Ministério da
Ciência, Tecnologia
e Inovação

GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

sid.inpe.br/mtc-m19/2012/03.09.18.28-RPQ

UM ESTUDO SOBRE ARQUITETURAS DE SIMULADORES DE SATÉLITES

Ana Maria Ambrósio
Joaquim Pedro Barreto

Relatório Técnico de Pesquisa em Simuladores de Satélites.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3BG6628>

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

CONSELHO DE EDITORAÇÃO E PRESERVAÇÃO DA PRODUÇÃO INTELECTUAL DO INPE (RE/DIR-204):
Presidente:
Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)
Membros:
Dr. Antonio Fernando Bertachini de Almeida Prado - Coordenação Engenharia e Tecnologia Espacial (ETE)
Drª Inez Staciarini Batista - Coordenação Ciências Espaciais e Atmosféricas (CEA)
Dr. Gerald Jean Francis Banon - Coordenação Observação da Terra (OBT)
Dr. Germano de Souza Kienbaum - Centro de Tecnologias Especiais (CTE)
Dr. Manoel Alonso Gan - Centro de Previsão de Tempo e Estudos Climáticos (CPT)
Drª Maria do Carmo de Andrade Nono - Conselho de Pós-Graduação
Dr. Plínio Carlos Alvalá - Centro de Ciência do Sistema Terrestre (CST)
BIBLIOTECA DIGITAL:
Dr. Gerald Jean Francis Banon - Coordenação de Observação da Terra (OBT)
REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:
Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)
Yolanda Ribeiro da Silva Souza - Serviço de Informação e Documentação (SID)
EDITORAÇÃO ELETRÔNICA:
Ivone Martins - Serviço de Informação e Documentação (SID)

Ministério da
Ciência, Tecnologia
e Inovação

GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

sid.inpe.br/mtc-m19/2012/03.09.18.28-RPQ

UM ESTUDO SOBRE ARQUITETURAS DE SIMULADORES DE SATÉLITES

Ana Maria Ambrósio
Joaquim Pedro Barreto

Relatório Técnico de Pesquisa em Simuladores de Satélites.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3BG6628>

INPE
São José dos Campos
2012

RESUMO

Simuladores de satélite desempenham um importante papel no ciclo de vida de uma missão espacial. Esforços para diminuir custos e tempo de desenvolvimento são importantes para o sucesso dos mesmos. Uma arquitetura corretamente dimensionada pode ajudar na obtenção destes objetivos. Por conta desta importância, um estudo sobre arquitetura de satélites foi conduzido, utilizando-se diversas missões espaciais no âmbito da ESA (Agência Espacial Européia) para identificar suas aplicações e seus respectivos simuladores. Este estudo levou a um padrão sendo adotado, SMP (*Simulation Modeling Platform*), objetivando o incremento do reuso, portabilidade e interoperabilidade, cujas características são apresentadas neste trabalho. O SMP define um conjunto mandatório de interfaces a serem implementadas por qualquer simulador em desenvolvimento dentro das regras do padrão. Além disso, técnicas de engenharia de software como desenvolvimento baseado em componentes, padrões de projeto, desenvolvimento de software baseado em linha de produtos e arquitetura baseada em modelos contribuem para a flexibilidade e reusabilidade de um simulador de satélite. Este relatório apresenta o estudo relacionado às arquitetura de simuladores de satélite, o padrão SMP bem como descreve as tecnologias listadas.

ABSTRACT

Satellite simulators play an important role through a space mission life cycle. Efforts to mitigate costs and save developing time are crucial to the success of satellite simulators. A well designed architecture can help to achieve such purposes. Due to this importance, a satellite simulator architecture study was conducted using several ESA (European Space Agency) space missions to identify its applications and its respective simulators. This study has led to a standard being adopted to increase software reuse, portability and interoperability: SMP (*Simulation Modeling Platform*), whose characteristics are presented in this work. The SMP defines a set of mandatory interfaces to be implemented by a simulator being developed under this standard. Furthermore, software engineering techniques like component-based development, design patterns, software product line and model driven architecture can contribute to flexibility and reusability of a satellite simulator architecture. This report presents a study concerning satellite simulators architectures, th standard related as well as it depicts the technologies listed.

Ministério da
Ciência, Tecnologia e Inovação

GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

LISTA DE FIGURAS

Figura 2.1 - Arquitetura do SIMSAT. ADAPTADO DE: (HOMEM et al., 2006)..... 15
Figura 2.2 - Arquitetura do CSIM. ADAPTADO DE: (HOMEM et al., 2006)..... 18
Figura 2.3 - Arquitetura do simulador XMM. ADAPTADO DE (CÔME; IRVINE, 1998)..... 23
Figura 2.4 - Infraestrutura do simulador XMM. ADAPTADO DE (BODEMANN, 2004)..... 26
Figura 2.5 - Arquitetura do framework SimGen. 28
Figura 2.6 - Arquitetura do simulador Lisa Pathfinder (LPF)..... 31
Figura 2.7 - Arquitetura do simulador para segmento solo. 34
Figura 2.8 - Arquitetura da primeira versão do simulador SIMC3. 36
Figura 3.1 - Arquitetura do padrão SMP. ADAPTADO DE: (ECSS, 2011)..... 41
Figura 3.2 - Arquitetura típica de um simulador SMP. 43
Figura 3.3 - Diagrama hierárquico de componentes do SMP.FONTE: (ECSS, 2011) 44
Figura 4.1 - Arquitetura MDA. ADAPTADO DE (OMG, 2003) 54

LISTA DE TABELAS

Tabela 2.1 - Lista de atividades e requisitos de alta fidelidade..... 20
Tabela 2.2 – Objetivo e características dos simuladores de satélites..... 38
Tabela 2.3 – Detalhes dos Simuladores de satélites..... 40

SUMÁRIO

Pág.

RESUMO **5**

ABSTRACT **6**

SUMÁRIO **9**

INTRODUÇÃO **11**

1 - ARQUITETURAS DE SIMULADORES DE SATÉLITES **14**

1.1 - SIMSAT: núcleo de simulador de satélites 14
1.2 - CSIM: Simulator para a constelação de satélites GALILEO 16
1.3 - Simulador XMM (X-ray Multi Mirror) 19
1.4 - SimGen (Simulation Generation) 26
1.5 - Simulador LISA (Laser Interferometer Space Antenna) Pathfinder 29
1.6 - Simulador para o Segmento Solo 32
1.7 - Simulador SIMC3 35
1.8 - Sumário das características dos simuladores de satélites estudados 38

2 - O PADRÃO SMP **41**

3 - TÉCNICAS DE ENGENHARIA RELACIONADAS À REUTILIZAÇÃO DE SOFTWARE **47**

3.1 - Componentes de software 50
3.2 - Padrões de projetos (Design Patterns) 51
3.3 - Linha de produto de software (Product Line) 52
3.4 - Arquitetura Dirigida a Modelos (Model Driven Architecture - MDA) 53

4 - CONCLUSÃO **55**

REFERÊNCIAS BIBLIOGRÁFICAS **57**

INTRODUÇÃO

Um simulador de satélite é um software ou um sistema de software e hardware que representa o satélite físico real. Um simulador de satélite é utilizado em diferentes fases do ciclo de vida de uma missão espacial, tais como na integração e testes do modelo de engenharia do satélite, na integração e teste do Sistema de Monitoração e Controle de um Centro de Controle de Satélites, na avaliação de operações de contingências e também no treinamento dos operadores de satélite (EICKHOFF, 2009).

Considerando-se que toda missão espacial requer a construção de simuladores de satélite, a questão da reutilização como uma forma de reduzir os custos da missão é de fundamental importância. A reutilização de simuladores de satélite pode dar-se em duas vertentes diferentes. Na primeira, o simulador é reutilizado em várias fases de verificação e validação dentro de uma mesma missão. Na segunda, a reutilização se dá entre missões, uma vez que de um satélite para outro há vários pontos em comum (REGGESTAD et al., 2004).

Para melhor reutilizar um simulador de satélites, os seguintes aspectos merecem estudo aprofundado: (i) a abordagem de desenvolvimento do simulador, (ii) a forma como seus elementos representados por modelos interagem, (iii) o grau de fidelidade na representação, (iv) a presença ou não de hardware “in the loop”, (v) o grau de acoplamento entre os elementos a serem simulados e finalmente, (vi) o requisito de inserção e exclusão destes elementos a qualquer momento, de forma prática e rápida.

Relacionada à reutilização está a padronização. Com o objetivo de padronizar o desenvolvimento de simuladores de satélites, a Agência Espacial Européia (do inglês European Space Agency (ESA)) propôs um padrão chamado SMP (Simulation Model Portability) (ESA, 2005) que passou a ser denominado

SMP1 em sua primeira versão. Nesta versão as limitações nos mecanismos de comunicação e a não utilização de técnicas de orientação a objetos levaram a proposição de uma segunda versão, chamada então SMP2 (ESOC, 2005).

Esta nova versão do padrão utiliza técnicas e mecanismos para elaboração de uma arquitetura de software baseada em componentes, visando à independência de plataforma (portabilidade), interoperabilidade e reuso dos elementos do simulador. Entre as técnicas de software utilizadas estão, Orientação a Objetos, UML (Unified Modeling Language), Componentes de Software e Repositórios de Software. Este padrão define a modelagem do software como um conjunto de componentes, padroniza as interfaces entre os mesmos e fortalece, desta forma, a interoperabilidade e o reuso dos componentes.

A arquitetura do padrão SMP2 define uma camada de simulação, formada pela integração dos modelos necessários à criação do simulador e uma camada de ambiente de simulação, que interage com o simulador, fornecendo serviços básicos necessários à simulação tais como temporização, armazenamento de mensagens, tratamento de eventos, escalonamento, etc. Os componentes de software podem ser modelos e serviços, os quais implementam funcionalidades que podem ser fornecidas ou consumidas. Interfaces de software são definidas para conexão entre os elementos, permitindo interação e acoplamento com outros componentes, de tal forma que possam ser facilmente agrupados. Para cada simulador, dependendo da configuração desejada, um conjunto de componentes é selecionado e aglutinado. Um componente também pode possuir mais de uma versão, o que implica que uma determinada instância do simulador pode ser formada incluindo uma determinada versão de um componente. Esta solução promove fortemente a reutilização.

O SMP2 foi submetido à ECSS (do inglês European Cooperation for Space Standardization) para se tornar uma norma. Em janeiro de 2011 a ECSS disponibilizou um Memorando Técnico sob a denominação de Simulation Modelling Platform (SMP) (ECSS, 2011).

Este relatório apresenta um estudo sobre arquiteturas de simuladores de satélites encontradas na literatura. Um resumo das características do padrão SMP e uma discussão de técnicas aplicáveis à reutilização de sistemas de software são também apresentados.

O relatório está organizado da seguinte maneira: a seção 2 apresenta um resumo de arquiteturas de simuladores de satélites. No final desta seção, é apresentada uma síntese das principais características de cada simulador e o custo das missões nas quais os simuladores foram utilizados. A seção 3 descreve o SMP, enquanto que a seção 4 discute técnicas de Engenharia de Software aplicáveis à reutilização. A seção 5 traz a conclusão.

1 - ARQUITETURAS DE SIMULADORES DE SATÉLITES

Esta seção apresenta um resumo das principais características de simuladores desenvolvidos ou em desenvolvimento, encontrados na literatura.

1.1 - SIMSAT: núcleo de simulador de satélites

O SIMSAT consta de um núcleo de simulador, o qual provê uma estrutura genérica para simuladores desenvolvidos no âmbito do Centro de Operações de Satélites da ESA (ESOC) (HOMEM et al., 2006). Esta estrutura engloba uma interface com o usuário do simulador e um núcleo de serviços que permitem aos modelos do simulador interagirem com a infraestrutura de simulação. Entre estes serviços estão: núcleo de tempo real, escalonamento de eventos, gerenciamento de dados, *log* de mensagens, comandos, modelagem de equipamentos de estações terrenas como codificadores de telecomandos e decodificadores de telemetrias. O SIMSAT também possui uma biblioteca de modelos e componentes genéricos, que podem ser utilizados no desenvolvimento de um simulador operacional específico, como modelos de solo, modelo dinâmicos de órbita e veículos espaciais. Também provê um conjunto de emuladores para processadores, visando emular o hardware dos computadores de bordo. A versão 3.0 do SIMSAT é compatível com o padrão SMP1. Já a versão 4.0 deverá ser totalmente compatível com o padrão SMP2.

