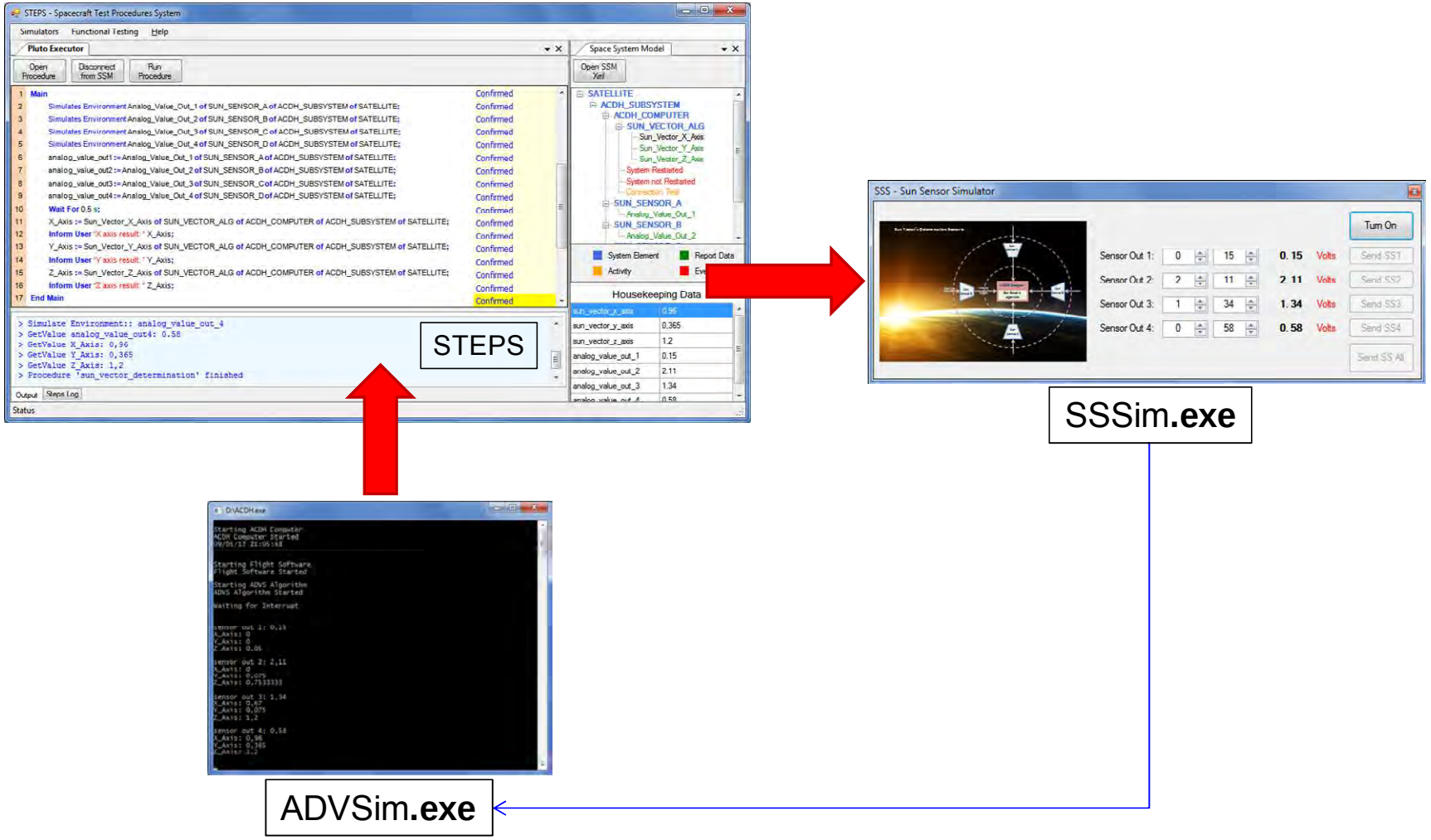

Figura 6.8 – Visualização macro da utilização do protótipo STEPS sob o estudo de caso

6.5.4. O Funcionamento da Arquitetura

Uma vez disponíveis o procedimento (*.pluto) e o modelo (*.ssm), o primeiro passo é o carregamento do modelo ao *SSM Instance* e em seguida o carregamento do procedimento. É necessário que o modelo já esteja instanciado em memória para dar início à execução.

Como descrito na Seção 6.1, a interpretação do procedimento é feita linha por linha. Para cada linha, o *PLUTO Interpreter* faz a verificação das palavras reservadas, objeto(s) referenciado(s), valor(es) e faz a montagem da respectiva solicitação, *Get* ou *Set*, para envio ao *SSM Instance*.

Ao receber a solicitação, o *SSM Instance* realiza uma busca recursiva para identificação do objeto de interesse e realiza a operação (escrita ou leitura). A busca é recursiva pelo fato da estrutura computacional, instanciada em memória pelo *SSM Instance*, manter os objetos em conformidade com o padrão SSM.

Para compreender com mais clareza o relacionamento entre PLUTO e SSM e demonstrar passo a passo como o *PLUTO Interpreter* se comunica com o modelo, um subconjunto de instruções do procedimento descrito para o estudo de caso, apresentado no Apêndice B, é ilustrado na Figura 6.9.

Junto desse subconjunto de instruções, encontra-se também a estrutura SSM, descrita em formato XML, correspondente a tais instruções. Duas instruções do procedimento estão sendo relacionadas com seus respectivos objetos para facilitar o entendimento.

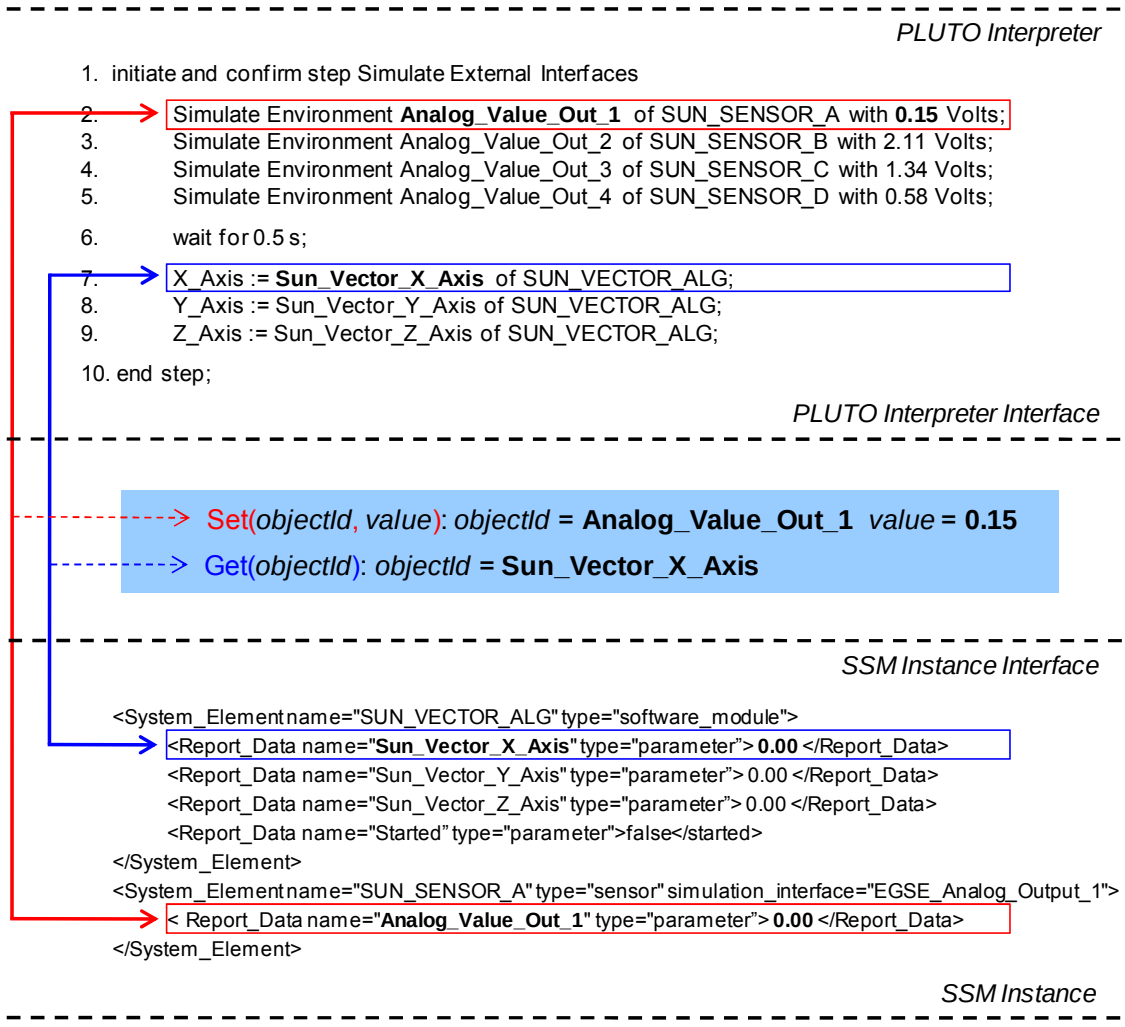

Figura 6.9 - Rastreamento entre os objetos SSM referenciados no procedimento e a estrutura SSM alocada em memória pelo SSM Instance.