A Figura 2.1 mostra a arquitetura do núcleo de simulação SIMSAT.

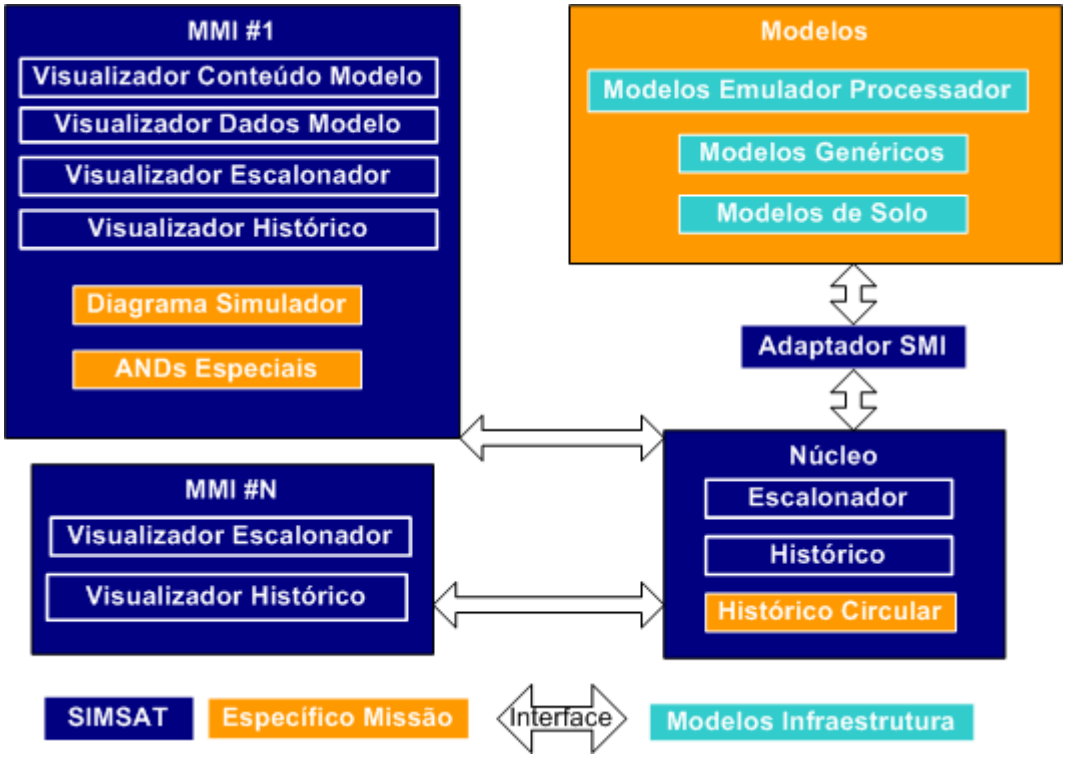

Figura 2.1 - Arquitetura do SIMSAT. ADAPTADO DE: (HOMEM et al., 2006)

A arquitetura do SIMSAT consta de quatro elementos principais: interfaces gráficas, do inglês, *Man Machine Interface* (MMI); o Núcleo da simulação, os Modelos e um adaptador entre os modelos e o núcleo, o Adaptador SMI (*Simulation Model Interface*). Nesta figura, observa-se que diferentes interfaces gráficas podem conectar-se ao núcleo de simulação, para controle e visualização dos dados. O Adaptador SMI entre os modelos e o núcleo permite que os modelos sejam interligados com a simulação em tempo de execução.

O núcleo de simulação inclui serviços como escalonador e armazenamento histórico, sendo que serviços customizados podem ser adicionados.

Um conjunto de modelos e componentes como emuladores de processadores, modelo de estações terrenas do sistema de solo, modelo de órbita e um pacote

genérico do subsistema elétrico são fornecidos para o desenvolvimento de simuladores baseados no SIMSAT.

Desta forma, o núcleo SIMSAT juntamente com os modelos específicos de uma missão (Específico Missão) e os modelos estruturais (Modelos Infraestrutura) previamente incluídos no SIMSAT permite a criação de diferentes simuladores favorecendo o reuso. A utilização do padrão SMP reforça ainda mais a interoperabilidade e reuso para desenvolvimento de simuladores de satélites no âmbito da ESA.

1.2 - CSIM: Simulator para a constelação de satélites GALILEO

O objetivo principal do simulador CSIM (HOMEM et al., 2006) é prover um ambiente completo e confiável para validação do segmento solo responsável pelo controle da constelação de satélites chamada GALILEO. Este simulador também será usado para treinamento dos operadores, validação dos procedimentos de controle dos satélites tanto no lançamento e nas órbitas iniciais (LEOP - *Launch and Early Orbit Phase*), quanto na fase de operação nominal. Um outro objetivo deste simulador é a investigação de anomalias e validação do software de bordo antes de sua carga no computador de bordo de voo.

O principal desafio deste simulador é representar múltiplos modelos de satélite coordenados em uma constelação. Até 51 satélites e as estações terrenas associadas poderão ser simulados. Cada modelo de satélite deve ser executado individualmente, porém coordenados por uma central de simulação. O simulador deve ainda considerar que cada satélite simulado é diferente em relação à função do software de bordo, a estados específicos e ao gerenciamento a partir de banco de dados específicos.

Entre os desafios tecnológicos deste simulador, pode-se destacar:

a) suporte a múltiplos satélites (constelação).

b) convivência entre modelos funcionais e modelos emulados.

c) missão de longa duração.

d) modelos de múltiplas estações terrenas, cada uma oferecendo contato com diferentes satélites.

A especificação do simulador CSIM requer compatibilidade com o padrão SMP2 e deve fazer uso da infraestrutura genérica do SIMSAT, apresentado na seção 2.1 deste documento. Em função da necessidade de prover um ambiente distribuído de simulação, foram desenvolvidos adaptadores para a interface dos modelos do SIMSAT.

A Figura 2.2 apresenta a arquitetura do simulador CSIM.

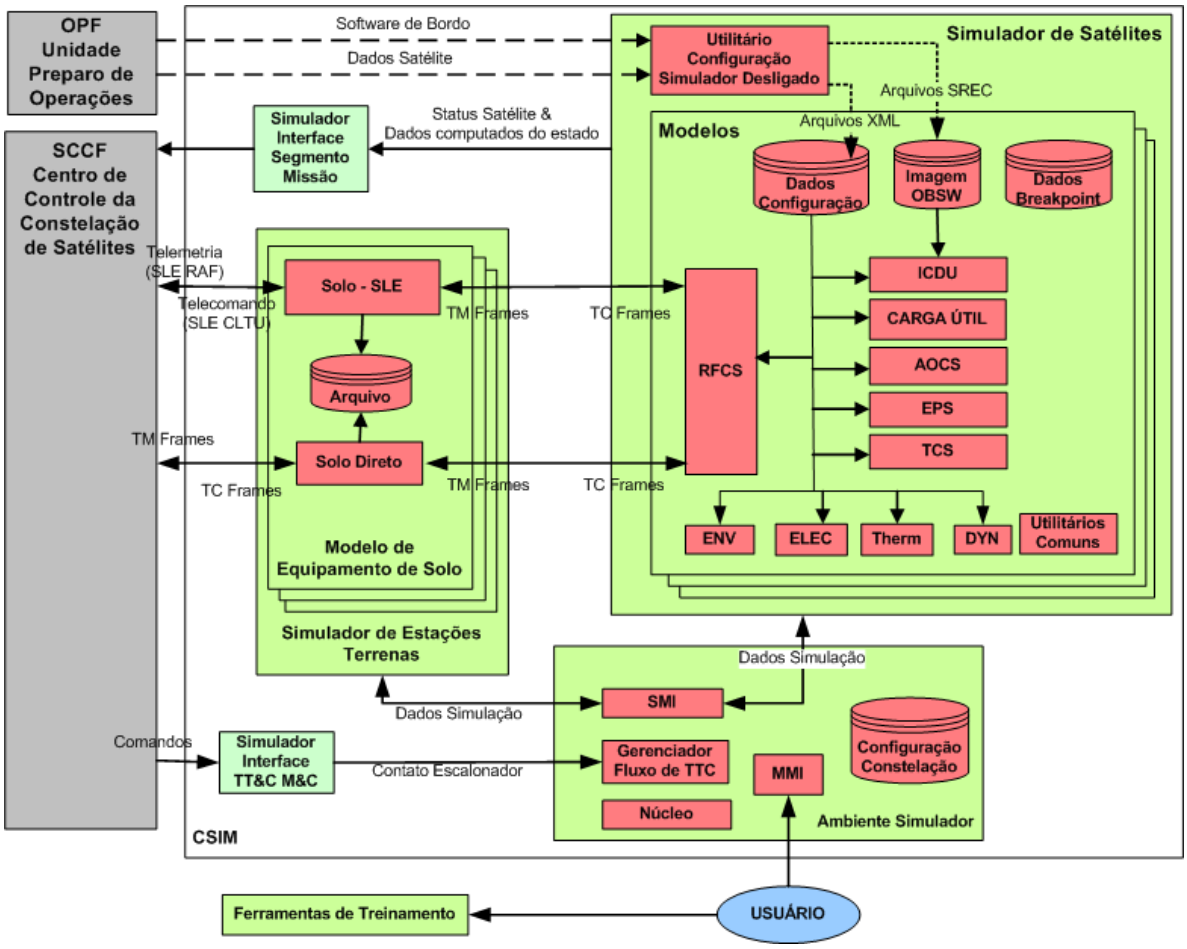

Figura 2.2 - Arquitetura do CSIM. ADAPTADO DE: (HOMEM et al., 2006)

Esta arquitetura é composta de três componentes funcionais principais: o Ambiente Simulador, o Simulador das Estações Terrenas e o Simulador de Satélites. Nesta arquitetura, o núcleo do simulador é o Ambiente Simulador (*Simulator Environment*) cuja base é o SIMSAT. Neste Ambiente encontram-se a base de dados de Configuração da Constelação, o Núcleo *do* SIMSAT, a interface com o usuário (*Man Machine Interface* - MMI), além da interface entre o SIMSAT e os modelos simulados e o simulador da estação terrena, ambos feitos via interface de simulação de modelos, a SMI (*Simulation Model Interface*). O Ambiente também mantém um link com a estação terrena de controle da constelação (*Spacecraft Constellation Control Facility* - SCCF)

através do Simulador Interface TT&C M&C, mostrado à esquerda na figura 2.2, por meio do qual são enviados macro comandos para o simulador.

O Simulador da Estação Terrena envia telemetrias ao SCCF e recebe telecomandos, tanto diretamente quanto via os serviços de transferência (*Space Link Extension* - SLE) do CCSDS. Cada um dos satélites simulados possui um sistema de controle de radio freqüência (*Radio Frequency Control System* - RFCS) para implementar esta comunicação.

No componente Simulador de Satélite são encontrados os subsistemas de controle de atitude e órbita (*Attitude and Orbit Control System* - AOCS), térmica (*Thermal Control System* - TCS), suprimento de energia (*Electrical Power System* - EPS), controle e gravação de dados (*Integrated Control and Data Unit* - ICDU), cargas úteis, bem como outros itens gerais vinculados ao satélite como ambiente espacial (ENV), elétrica (ELEC), controle térmico (Therm) e dinâmica de voo (DYN). Também estão inclusas as bases de dados e o software do computador de bordo (OBSW). Todo o software do computador de bordo é executado em um processador virtual ERC-32 de 64 bits, cujo emulador é provido pelo SIMSAT. Possui interface com o SCCF através de um simulador de interfaces do segmento de missão, enviando dados computados do estado do simulador.

O simulador também possui interface com a unidade para preparação de operações (*Operation Preparation Facility* - OPF) que oferece dados de entrada para configuração do simulador.

1.3 - Simulador XMM (X-ray Multi Mirror)

A missão XMM consiste de um telescópio de três espelhos, operando na faixa inferior de raios X do espectro eletromagnético. Nesta missão, o segmento lançador baseou-se no foguete Ariane 5. O segmento solo consiste do Centro

de Operação da missão (MOC) e do Centro de Operação Científica da missão (SOC), além de outros itens como estações terrenas e demais estruturas físicas.

Para testar e validar os sistemas do Centro de Operação da Missão (MOC) e Centro de Operações Cientificas (SOC) foi desenvolvido o simulador XMM (CÔME; IRVINE, 1998). Inicialmente a previsão era desenvolver dois simuladores separados, um para operações de missão (MOCSIM) e outro para operações de ciência (SOCSIM). Porém, em função do curto prazo, houve uma junção das especificações e apenas um simulador foi desenvolvido, englobando as duas funcionalidades. O simulador XMM enquadra-se na categoria de simulador de tempo real e é baseado no SIMSAT.

Para efeito de comparação, a tabela 2.1 mostra as atividades e os requisitos de alta fidelidade previstos para os simuladores, considerados individualmente (MOCSIM e SOCSIM).

Tabela 2.1 - Lista de atividades e requisitos de alta fidelidade

	MOCSIM	SOCSIM
Atividades	Teste e validação do Sistema de Controle da Missão XMM (XMCS) e do Sistema de Dinâmica de Vôo (FDS)	Teste e validação do Sistema de Controle Científico XMM (XSCS)
	Validação da base de dados operacional do Centro de Operação do XMM	Validação da base de dados operacional SOC XMM
	Teste e validação dos procedimentos das operações de	Teste e validação dos procedimentos de operação dos

	vôo (nominal e contingência)	experimentos (nominal e contingência)
	Validação dos procedimentos para suporte ao Sistema de Validação de Testes e testes fim a fim.	Validação dos procedimentos para suporte ao Sistema de Validação de Testes e testes fim a fim.
	Treinamento da equipe de operação de vôo	Treinamento da equipe de operação científica
Alta Fidelidade	Simulação fiel do satélite XMM – emulação do software da CDMU (Unidade de Gerenciamento e Controle de Dados) e do ACC (Controle de Atitude)	Modelagem funcional fiel do OBDH
	Modelagem funcional dos instrumentos, com modelagem precisa da geração de telemetria de housekeeping e modelagem simulada das telemetrias científicas	Emulação em software dos vários processadores controladores de instrumento para permitir a execução dos instrumentos (apenas um instrumento deve ser simulado por vez)
	Interface idêntica ao equipamento da estação terrena para envio de telecomando e recepção de telemetria	Modelagem fiel dos instrumentos
	Possibilidade de introduzir falhas simuladas ou efeitos especiais em	Playback dos dados científicos armazenados

	todos os subsistemas durante a execução de uma simulação	
		Possibilidade de definir, em tempo de execução, qual instrumento é emulado
		Possibilidade de introduzir falhas ou efeitos especiais em todos os instrumentos (equipamentos)

A Figura 2.3 apresenta a arquitetura do simulador XMM.

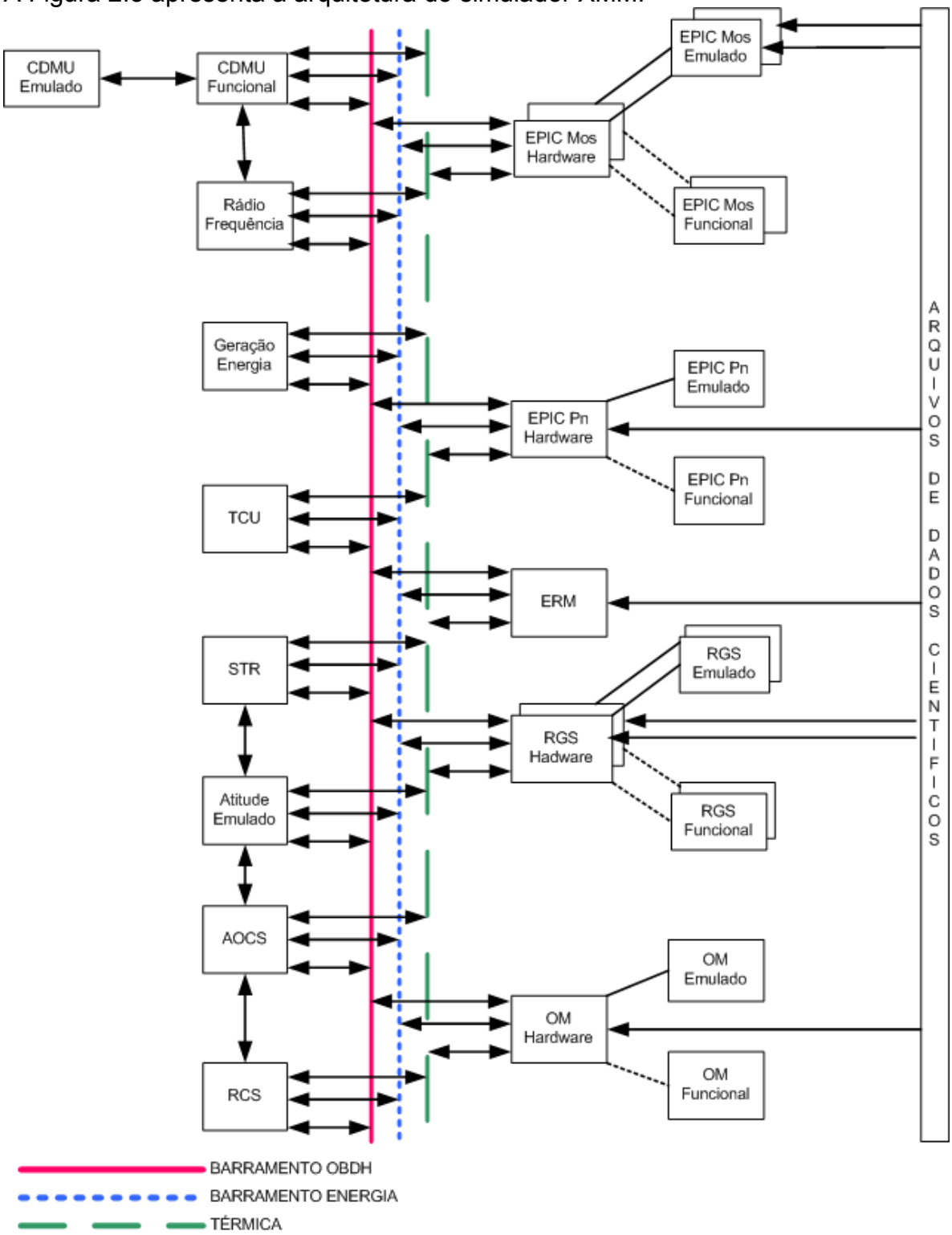

Figura 2.3 - Arquitetura do simulador XMM. ADAPTADO DE (CÔME; IRVINE, 1998)

À direita da figura, algumas cargas úteis da missão XMM, como a câmara EPIC (*European Photon Imaging Camera*), o espectômetro RGS (*Reflection Grating Spectrometer*), o monitor de radiação ERM (*EPIC Radiation Monitor*) e o monitor óptico OM (*Optical Monitor*) são mostrados. Com exceção do ERM, cujo modelo incluso é apenas funcional, os demais contém um modelo emulado, além do modelo funcional.

Ao centro da figura aparecem os barramentos relativos ao OBDH, a energia e a térmica. E à esquerda estão os subsistemas do satélite a serem incluídos no simulador.

Dentre as principais características do simulador, destacam-se:

a) cada subsistema é visto como um 'módulo', com um ponto de conexão ao OBDH, com pontos de conexão à rede elétrica e pontos de conexões aos elementos de simulação térmica. Esta é uma importante característica, uma vez que permite incrementos no simulador para adicionar novos subsistemas e funcionalidades. Integrar um novo subsistema ao simulador implica conectá-lo ao bus do OBDH, à rede elétrica e à rede térmica.

b) o simulador é decomposto em subsistemas, como por exemplo, OBDH, subsistema de potência, térmica, controle de reação (Reaction Control Subsystem - RCS), sensor de estrelas (Star Tracker - STR), unidade de telecomando (TCU), etc. As interfaces entre os subsistemas no simulador são identificadas no satélite real, no que se refere a projeto do satélite, documentos, etc.

c) as interfaces entre os subsistemas são minimizadas para facilitar o projeto do software e simplificar a integração.

d) os modelos de instrumentos são configuráveis como modelos funcionais ou emulados. Por razões de similaridade e eficiência, ambos usam a mesma memória e acessam os modelos de hardware através da mesma interface de entreda e saída (I/O).

e) existe um decodificador de telemetria que processa pacotes produzidos pela unidade de gerenciamento e controle de dados, a CDMU (Control and Data Management Unit). A arquitetura passou a considerar ambas (emulada e funcional), como previsto antes da união das duas especificações do simulador. Parâmetros analógicos de telemetrias são mostrados no formato bruto e também em unidade de engenharia. O decodificador permite a operação do simulador sem o centro de controle, facilitando os testes e a aceitação.

f) para validação das funcionalidades das cargas úteis EPIC e RGS, um conjunto de arquivos de dados científicos pré processados é disponibilizado. Dependendo do modo de operação do instrumento, os dados podem sera cessados pelo mesmo e enviados posteriormente ao segmento solo como telemetrias.

A Figura 2.4 mostra a integração com o SIMSAT e as interfaces externas de configuração através dos arquivos de dados.

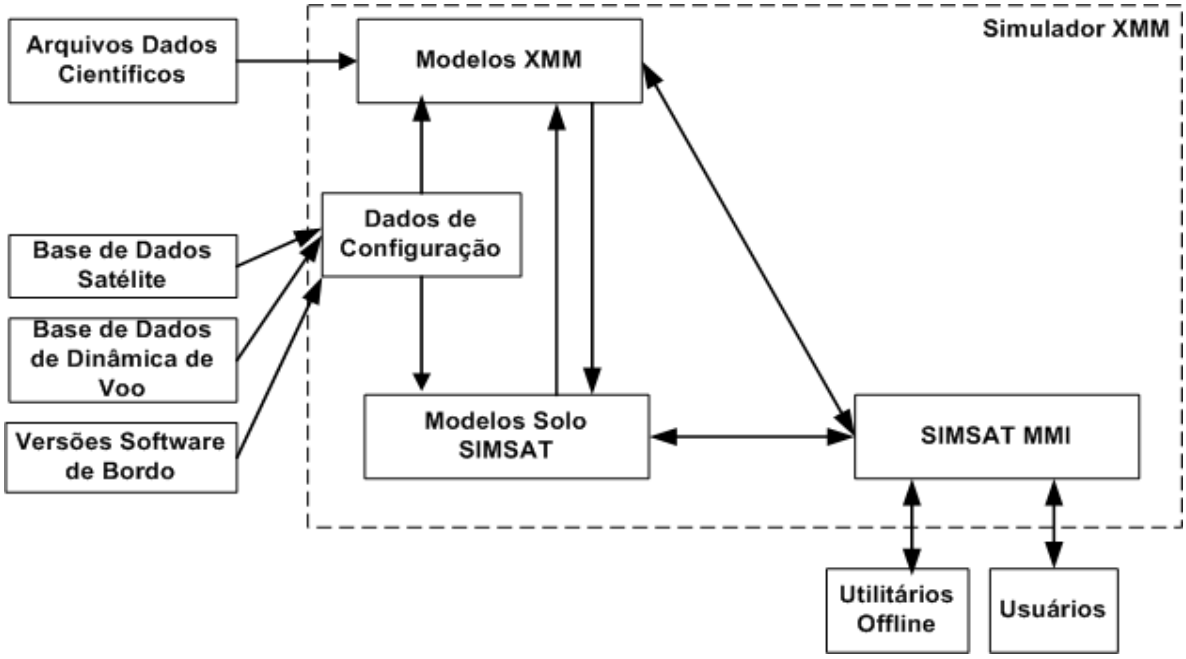

Figura 2.4 - Infraestrutura do simulador XMM. ADAPTADO DE (BODEMANN, 2004)

O simulador permite que se configure qual dos modos, emulado ou funcional, dos instrumentos será utilizado. Ao todo, sete possíveis configurações estão disponíveis para o simulador, podendo ser convenientemente escolhidas através da interface visual (MMI) do SIMSAT.