Ao interpretar a instrução número 2 do procedimento, o *PLUTO Interpreter* identifica o comando *Simulate Environment* e se certifica que o mesmo é um comando que requer uma solicitação do tipo *Set*. Em seguida, faz a identificação do respectivo objeto, que nesse caso é o '*Analog_Value_Out_1*'. Essa instrução também conta com a palavra '*with*', que aponta para um valor real destinado para escrita no objeto '*Analog_Value_Out_1*'.

Ao realizar esses passos, o *PLUTO Interpreter* alimenta a solicitação *Set* da seguinte forma: '*Set*(Analog_Value_Out_1_id, 0.15)', e envia ao *SSM Instance*, conforme ilustra a Figura 6.9.

O objetivo do comando *Simulate Environment* é solicitar a comunicação com algum equipamento do ambiente externo. Para que o *SSM Instance* realize essa comunicação, a propriedade '*simulation_interface*' é a referência para a interface configurada dentro do componente *External Interfaces*, componente este apresentado na Seção 5.1.3.

A propriedade '*simulation_interface*' é opcional e deve ser atribuída em objetos do tipo *System Element*, que no presente caso da Figura 6.9 é o *System Element* que representa o sensor solar A, cujo nome atribuído é *'SUN_SENSOR_A'*. Essa propriedade é a referência para a interface configurada no componente *External Interfaces*. Nesse caso, todos os objetos declarados em *'SUN_SENSOR_A'* que receberem uma solicitação do tipo *Set*, deverão registrar o valor e enviá-lo ao componente *External Interfaces*, que por sua vez se comunicará com o respectivo equipamento do ambiente externo.

Ao interpretar a instrução número 7 do procedimento, o *PLUTO Interpreter* se certifica que a mesma é uma instrução de leitura. Nesse caso, a solicitação do tipo *Get* é utilizada. A operação é a mesma feita para a solicitação do tipo *Set*. A diferença encontra-se na passagem de parâmetro, onde a *Get* passa apenas a identificação do objeto, que nesse caso é o '*Sun_Vector_X_Axis*'. A solicitação *Get* fica da seguinte forma: '*Get*(Sun_Vector_X_Axis_id)'.

Ao ser recebida, o *SSM Instance* disponibiliza o valor que está registrado naquele momento no objeto '*Sun_Vector_X_Axis*'. Durante a execução, os objetos do SSM estão sujeitos a assumirem diferentes valores, pois podem registrar valores solicitados via procedimento ou recebidos do sistema, subsistema ou equipamento que está subordinado à respectiva execução.

No contexto da Figura 6.6 e Figura 6.7, ao relacionar o funcionamento da arquitetura com o caminho *(loop)* realizado com a execução do procedimento, temos que, a instrução de número 2, da Figura 6.9, solicita a simulação de um dos sensores solares. O interpretador envia essa solicitação para a instância do modelo que por sua vez envia o valor 0.15 ao simulador SSSim. Esse simulador, por sua vez, reconhece-o como um estímulo para envio do mesmo ao simulador do algoritmo, o ADVSim.

O ADVSim recebe o valor 0.15, gera os parâmetros dos eixos X, Y e Z e envia-os ao STEPS. O módulo *SSM Instance* do STEPS é o módulo que os recebe e registra-os em seus respectivos objetos. No presente exemplo da Figura 6.9, o objeto '*Sun_Vector_X_Axis*' equivalente à instrução de número 7, mantém o registro do parâmetro gerado para o eixo X.

7 CONCLUSÃO

O objetivo dessa dissertação de mestrado foi definir uma arquitetura com os recursos necessários para o desenvolvimento de um sistema computacional capaz de permitir o preparo e a execução automática de procedimentos de teste com a PLUTO.

O presente Capítulo apresenta quais foram os resultados atingidos, principais contribuições, trabalhos submetidos, publicados e novos trabalhos que podem ser realizados no âmbito de trabalhos futuros.

7.1. Principais Contribuições

A arquitetura proposta foi definida com características modulares e seus módulos se comunicam por meio de interfaces bem definidas. São módulos que permitem o preparo, execução, monitoramento e registro dos resultados provenientes da execução de procedimentos PLUTO.

A partir dessa arquitetura é possível:

1. Gerenciar a modelagem do SSM;
2. Gerenciar o preparo de procedimentos PLUTO;
3. Interpretar e executar procedimentos PLUTO com a presença de um modelo do sistema, subsistema ou equipamento sob teste ou operação;
4. Definir eventos customizados, como forma diferenciada para monitorar o comportamento do sistema;
5. Manter comunicação paralela com equipamentos do ambiente externo por meio dos procedimentos;
6. Monitorar o status da execução de procedimentos PLUTO;

7. Registrar os dados provenientes da execução dos procedimentos em um repositório de dados, para possibilitar a geração de conhecimento.

Como mencionado na Seção 1.1, em motivação, para adotar a PLUTO como padrão de linguagem, é necessário que se tenha uma estrutura bem definida, capaz de manter: um interpretador da linguagem; um gerenciador do modelo em execução; ferramentas computacionais para apoio ao preparo e execução; e interfaces de comunicação bem definidas.

A presente arquitetura cobre todas essas necessidades, e podem ser rastreadas na arquitetura ilustrada na Figura 5.1, da seguinte forma:

- **Interpretador:** corresponde ao módulo *PLUTO Interpreter*;
- **Gerenciador do modelo:** corresponde ao módulo *SSM Instance*;
- **Sistemas computacionais de apoio:** correspondem aos módulos *PLUTO Manager* e *SSM Manager*;
- **Interfaces de comunicação entre os elementos:** correspondem às interfaces *PLUTO Interpreter Interface*, *SSM Instance Interface, SSM/SUT Interface e External Interfaces*.

Essa arquitetura foi definida para procedimentos de teste, mas pode ser facilmente adaptada para uso em operações, pelo fato da PLUTO poder ser utilizada em atividades tanto de teste quanto operação.

Um protótipo, o chamado *Spacecraft Test Procedures System* (STEPS), foi desenvolvido para exercitar a arquitetura. Foi possível analisar como um procedimento PLUTO deve ser descrito e demonstrar o comportamento dos módulos dessa arquitetura durante seu funcionamento. Essa etapa também conta com a modelagem do *Space System Model* (SSM), o modelo necessário para uso da linguagem.

O protótipo do STEPS foi utilizado de maneira experimental, a partir do estudo de caso, definido sob um computador de bordo com as atribuições *Attitude Control and Data Handling* (ACDH), chamado Computador de ACDH. A partir desse estudo de caso foi possível modelar o SSM e descrever um procedimento na sintaxe da linguagem, para então implementar os módulos da arquitetura responsáveis pela execução.

Os módulos que foram implementados no protótipo foram: *PLUTO Interpreter*; *SSM Instance*; *PLUTO Manager* e *SSM Manager*, para leitura e exibição dos dados do procedimento e do modelo durante a execução; as interfaces entre tais módulos; e o componente *External Interfaces*.

Uma instrução chamada *Simulate Environment*, foi definida para interação com os equipamentos que interagem com o ambiente externo. Para isso o componente *External Interfaces* foi definido para permitir flexibilidade na modelagem e definição das interfaces de comunicação com tais equipamentos. Esse componente faz parte do módulo que gerencia a instância do modelo, o *SSM Instance*. Até o presente momento, não existem referências que descrevam a utilização da PLUTO para esse tipo de aplicação.

No Capítulo 4 foi apresentado um panorama da estrutura de preparo e execução dos procedimentos da ESA, junto aos seus mecanismos e ferramentas computacionais de apoio. Isso permitiu apresentar como a estrutura da ESA encontra-se atualmente e como a Agência está trabalhando para que tal estrutura torne-se um sistema único, para uso em todas as suas futuras missões.

A partir da estrutura da ESA, uma correlação de equivalência foi realizada entre as características da arquitetura do STEPS e as características de um sistema presente em tal estrutura, o chamado ASE, descrito na Seção 4.3.2. O objetivo desse sistema é apoiar a automatização dos procedimentos de atividades de teste da ESA.

Essa correlação pode ser conferida na ilustração da Figura 5.7 e por meio da Tabela 5.1. Ambos, tanto o STEPS quanto o ASE se aplicam aos mesmos objetivos, porém, a arquitetura do STEPS apresenta alguns diferenciais. Dentre eles, encontram-se o componente *External Interfaces*, o comando *Simulate Environment* e o *Identify Events*. A característica que se destaca na arquitetura do ASE é a presença do elemento *Logger*, mas com um maior nível de detalhamento, alcançado a partir de trabalhos futuros, permitirá incluir recurso semelhante na arquitetura do STEPS.

Outra contribuição encontra-se na pesquisa realizada, também no âmbito do 'estado da arte', para o levantamento das linguagens mais utilizadas na área espacial. Nessa etapa foi possível identificar, na Figura 2.2, quais as fases e atividades de uma missão espacial permitem o uso de alguma linguagem e apresentar dados relativos à disponibilidade e utilização das linguagens apresentadas na Tabela 2.1.

7.1.1. Trabalhos Submetidos e Publicados

Dentre os trabalhos submetidos e publicados encontra-se o (PEREIRA; KUCINSKIS; FERREIRA, 2011), publicado com apresentação oral no II Workshop em Engenharia e Tecnologias Espaciais. Esse artigo é resultado dos primeiros passos, necessários para o entendimento das características da linguagem e amadurecimento das idéias referentes aos recursos necessários para o estabelecimento da arquitetura.

O artigo (PEREIRA, 2011), submetido e publicado foi apresentado de forma oral no *Fifth Latin-American Symposium on Dependable Computing* (LADC). Esse artigo foi submetido e publicado na sessão de estudantes e pode ser gerado durante o período de cumprimento dos créditos, por meio da disciplina chamada Verificação e Validação de Sistemas Espaciais.