1.4 - SimGen (Simulation Generation)

O Simulador SimGen é um *framework* para criação de simuladores baseado em parâmetros, com o objetivo de prover uma flexibilidade que um simulador baseado em hardware não possui (ROLET; CROENNE, 2006). Os parâmetros, por sua vez, estão armazenados em uma base de dados. Na criação de um simulador, apenas os parâmetros desejados são importados e disponibilizados para serem utilizados através de uma interface gráfica. O SimGen pode ser usado no desenvolvimento de simuladores para sistemas espaciais e treinamento de operadores. Este *framework* foi utilizado para desenvolvimento dos seguintes produtos:

a) *Columbus Simulator*, para treinamento de controle de vôo, validação de procedimentos de vôo e testes dos subsistemas do sistema de monitoração e controle (MCS).

b) *Payload Simulator*, para modelagem *onboard* de *payloads* e interação com o simulador Columbus.

c) *Terrasar–X Simulator*, que injeta dados de telemetria em um modelo 3D do satélite para acompanhamento de sua trajetória em tempo real ou simulado.

d) *3D Computer Based Training*, para aprendizado através da interação com um modelo 3D e suas reações.

O *framework* é baseado em blocos escritos na linguagem Java, de forma a torná-lo independente em termos de hardware. Inclui uma ferramenta de edição de script, permitindo a criação de simuladores sem necessidade de codificação. Na edição de um script é possível inserir blocos de simulação, seja de forma centralizada ou distribuída, sendo que o SimGen se encarrega da coordenação entre os módulos.

A Figura 2.5 mostra a arquitetura do *framework* SimGen.

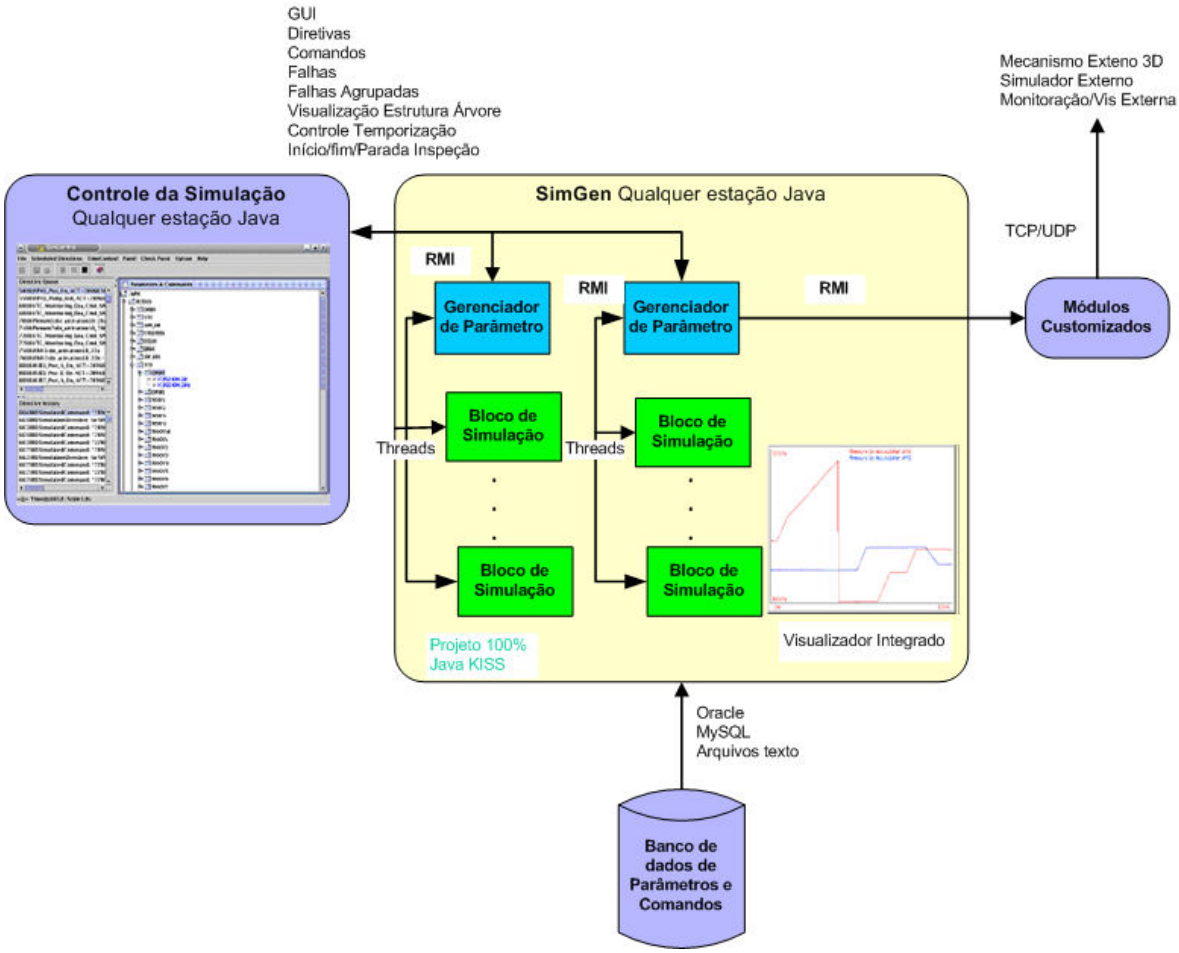

Figura 2.5 - Arquitetura do framework SimGen.

ADAPTADO DE: (ROLET; CROENNE, 2006)

Uma simulação baseada no SimGen pode ser centralizada ou distribuída. No caso de se optar por uma solução em ambiente distribuído, o protocolo RMI (do inglês *Remote Method Invocation*) da linguagem Java é utilizado. Toda comunicação e propagação entre os blocos de simulação e outros módulos (interface gráfica, controle da simulação, etc.) é feita com base no RMI. Estes blocos se comunicam com o SimGen através de uma API (*Application Programming Interface*). Módulos customizados para comunicação TCP/UDP estão disponíveis, permitindo diversidade na visualização dos dados de

simulação. Os parâmetros (telemetria e telecomandos) disponíveis são armazenados em bases de dados, que podem ser customizadas na criação de simuladores através de uma ferramenta de edição de *scripts*.

A arquitetura prevê uma interface gráfica (GUI) para edição dos arquivos e funcionalidades para execução dos *scripts*. Uma vez que o SimGen é baseado em parâmetros e valores, estes dados são armazenados em banco de dados padrões.

A arquitetura do SimGen permite o uso de diferentes gerenciadores de banco de dados, como por exemplo: MySQL, Oracle, MS Access, etc.

A GUI fornece facilidades de importação dos dados para criação de simuladores customizados. A visualização em árvore permite uma rápida visualização dos parâmetros bem como facilidade na gestão dos mesmos. Qualquer modificação nos dados será percebida na próxima carga dos mesmos, sem que nenhuma modificação no código seja necessária.

1.5 - Simulador LISA (Laser Interferometer Space Antenna) Pathfinder

A missão LISA Pathfinder (LPF) (DELHAISE; BRU, 2006) está sendo desenvolvida para validar a tecnologia necessária para se alcançar a sensibilidade de detecção de ondas gravitacionais no espaço, requerida à missão LISA, cujo lançamento está previsto para 2014.

O simulador operacional para a missão LPF tem o objetivo de validar e testar o segmento solo, provendo telemetrias de forma realística, em resposta a telecomandos enviados, e implementar as interfaces com o centro de missão. Também deverá validar a base de dados do satélite e os procedimentos operacionais, treinar a equipe de controle e validar modificações no software de bordo.

Este simulador inclui três elementos principais:

a) o modelo do satélite (plataforma e cargas úteis) - o qual simula resposta a telecomandos, geração de telemetria, funcionalidade de tempo real para todos os subsistemas, emulação do computador de bordo e dos *payloads*, ainda que com funcionalidade limitada.

b) ambiente espacial - o qual simula de forma realista as órbitas, com períodos de AOS (do inglês *Acquisition Of Signal*) e de LOS (do inglês *Loss Of Signal*) para as estações, eclipse solar e lunar, correta execução de manobras orbitais, sensores de estrela, Terra e Sol.

c) equipamentos e links nas estações de recepção de telemetria e envio de telecomando, bem como interfaces de comunicação.

A Figura 2.6 mostra a arquitetura do simulador LPF.

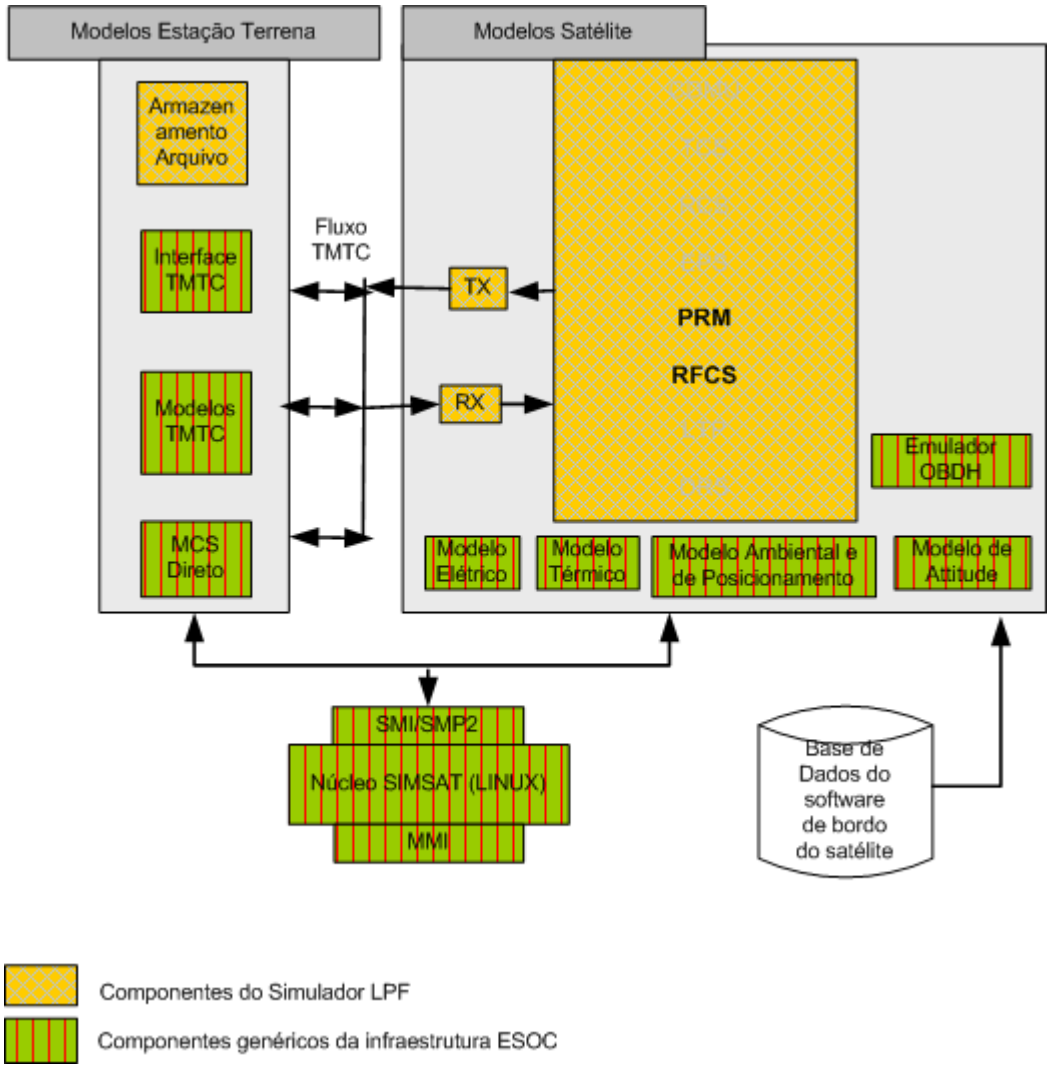

Figura 2.6 - Arquitetura do simulador Lisa Pathfinder (LPF).

ADAPTADO DE: (DELHAISE; BRU, 2006)

A arquitetura proposta considera a aplicação de elementos de projeto de software e de códigos, proveniente de simuladores anteriores.

O simulador deve fazer reuso do núcleo de simulação SIMSAT existente no ESOC (*European Space Operation Center*) e do software de infraestrutura chamado SCOS-2000 (Sistema de Controle de Missão genérico) (COUCH et al., 2006).

Também deverá ser compatível com o SMP2 em função do reuso do SIMSAT. Estas partes ilustradas na Figura 2.6 são:

a) modelos da Estação Terrena: Interface TMTC, Modelos TMTC, MCS Direto.

b) modelos do Satélite: Modelo Elétrico, Modelo Térmico, Modelo Ambiental e de Posicionamento, Modelo de Atitude e Emulador.

c) interface gráfica de gerenciamento (MMI), Núcleo SIMSAT e SMI/SMP2.

Os blocos referentes a armazenamento de arquivos, transmissão de telemetrias (TX) e recepção de comandos (RX), sistema de controle de rádio freqüência (RFCS) e gerenciamento de cargas úteis (PRM) são de desenvolvimento específico para o simulador LPF, como ilustrado na legenda da Figura 2.6.

Os modelos das estações terrenas incluem o sistema de telemetria e telecomando (TMTC) e também a interface com o Centro de Missão (MCS).

1.6 - Simulador para o Segmento Solo

O simulador para Segmento Solo apresentado por Stellato (STELLATO; ROMANI, 2005) tem por objetivo simular diferentes configurações de segmento solo, incluindo definição de cenários, execução do plano de voo e análise dos

dados. Este simulador pode ser utilizado para análise de facilidade de segmento solo para diferentes missões.

Esta ferramenta em software permite: (i) avaliar questões como tamanho e crescimento da rede de dados, (ii) robustez na arquitetura do segmento solo para gerenciamento de falhas e mudanças na configuração do segmento solo, (iii) avaliar cenários de atuação nominal, de contingência e de crise, (iv) checar a performance em termos de propagação de órbita, levando-se em conta recursos do sistema e tempo de execução das atividades, (v) treinar operadores para procedimentos de manutenção, entre outras.

Do ponto de vista operacional, a execução deste simulador requer a definição de um cenário que por sua vez é composto de elementos dos segmentos solo e espaço, bem como suas interligações. Um plano de voo é requisitado ao Centro de Missão e, após a obtenção do mesmo, a simulação é executada. O simulador gera dados de saída conforme esperado pelo Centro de Missão, no mesmo formato e taxa.

A análise de dados gerados permite, então, que se faça uma verificação do segmento solo simulado.

A figura 2.7 mostra a arquitetura do simulador.

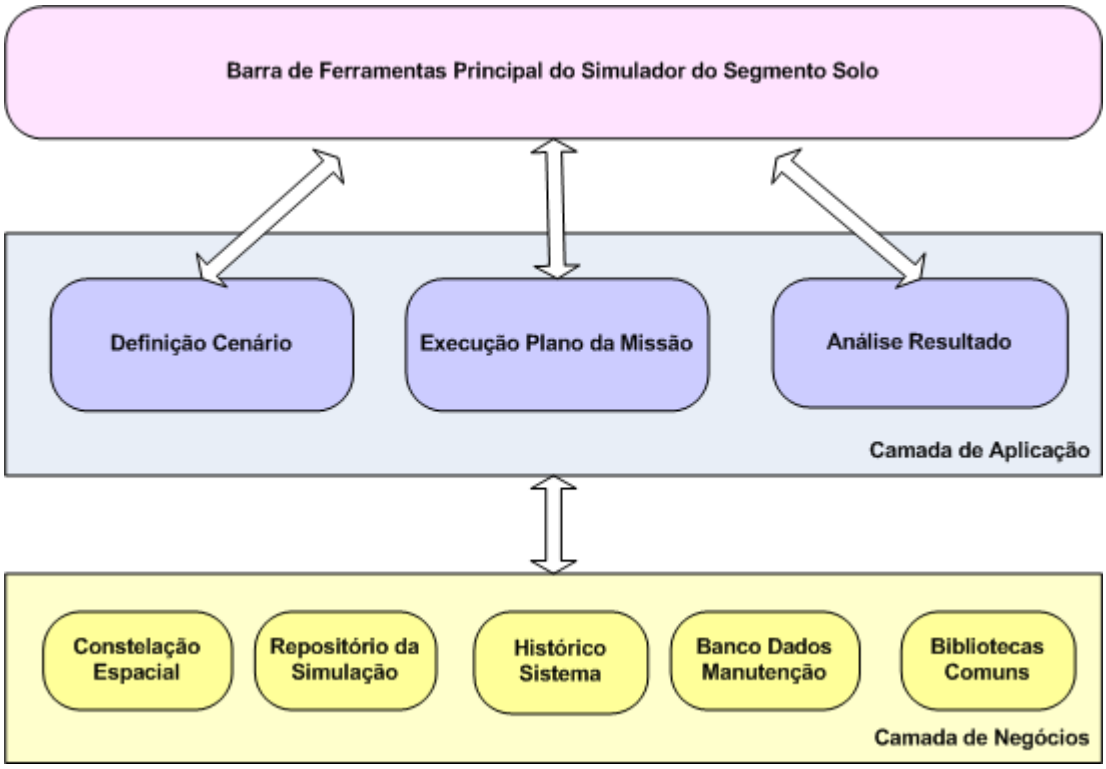

Figura 2.7 - Arquitetura do simulador para segmento solo.

ADAPTADO DE: (STELLATO; ROMANI, 2005)

Três camadas compõem a arquitetura: a interface com o usuário, a definição, controle e análise da simulação (aplicações) e, finalmente, a camada com os componentes responsáveis pelo domínio do problema.

A interface com o usuário, representada pelo retângulo arredondado superior, é responsável pela execução geral da simulação e oferece acesso a todas as ferramentas disponíveis.

A Camada de Aplicação contém opções para definição e execução do cenário e posteriormente a análise. Na definição, ocorre a preparação de um plano que contenha os parâmetros e suas respectivas ligações, bem como recursos

extras para execução. Os dados são obtidos de um repositório contendo todos os parâmetros. Já a execução de um plano inclui a definição do que é necessário para a simulação, como o cenário, plano da missão e os dados dinâmicos de vôo. A ferramenta de execução perfaz uma estimativa dos comandos a serem enviados com base nos planos da missão e nas manobras de rotina. Após a simulação do envio dos telecomandos, o simulador avalia se as aquisições e downloads previstos no plano da missão são factíveis, considerando falhas e janelas de tempo. Em caso positivo, simula a aquisição dos dados. A análise do resultado conta com ferramentas para geração de gráficos e relatórios, bem como navegação através dos dados para comparação. Também oferece interface para ferramentas externas, como o Matlab.

A Camada de Negócios contém um conjunto de bibliotecas usadas pelas ferramentas de definição, execução e análise das simulações.

1.7 - Simulador SIMC3

O simulador SIMC3 (AMBROSIO et al., 2006) simula o satélite de sensoriamento remoto chamado CBERS3 (China and Brazil Earth Resource Satellite 3). Este satélite está sendo desenvolvido através de uma parceria entre China e Brasil. O CBERS3 é um satélite para fornecimento de imagens do sistema terrestre que são utilizadas para mapeamento de queimadas e desmatamento, monitoramento de recursos hídricos e áreas agrícolas, ocupação urbana e do solo, entre outros campos de aplicação.

O simulador SIMC3, desenvolvido pelo INPE, tem por objetivo proporcionar uma ferramenta para treinar os operadores dos satélites CBERS3 e 4, validar procedimentos de voo durante o ciclo de vida do satélite e validar o software de controle de satélites a ser utilizado no Centro de Controle de Satélites do INPE.

A Figura 2.8 mostra a primeira versão da arquitetura do simulador SIMC3.

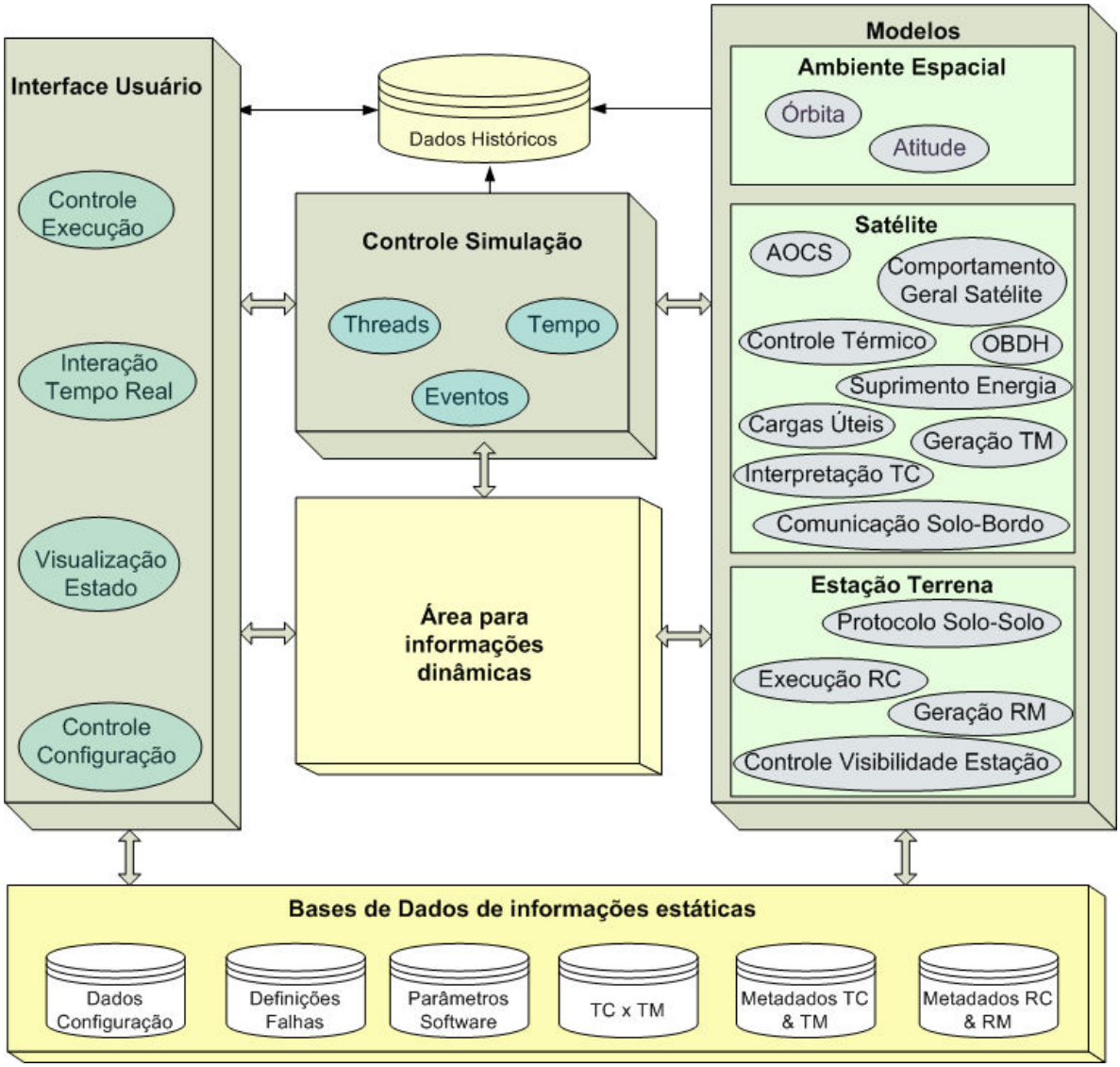

Figura 2.8 - Arquitetura da primeira versão do simulador SIMC3.

ADAPTADO DE: (AMBROSIO et al., 2006)

A arquitetura proposta compõe-se de três grandes blocos de software: Interface com o Usuário, Controle da Simulação e Modelos.

A Interface com o Usuário inclui os controles para início, pausa, reinício e fim de uma simulação. Também permite a escolha de uma configuração, a interação em tempo real para alteração de parâmetros da simulação e visualização da evolução dos valores dos parâmetros simulados. Entende-se por configuração o conjunto de valores iniciais de todos os parâmetros de simulação.