Com a implementação do protótipo do STEPS e a aplicação do mesmo sob um estudo de caso, o artigo (PEREIRA; FERREIRA; KUCINSKIS, 2012) foi

publicado no *12th International Conference on Space Operations* (Spaceops 2012) com apresentação via pôster.

O período da disciplina Introdução à Engenharia de Sistemas Espaciais, teve como resultado uma publicação, o *paper* (PEREIRA; REIS; LOUREIRO, 2010) na revista *Product: Management & Development* do Instituto de Gestão de Desenvolvimento do Produto da Universidade Federal de São Carlos.

7.2. Trabalhos Futuros

Entre as sugestões de trabalhos futuros, com o objetivo de dar continuidade ao desenvolvimento da arquitetura do STEPS, destacam-se:

- Especificar com maior nível de detalhe o interpretador para a PLUTO de modo que seja usado tanto em procedimentos de teste quanto de operação. Isso inclui a especificação de um mecanismo de gerenciamento da execução de procedimentos agendados.

  Em paralelo, desenvolver uma *Integrated Development Environment* (IDE), para apoio ao preparo e acompanhamento do status da execução, de modo que seja integrada aos requisitos de teste e que a validação dos procedimentos seja feita automaticamente. Essa IDE pode ser desenvolvida para que os procedimentos sejam criados graficamente, como o exemplo do *Flowcharter*, apresentado na Figura 4.5. O trabalho de (SCHWAB; EILENBERGER; BORG, 2012) pode ser usado como referência para a validação dos procedimentos.

- Desenvolver uma IDE que permita a modelagem da estrutura lógica e definição dos dados de um SSM, em conformidade com o padrão ECSS-E-ST-70-31C, especificado em (ECSS, 2003). Estudar a possibilidade da geração da estrutura do SSM a partir de um modelo UML. Incluir no modelo UML informações que permitam a geração do SSM no mesmo formato XML apresentado na Figura 5.3 e no Apêndice A.3. Isso

possibilitará também, definir em qual etapa do processo de elaboração e execução de um Plano de Teste o uso do STEPS se enquadra.

O Apêndice C apresenta um protótipo desenvolvido para o gerenciamento da criação da estrutura lógica de modelos SSM. Esse protótipo foi desenvolvido durante o cumprimento dos créditos, na disciplina Processos de Desenvolvimento de Software.

- Definir um módulo, junto aos módulos de preparo dos procedimentos, que tenha ligação direta com o *SSM Instance*, para permitir que o usuário defina graficamente as interfaces de comunicação com equipamentos do ambiente externo via *External Interfaces*.

- Segue também a sugestão para o desenvolvimento de um banco de dados único para centralizar e estruturar os dados das missões espaciais do INPE. Esse banco de dados pode ser modelado em conformidade com o SSM. Os dados que são gerados desde o início da concepção de projeto de um satélite podem ser armazenados nessa base de dados. O objetivo é integrar, de maneira bem estruturada, os dados gerados em todas as etapas de desenvolvimento da missão.

7.3. Considerações Finais

Algumas tecnologias que hoje são utilizadas na integração e testes dos equipamentos e subsistemas dos satélites do INPE podem estar chegando ao final do seu ciclo de vida. Por isso, existe uma necessidade voltada à atualização dessas tecnologias, como é o caso do sistema de execução de procedimentos descritos com a ETOL (MELTON; HUBSCHER, *et al*., 1996), utilizado atualmente pelo LIT.

Para tanto, a inviabilidade da atualização de ferramentas existentes encontra-se no fato de que foram desenvolvidas já há alguns anos, ou seja, podem ser tecnologias que já estejam obsoletas.

A atualização de um recurso já existente pode ter um custo-benefício não tão satisfatório, visto que novas tecnologias e soluções podem ser desenvolvidas com propósito de sempre melhorar a maneira com que as atividades espaciais são executadas. Isso viabiliza iniciativas para o desenvolvimento de novas tecnologias.

A PLUTO é a linguagem de interesse, pelo fato do INPE já utilizar os padrões da ECSS no desenvolvimento de suas atividades espaciais. O uso da PLUTO em suas atividades permitirá que o Instituto continue mantendo suas missões alinhadas aos padrões vigentes da área espacial, principalmente aos padrões da ECSS.

A utilização da PLUTO como padrão para elaboração e aplicação de procedimentos de testes e de operação de satélites do INPE pode proporcionar os seguintes benefícios:

1. Permitir que o INPE continue mantendo suas missões alinhadas aos padrões vigentes da área espacial, em especial aos padrões da ECSS.

2. Desenvolver seu sistema de gerenciamento de procedimentos e uma série de ferramentas computacionais para auxílio às atividades que são executadas por meio de procedimentos.

3. Reutilização de procedimentos de teste em diferentes níveis (sistema, subsistema ou equipamento) e fases do processo de desenvolvimento. O processo de desenvolvimento, que se inicia desde um equipamento até a AIT do satélite como um todo, poderá contar com o apoio do mesmo padrão de linguagem em todas as etapas;

4. Aumentar o intercâmbio de informações entre as atividades de teste e operação. A utilização da PLUTO também permite a construção de modelos do sistema por meio do SSM e isso viabiliza o armazenamento de históricos de dados de maneira organizada, conforme as

características da estrutura de um SSM. Isso pode facilitar a geração de conhecimento sobre o funcionamento do sistema que pode ser utilizado em outros projetos ou até mesmo em futuras missões;

Dentre as características da PLUTO, uma que se destaca é sua sintaxe, cuja semelhança está próxima da linguagem natural.

Não existe interpretador disponível para a linguagem. A ESA, por exemplo, está trabalhando com a PLUTO há alguns anos e ainda não possui um interpretador suficientemente capaz de cobrir todo o potencial da linguagem. O que explica isso é o fato da ESA já possuir uma estrutura bem definida e ainda utilizá-la apenas para fins experimentais.

Tendo em vista que o INPE atualmente não adota nenhuma linguagem para descrição de procedimentos em todas as fases do desenvolvimento e operação, é recomendável que o INPE estabeleça um padrão de linguagem que possa ser utilizado em todas as fases de suas missões.

A PLUTO é uma linguagem especificada a partir da experiência adquirida no decorrer do desenvolvimento de missões espaciais nas últimas décadas, principalmente pela ESA e indústria européia.

REFERÊNCIAS BIBLIOGRÁFICAS

ADAMSON, K. et al. ADM-AEOLUS mission planning re-use, autonomy and automation. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2010, Huntsville, AL, USA. **Proceedings…** Huntsville: AIAA, 2010. DOI: 10.2514/6.2010-1968 eISBN: 978-1-62410-164-9.

AHLGREN, N. et al. PRISMA mission extension: adapting mission operations to new and changing mission objectives. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

BEZICK, S. M.; PUE, A. J.; PATZELT, C. M. Inertial navigation for guided missile systems. **Johns Hopkins APL Technical Digest**, Washington, USA, v. 28, n. 4, p. 331-342, 2010. ISSN: 0270-5214

CENTRO DE GESTÃO E ESTUDOS ESTRATÉGICOS (CGEE). **Tecnologia inercial no Brasil 2007-2010** : a rota para seu estabelecimento na indústria. Brasília, Ministério de Ciência e Tecnologia, DF, 104 p., 2006.

CENTRO DE GESTÃO E ESTUDOS ESTRATÉGICOS (CGEE). **Materiais avançados no Brasil 2010-2022**. Brasília, Ministério da Ciência, Tecnologia e Inovação, DF, 361 p., 2010. ISBN: 978-85-60755-25-7.

CONNORS, E. M. et al. Static verification of spacecraft procedures. In: AIAA INFOTECH@AEROSPACE CONFERENCE. 2009, Washington, USA. **Proceedings…** Washington: AIAA, 2009. DOI: 10.2514/6.2009-2033 eISBN: 978-1-60086-979-2.

CROCE, F.; SIMONIC, A. ECSS E-70-32 test platform features and applicability area. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 10., 2008, Heidelberg, Alemanha. **Proceedings…** Heidelberg: AIAA, 2008. DOI: 10.2514/6.2008-3417 eISBN: 978-1-62410-167-0.

EUROPEAN COMMITTEE FOR SPACE STANDARDIZATION (ECSS). **Ground systems and operations** - monitoring and control data definition. Noordwijk, Holanda: ECSS, 2003. (ECSS-E-ST-70-31C)

EUROPEAN COMMITTEE FOR SPACE STANDARDIZATION (ECSS). **Test and operations procedure language**. Noordwijk, Holanda: ECSS, 2008. (ECSS-E-ST-70-32C)

EUROPEAN COMMITTEE FOR SPACE STANDARDIZATION (ECSS). **Space system data repository**. Noordwijk, Holanda: ECSS, 2011. (ECSS-E-TM-10-23A)

EUROPEAN COMMITTEE FOR SPACE STANDARDIZATION (ECSS). **Space project management** - project planning and implementation. Noordwijk, Holanda: ECSS, 2009. (ECSS-M-ST-10C)

EICKHOFF, J. **Onboard computers, onboard software and satellite operations:** an introduction. 1. ed., London, New York: Editora Springer Series in Aerospace Technology, 2012. 282 p. ISSN 1869-1730 DOI: 10.1007/978-3-642-25170-2.

FRITZ, M. et al. Low cost control and simulation environment for the "flying laptop", a university microsatellite. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 11., 2010, Huntsville, Alabama USA. **Proceedings…** Huntsville: AIAA, 2010. DOI: 10.2514/6.2010-2294 eISBN: 978-1-62410-164-9.

HAUCK, T. F.; FINNIGAN, J. V. Use of the ground support equipment operating system (GSEOS) software on the messenger mission - a case study. In: INTERNATIONAL SYMPOSIUM ON REDUCING THE COST OF SPACECRAFT GROUND SYSTEMS AND OPERATIONS (RCSGSO), 5., 2003, Pasadena, CA, USA. **Proceedings…** Pasadena, 2003.

HEINEN, W.; REID, S.; PEARSON, S. Mission operations preparation environment: a new approach for the future. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Plano diretor 2011-2012**. São José dos Campos, SP, 57 p., 2011.

KARLSSON, T. et al. PRISMA mission control: transferring satellite control between organisations. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

KOLLER, M. ESOC earth observation missions and the automation of operational routine tasks. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 11., 2010, Huntsville, Alabama USA. **Proceedings…** Huntsville: AIAA, 2010. DOI: 10.2514/6.2010-2308 eISBN: 978-1-62410-164-9.

LANNES, C. et al. A new generation of monitoring and control system for ESTRACK. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

LARSON, W. J.; WERTZ, J. R. **Space mission analysis and design**. 3. ed. Torrance, CA, USA: Microcosm, Inc. and Kluwer Academic Publishers, 1999. 969 p. ISBN-13 978-1881883104.

LORETUCCI, A. et al. Mission automation infrastructure tools at ESOC. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 10., 2008, Heidelberg, Alemanha. **Proceedings…** Heidelberg: AIAA, 2008. DOI: 10.2514/6.2008-3340 eISBN: 978-1-62410-167-0

MAK, R. **Writing compilers and interpreters**: a modern software engineering approach. 3ª. ed. Indianápolis, Indiana, EUA - Published simultaneously in Canada: Wiley Publishing, Inc., 2009. Disponível em: <http://www.apropos-logic.com/wci/>. Acesso em 06 set 2012.

MELTON, B. et al. ESA's ETOL software in international markets. **ESA Publications Bulletin**, v. 85, p. 74-78, 1996.

MIMS, T. L. **Use of spacecraft command language for advanced command and control applications**. Orlando, Florida, USA, 2008. 5 p. NASA Technical Reports Server (NTRS).

MORELLI, G.; BOULEAU, F. Unleashing the full power of today's technologies for flight procedures automation. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 11., 2010, Huntsville, Alabama USA. **Proceedings…** Huntsville: AIAA, 2010. DOI: 10.2514/6.2010-2286 eISBN: 978-1-62410-164-9

MUSLINER, D. J.; PELICAN, M. J. S.; SCHLETTE, P. J. Verifying equivalence of procedures in different languages: preliminary results. In: INTERNATIONAL CONFERENCE ON AUTOMATED PLANNING AND SCHEDULING (ICAPS), 19., 2009, Thessaloniki, Grécia. **Proceedings…** Thessaloniki: AAAI, 2009.

OUSTERHOUT, J. K. **Tcl and the Tk toolkit.** 1. ed. Berkeley, CA: Addison-Wesley Publishing Company, Inc, 1993. v. I, 134 p. ISBN: 0-201-63337-X.X

PEARSON, S. et al. An integrated development and validation environment for operations automation. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

PEREIRA, T. D. An architecture for test procedures execution applied to a satellite's on-board data handling computer (OBDH). In: LATIN-AMERICAN SYMPOSYUM ON DEPENDABLE COMPUTING (LADC), 5., 2011, São José dos Campos, SP, Brasil. **Proceedings…** São José dos Campos, Brasil: IEEE, 2011. 1 CD.

PEREIRA, T. D.; FERREIRA, M. G. V.; KUCINSKIS, F. D. N. Using the PLUTO language on functional tests of a brazilian satellite's on-board data handling computer. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

PEREIRA, T. D.; KUCINSKIS, F. D. N.; FERREIRA, M. G. V. Padronização dos procedimentos de teste para computadores de supervisão de bordo de satélites. In: WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS (WETE), 2., 2011, São José dos Campos, SP, Brasil. **Anais...** São José dos Campos: INPE, 2011. DVD. ISSN: 2236-2606. Disponível em: http://urlib.net/J8LNKAN8RW/39RQ9JL. Acesso em: 03 jul. 2012.

PEREIRA, T. D.; REIS, M. F. S.; LOUREIRO, G. The use of system concurrent engineering to develop a configuration system for automobile interior. **Product: Management & Development**, São Carlos, v. 8, n. 2, p. 155-165, 2 Dezembro 2010. ISSN 1676-4056.

REID, S. et al. MOIS student edition and the flying laptop. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

RHEA, S. S. A. Manufacturing and operations information system. **Rhea Group**, 2012. Disponivel em: <http://www.rheagroup.com/en/x/28/mois>. Acesso em 16 Set 2012.

SCHWAB, A.; EILENBERGER, R.; BORG, W. Z. OBCPs - an integrated part of the BepiColombo autonomy and flexibility. In: INTERNATIONAL CONFERENCE ON SPACE

OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

SEYMOUR, M. The PLUTO operations procedure language and its use for RADARSAT-2 mission operations. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 8., 2004, 17-21 May, Montréal, Canada. **Proceedings...** Montreal, 2004.

WALSH, A. et al. The european ground systems common core (EGS-CC) initiative. In: INTERNATIONAL CONFERENCE ON SPACE OPERATIONS (SPACEOPS), 12., 2012, Stockholm, Suécia. **Proceedings…** Stockholm: AIAA, 2012.

WANG, G. et al. Design and performance test of spacecraft test and operation software. **Acta Astronautica**, v. 68, p. 1774-1781, Beijing, 6 February 2011. DOI: 10.1016/j.actaastro.2011.02.002 ISSN: 0094-5765.

GLOSSÁRIO

Atividades de Operação – são as atividades de controle e monitoramento do satélite, realizadas a partir da terra durante sua operação.

Atividades de Teste – são as atividades realizadas com o objetivo de executar o sistema, ou o subsistema, ou o equipamento, para encontrar possíveis falhas e exercitar seu comportamento operacional.

Atividades Espaciais – termo que se refere às atividades realizadas durante todo o ciclo de vida de uma missão espacial.

Electrical Ground Support Equipment **(EGSE)** – equipamento composto de hardware, software e interfaces de comunicação com o sistema espacial, cujo objetivo é a realização de testes funcionais, elétricos e de desempenho durante seu desenvolvimento.

Housekeeping – são dados que refletem o status operacional do sistema espacial, quando em operação, para uso durante as atividades de operação.

Interpretador – sistema computacional criado para converte em código executável um código-fonte descrito na sintaxe de uma linguagem interpretada.

Linguagem de Procedimentos – linguagem utilizada para especificar, de forma textual (semelhante a um *script*) os *steps* de um procedimento de teste ou de operação. Uma linguagem utilizada em procedimentos de teste ou de operação é equivalente a uma linguagem de *scritps*.

Parser – o *parser* verifica se o código-fonte descrito por meio de alguma linguagem está sintaticamente correto, em outras palavras, o *parser* verifica cada *token* gerado durante a leitura do código-fonte. O *Parser* sinaliza erros de sintaxe.

Procedimento – são meios de interagir com o sistema espacial, em nível de sistema, de subsistema e de equipamento, com o objetivo de se atingir um objetivo ou um conjunto de objetivos. Na prática, entende-se como uma sequência lógica de *steps* a serem executados durante a realização dos testes ou da operação.

Reader – o *Reader* faz a leitura do código-fonte, caractere por caractere, e realiza a construção dos *tokens* para que em seguida sejam usados pelo *Parser*.

Roda de Reação – roda instalada com o eixo fixo na estrutura da plataforma orbital, projetada para uso nas manobras do satélite. As rodas de reação controlam a direção (posicionamento angular) do satélite.

Sensor de Estrela – adquire dados do espaço para a identificação de pontos (identificação de estrelas) para uso como referência posicional, durante o controle de atitude do satélite.

Sensor Solar – adquire dados referentes à intensidade dos raios solares refletidos na plataforma do satélite. Usado para a identificação direcional do sol, para a realização de uma possível manobra no espaço, como por exemplo: posicionar o painel solar em direção aos raios solares para a geração de energia.

Tecnologias *ad-hoc* – são tecnologias desenvolvidas para uso em casos específicos.

Telecomando – é um pacote de dados gerado em solo para ser executado remotamente. O mesmo é transferido por meio de um sistema de telecomunicação para um dado equipamento embarcado no satélite.

Telemetria – é um pacote de dados gerado pelo satélite e enviado para solo. Uma telemetria consiste em medidas de sensores, dados de *housekeeping* e

dados da missão espacial. As telemetrias são enviadas para o solo por meio de um sistema de telecomunicação.

Token – os *tokens* são construídos durante a leitura do código-fonte pelo *Reader*. Na sintaxe da PLUTO, alguns exemplos de *tokens* são: *preconditions*; *main*; *initiate*; *confirm*; *step*; *end*; *if-then-else*, *while*; etc. Os *tokens* são as palavras reservadas da linguagem.