O bloco Modelos engloba o conjunto dos modelos dos subsistemas do satélite a serem simulados, os algoritmos que mapeiam o ambiente espacial e o comportamento das estações terrenas.

Finalmente, o Controle da Simulação inclui os mecanismos de software necessários à execução do simulador, tais como controle de *threads*, temporização, controle de eventos, manipulação controlada dos parâmetros, armazenamento das mensagens de *log* para histórico da simulação, controle de criação e comunicação entre os modelos, entre outros.

Com o objetivo de agilizar a comunicação entre os componentes do simulador, o acesso a discos e banco de dados é minimizado pela carga de todos os parâmetros necessários para a simulação para a memória principal do ambiente de simulação. A arquitetura é do tipo *blackboard*, na qual a comunicação é centralizada em uma área de memória comum a todos os modelos e o controle de leitura e escrita é realizado pelo bloco Controle da Simulação.

As bases de dados armazenam as informações estáticas como, configurações da simulação, descrição dos parâmetros, lista de falhas, os formatos e

conteúdos dos pacotes de telemetria e de telecomandos manipulados durante uma rodada de simulação.

1.8 - Sumário das características dos simuladores de satélites estudados

Um resumo do estudo sobre simuladores de satélites é apresentado nas tabelas 2.2 e 2.3. A tabela 2.2 mostra os objetivos e as características dos simuladores, enquanto que a tabela 2.3 sumariza, para os diversos simuladores listados, o autor ou a empresa que desenvolveu o simulador, a data em que o simulador tornou-se disponível para uso, o estado do desenvolvimento, o satélite que ele simula e o custo previsto da missão.

Tabela 2.2 – Objetivo e características dos simuladores de satélites

Nome	Objetivo	Características
CSIM	Sistema global de navegação via satélite	- Usa SIMSAT - Compatível com SMP2 - Simula múltiplos satélites - Emula o processador do OBDH
Simulador XMM	Testar e validar os centros de controle de missão e de dados científicos do satélite XMM	- Usa SIMSAT - Compatível com SMP2 - Engloba Centro de Missão e Centro de Dados Científicos
SimGen	Prover um framework genérico para desenvolvimento de	- Parametrizado - Baseado em blocos desenvolvidos em

	simuladores	Java - Independe do hardware -Apresenta editor de scripts com interface gráfica
Simulador LPF	Testar o segmento solo, validar a base de dados do satélite e os procedimentos operacionais, bem como validar o software de bordo	- Usa SIMSAT - Compatível com SMP2 - Simula satélite e ambiente espacial
Segmento Solo	Executar análises da configuração de um segmento solo de missão	- Configurável - Reutilizável durante toda a missão
SIMC3	Testar e validar procedimentos e validar os softwares usados no Centro de Controle de Software, além de treinar os operadores	- Desacoplamento entre modelos - Configurações e parâmetros armazenados em bases de dados reconfiguráveis - Reuso de software

Tabela 2.3 – Detalhes dos Simuladores de satélites

Nome	Autor/Empresa	Data Início Uso	Status	Satélite	Custo da missão
CSIM	SciSys, Vega, Critical, SkyTek	2011	Dois satélites para validação em órbita lançados	Constelação Galileo	Mais de 3.4 bilhões de euros
Simulador XMM	ESA/ESOC, Vega	1999	Concluído	XMM	689 milhões de euros
SimGen	SESS	2004	Concluído	Genérico	NA
Simulador LPF	ESOC, EADS Astrium	2011	Não finalizado	LPF	240 milhões de dólares
Segmento Solo	Dataspazio	-	Concluído	Segmento Solo	NA
SIMC3	INPE	-	Em desenvolvimento	CBERS-3	340 milhões de dólares (FONSECA, 2011)

2 - O PADRÃO SMP

O padrão SMP (*Simulation Modeling Platform)* visa promover a portabilidade, interoperabilidade e reuso entre diferentes ambientes e sistemas de simulação de satélites (ECSS, 2011).

A Figura 3.1 apresenta a visão de alto nível do SMP.

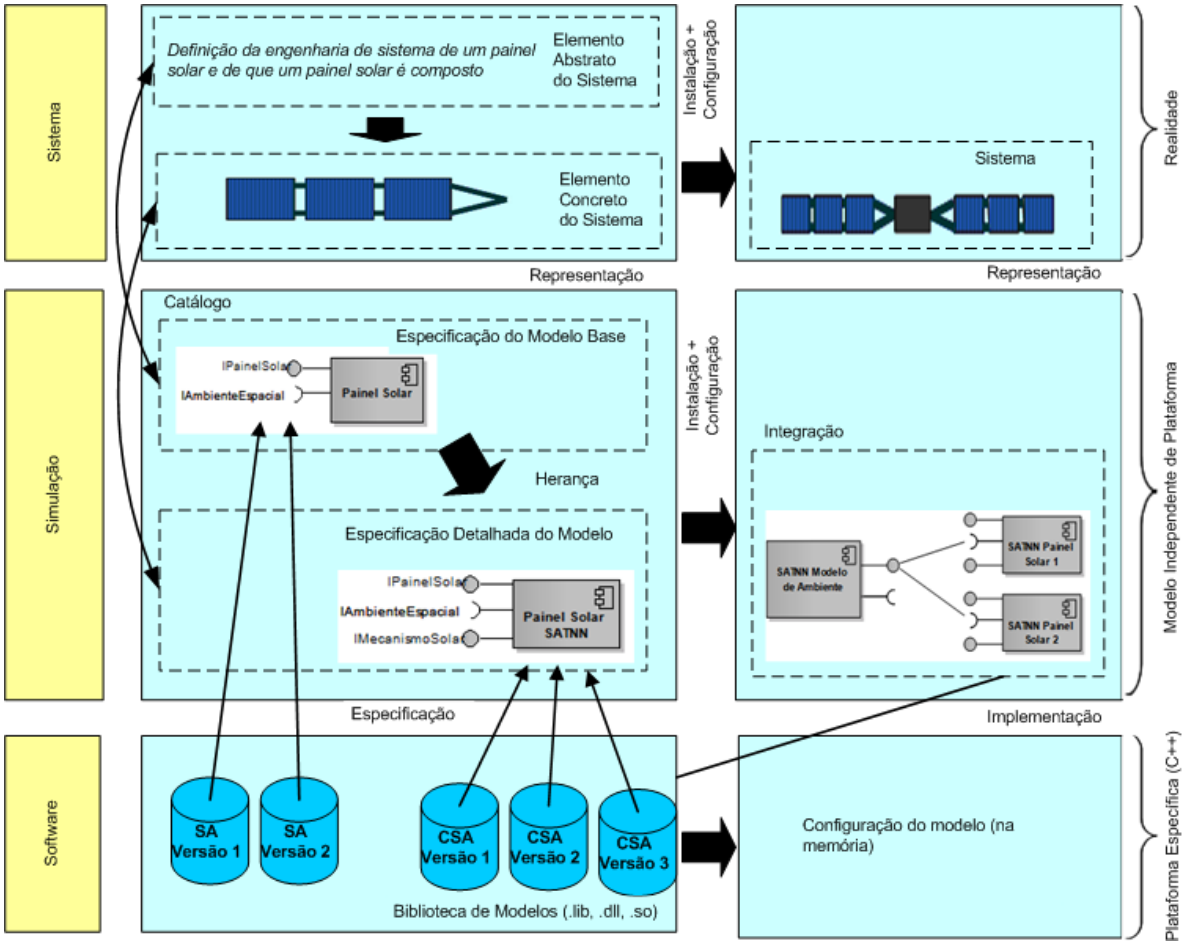

Figura 3.1 - Arquitetura do padrão SMP. ADAPTADO DE: (ECSS, 2011)

A arquitetura do padrão é decomposta em três camadas, sendo que a primeira camada representa o mundo real a ser modelado (*reality*), a segunda camada apresenta a especificação dos modelos através de uma plataforma independente (PIM - *Platform Independent Model*), não levando em conta como serão desenvolvidos e executados. A terceira camada apresenta os objetos já em plataforma específica (PSM - *Platform Specific Model*) para desenvolvimento e execução.

A Figura 3.1 acima também apresenta duas colunas sendo que a primeira corresponde às definições dos artefatos e a segunda a instâncias dos mesmos, se tomarmos como analogia a Orientação a Objetos.

O objetivo principal do SMP é a promoção da independência de plataforma para os modelos. Esta independência é obtida pela definição de um conjunto de modelos, independentes de plataforma, que depois possam ser mapeados para uma plataforma específica.

Outros fundamentos usados no SMP que reforçam o reuso e a portabilidade são: (i) a separação entre projeto e execução (demonstrado nas colunas da figura anterior); (ii) configuração dinâmica; (iii) extensivo uso de interfaces, componentes e herança (recurso de orientação a objeto).

O padrão SMP possibilita a criação de objetos através de uma configuração dinâmica, o que permite a troca de modelos em tempo de execução, por exemplo. Uma vez que as interfaces sejam idênticas, é possível que seja feita a substituição de um modelo por outro modelo com maior fidelidade ou até mesmo o chaveamento de um modelo de software que simula um hardware por um equipamento real (*hardware in the loop*).

A arquitetura típica de um simulador baseado no padrão SMP possui uma camada contendo os modelos dos subsistemas do satélite a ser simulados, uma camada de serviços de simulação obrigatórios e uma camada de controle da simulação (ambiente de simulação). A Figura 3.2 mostra esta arquitetura.

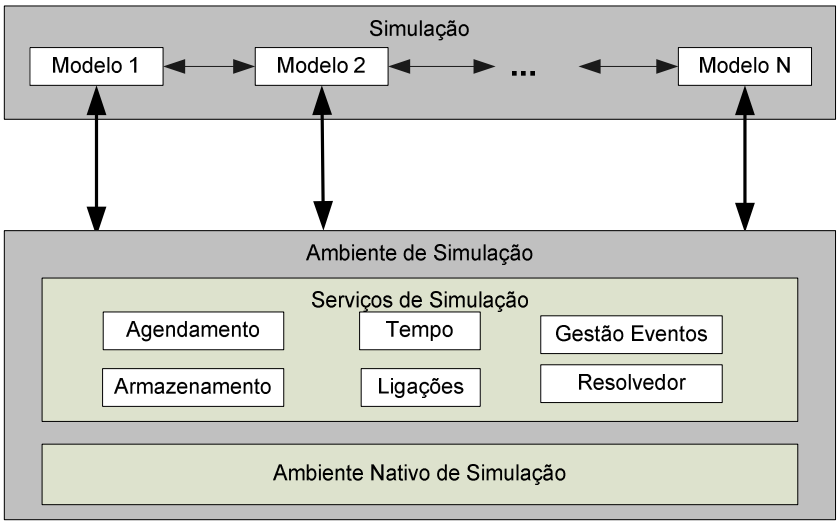

Figura 3.2 - Arquitetura típica de um simulador SMP.

ADAPTADO DE: (ECSS,2011)

O núcleo da simulação é representado pelos modelos. Os Serviços de Simulação fazem parte da camada Ambiente de Simulação, a qual provê facilidades para os modelos. Ainda nesta camada de Ambiente, eventuais recursos nativos de simulação já existentes podem ser reaproveitados, através de um encapsulamento dos mesmos, para que se adaptem ao padrão de interfaces de componentes do SMP.

A arquitetura do SMP é baseada em componentes. Tanto modelos como serviços são vistos como componentes, o que permite um tratamento igualitário entre eles. Todas as conexões entre componentes e entre serviços e

componentes são realizadas através de um conjunto predefinido de interfaces. Todo componente pode abrigar uma árvore hierárquica abaixo de si, através de composições. A interface de componente provê métodos para navegação através da hierarquia construída, o que permite recuperar todas as informações dos componentes das composições. A Figura 3.3 mostra o diagrama de componentes.