APÊNDICE A – O SPACE SYSTEM MODEL (SSM) CRIADO PARA O ESTUDO DE CASO

A.1 Os System Elements (SE)

Tabela A.1 – System Elements (SE) identificados para composição da estrutura do SSM

Id. Obj.	Nome	Tipo do SE	Descrição	Ref. Obj.
SE1	ACDH SUBSYSTEM	system	Subsistema de Controle de Atitude e Supervisão de Bordo.	nenhum
SE2	ACDH COMPUTER	equipment	Computador de bordo mantido como principal equipamento do Subsistema de Controle de Atitude e Supervisão de Bordo.	SE1
SE3	SUN VECTOR ALGORITHM	software module	Algoritmo de determinação dos vetores X, Y e Z para uso no controle de atitude.	SE2
SE4	SUN SENSOR A	sensor	Sensor de recepção solar instalado na posição 1	SE2
SE5	SUN SENSOR B	sensor	Sensor de recepção solar instalado na posição 2	SE2
SE6	SUN SENSOR C	sensor	Sensor de recepção solar instalado na posição 3	SE2
SE7	SUN SENSOR D	sensor	Sensor de recepção solar instalado na posição 4	SE2

A.2 Os Report Data (RD)

Tabela A.2 – Report Data (RD) identificados para composição do SSM

ID	Nome	Tipo do RD	Descrição	Ref. Obj.
RD1	Analog Value Output 1	parameter	Valor da intensidade de luz solar refletida na posição 1 da plataforma, adquirida pelo SUN SENSOR A.	SE4
RD 2	Analog Value Output 2	parameter	Valor da intensidade de luz solar refletida na posição 2 da plataforma, adquirida pelo SUN SENSOR B.	SE5
RD 3	Analog Value Output 3	parameter	Valor da intensidade de luz solar refletida na posição 3 da plataforma, adquirida pelo SUN SENSOR C.	SE6
RD 4	Analog Value Output 4	parameter	Valor da intensidade de luz solar refletida na posição 4 da plataforma, adquirida pelo SUN SENSOR D.	SE7
RD5	Sun Vector X Axis	parameter	Parâmetro de saída gerado pelo ADVS. Fornece o valor de posicionamento do eixo X.	SE3
RD6	Sun Vector Y Axis	parameter	Parâmetro de saída gerado pelo ADVS. Fornece o valor de posicionamento do eixo Y.	SE3
RD7	Sun Vector Z Axis	parameter	Parâmetro de saída gerado pelo ADVS. Fornece o valor de posicionamento do eixo Z.	SE3

A.3 A estrutura do modelo gerada em formato XML

```
<?xml version="0.1"?>

<System_Element
      id="01"
      name="SATELLITE"
      object_type="system_element"
      type="system">

      <System_Element
            id="02"
            name="ACDH_SUBSYSTEM"
            object_type="system_element"
            type="subsystem">

            <System_Element
                  id="03"
                  name="ACDH_COMPUTER"
                  object_type="system_element"
                  type="equipment">

                  <System_Element
                        id="08"
                        name="SUN_VECTOR_ALG"
                        object_type="system_element"
                        type="software_module">

                        <Report_Data
                              id="09"
```

```
            name="Sun_Vector_X_Axis"
            object_type="report_data"
            type="parameter"/>

      <Report_Data
            id="10"
            name="Sun_Vector_Y_Axis"
            object_type="report_data"
            type="parameter"/>

      <Report_Data
            id="11"
            name="Sun_Vector_Z_Axis"
            object_type="report_data"
            type="parameter"/>

      <Report_Data
            Id=”29”
            name="Started"
            type="parameter">
            false
      </Report_Data>

</System_Element>

<Event
      Id=”28”
      name= "Switch_On">
      false
</Event>
```

```
</System_Element>

<System_Element
      id="04"
      name="SUN_SENSOR_A"
      object_type="system_element"
      type="sensor"
      simulation_interface="EGSE_Analog_Output_1">

      <Report_Data
            id="12"
            name="Analog_Value_Out_1"
            object_type="report_data"
            type="parameter">
            0.00
      </Report_Data>

</System_Element>

<System_Element
      id="05"
      name="SUN_SENSOR_B"
      object_type="system_element"
      type="sensor"
      simulation_interface="EGSE_Analog_Output_2">

      <Report_Data
            id="13"
            name="Analog_Value_Out_2"
            object_type="report_data"
            type="parameter">
```

```
            0.00
        </Report_Data>

</System_Element>


<System_Element
        id="06"
        name="SUN_SENSOR_C"
        object_type="system_element"
        type="sensor"
        simulation_interface="EGSE_Analog_Output_3">

        <Report_Data
            id="14"
            name="Analog_Value_Out_3"
            object_type="report_data"
            type="parameter">
            0.00
        </Report_Data>

</System_Element>

<System_Element
        id="07"
        name="SUN_SENSOR_D"
        object_type="system_element"
        type="sensor"
        simulation_interface="EGSE_Analog_Output_4">

        <Report_Data
```

```
                    id="15"
                    name="Analog_Value_Out_4"
                    object_type="report_data"
                    type="parameter">
                    0.00
                </Report_Data>

            </System_Element>

        </System_Element>

</System_Element>
```

APÊNDICE B – O PROCEDIMENTO PLUTO DESCRITO PARA O ESTUDO DE CASO

B.1 Procedimento descrito na sintaxe da PLUTO

```
procedure 'Sun Vector's Determination Algorithm Monitoring'

declare
     variable real X_Axis := 0.0 deg;
     variable real Y_Axis := 0.0 deg;
     variable real Z_Axis := 0.0 deg;
end declare

preconditions
     wait until Battery1_Capability <= 40 %;
end preconditions

main
     in the context of ACDH_SUBSYSTEM do

     // inicialize Sun's Vector Determination Algorithm (SVDA)
     initiate and confirm step Start_SVDA

          declare
               variable boolean initialized := false;
          end declare;

          main
               Initiate SUN_VECTOR_ALG;

               initialized := Started of SUN_VECTOR_ALG of
                              ACDH_COMPUTER of ACDH_SUBSYSTEM;
```

```
            if (initialized) then
                  inform user "SVDA Initialized";
            else
                  inform user "SVDA not Initialized";
            end if;

      end main;

end step;

in case
      confirmed : continue;
      not confirmed : terminate;
end case;


// initialize the Sun Sensor A
initiate and confirm Step Switch_On_SensorA

in case
      confirmed : continue;
not confirmed : terminate;
end case;

// initialize the Sun Sensor B
initiate and confirm Step Switch_On_SensorB

in case
      confirmed : continue;
      not confirmed : terminate;
```

```
end case;

// initialize the Sun Sensor C
initiate and confirm Step Switch_On_SensorC

in case
     confirmed : continue;
     not confirmed : terminate;
end case;

// initialize the Sun Sensor D
initiate and confirm Step Switch_On_SensorD

in case
     confirmed : continue;
     not confirmed : terminate;
end case;

// Start the step that communicates with sensors
initiate step Simulate_External_Interfaces


     Simulate Environment Analog_Value_Out_1 of
                                        SUN_SENSOR_A with
                                        0.15 Volts;
     Simulate Environment Analog_Value_Out_2 of
                                        SUN_SENSOR_B with
                                        2.11 Volts;
     Simulate Environment Analog_Value_Out_3 of
                                        SUN_SENSOR_C with
                                        1.34 Volts;
     Simulate Environment Analog_Value_Out_4 of
                                        SUN_SENSOR_D with
                                        0.58 Volts;
end step;
```

```
      initiate step Verify_Sun_Vector_Out

            wait for 0.200 s;

            // obtains the SVDA values out from SSM
            X_Axis := Sun_Vector_X_Axis of SUN_VECTOR_ALG;
            Y_Axis := Sun_Vector_Y_Axis of SUN_VECTOR_ALG;
            Z_Axis := Sun_Vector_Z_Axis of SUN_VECTOR_ALG;

            inform user "X Axis" X_Axis;
            inform user "Y Axis" Y_Axis;
            inform user "Z Axis" Z_Axis;

      end step;

      in case
            confirmed : continue;
            not confirmed : terminate;
      end case;

      end context;

end main;
end procedure.
```

APÊNDICE C – UM PROTÓTIPO PARA O GERENCIAMENTO DA CRIAÇÃO DA ESTRUTURA DE MODELOS SSM

C.1 Diagramas Elaborados por meio da Unified Modeling Language (UML)