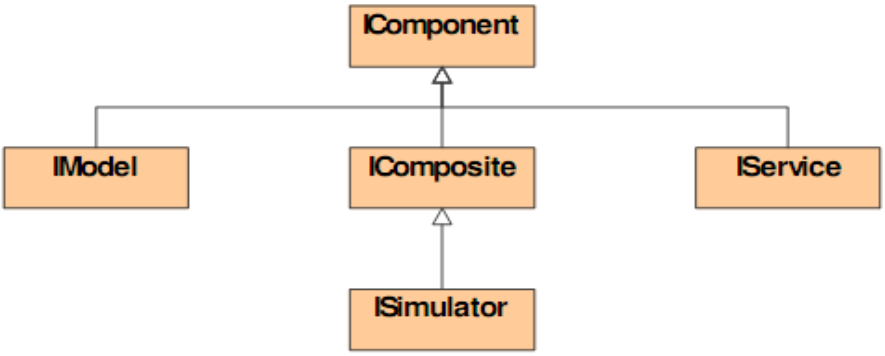

Figura 3.3 - Diagrama hierárquico de componentes do SMP.FONTE: (ECSS, 2011)

Os modelos (interface IModel) e serviços (interface IService) se equivalem, o mesmo ocorrendo com as composições (interface IComposite), que são agrupamentos de modelos. A instância do simulador (interface ISimulator) não possui um componente pai, sendo ela a raiz da árvore.

Este padrão define quatro serviços obrigatórios e outros opcionais. Também é possível a criação de serviços pelo usuário, desde que respeitando a interface padrão para serviços (IService). Os serviços mandatórios são:

a) temporização: provê quatro tipos de tempo para a simulação - tempo da simulação, tempo da missão, tempo da época (*epoch time*) e tempo UTC (computador).

b) gerenciamento de eventos: provê um mecanismo para disparo de eventos globais assíncronos.

c) armazenamento: armazena todas as mensagens geradas por modelos ou serviços.

d) escalonamento: executa eventos de forma cíclica ou em intervalos de tempos específicos.

e) ligações: mantém a lista de todos os vínculos entre os componentes da simulação, permitindo que a remoção dos mesmos seja feita de forma segura.

f) resolvedor: permite resolver as referências a outros componentes pelo nome. A referência pode ser tanto absoluta quanto relativa a outro componente.

A criação do ambiente de simulação com os componentes, seus relacionamentos e escalonamentos para execução pode ser feita de duas formas: estática e dinamicamente.

Na criação estática, todos os passos para criação do ambiente, até o momento em que o simulador esteja pronto para execução, são feitos com base em código fixo. Ou seja, qualquer modificação que venha a ser feita posteriormente implicará em modificação no código de criação e recompilação.

A criação dinâmica permite que os passos para a criação do ambiente de simulação sejam definidos em arquivos externos, escritos em uma linguagem própria, baseada em XML (XML, 2006). Desta forma, os passos da criação podem ser modificados em tempo de execução. Com isso, fica facilitada a tarefa de substituir, por exemplo, um dado componente, em tempo de execução. O SMP usa uma linguagem de definição de modelos para simulação

(SMDL - *Simulation Model Definition Language*) que permite criar catálogos de modelos, relações de dependências entre modelos, escalonamentos, integração, empacotamento e documentação.

Cada um destes catálogos contém um conjunto de informações, que estão agrupadas em:

a) modelos: contém as definições dos modelos, seus tipos e atributos. Os modelos podem estar aninhados, formando uma hierarquia.

b) dependências: as instâncias dos modelos estão conectadas, usando interligações baseadas em regras e restrições definidas para os modelos.

c) escalonamentos: define como as instâncias dos modelos serão escalonadas para execução, tanto com base em tempos pré-definidos quanto em eventos cíclicos. Define também as tarefas que serão disparadas quando das execuções.

d) integração: define quais instâncias ou tipos serão integrados em uma biblioteca ou um arquivo binário. Embora normalmente haja apenas uma instância de um mesmo modelo, outras poderão ser criadas, permitindo, assim, diferentes versões de um mesmo modelo.

e) documentação: refere-se ao ambiente a ser criado por estes catálogos. Seu objetivo é interligar todos os catálogos para criação do ambiente de simulação.

3 - TÉCNICAS DE ENGENHARIA RELACIONADAS À REUTILIZAÇÃO DE SOFTWARE

O desenvolvimento de um simulador de satélites em software envolve a análise de diversos itens, como tempo, custo, esforço, tecnologia e manutenção, que devem, tanto quanto possível, serem equilibrados com o objetivo de se minimizar os impactos negativos.

Desta forma, tecnologias que possam diminuir os impactos destas variáveis na produção de software são buscadas com afinco. Para simuladores de satélites, o prazo e o custo são elementos cruciais, uma vez que estes produtos devem, via de regra, ser entregues antes dos demais, para auxílio em etapas de verificação, validação e operação das missões.

Outro detalhe acerca do desenvolvimento de simuladores de satélites é o fato de existirem recorrências e similaridades patentes entre os diversos satélites de uma mesma missão ou entre satélites de diferentes missões. Desta forma, muita repetição pode ocorrer, o que implica afirmar que um mecanismo que aproveite estas similaridades pode ser muito útil na redução de prazo e custo.

Esta seção discute algumas tecnologias de engenharia de software que estão relacionadas as questões de maximizar o reuso de software.

O reuso de software é a tecnologia a ser explorada para que se obtenha simultaneamente a minimização dos impactos negativos dos itens mencionados e o aproveitamento da repetição de características comuns entre diferentes satélites em uma mesma missão e entre missões.

Segundo (PRESSMAN, 2005), "reuso é simplesmente um procedimento qualquer para se produzir (ou que ajude a produzir) um sistema através da reutilização de qualquer elemento advindo de um esforço anterior”. A questão então é o que reutilizar e quais procedimentos que, se seguidos, resultarão em

sucesso no reuso. O mesmo autor também afirma que "reuso é uma atividade e não um objeto".

Há tanto vantagens como desvantagens no reuso de software conforme descrito em (SOMMERVILLE, 2010).

Entre as vantagens do reuso, destacam-se:

a) aumento da confiabilidade - os componentes já foram testados em diferentes ambientes, permitindo que falhas de projeto e implementação já tenham sido sanadas, reduzindo, portanto, o número de falhas quando do reuso.

b) redução de riscos de desenvolvimento - válido especialmente para reuso de subsistemas, com a redução das incertezas da estimativa de custo.

c) observância de padronização - componentes que implementem padrões, como, por exemplo, interface gráfica, podem ser reutilizados, padronizando toda a interface com o usuário.

d) aceleração do desenvolvimento - o reuso de componentes permite maior rapidez na implementação, uma vez que os tempos de desenvolvimento e validação são reduzidos.

Por outro lado, há questões relativas ao reuso que podem tornar-se problemas. Entre as principais, destacam-se:

a) aumento do custo de manutenção - se o código fonte do componente não está disponível, então os custos com manutenção tendem a aumentar em função de incompatibilidade com eventuais evoluções do sistema.

b) falta de ferramentas de suporte - ferramentas CASE geralmente não suportam desenvolvimento voltado para reuso.

c) síndrome "não-feito-aqui" - alguns engenheiros de software preferem reescrever os componentes ao acreditar que podem melhorá-los. Isto acontece em parte em função da confiabilidade e também porque consideram o desenvolvimento como sendo mais desafiador que o reuso.

d) manutenção de biblioteca de componentes - preencher uma biblioteca de componentes pode ser muito caro se não houver garantia de que outros desenvolvedores a usarão.

e) encontrar e adaptar componentes reusáveis - os componentes precisam ser encontrados na biblioteca, entendidos e, algumas vezes, adaptados para um novo ambiente.

O reuso de software pode ser feito em diferentes níveis, sendo que os principais são componentes, *frameworks* e COTS (*Commercial-Off-The-Shelf*).

Os *frameworks* são estruturas complexas, compostas de um conjunto de classes e interfaces entre elas. Normalmente não formam uma aplicação por si só. A criação de aplicações baseadas em *frameworks* consiste em agregar classes derivadas de suas classes abstratas. Todavia, a estrutura de um *framework* inclui uma complexidade que pode torná-lo negativo, devido ao esforço que se requer para o entendimento.

Os COTS normalmente são aplicações ou sistemas completos, reutilizados dentro de um sistema maior. Exemplo de um produto COTS tradicionalmente reutilizado é o sistema de gerenciamento de bancos de dados. Dificilmente uma empresa irá desenvolver um sistema de gerenciamento de banco de

dados próprio. Os ganhos em usar um sistema COTS são imensos, já que muito tempo de projeto e uso pode ser economizado ao se incorporar um sistema robusto no projeto. Porém, dificuldades com suporte do fornecedor, evolução do produto e interoperabilidade podem tornar seu reuso um tanto quanto difícil.

A seguir, serão detalhados conceitos e técnicas que auxiliam melhor a aplicação do reuso em software.

3.1 - Componentes de software

O uso de componentes representa um grande passo na modularização de sistemas de software, pois, permite o encapsulamento de funcionalidades, tornando-as independentes. Com isso, um componente que executa uma determinada função pode ser reutilizado inúmeras vezes, como parte de outros sistemas.

Um componente caracteriza-se por executar uma tarefa de forma bem delineada e apresentar interfaces muito bem definidas para comunicação com o sistema que o utilizará. Estas interfaces podem ser de entrada (*require*) ou saída (*provide*). A interface de entrada define o que o componente precisa receber do sistema que o usará, de forma a executar sua função. A interface de saída é a resposta que o componente devolve ao sistema requisitante, com base nos dados recebidos na entrada.

A forma como o componente executa sua funcionalidade não precisa ser conhecida por quem for utilizá-lo. Aspectos de projeto e implementação do componente são internos ao mesmo e irrelevantes aos usuários.

Desenvolver componentes voltados para reuso apresenta alguns desafios, uma vez que este processo implica adicionar complexidade ao componente, restringindo a sua usabilidade. Esta, por sua vez, será tanto maior quanto mais simplificadas forem as interfaces para uso do componente. Portanto, o projeto e a implementação de componentes requerem um balanceamento entre simplicidade no uso e uma eventual complexidade no domínio do problema implementado pelo componente.

3.2 - Padrões de projetos (Design Patterns)

O uso de componentes pode, em alguns casos, sofrer restrições em função de uma prévia definição e implementação de interfaces ou tipos. Neste caso, a reutilização de um componente pode tornar-se inviável ou reduzir sua eficiência. Uma solução para sobrepor esta limitação é o uso de *padrões de projeto* (GAMMA et al., 1994), que são mais abstratos e não incluem detalhes de implementação. Um padrão é uma descrição do problema e sua solução pode ser adaptada a diferentes situações.

Um padrão necessita incluir quatro informações:

a) um nome.

b) um problema no qual o padrão é geralmente aplicado.

c) as características do padrão.

d) as conseqüências da aplicação do padrão.

Padrões de projeto estão intrinsecamente ligados a orientação a objetos. Além disso, os padrões de projeto requerem um bom conhecimento para seu uso, como a habilidade de reconhecer situações genéricas onde um padrão possa ser utilizado.

3.3 - Linha de produto de software (Product Line)

Esta solução para reuso consiste em um conjunto de aplicações que compartilham uma arquitetura comum para um determinado domínio. Para satisfazer um segmento particular do domínio, algumas aplicações necessitam ser especializadas, o que pode requerer a criação de componentes adicionais ou a modificação de algum já existente. Todavia, o núcleo comum da família de aplicações é reutilizado.

Os tipos de especializações para uma família de produtos podem ser agrupados em:

a) especialização por plataforma: diferentes versões da aplicação são desenvolvidas para diferentes plataformas. A arquitetura comum é reutilizada, devendo-se apenas modificar os componentes que fazem interface com o hardware e o sistema operacional.

b) especialização de configuração: diferentes versões da aplicação são criadas para tratar diferentes dispositivos periféricos. Neste caso, os componentes que fazem interface com os periféricos deverão ser modificados.

c) especialização funcional: novas versões da aplicação devem ser criadas para tratar diferentes requisitos. Neste caso, os componentes que atendem as funcionalidades relacionadas aos requisitos deverão ser modificados. Pode haver a necessidade de se desenvolver novos componentes, caso os requisitos não estejam contemplados pela arquitetura do domínio.

Para maximizar o reuso, a arquitetura do sistema deve separar as funcionalidades comuns e as funcionalidades modificáveis quando da customização da linha de produtos.