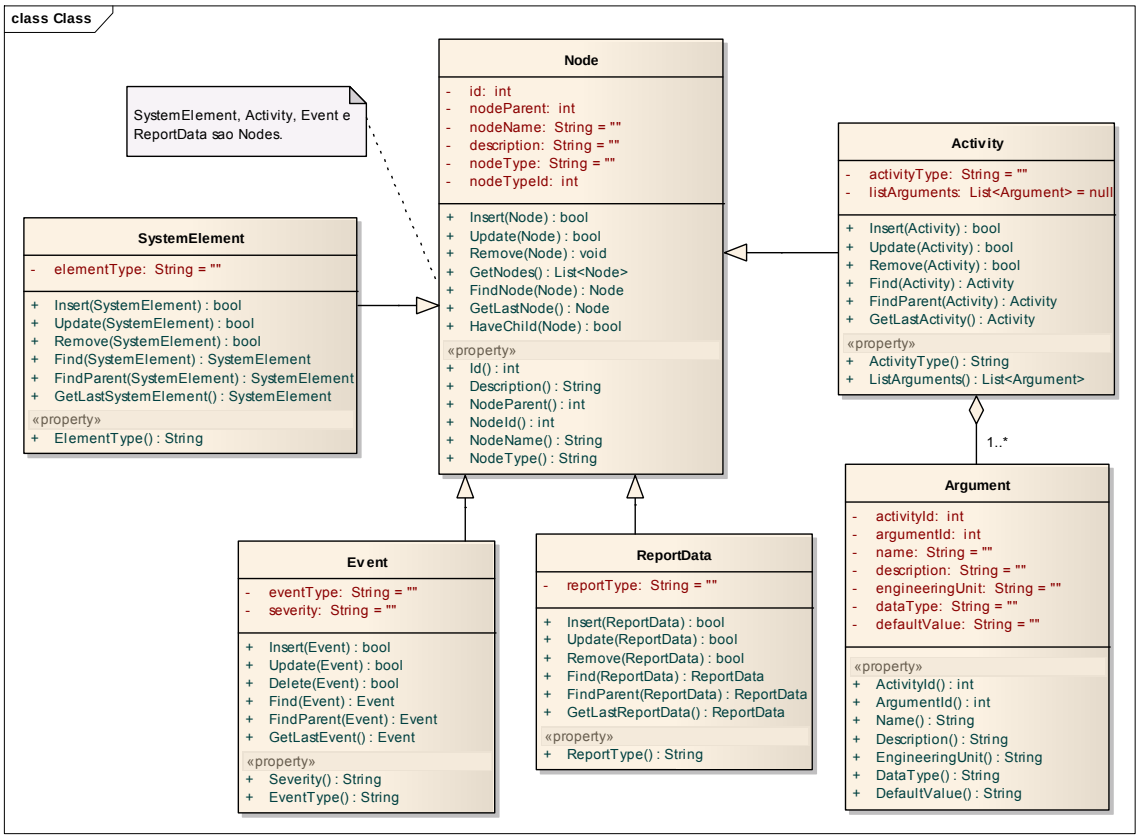

Figura C.1 – Diagrama de classes para gerenciar a criação da estrutura de modelos SSM no SSM Database

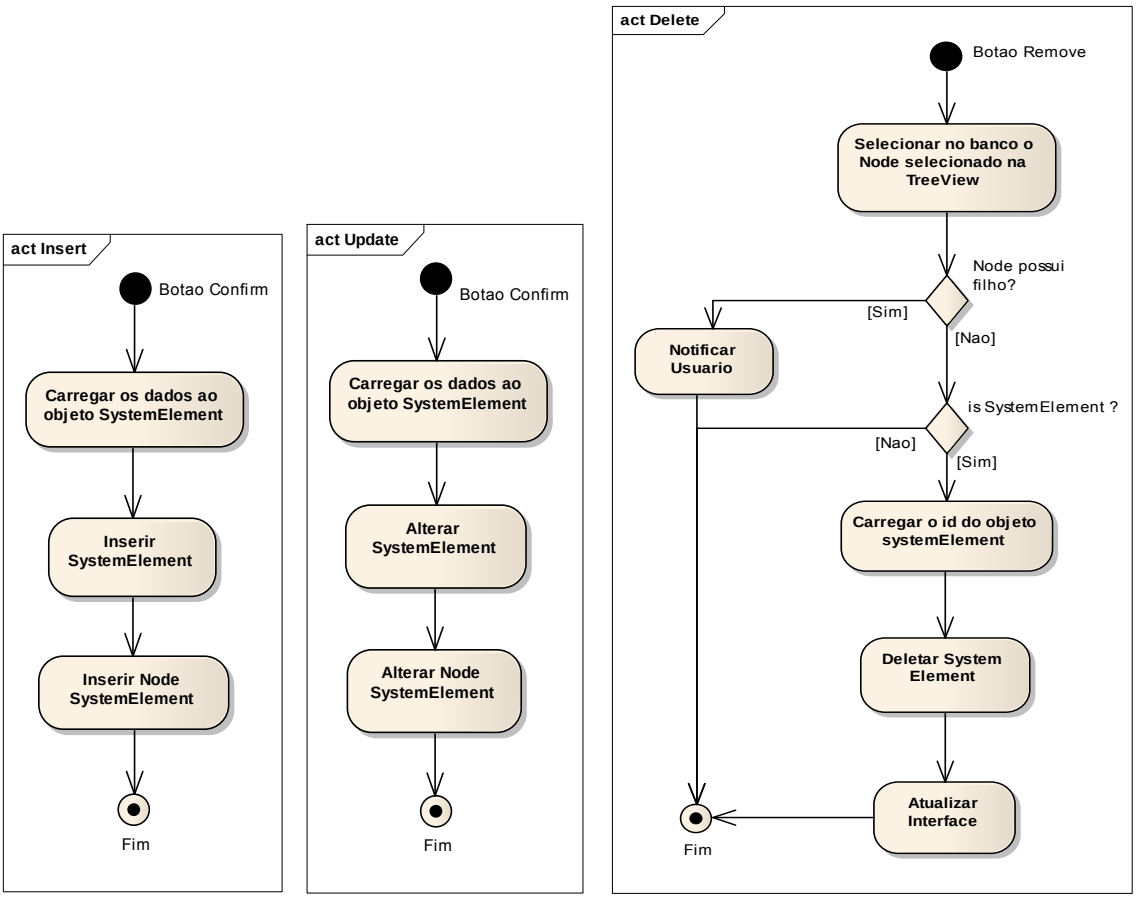

Figura C.2 – Diagramas de atividades para inserir, atualizar e remover um objeto do tipo SE no SSM Database

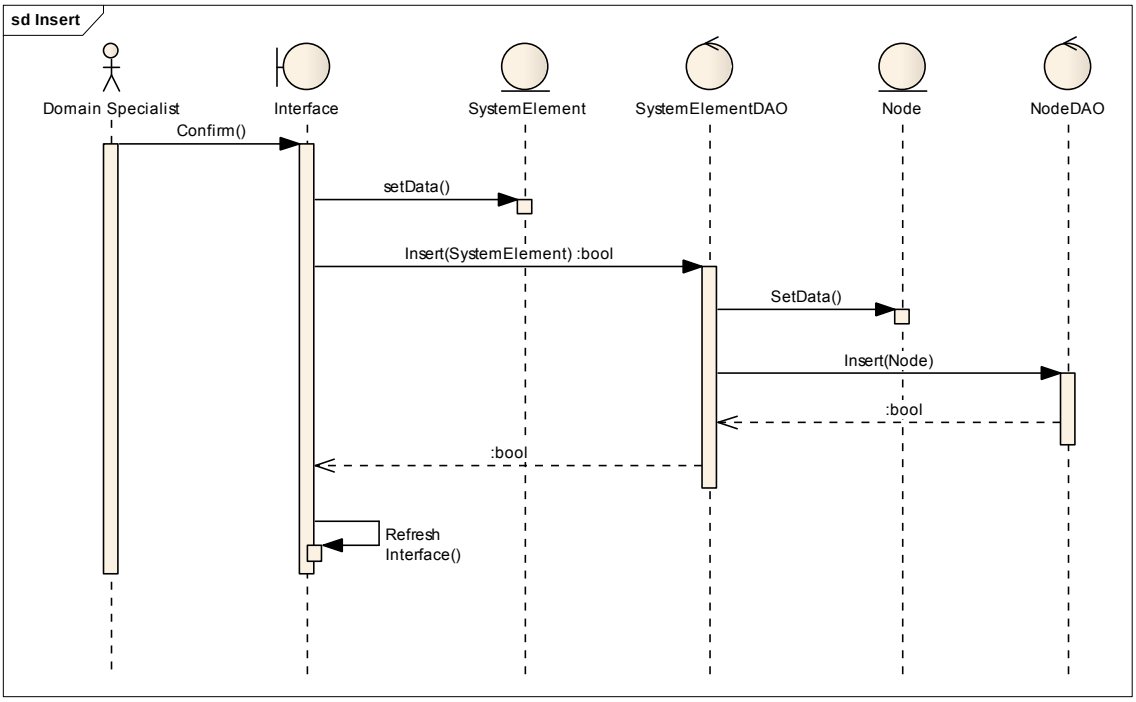

Figura C.3 – Diagrama de sequência para inserir um objeto do tipo SE

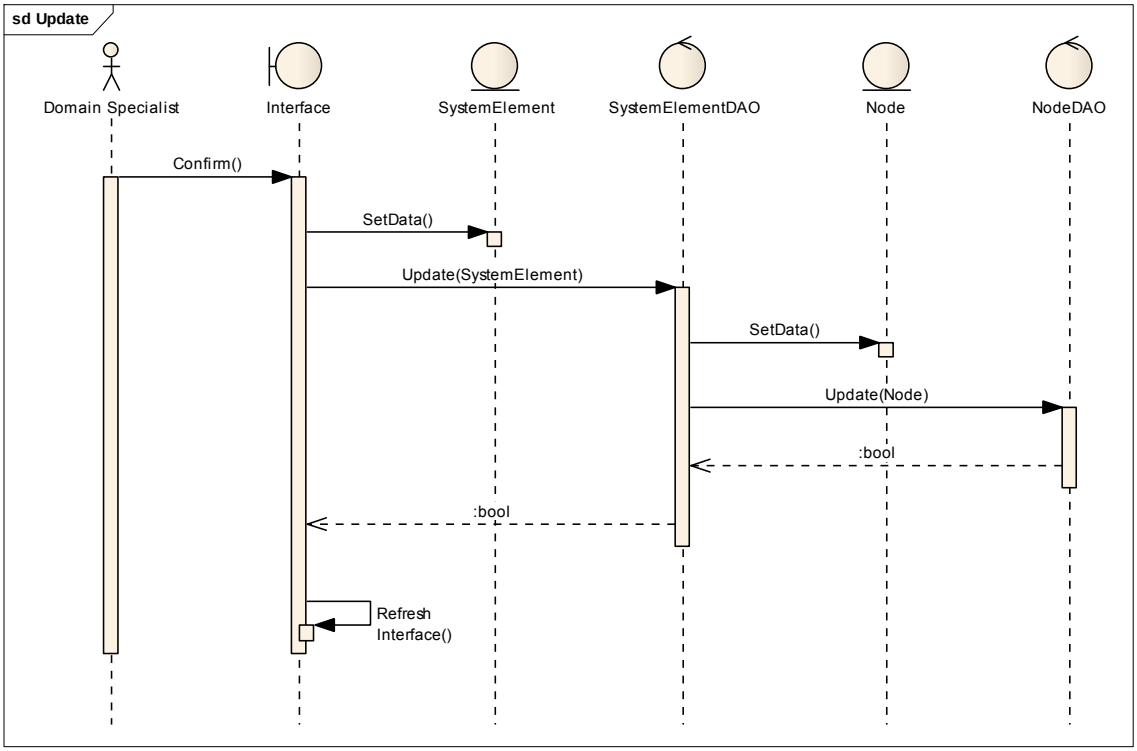

Figura C.4 – Diagrama de sequência para atualizar um objeto do tipo SE

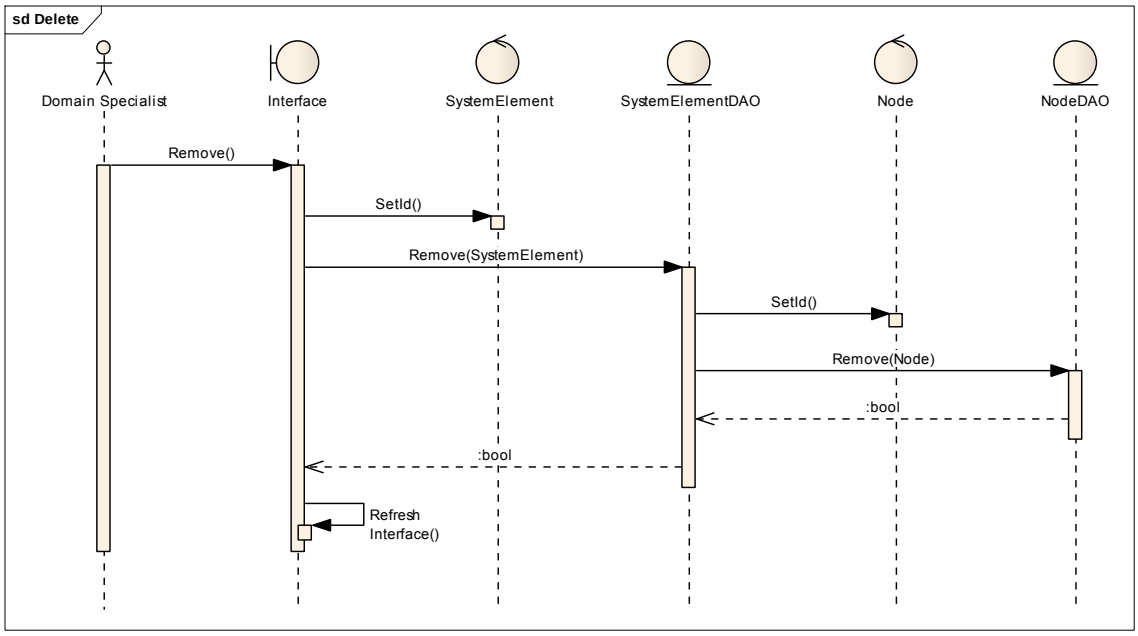

Figura C.5 – Diagrama de sequência para remover um objeto do tipo SE

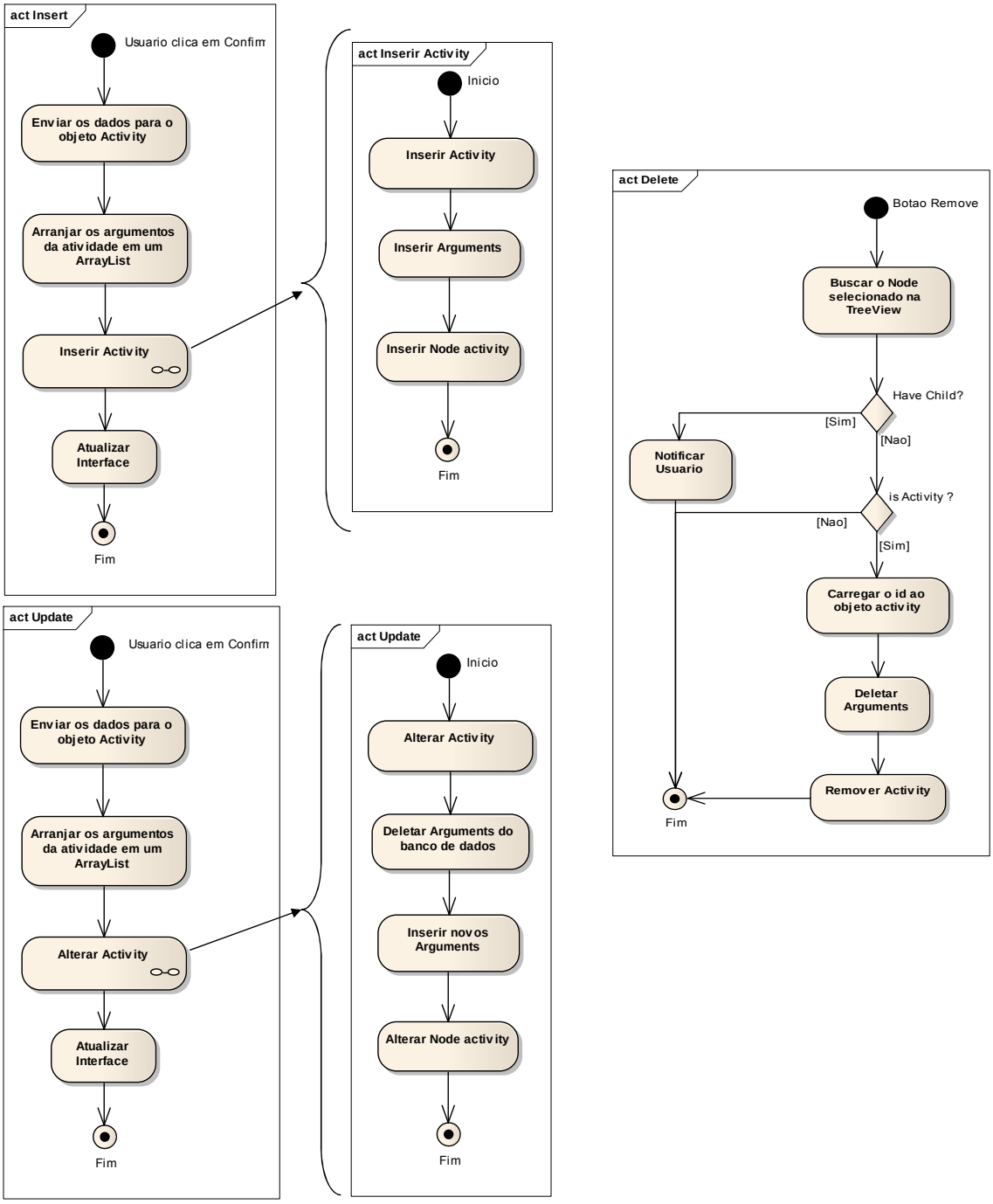

Figura C.6 – Diagramas de atividades para inserir, atualizar e remover um objeto do tipo Activity no SSM Database