3.4 - Arquitetura Dirigida a Modelos (Model Driven Architecture - MDA)

A arquitetura dirigida a modelos trata de um processo de desenvolvimento (OMG, 2003) criado em 2001 pelo *Object Management Group* que propõe uma solução que separa os detalhes da especificação de um sistema dos detalhes da implementação. A idéia central é a definição de um conjunto de modelos independentes de plataforma. A UML (*Unified Modeling Language*) é usada para descrever os modelos. Para representar um conjunto de modelos é necessária uma abstração ainda superior à modelagem, que é a metamodelagem, já que um metamodelo é um modelo de um modelo. A arquitetura MDA usa uma metalinguagem definida como *Meta Object Facility* (MOF). Esta família de metamodelos define um repositório de artefatos. Também é usado o *Common Warehouse Metamodel* para definição de metamodelos.

Uma vez criada a família de metamodelos independentes, um conjunto de regras de transformação é usado na conversão para uma plataforma específica. A Figura 4.1 mostra a arquitetura MDA, sendo que o anel externo mostra as plataformas para as quais o núcleo central de modelos pode ser convertido.

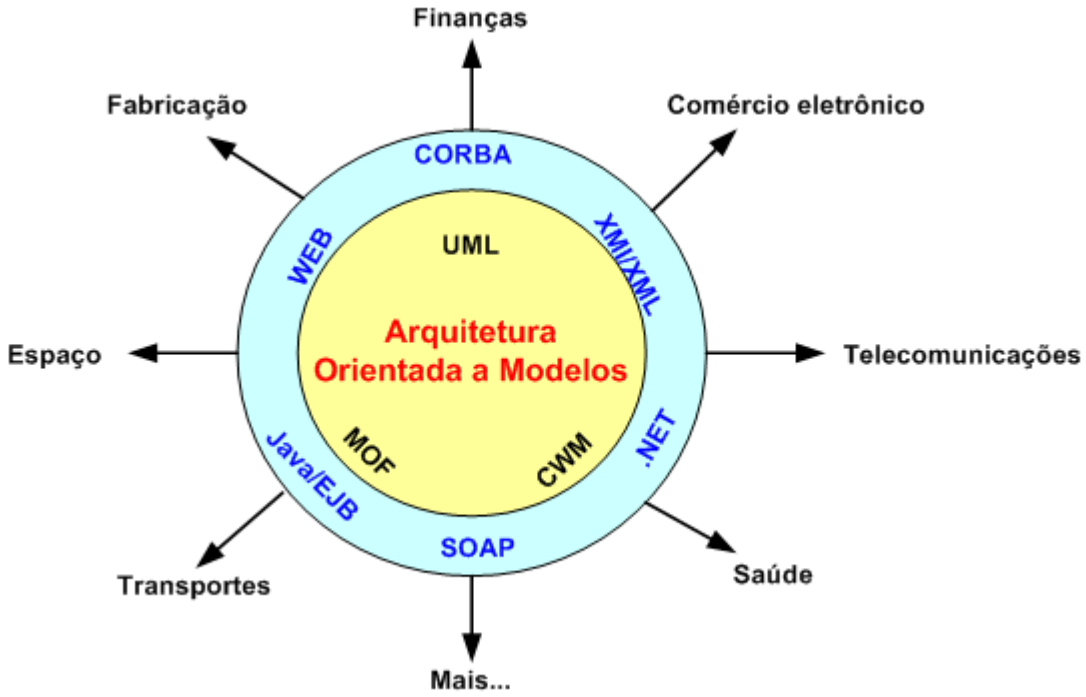

Figura 4.1 - Arquitetura MDA. ADAPTADO DE (OMG, 2003)

Este direcionamento a modelos guia todo o processo de entendimento, projeto, desenvolvimento, distribuição, operação, manutenção e modificação dentro da MDA.

O padrão SMP trata o conceito de independência de plataforma de maneira análoga ao MDA. No SMP, é definido um catálogo de modelos independente de plataforma, que é mapeado posteriormente para uma plataforma específica.

4 - CONCLUSÃO

O estudo sobre arquiteturas de simuladores de satélites foi elaborado com o objetivo de se tomar conhecimento e comparar soluções correntes em termos de simuladores de satélites que estão sendo desenvolvidos no âmbito da Agência Espacial Européia (ESA) com relação ao uso do padrão SMP (*Simulation Modelling Platform*). A arquitetura do simulador de satélites em desenvolvimento no INPE também é apresentada e comparada.

Neste estudo buscou-se elencar arquiteturas de simuladores desenvolvidos ou em desenvolvimento e foram observadas as soluções sendo implementadas, e as tecnologias adotadas.

Uma conclusão deste estudo é a evidente importância dos simuladores nas missões espaciais, suas aplicações em diferentes momentos do desenvolvimento de um satélite e a importância do reuso para diminuir custos e prazos, uma vez que o simulador, via de regra, deve estar pronto antes do lançamento do satélite que ele simula.

Entre as aplicações principais dos simuladores destacam-se:

a) testar e validar centros de controle de missão.

b) testar software de bordo ante da carga no satélite.

c) treinar operadores.

d) avaliar operações de contingência.

O estudo mostrou também um simulador diferenciado, por não estar vinculado diretamente a um satélite, o Simulador de Segmento Solo. Este simulador é destinado a validar a robustez e os limites de performance dos componentes de um segmento solo.

Para se atingir os objetivos de diminuição de custos e prazos através do reuso, a ESA, primeiramente definiu um núcleo de simulação chamado SIMSAT. Mais tarde, visando a uniformidade no desenvolvimento de simuladores, a ESA propôs um padrão para desenvolvimento de simuladores, o chamado SMP. Nesta altura, o núcleo de simulação SIMSAT passou por ajustes para se tornar compatível com o padrão SMP.

O estudo mostra que praticamente todos os simuladores no âmbito da ESA utilizam o SIMSAT e, como este foi adaptado ao padrão SMP, também os simuladores deverão ser compatíveis com o padrão. Considerando também que o SMP explicitamente promove reuso, portabilidade e interoperabilidade, os simuladores desenvolvidos sob o padrão beneficiar-se-ão destas características.

Quanto às técnicas de engenharia de software discutidas neste relatório, muitas delas apóiam diretamente o padrão SMP. Estas técnicas poderão ser utilizadas no desenvolvimento de simuladores para torná-los ainda mais reutilizáveis.

Finalmente, o estudo das arquiteturas de simuladores recentes adotadas em projetos de simuladores criados para a Agência Espacial Européia (ESA) permitiu comprovar o esforço na reutilização através da criação de um núcleo comum de simulação, na busca pela uniformização do desenvolvimento através da proposição de um padrão específico para tal e o uso de simuladores de satélites para diversos objetivos dentro de uma missão.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBROSIO, A. M.; CARDOSO, P. E.; ORLANDO, V.; BIANCHI NETO, J. Brazilian satellite simulators: previous solutions trade-off and new perspectives for the CBERS program. In: CONFERENCE ON SPACE OPERATIONS, 9., 2006, Rome. **Proceedings…** Rome, 2006, p. 7. CD-ROM. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m16@80/2006/08.21.15.01>. Acesso em 27 mai 2011.

BODEMANN, C. Software simulation for the XMM mission. In: **vega white paper** , 2004.

CÔME, H.; IRVINE, M. The XMM Simulator - The technical challenges. **ESA Bulletin**, v 96, 1998. Disponível em

< http://www.esa.int/esapub/bulletin/bullet96/COME.pdf>. Acesso em 30 nov 2011.

COUCH, M.; CINA, G.; GONZALES, O.; MONTRONI, G.; PECCHIOLI, M. SCOS-2000 Release 5, A milestone in the evolution of the MCS infrastructure at ESOC. In: CONFERENCE ON SPACE OPERATIONS, 9., 2006, Rome. **Proceedings…** Rome, 2006. Disponível em <http://pdf.aiaa.org/getfile.cfm?urlX=5%3A7I%276D%26XZ%22O%23S0WUWT%5B%5EPK%3B%3A4JL%22%0A>. Acesso em 29 nov 2011.

DELHAISE, F.; BRU, T. Innovative concepts to reduce costs of Mission Control and Simulator for Lisa Pathfinder. In: CONFERENCE ON SPACE OPERATIONS, 9., 2006, Rome. **Proceedings…** Rome, 2006. Disponível em: <http://pdf.aiaa.org/getfile.cfm?urlX=5%3A7I%276D%26XZ%22O%22S%20WUWT%5B%5EPK%3B%3A7%2AX%2C%0A. Acesso em 29 nov 2011.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Simulation modelling platform.** 2011.Disponível em: <http://www.ecss.nl/>. Acesso em: 19 jul 2011.

EICKHOFF, J. **Simulating spacecraft systems**. Berlin: Springer Aerospace Technology, 2009. ISBN: (978-3-642-01275-4).

EUROPEAN SPACE AGENCY (ESA). **Simulation model portability**. 2001. Disponível em: <http://www.esa.int/TEC/Modelling_and_simulation/TEC2DCNWTPE_0.html>. Acesso em: 14 jul 2011.

EUROPEAN SPACE OPERATIONS CENTRE (ESOC). **SMP 2.0 handbook**. EGOS-SIM-GEN-TN-0099, Issue 1, Revision 2, 2005. Disponível em: <http://www.eurosim.nl/support/manual_4_0/pdf/SMP_2.0_Handbook-1.2.pdf>. Acesso em: 19 jul 2011.

FONSECA, L. INPE como provedor de dados de observação da terra gratuitos. In: **MundoGEO#Connect**, 2011. Disponível em: <http://mundogeoconnect.com/2011/arquivos/palestras/leila_fonseca-inpe_como_provedor_de_dados_gratuitos_de_observacao_da_terra.pdf >. Acesso em 14 out 2011.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns:** elements of reusable objected-oriented software. 12. ed. 1997. ISBN (0-201-63361-2).

HOMEM, M. T; PIGNÉDE, M; MERRI, M.; REGGESTAD, V; PIDGEON, A.; MATUSSI, S. The GALILEO simulator: a major step in software technology from single spacecraft to constellation simulators. In: CONFERENCE ON SPACE OPERATIONS, 9., 2006, Rome. **Proceedings…** Rome, 2006. Disponível em

<http://pdf.aiaa.org/getfile.cfm?urlX=5%3A7I%276D%26XZ%22C%2BR%20%5FUWT%5B%5EPK%3B%3A7%3AP%26%0A>. Acesso em 29 nov 2011.

OBJECT MANAGEMENT GROUP (OMG). **MDA guide version 1.0.1**, 2003. Disponível em: <http://www.omg.org/mda/>. Acesso em: 14 nov 2010.

PRESSMAN, R. S. **Software engineering, a practioner´s approach**. Mcgraw-Hill, 2005.

REGGESTAD, V.; GUERRUCCI, D.; EMANUELLI, P.P; VERRIER, D. Simulator development: the flexible approach applied to operational spacecraft simulators. In: CONFERENCE ON SPACE OPERATIONS, 8., 2004, Montreal, 2004. **Proceedings…** Monteral: AIAA, 2004. Disponível em:

<http://www.aiaa.org/spaceops2004archive/downloads/papers/SPACE2004sp-template00139F.pdf >. Acesso em: 29 nov 2011.

ROLET, E.; CROENNE, D. SimGen: a future standard for simulations?. In: CONFERENCE ON SPACE OPERATIONS, 9., 2006, Rome**. Proceedings…** Rome, 2006. Disponível em:

<http://pdf.aiaa.org/getfile.cfm?urlX=5%3A7I%276D%26XZ%22C%2CS%40SUWT%5B%5EPK%3B%3A7%3AP%21%0A. Acesso em 29 nov 2011.

SOMMERVILLE, I. **Software engineering**. 6. ed. New York: Addison-Wesley, 2010.

STELLATO, S.; ROMANI, E. Ground segment simulator: a tool for mission design and life cycle management. In: DATA SYSTEMS IN AEROSPACE CONFERENCE, 2005, Edinburgh, UK. **Proceedings…** Edinburgh, 2005 . CD-ROM, p. 39. ISBN (92-9092-913-8)

EXTENSIBLE MARKUP LANGUAGE (XML). **Extensible markup language version 1.1**, 2006. Disponível em <http://www.w3.org/TR/xml11/>. Acesso em 26 Ago 2011.