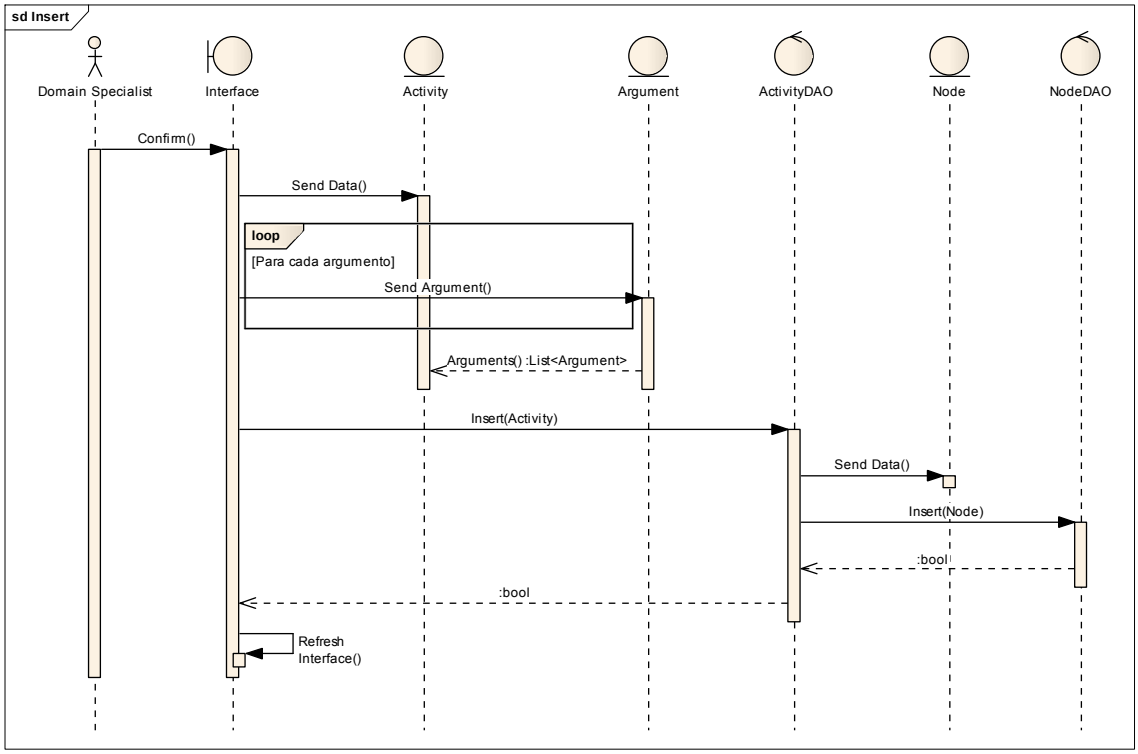

Figura C.7 – Diagrama de sequência para inserir um objeto do tipo Activity

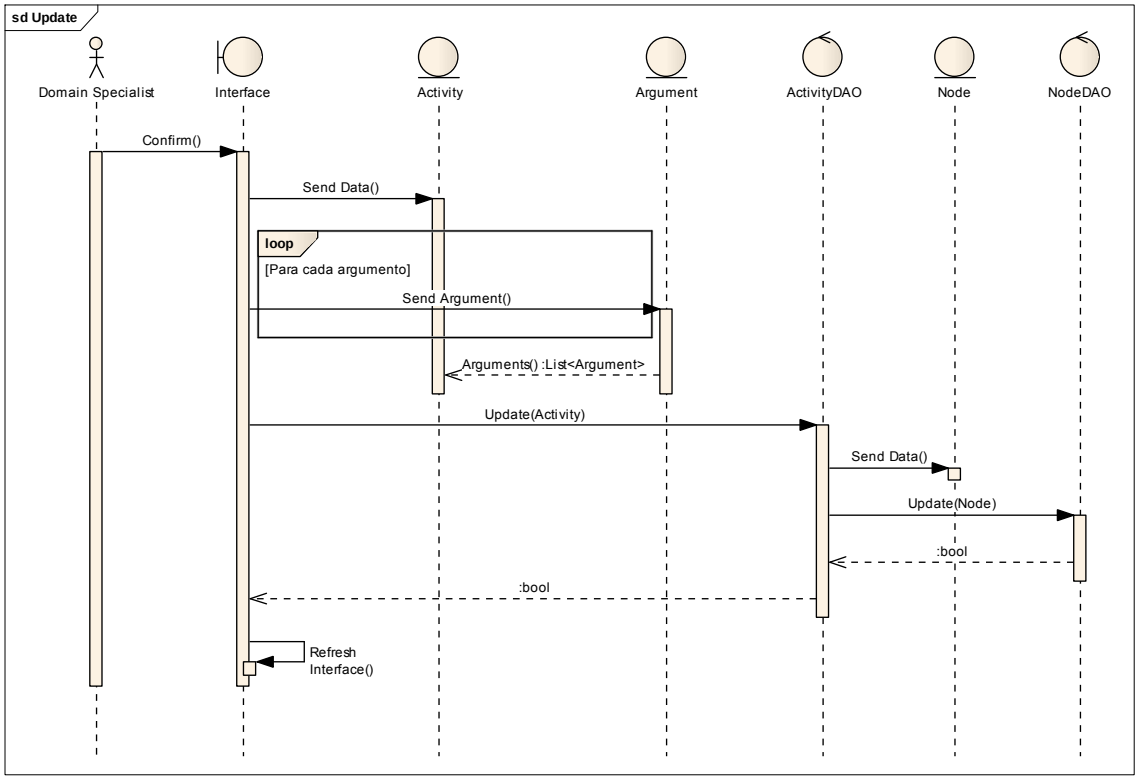

Figura C.8 – Diagrama de sequência para atualizar um objeto do tipo Activity

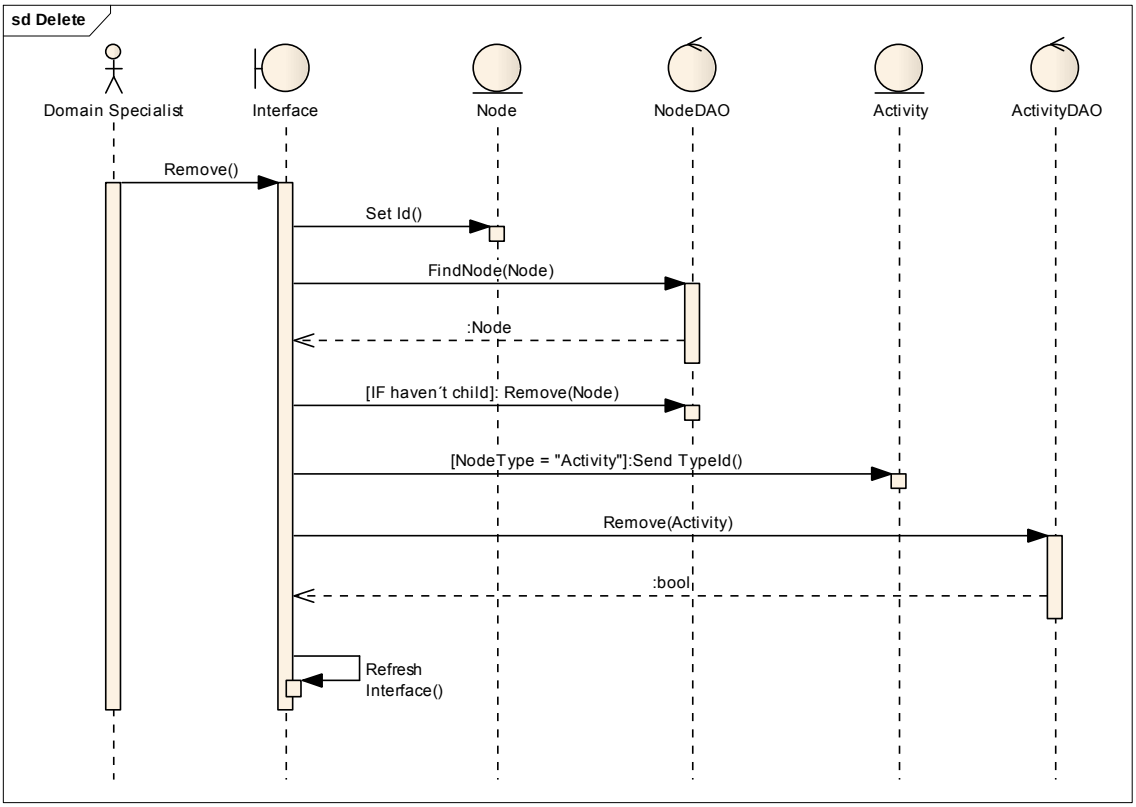

Figura C.9 – Diagrama de sequência para remover um objeto do tipo Activity

C.2 Protótipo de uma Interface Gráfica para Gerenciamento da Criação da Estrutura de Modelos SSM

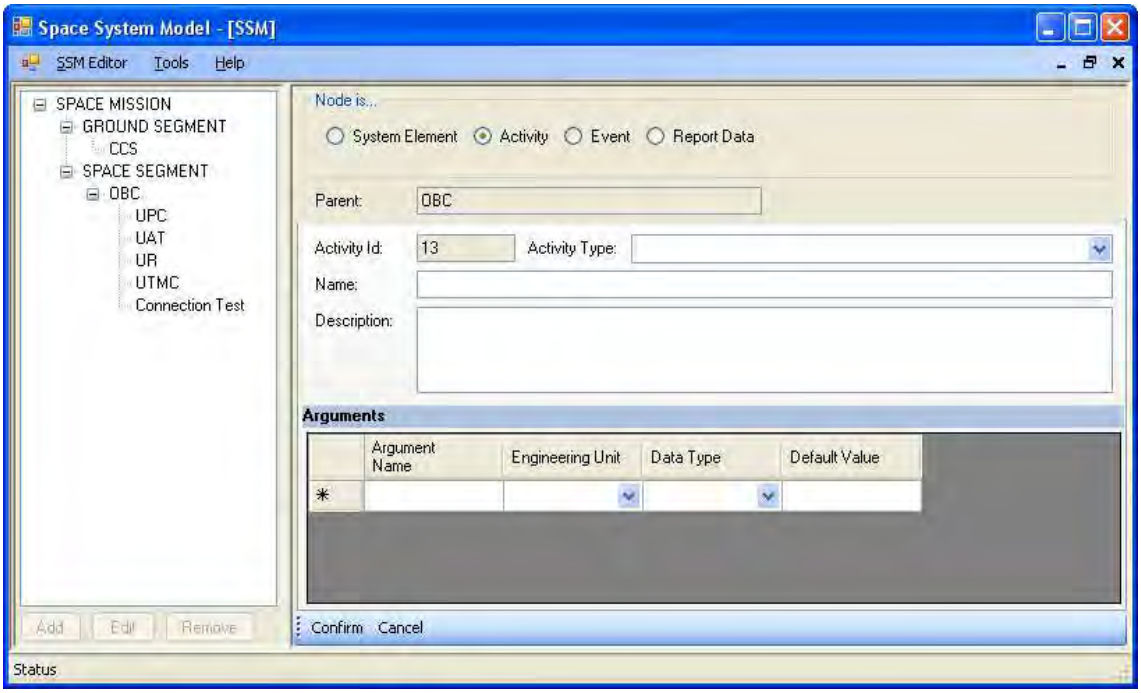

Figura C.10 – Protótipo de uma interface gráfica para gerenciamento da modelagem de um SSM